UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

Metode de învățare profundă în calculul afectiv pentru identificarea expresiilor faciale și emoțiilor din voce

Coordonator ştiinţific
Prof. dr. Czibula Gabriela

Student
Alex-Răzvan Ispas

2022

BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DIPLOMA THESIS

Deep learning approaches for affective computing in facial expression and speech emotion recognition

Scientific supervisor
Prof. dr. Czibula Gabriela

Student
Alex-Răzvan Ispas

2022

Abstract

The main subject of this thesis is the field of Affective Computing. This thesis addresses the tasks of facial expression recognition and speech emotion recognition which are two of the most important tasks in the development of Affective Computing.

One of the main contributions of this thesis is the spectrogram image preprocessing approach that is used to fine-tune deep convolutional neural network architectures on RAVDESS audio dataset. Our model achieves a better performance than the current state of the art, having an average accuracy of 74.51%. A comparison with other preprocessing methods was also conducted. Regarding the facial expression recongition task, we introduce a preprocessing method with image interpolation and histogram equalization to increase the performance of fine-tuning on deep convolutional neural network architectures. The best performing model reached an average accuracy of 70.6%. Both methods are an element of originality.

Another original contribution is the design of *FASEL* which is an educational application that people can use to monitor their emotional states. With this application, people can test their cognitive skills by solving different tests and they can also monitor their emotional states, once they submit their answers. The purpose of the application is to promote a for of education centred on human feelings and needs.

This work represents the outcome of my original research. I have not published nor accepted any additional assistance from unauthorized sources on this work.

Alex Răzvan Ispas

Cluj-Napoca
03-06-2022

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Affective Computing represents the examination and development of intelligent systems which can interpret, identify, analyze and even simulate human feelings [Pic00]. The branch was born in 1995 when *Ross Picard* published her paper about *Affective Computing* where she motivated the desire of providing machines emotional intelligence as well as the ability to simulate **empathy** in order to make computers genuinely intelligent. There are different applications of *affective computing* in *artificial intelligence*, including *facial expression recognition* (FER), *speech emotion recognition* (SER) and *sentiment analysis*.

Due to the fact that the world is being exposed to continuous changes, humans find it difficult to adapt to the exponential growth of information in the society. Therefore, there are major discussions related to the subjects children are supposed to learn in schools. The knowledge which is taught in schools can become irrelevant in just a couple of years as big data algorithms are becoming stronger, replacing many human jobs and activities. That results into a lot of confusion and dysfunctional emotions which seems like people do not know how to manage them.

Daniel Goleman claims that, in the future, emotional intelligence will be a discipline that will be studied on a regular basis in all institutions [Gol96]. The fact that nowadays education is focused mostly on the students, rather than on the way teachers hold their lessons, is another reason of why we are heading to that future. In other words, the demand for emotional education starts to increase.

The contribution of this thesis comes from the image spectrogram fine-tuning approach on *RAVDESS* dataset in which the mel spectrograms are being converted from decibels to pixels in order to achieve a better performance on the pre-trained deep convolutioal neural network architectures. The speech model sets a new state of the art for *RAVDESS* dataset with an average of accuracy of 74.51%. Regarding facial expression recognition, our original contribution is the combination of the deep convolutional neural network architectures, that are used in the current state of the art on *FER2013* dataset, with preprocessed versions of the images that were interpolated and equalized through the pixel histogram equalization in order to achieve a fast and accurate training.

To validate the results, from an application perspective, we are also going to present *FASEL*, a system which tries to promote education centered on students' feelings and needs. The system monitors the primary six human emotions, through facial expressions and voice, in order to analyze the emotional states of the students while taking tests and quizzes. The data can be used by the students to monitor and receive self awareness, regarding their emotional states, and to commence actions for controlling their own feelings, before attending to a real test.

The rest of the thesis was structured as follows. Chapter 1 is presenting the background that was used in the thesis as well as the related work, meanwhile Chapter 2 presents the research related to the model used for *speech emotion recognition*. Chapter 3 presents the research regarding the model that solves the *facial emotional recognition* task and the details for the *FASEL* software is being presented in Chapter 4. The conclusion (Chapter 4.3.6) summarises the thesis and proposes future improvements at the end.

Chapter 1

Background

Affective Computing is a new field of research with high promises related to the well-being of humans. However, it faces many challenges which are concerned with the way people express their own feelings and needs.

This chapter will introduce the main concepts and notions which are used in this work as well as the review for the literature which connects with the two tasks of this thesis.

1.1 Speech Emotion Recognition

This chapter will present the background for Speech Emotion Recognition in the context of *RAVDESS* dataset

1.1.1 Psychological and Speech Perspective

From a **psychological perspective**, emotions can be perceived as a complex experience which implies consciousness, bodily sensation and even behaviour. They imply a synthesis of people's subjective experiences and behaviour, as well as their neurochemical body activities. In spite of the fact that more than 300 emotions were identified by researchers, many agreed on the principle of the *Pallete theory* which claims that all emotions are divided from six primary emotions the same way colours are divided from the three primary colours [EAKK11]. Those six emotions are anger, fear, happiness, sadness and surprise. From a **speech signal perspective**, the psychological and physiological changes are caused by different emotional experiences, which result into specific actions. Due to the fact that speech is one of the outcomes of emotional states, specific information can be collected by analyzing the vocal tract systems and the characteristics of the excitation source [KK13].

1.1.2 Dataset

For emotional speech databases, there is an issue regarding the quality of the database utilized in the development. There are three categories of speech corpora according to [KK13]:

- **Actor based** emotional speech databases are the most common corpora used in development. The advantage comes from the fact that the results can have an easier comparison. It also provides a wide range of emotions. A major disadvantage is the fact that acting speech illustrates the way emotions are meant to be expressed in the ideal scenario, not the way they are expressed in reality. In other words, they are episodic and they cannot be applied in many situations from real life.
- **Elicited (Induced)** emotional speech databases are much closer to the natural databases. They contain contextual information. However, most of the samples are artificial. A problem could be the fact that not all emotions might be present. The speaker might struggle to sound natural, if they know they are recorded.
- **Natural** emotional speech databases contains recordings which are naturally expressed during the recording. They are the most useful option for modeling real world emotions. The problem is once again the fact that not all emotions might be available. They also contain background noise and multiple emotions at the same time.

In the context of our problem, we are going to use *RAVDESS*, which is an actor based dataset. It offers support for both speech and multi-modal research. However, this thesis will focus straightly on the SER task when experiments will be performed on *RAVDESS*. It contains 7356 files recorded by 24 actors of both genders and it focuses on the six primary emotions. Therefore, RAVDESS is a suitable option for the task of speech emotion recognition.

1.1.3 Applications

There are many applications of speech emotion recognition during the daily life. Perhaps, one of the most used applications is the improvement of human machine interaction by making the speech more natural [LN05]. In the medical field, the emotional data of a patient's speech could be used by doctors in order to diagnose different mental disorders [KK13]. Even in education, the field on which this thesis is focused, teachers can use the data to adapt their lessons based on students' feelings and needs [VK03].

1.1.4 Challenges

There are a number of issues in the research process of speech emotion recognition. The main one comes from the fact that the word *emotion* is subjective and often perceived by different fields as being *vague*. Due to the fact that it is a process that comes spontaneously for each individual without any kind of effort, makes it difficult to define *emotion* objectively [SC06]. For instance, for the humanist branch of psychology, there is no difference between *emotion* and *sentiment*. For them, the process of *sentiment analysis* could be called *evaluation*. Therefore, it is challenging to come with a concrete definition.

1.2 Facial Expression Recognition

This chapter will present the background for Facial Expression Recognition in the context of *FER2013* dataset.

1.2.1 Pattern Recognition Problem

It was noticed that the facial reactions provide a good affinity of the emotional arousal. That is one of the main reasons which determined research teams to locate the emotions of humans in report to their expression. In spite of the fact that a concrete mapping between the affinity of emotion and a specific emotional state has not been proved, many researchers agree that the emotions displayed from facial expressions are universal.

The emotional reactions of a human could be conveyed through distinctive ways. If only one way of expressing emotions is taken into consideration when they are being recognized, the approach is called *unimodal*. Many times, emotions are not expressed properly from a specific channel. That results into frequent misclassifications. One way through which the phenomenon could be avoided is to make a prediction from multiple sources. The method is also known as *multimodal* emotional recognition [KC15]. In this thesis, we are going to focus on the *unimodal* approach in both scenarios.

1.2.2 Facial Methods

Regarding the methods which are used in the process of facial expression recognition, the dominance is asserted by two methods. The first one is the holistic method, which takes a global approach by attempting to replicate the whole human anamorphosis. The second method is the analytical approach which are locating people's eyebrows, mouth, nose and other features which are shared by every human. The main objective is to create geometrical connections between the features of the face to provide an accurate prediction of the face expressions [ACL04].

In the analytical method, the process of feature extraction is made by using two methods: the geometrical methods and the appearance methods. The geometrical method attempts to identify the face geometry by splitting it into simple shapes meanwhile the appearance method is relying on image filters in order to create feature vectors. [GPH18]

1.2.3 Dataset

The experiments will be performed on *FER2013*. It was firstly presented in the challenge of ICML 2013 workshop on facial expression recognition [GEC+13]. It was built by using the Google image API which could find relevant images based on emotional keywords. Challenges occur in the dataset due to the fact that it contains many faces which differ in in age as well as pose. Even humans face problems to identify the emotions as the average accuracy of a human is 65 ± 5 [GPH18]. Figure 1.1 displays some samples from *FER2013* dataset.

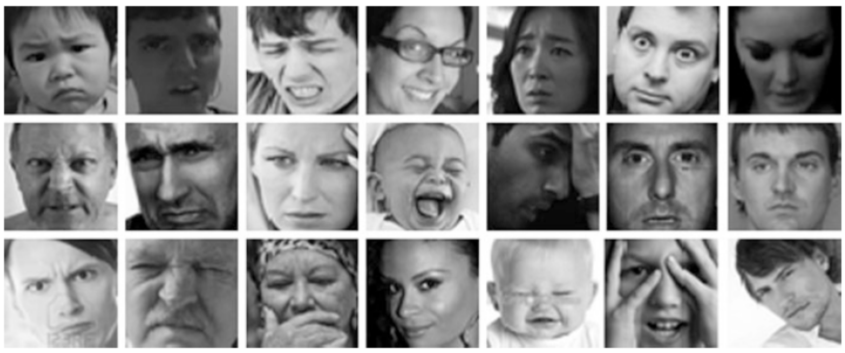


Figure 1.1: Samples from FER2013 [GPH18]

1.3 Deep Learning

This section will present the reasons why deep learning is preferred for the tasks of speech emotion recognition and facial expression recognition. A mathematical formalisation will be provided for a general problem that is solved with deep learning.

1.3.1 Why Deep Learning?

In order to understand properly why someone might choose deep learning as a solution to an artificial intelligence problem, it is important to take a step back and look at the types of data that people use. There are mainly two groups of data:

- **Structured data** which is mainly data that was handled by humans. They are usually stored into databases and are used in problems that are closer to data science. Most of the time, these sets of data can be fit with traditional machine learning methods
- **Unstructured data** which consist of images, sounds or texts. These are more complex representations of data and they require deeper methods to achieve a high performance.

Another reason which makes deep learning get over the competition is the fact that in the last 30 years people collected more data than what traditional methods could handle. The more data is collected and the larger a neural network becomes, the higher the chances are to achieve better performance [GBC16]. In other words, scale is one of the factors which drives deep learning into progress. Figure 1.2 shows the scale of performance in report with the amount of data for traditional machine learning and neural networks of different size.

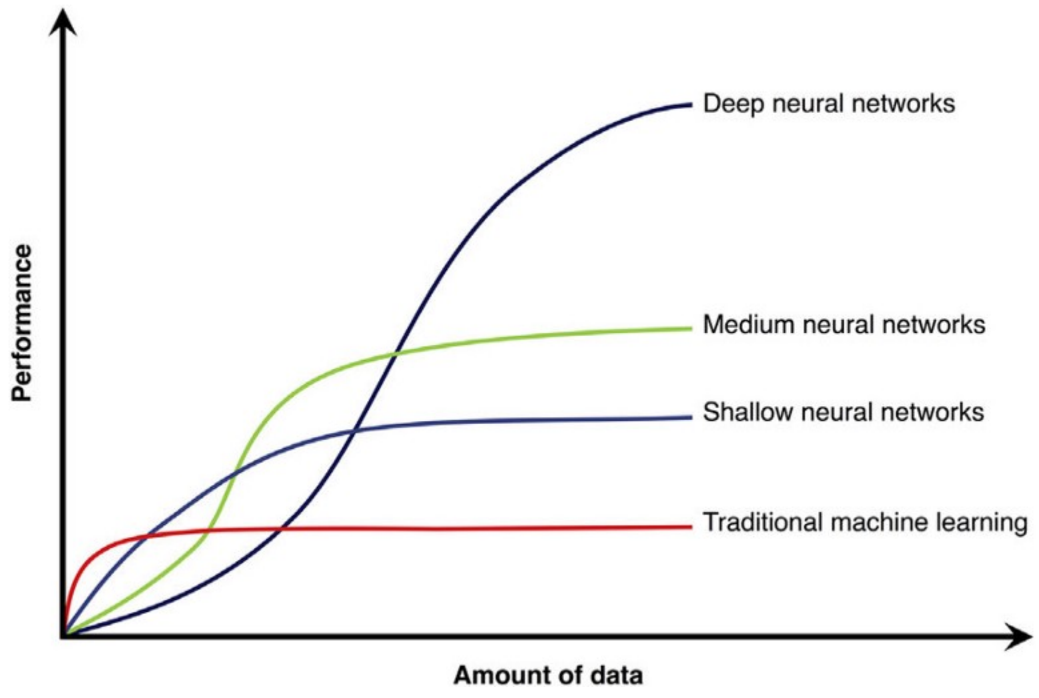


Figure 1.2: Deep Learning performance scale [Joo19]

1.3.2 Deep Learning Problem Formalisation

In a deep learning problem where we want to make a prediction, there is a dataset (X, Y) where X is the input data with $X \in R^n$ and y contains the labels that are being predicted with $Y \in \{1,2...,m\}, m$ being the total number of labels. There will be n pair of training examples $(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)})...(x^{(n)},y^{(n)})$. A deep learning problem provides the input and the target. The task is to find the function that is represented by the neural network, which can fit the data.

Let $w = \{w^{[1]}, w^{[1]}, w^{[2]}...w^{[l]}\}$ be the weights on each layer, $b = \{b^{[1]}, b^{[1]}, b^{[2]}...b^{[l]}\}$ the bias values for each layer and $a = \{a^{[1]}, a^{[1]}, a^{[2]}...a^{(l)}\}$ the activation function for each layer. Each hidden layer has multiple neurons. Therefore, let w be defined as $w^{[l]} = \{w_1^{[l]}, w_2^{[l]}, w_3^{[l]}...w_k^{[l]}\}$, where l represents the lth layer in the neural network meanwhile k represents the kth neuron in the respective layer. The same notation applies to both the bias and the activation functions.

The forward propagation has two steps for each hidden layer. Let $z^{[l]} = w^{[l]T} * a^{[l-1]} + b^{[l]}$ be the information propagated for the neurons of the lth layer. Dataset X is used in the case of a^0. The second step is $a^{[l]} = g(z^{[l]})$, where $g^{[l]}$ is the activation function for the lth layer in the neural network. In general, the activation will be Relu which stands for *Rectifier Linear Unit* and has the formula $ReLU(x) = max(0,x)$. The final layer will be activated by *Softmax* which has the formula Softmax$(z_i^{[l]}) = \frac{\exp(z_i^{[l]})}{\sum_j \exp(z_j^{(l)})}$.

Due to the fact that our models contain multiple labels, let the cross entropy loss function be defined as $CE(y,\hat{y}) = -\sum_{j=1}^{M} y_j \log(\hat{y}_j)$, where y is the ground truth of label j and $\hat{y}$ is the predicted probability observation for label j. The cost function is defined as $\mathcal{J}(w,b) = \frac{1}{n}\sum_{i=1}^{n} CE(\hat{y}^{(i)}, y^{(i)})$, which shows the performance of a neural network on the training set. The goal of the neural network is to minimise the result of the cost function as much as possible.

The process of learning can be performed through backpropagation which consists of calculating the gradient $dW^{[l]}$ and $db^{[l]}$ for the weights and bias of each layer. The weights and bias for the lth layer are updated with the following two formulas: $W^{[l]} = W^{[l]} - \alpha * dW$ and $b^{[l]} = b^{[l]} - \alpha * db$, where α represents the learning rate of the neural network.

1.4 Related Work

The field of affective computing is relatively young, having a gradual shift to deep learning due to the fact that it proved to be more efficient while working with unstructured data. This section will present the current literature for SER and FER, as well as other methods that can be used for a possible improvement.

1.4.1 SER Literature

In spite of the fact that RAVDESS is a multimodal dataset for emotional recognition [LR18], there were different approaches which focused solely on audio. There were deep covolutional neural network architectures which attempted to combine multiple features, including the MFCC, chromagram, mel spectrogram, contrast as well as the tonnetz representation [IDY20]. In other researches, they tried to increase the performance by combining CNN feature extractors with Long Short Term Memory (LSTM) cells [ZJL19]. That resulted into a better understanding of the contextual dependencies in the audio features. The best performing model on RAVDESS's audio is *ERANN* [VBV21], which implemented modified residual blocks to boost the inference efficiency by increasing the stride size from 2 to 4. Another remarkable approach in the field of audio recognition, was to convert the mel spectrograms from decibel to images in order to use pre-trained deep convolutional neural network architectures, such as VGG19, to increase the feature extraction process [DSL+18]. Even though the method was tested on datasets of sound recogition, it can be replicated for emotional recongition tasks.

1.4.2 FER Literature

In the case of FER2013 dataset, deep convolutional neural network architectures are the preferred option in the recent years. Indeed, the winner of the ICML workshop, "Challenges in Representation Learning", relied on a support vector machine solution with a L2 loss function [GEC+13]. A Romanian team also provided an efficient solution which did not rely on any feature learning and used multiple kernel learning instead [IPG13]. It was the best solution without feature learning and the fourth best solution in the whole contest. Three years later, a research team from Vienna tried to enhance the performance on the dataset by implying face detection, facial landmark detection and contrast correction in combination with deep convolutional neural network architectures such as VGG, Xception and ResNet [PK16]. Every architecture performed better than the previous competitors, the best architecture being VGG which had an accuracy 72.4%. An additional ensemble with 8 deep CNNs was performed which increased the accuracy to 75.2%. In spite of the remarkable performance, an ensemble model which consists of 8 deep convolutional neural networks consumes quite a lot of resources. Recent attempts made hyperparameter optimisations to the convolutional layers to achieve a similar performance with a lighter version of VGG architecture [VGG21]. Other researches performed in the recent years made experiments of combining self-attention modules and convolutional neural networks. LHC-Net [PBBG21] implemented local multi head channels in the architecture of ResNet34v2 and reached the performance of 74.42%. The

LHC block is flexible and it can be implemented in multiple architectures.

Chapter 2

SIASER: Spectogram Image Approach for Speech Emotion Recognition

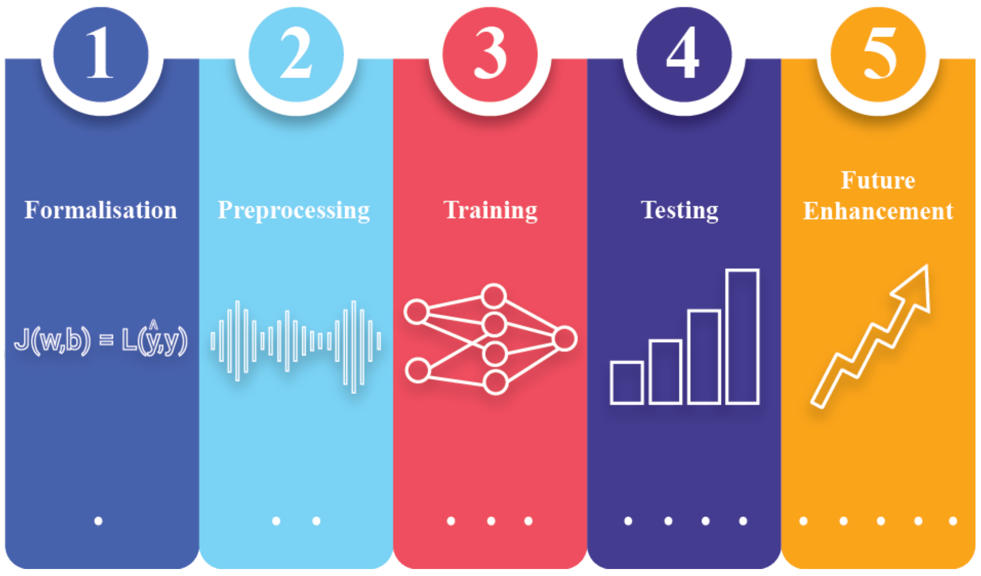


Figure 2.1: SER methodology summary

The main challenge regarding speech emotion recognition comes from the fact that, in spite of the utilised representation for the given data, the information is not always complete.

The *mel-frequency cepstral coefficients* (MFCC) provide a wide structure of the analyzed spectrum. Moreover, they are suitable for models that require either speech or music preprocessing. On the other hand, they lack robustness to noise, compared to the *gammatone frequency cepstral coefficients* (GFCC) [ZW13].

Chromagrams are useful for observing the pitch sound of each person. The pitch sound property can illustrate the height of the sounds, providing a better judgement [KNSS19]. Likewise, they can efficiently approximate the equal-tempered scale value. [She64].

Mel spectrogram is a closer representation of the way humans perceive sounds as they interpret them in a logarithmic way. However, it combines all sounds assuming that they belong to a single source, instead of separating them on distinct layers [Rot18].

In this chapter we are introducing the methodology for our approach on *RAVDESS* dataset. Figure 2.1 illustrates the summary of our methodology.

2.1 Formalisation

.

The approached problem in this chapter is defined in the following way: there are n audio files recorded by different professional actors of both genders who vocalised a number of statements. There are m emotions which the actors reproduced while repeating the statements. The goal is to create a model that can identify the emotions based on people's vocal statements.

The mathematical model of the problem is defined like following: let $\mathcal{A} = \{A_1, A_2, A_3, A_4 \cdots, A_n\}$ represent the domain of audio files used in the problem. Furthermore, let $\mathcal{E} = \{happy, sad, angry, fearful, surprise, disgust\}$ be the array of emotions which is disposed. Let $\mathcal{R} = \{R_1, R_2, R_3, R_4 \cdots, R_n\}$ be the domain of representations that the audio files can take and P defined on $\mathcal{P}\colon \mathcal{A} \to \mathcal{R}, M(a) = r$ is the preprocessing function which can transform an audio file in one of the supported features.

Now let us assume that there is a model M which predicts the emotion of the given audio file, defined on $\mathcal{M}\colon \mathcal{R} \to \mathcal{E}, M(r) = e$ where $a \in \mathcal{R}$ and $e \in \mathcal{E}$. The model is going to be a neural network whose cost function $\mathcal{J}(w, b) = \frac{1}{n}\sum_{i=1}^{n} CE(\hat{y}^{(i)}, y^{(i)})$ is going to be minimised for a better performance. Finally, there is $\mathcal{S} = \{accuracy, precision, recall, F1 - score\}$ which represent the metrics that are used to test the performance of the model. It is known that the audio files have a single channel when they are fed into the model.

2.2 Data Preprocessing and Representation

Considering the fact that many audio features can contribute to the performance of the model, there are more preprocessing techniques which should be taken into account.

2.2.1 Mean Average

It was highlighted above that each feature has its own advantages and disadvantages. Therefore, a possible solution could be to combine a wider range of audio features instead of testing only one, as they offer a wide range of sound characteristics. This can provide higher chances to obtain proper information regarding each sound sample [IDY20]. Therefore, the first method which could be used is the *mean average values* of each y-axis for all three features as shown in Algorithm 1. The y-axis reflects the measurement unit of each representation while the x-axis represents the time. Beside that, the audio is converted to a single channel to reduce the complexity.

```
Input:  files - audio files
Output: results - the list with mean average of MFCC, chroma features and
        mel spectrogram for each audio file

$results \leftarrow []$

for file in files do
    $soundfile \leftarrow readSoundfile(file)$
    $soundfile \leftarrow toMono(soundfile)$

    $features \leftarrow []$

    $mfccs \leftarrow mean(mfcc(soundfile))$
    $chroma \leftarrow mean(chroma(soundfile))$
    $mel \leftarrow mean(melspectrogram(soundfile))$
                        /* getting the mean for the three features */

    $features \leftarrow stack(features, mffccs)$
    $features \leftarrow stack(features, chroma)$
    $features \leftarrow stack(features, mel)$
                                               /* stacking the features */

    $results \leftarrow append(results, features)$
end
```
Algorithm 1: Mean average features algorithm

2.2.2 Pure Spectrogram

As mel spectrogram scale supports a consistent representation for the scale of human auditory system [SB17], experiments that used solely mel spectrograms were applied in order to find features that are more relevant than others. Both, *the mean average value* and the *classic representation* were taken into account during the experiments. For *pure mel spectrograms*, an additional feature scaling is applied in order to reduce the domain between 0 and 1 and ease the process of learning.

2.2.3 Spectrogram Image

A final attempt which is pursued in this work is to convert the mel spectrograms from decibels into image pixels so that they can be used for fine-tuning in deep convolutional neural networks architectures [DSL+18]. If some audio files will be longer than others, a padding consisting of white pixels will be added to match the size of the longest audio file.

The representation of the spectrogram will contain the frequencies in hertz during time, for each audio file, meanwhile chroma features will represent the pitch height of the twelve classes used in the octave invariance [KNSS19]. Regarding the mel spectrogram, there will be two representations which will be used in the fine tuning approaches.

The first representation will be provided by the algorithm of extracting the mel features from an acoustic signal. First of all, the audio signal is converted from time to frequency with the help of *short-time fourier transform*. Afterwards, the spectrogram envelope is produced by the Mel filter banks. Finally, the envelope is converted from frequency to decibels to provide a logarithmic representation which is close to the human auditory system. The process can be observed in Figure 2.2.

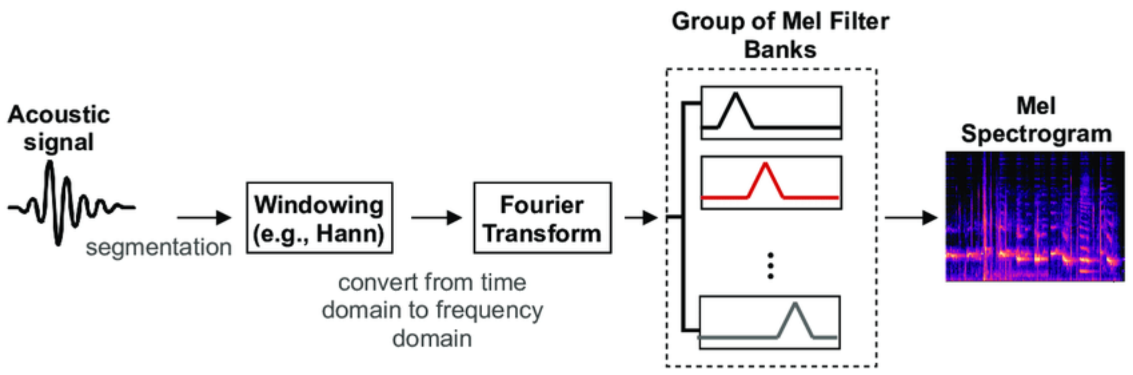


Figure 2.2: The process of extracting the mel spectrogram [HFQ+21]

The second representation will include the previous algorithm, including the conversion into decibels. Additionally, the frequency will be converted into pixels, so that the spectrograms can be fed into pre-trained convolutional neural network architectures. After the mel spectrogram is produced, the frequency is converted with a logarithmic function and a small value is added to avoid the values of 0. Therefore, a *min-max scale* is being performed to fit the domain into an 8 bit range which is then used to represent the pixels for each human eye channel. Once the spectrogram is converted into pixels, the image is flipped to get the lower frequencies at the bottom of the image and the colours are inverted to spot the intense sides of the spectrogram with dark colours. The process can be observed in Algorithm 2.

```
Input: soundfile - audio file
Output: image - the image of the spectrogram

$mel \leftarrow melspectrogram(soundfile)$        /* getting the spectogram */
$mel \leftarrow log(mel + 1e-9)$   /* converting the spectogram into decibels
 and adding a small value to avoid log(0) */

$image \leftarrow minMaxScale(image)$  /* scale the values between 0 and 255
 to obtain the pixels) */
$image \leftarrow flip(image)$  /* flip the image to put lower frequencies at
 the bottom */
$image \leftarrow invertPixels(image)$     /* invert pixel colours to make the
 intense regions dark */
```

Algorithm 2: Spectrogram to greyscale image

As many of the deep convolutional neural networks that will be used were trained on images which had three channels, every image will be converted to three channel grayscale images. As each audio file in *RAVDESS* ranges between 4 and 5 seconds, there will be differences in the width of each image. That is why an additional padding will be included in order to be certain that each image has the same *height* and *width* and to make sure that the training set has the same input size for each sample. Figure 2.3 shows the preprocessed spectogram image.

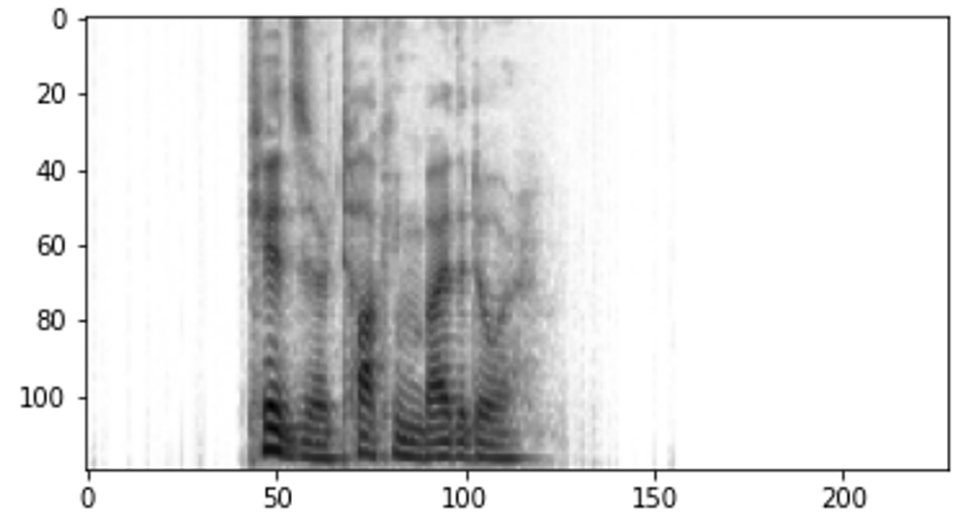


Figure 2.3: Mel spectogram converted into grayscale image with padding

2.3 Training

There were three neural network architectures which were used in the process of training. Due to the fact that we have three preprocessing methods, various combinations were performed in order to obtain high results.

Most of the experiments used *Adam* as the main optimizer. Compared to other stochastic optimizers which do not adapt during the training, *adam* relies on the *momentum* to accelerate the values of the gradient descent and by the *root mean square propagation* which is adapting the learning rate based on the recent changes in the gradient's magnitude [KB15]. Likewise, the model can convergence much faster and use a lower amount of resources to reach the desired performance.

From a mathematical point of view, *Adam* calculates both the exponentially weighted average of the previous gradients as well as the the exponentially weighted average of the squares in the previous gradients. Let v and s be the two vectors where the two exponentially weighted averages are stored for each layer. Now, for each hidden layer $l = 1, ..., L$:

$$\begin{cases} v_{dW^{[l]}} = \beta_1 v_{dW^{[l]}} + (1-\beta_1)\frac{\partial \mathcal{J}}{\partial W^{[l]}} \\ v^{corrected}_{dW^{[l]}} = \frac{v_{dW^{[l]}}}{1-(\beta_1)^t} \\ s_{dW^{[l]}} = \beta_2 s_{dW^{[l]}} + (1-\beta_2)(\frac{\partial \mathcal{J}}{\partial W^{[l]}})^2 \\ s^{corrected}_{dW^{[l]}} = \frac{s_{dW^{[l]}}}{1-(\beta_2)^t} \\ W^{[l]} = W^{[l]} - \alpha \frac{v^{corrected}_{dW^{[l]}}}{\sqrt{s^{corrected}_{dW^{[l]}}}+\varepsilon} \end{cases}$$

where:

- t is the number of steps made by Adam;
- β_1 and β_2 are hyperparameters for the two mentioned;
- α is the optimizer learning rate;
- ε is a small number used to avoid a possible division by zero when the denominator is very small.

2.3.1 Neural Network Architecture

The first architecture is a simple neural network consisting of two dense layers of 30 neurons each and two dropout layers with a rate of 20% in order to avoid overfitting [Gyl19]. The output layer is a dense layer with as many neurons as emotions are supposed to be predicted. The activation function for each dense layer is ReLu. *Adam* was used as optimizer and categorical crossentropy for the loss function.

2.3.2 Convolutional Neural Network Architecture

The second architecture is a convolutional neural network. The architecture has a bigger depth compared to the previous one. It has two blocks that consist of both two convolutional layers and a max pooling layer. Following this, the information is being flattened to a one dimensional vector in order to connect and feed the information through two dense layers and a output layer. The first block has convolutional layers of kernel size 5, meanwhile the second one has kernel size filters of length 3. Each layer has 32 filters in total. The max pooling layers have a pool size and stride of 2. The dense layers consist of 512 neurons. The activation function, for each dense layer, is ReLU, meanwhile the output layer is activated by the softmax function. In order to boost the learning process for each dense layer, they were initialized with He as it helps a lot of models to converge, in comparison with Xavier method whose gradient values diminish whenever the neural network becomes deeper [HZRS15] . Figure 2.4 illustrates the layers of the architecture.

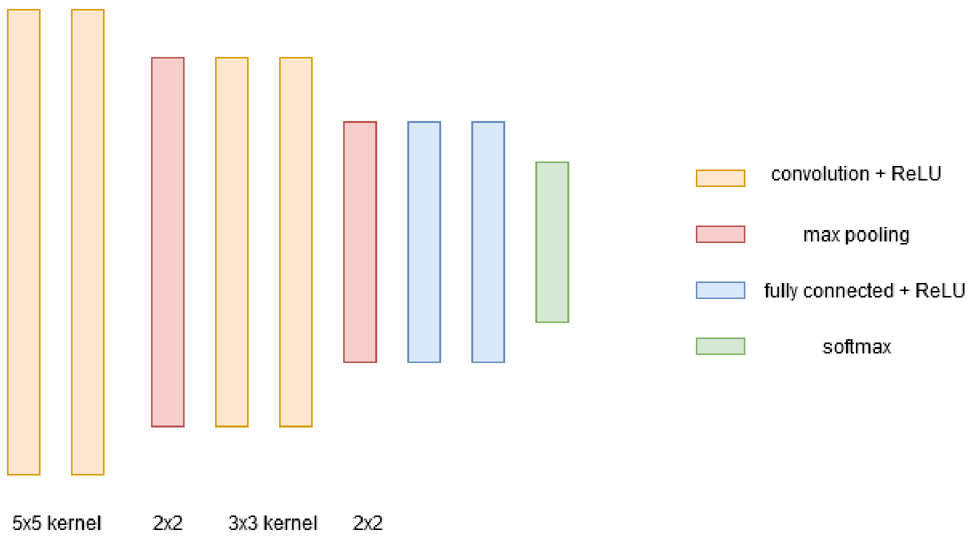


Figure 2.4: CNN architecture for spectrogram images

2.3.3 VGG19

Another approach was to perform transfer learning on a deep neural network. Therefore, the next architecture relied on the deep architecture provided by *VGG19* which is shown in Image 2.5. The architecture came up as the successor of *AlexNet-2012*. While *AlexNet-2012* contains convolutional layers of kernel size 11 and stride 4 [KSH12], *VGG19* uses 3 by 3 filters in the whole network. Two filters with kernel size 3 are followed by a filter with kernel size 5, resulting into a better receptive area when we shift to larger size kernels [SZ14]. Beside that, the number of weights are significantly reduced compared to *AlexNet*.

The input size was 120 by 228 pixels with three channels for each image as the *VGG19* architecture provided by Keras is constrainted by the fact that it was trained on ImageNet which is a dataset that contains three channel images [Cho21].

The weights of the VGG architecture were frozen, in order to learn spectrogram features, The architecture output is flattened and connected to two dense layers with 2000 neurons each and one dropout layers for regularization.

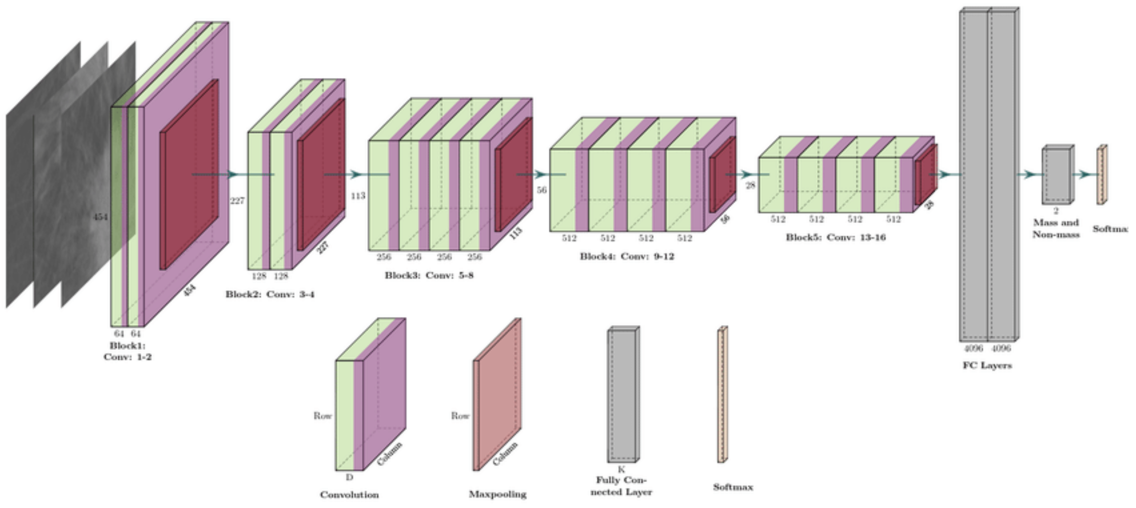


Figure 2.5: VGG19 architecture [HA+19]

2.4 Testing

This section will present the experimental setup and the result analysis for the experiments that were performed.

2.4.1 Experimental Setup

Considering that *RAVDESS* dataset contains only 7356 audio file, 80% of the data were used for the train set meanwhile 20% were left for the test set. Additionally, *k-fold cross validation* was performed to avoid the possibility of overfitting when the models are encountering general audio files [JWHT13].A mini-batch of 64 was implemented in order to converge faster to the desired loss error.

The first model was trained for 250 epochs with a learning of 0.01 and reached an accuracy of 68.23% on four emotions meanwhile a second training had an accuracy of 51.49% accuracy with the same set up and eight emotions.

There were two training experiments performed on the CNN architecture which used spectrogram images as the input data. One of them was performed with four emotions that were opposite on the spectrum (happy, calm, fearful, angry) while the other one used all emotions. In 10 epochs, the architectures achieved the average accuracy of 99.02% and 99.25% respectively.

After 30 epochs with a learning rate of 5e-5, the accuracy of VGG19 reached a 99.05% accuracy on the training set, meanwhile the loss reached the value of 0.0376. The final attempt for *greyscale spectogram images* was to fine-tune the VGG19 architecture by adjusting the weights for the current dataset. Thus, only the first 12 layers of the VGG19 architecture were frozen.

In order to make a clear comparison between the performance of the architecture while it uses spectrograms in decibels and while it uses spectrograms in pixels, we made one more experiment with the same set up to check the performance. This time, the architecture was loaded without *ImageNet* weights as they are not relevant for the distribution of our data. To accelerate the performance when the model converges, a learning rate decay scheduler was implemented to reduce the learning rate with a factor of 0.5 when the model does not increase the performance in less than three epochs.

2.4.2 Experimental Results

Table 2.1 shows the results for each experiment in report with the current state of the art. Beside *accuracy*, we also decided to calculate *precision*, *recall* and the *F1-score* to check the performance during both training and testing.

Accuracy is giving the report between the number of predictions which were correct and the total number of predictions. *Precision* is the number of predictions that are true positive divided by the total number of positive predictions. *Recall* represents the report between the predicted number of true positives and the sum between all true positives and false negatives [Rah21]. Finally, the *F1-score* is used to calculate the harmonic mean between the precision and recall. The F1-score formula is the following:

$$F1-score = 2 * \frac{precision * recall}{precision + recall}$$

There were high results for the fine-tuning approaches, this being the reason for trying an additional experiment with spectrograms in decibel scale. In order to make sure that we were not overfitting to a specific train/test split, k-fold validation was performed on the experiments which used spectrogram images and spectrograms in decibel scale. Like this, metrics such as the 95% *confidence intervals* (CIs) could be calculated in order to offer a more accurate result. Algorithm 3 illustrates the steps for performing the k-fold cross validation [Bro20].

In spite of the fact that the mean average method showed promising results for

Architecture	**Preprocessing**	**Nr emotions**	*Acc*	*Prec*	*Recall*	*F1*
Simple ANN	Mean Average	4	68.23	68.23	68.23	68.19
Simple ANN	Mean Average	8	51.49	51.49	51.49	51.49
Simple CNN	Spectrogram Image	4	64.43 ± 5.36	65.19 ± 5.17	63.91± 5.11	46.10
Simple CNN	Spectrogram Image	8	50.48 ± 3.7	52.29 ± 3.39	48.53 ± 4.9	45.49
VGG19 Transfer	Spectogram Image	8	59.71 ± 3.52	60.5 ±3.78	58.81 ± 3.56	57.25
VGG19 Fine-Tuning	**Spectogram Image**	**8**	**74.5 ± 1.3**	**75.66 ± 1.69**	**73.81 ± 1.13**	**77.97**
VGG19	Pure Spectrogram	8	74.44 ± 4.9	75.5 ± 4.11	73.6 ± 5.76	75.69
ERANN-6 [VBV21]	-	8	74.30	-	-	-

Table 2.1: Testing results for each experiment in report with the current state of the art. 95% CIs are used for the results.

```
**Input:** dataset - preprocessed dataset
**Input:** k - the number of folds
**Output:** score - calculated the average score for a specific metric

$shuffled \leftarrow shuffle(dataset)$ $groups \leftarrow splitGroups(dataset, k)$

$scores \leftarrow []$
**for** $group$ ***in*** $groups$ **do**
    $testSet \leftarrow group$ $trainingSet \leftarrow remainingGroups(groups, group)$

    $model \leftarrow model.fit(trainingSet)$ $metric \leftarrow model.predict(testSet)$

    $scores \leftarrow append(scores, metric)$
**end**

$score \leftarrow mean(scores)$
```

Algorithm 3: K-fold cross validation

opposite emotions, it faced difficulties in learning all fours emotions. It also requires more than 200 epochs to show some promising results while the other methods required 35 epochs on average.

On the other hand, the spectrogram fine-tuned method surpassed the state of the art accuracy with more than 4%. Overall, the methods which favour spectrograms perform much better than the ones which combine multiple features. However, there is still a high variance between the training and testing values due to the fact that the samples do not come from the same distribution. In addition, it is quite difficult to use such a model in a development scenario as the emotions conveyed by the actors are portraying emotions in the ideal shape.

2.5 Future enhancement

In this chapter, we proposed a spectrogram image based solution for *RAVDESS* dataset. We introduced an algorithm through which you can convert audio files to image spectrograms and we compared the performance of our fine-tuning approach with other data representations and state of the art models. The main aim of this research was to find methods through which audio models can be fine-tuned with pre-trained architectures.

For future work, experiments on natural speech data will be performed in order to check how well a model like this one can perform in a real scenario. Beside that, a late fusion approach between multiple architectures will be implemented to enhance the final prediction.

Chapter 3

FAFERII: Fine-tuning Approach for Facial Expression Recognition in Interpolated Images

One of the main challenges related to *FER2013* dataset comes from the fact that its labels are unbalanced [SPG19]. There are more than 8000 images for *happy* meanwhile, there are less than 600 images for *disgusted* and only 4000 for *surprised*. The average between the other emotion samples is around 5000 (Figure 3.1). In addition, all images are 48 by 48 pixels which makes it difficult for deeper networks to learn remarkable features.

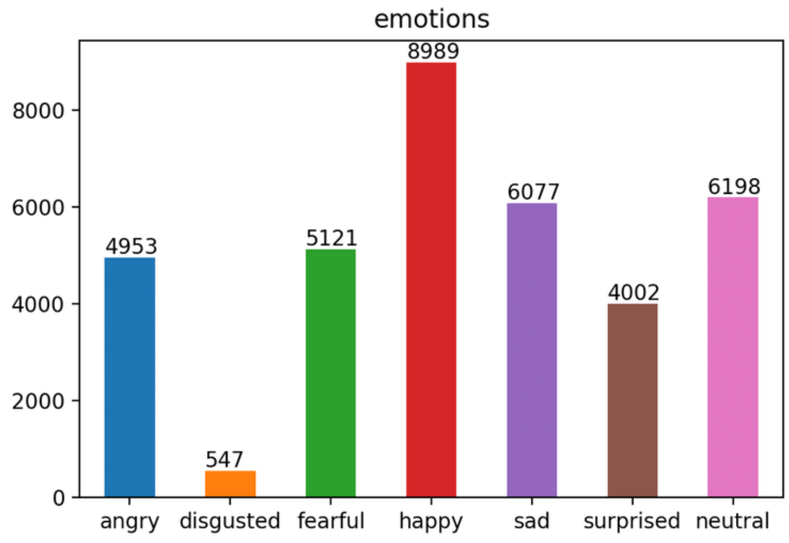


Figure 3.1: FER2013 emotion distribution [SPG19]

It is crucial to handle these two issues before training, as they are going to influence dramatically the process of training as it will be shown in the following chapters. This chapter introduces the methodology for our approach on *FER2013* dataset. Figure 3.2 illustrates the summary of our methodology.

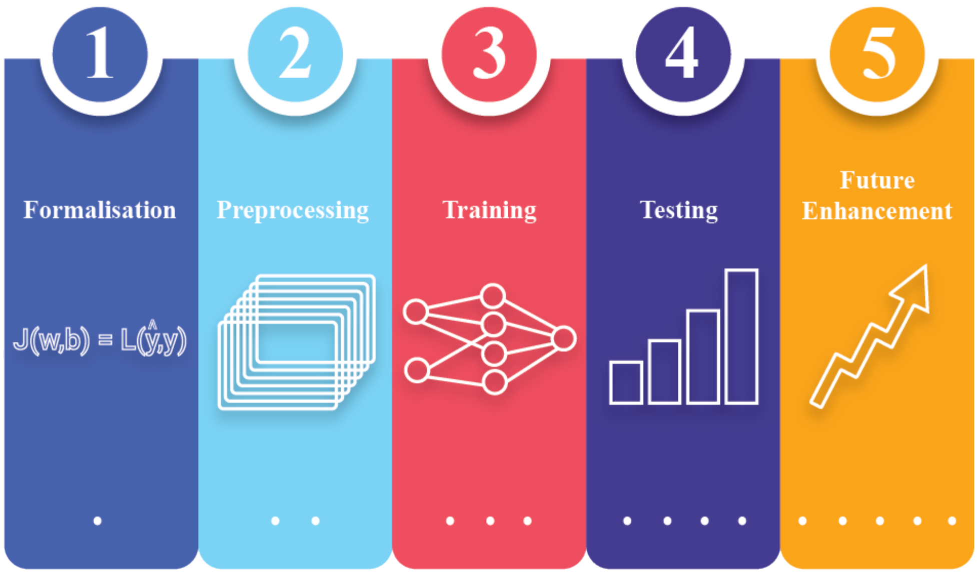


Figure 3.2: FER methodology summary

3.1 Formalisation

The problem is defined in the following way: there are n images collected from different sources on the internet, all of them representing one of the m given emotions. The task is to create a model which can predict the emotions based on the facial expressions made by people.

Let $\mathcal{I} = \{I_1, I_2, I_3, I_4 \cdots, I_n\}$ be the domain of given images that contain different human facial expressions. Now, let us define $\mathcal{E} = \{happy, sad, angry, fearful, surprise, disgust, neutral\}$ as the array of emotions which needs to be identified. There is a map function $\mathcal{B}\colon \mathcal{I} \to \mathcal{E}$ so that $\forall I \in \mathcal{I}, B(I) \in \mathcal{E}$ which ensures that every given image has a facial expression which belongs to one of the seven emotions. Let $\mathcal{R} = \{R_1, R_2, R_3, R_4 \cdots, R_n\}$ be the domain of representations that the images can take before training and P defined on $\mathcal{P}\colon \mathcal{I} \to \mathcal{R}, P(i) = r$ is the preprocessing function which can transform an image in one of the supported representations.

Now let us assume that there is a model M which predicts the emotion of the given image, defined on $\mathcal{M}\colon \mathcal{I} \to \mathcal{E}, M(i) = e$ where $i \in \mathcal{I}$ and $e \in \mathcal{E}$. Once again, the model is going to be a neural network whose cost function $\mathcal{J}(w,b) = \frac{1}{n}\sum_{i=1}^{n} CE(\hat{y}^{(i)}, y^{(i)})$ is going to be minimised for a better performance. . Finally, there is $\mathcal{S} = \{accuracy,$

$precision, recall, F1 - score\}$ which represents the metrics that are used to test the performance of the model. The images are given in a 48 by 48 pixel format.

3.2 Data Preprocessing and Representation

3.2.1 Data Augmentation

As it could be seen in Figure 3.1, there is a problem regarding the distribution of each label. If labels like *disgusted* are not taken into account, the sample average for each label is around 5500. Thus, procedures for augmenting the data will be implemented to reduce the data discrepancy.

For each image that belongs to *disgusted*, we will firstly generate a flipped image. Beside that, each image will be rotated with small angles of 15 degrees, to the left and right, in order to generate more data. To make the process easier, *OpenCV* was used due to the fact that it provides a lot of computer vision tools for image preprocessing. Algorithm 4 illustrates the steps for our data augmentation while Figure 3.3 shows the new data distribution. *FER2013* dataset contains some dark and blurred images which are producing noise. Therefore, some of them were eliminated manually and that is the reason why Figure 3.1 displays less data compared to previous distribution.

```
Input: files - image file paths
Output: augmented data for disgusted label

for file in files do
    if file in disgustedFolder then
        $image \leftarrow loadImage(file)$  /* loading the image from the given path */

        $flippedImage \leftarrow flip(image)$
        $leftImage \leftarrow rotateToLeft(image, 15)$
        $rightImage \leftarrow rotateToRight(image, 15)$  /* Generating flipped and rotate images */

        $saveToDisk(flippedImage)$
        $saveToDisk(leftImage)$
        $saveToDisk(rightImage)$  /* Saving augmented data to disk */
    end
end
```

Algorithm 4: The process of image augmentation for *disgusted* label

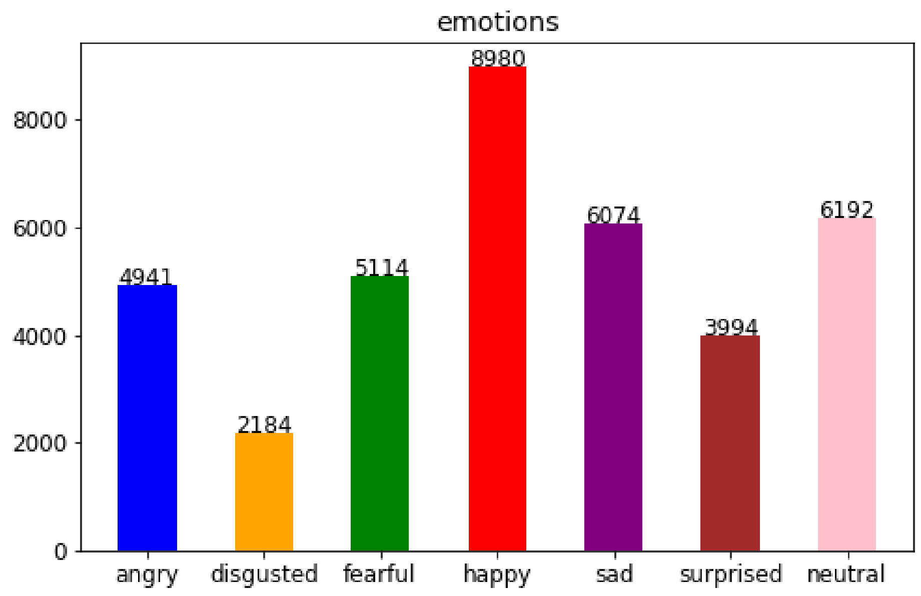


Figure 3.3: FER2013 dataset after data augmentation

3.2.2 Interpolation

It is difficult to obtain features on images that have only 48 pixels length because, in practice, most of the images that are produced consist of 300 ppi which is equivalent to 90,000 pixels. Also, the average size of images that were used for training deep convolutional neural network architectures is 224 pixels. Therefore, a method for increasing the size can ease the process of fine-tuning.

One problem that may occur when the image is being resized is the fact that a linear interpolation will result into a blurry version. To bring a clear transition, a bicubic interpolation is preferred as it can create *smoother* pixels. Once again, OpenCV provides a set of interpolated algorithms, including nearest neighbour, linear and bicubic interpolations. The difference between each interpolation type can be observed in Figure 3.4. The nearest neighbour method shows the reason of why there is a need for a smoother algorithm. The image becomes so blurry that it can no longer be considered as something that you can analyze in real life. The linear interpolation seems to approximate the values pretty accurately. However, the edges around the face remain blurred compared to the approximation of the bicubic interpolation. Therefore, the bicubic interpolation will be the preferred choice.

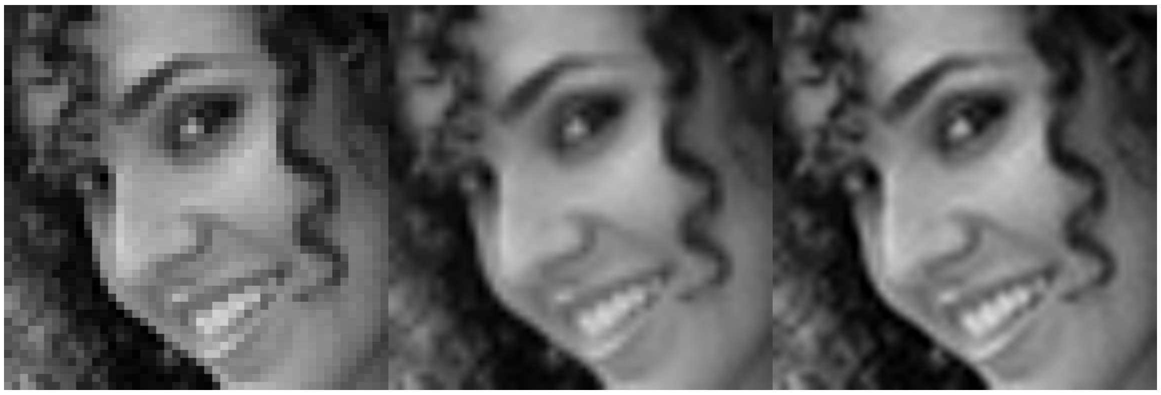


Figure 3.4: Comparison between nearest neighbour, linear and bicubic interpolation

3.2.3 Histogram Equalization

A final point that needs to be answered, regarding preprocessing, is the contrast. Many images seem washed out due to the fact that they lack contrast. This factor is another reason why the interpolation cannot reduce the amount of blur. The lack of contrast diminishes the amount of information which is provided. Even if the image seems normal for a human eye, the computer vision models will face difficulties in analyzing the spots when there is not a clear emphasise of light sources.

To answers this problem, a histogram equalization procedure will be performed. Histogram equalization is a process which adjusts the constrast of an image by spreading the most common intensities of the pixel values in order to allow the washed out areas receive more contrast.

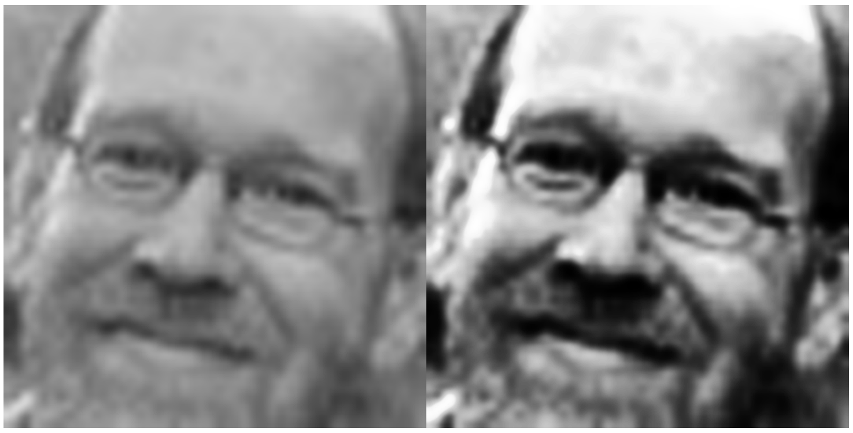


Figure 3.5: An image which received histogram equalization

Figure 3.5 show the difference between a blended image and an image with contrast. It can be observed that the lighter areas on the first image are challenging even for the human eye to be noticed. Once the pixels are scaled, the difference between each value in the image will be more visible.

3.2.4 Data Representation

Before feeding the preprocessed data into the network, a rescale is going to be performed. As it is shown in Algorithm 5, each image will be rescaled by an 1./255 factor which will shift the base from the pixel range of 0 and 255 into the range of 0 and 1. Beside that, a zoom and a shear process will be applied to create a wider range of angles and depth levels. Zooming and shearing are used with the intention of avoiding overfitting by providing a wide range of positions to generalise [Cho16].

```
Input:  files - image file paths
Output: rescaledData - the rescaled data

$rescaledData \leftarrow []$

for file in files do
    $image \leftarrow readImage(file)$
    $image \leftarrow rescale(image, 1./255)$  /* scaling the image between 0 and 1 range */
    $image \leftarrow zoom(image, range = 0.2)$  /* adding a random zoom with maximum range of 0.2 */
    $image \leftarrow shear(image, range = 0.2)$  /* adding a shearing transformation with a maximum random angle of 0.2 */

    $appendImage(rescaledData, image)$
end
```

Algorithm 5: The process of image rescaling

3.3 Training

In spite of the fact that some articles showed some good results with *stochastic gradient descend* and *nesterov* [KC21], the models trained with *adam* were performing much better. For additional reasons regarding *adam* as the preferred option, people can check Chapter 2.3. A scheduler for *learning rate decay* was implemented in order to reduce the step size once the models approach by the minimum value. That can help the model prevent the scenario where it can no longer converge. For the loss function, we used *categorical cross-entropy* as it combines both the *cross-entropy loss* and the *softmax activation* function.

Every proposed model uses *ReLU* as the activation function in the hidden layers as it is much cheaper to calculate the values due to its linearity. Regarding the architecture, there were many deep networks that were taken into account. The next chapters will briefly present each architecture.

3.3.1 VGGNet

The first architecture was *VGGNet* which, based on their authors, reached an accuracy of 73.28% in 300 epochs [KC21]. The architecture is a simplified version of *VGG19*, consisting of 4 convolution blocks and 3 linear layers. Each block contains two convolution filters which are activated by a ReLU function. Afterwards, a batch normalization layer and max-pooling layer are used in order to prevent a possible vanishing gradient.

3.3.2 Xception

According to [PK16], *Xception* architecture achieved an accuracy of 71.6% in approximately 300 epochs. The architecture is relying on *Depthwise Separable Convolutions* [Cho17]. Inception blocks are combining multiple features so that the next layers can decide on which features to focus. *Xception* adopts the principle of *Inception blocks*, but it applies it in the reverse way. Instead of compressing with a 1x1 convolution and then applying different filters, *Xception* takes the other way around. Due to the fact that *Keras's* implementation of *Xception* was constrained by the minimum image size of 128 pixels, an experiment with interpolated images of 144 pixels size was performed for 30 epochs.

3.3.3 ResNetV2

In [PK16] they achieved a performance of 72.4% accuracy by using *ResNet* architecture. One problem which *RestNets* are trying to overcome is the *vanishing gradient*. If an architecture has too many layers, the gradient starts to shrink closer to zero due to the fact that it calculates multiple derivatives. This results into the weights of the first layers not being updated and, consequently, not learning anything [PMB13]. ResNet connects the input of a layer straight to the output of other layers once they pass a couple of connections. Therefore, the gradient is able to flow through the skip connections and reach the initial layers [HZRS16a]. *ResNetV2* brings new residual units that shortcut connections and they are meant to ease the process of information propagation and to improve generalisation [HZRS16b].

3.3.4 DenseNet

DenseNet is trying to solve the same problem of *the vanishing gradient*. It reduces the connectivity pattern between each layer. What makes the difference between *DenseNets* and *ResNets* is the fact that the *DenseNets* avoid summing up the output features of each layer and concatenates the feature maps instead [HLVDMW17]. If each input is being concatenated, then each layers has access to the features maps from the preceding layers, offering a collective knowledge [Rui18]. The architecture can be seen in Figure 3.6.

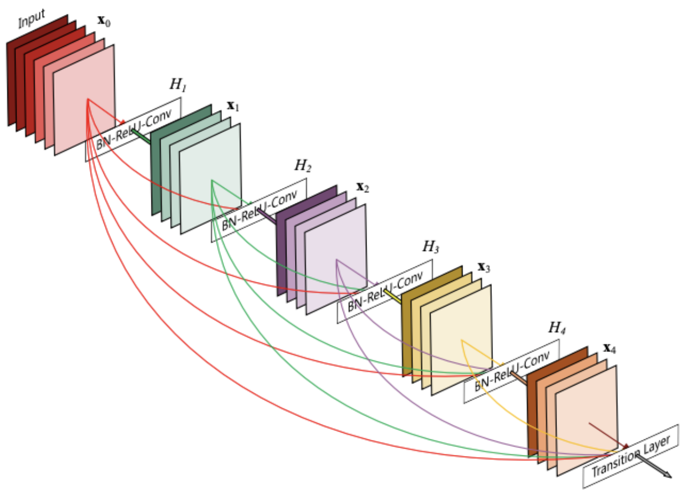


Figure 3.6: DenseNet architecture [HLVDMW17]

3.4 Testing

This section will present the experimental setup and the result analysis for the experiments that were performed.

3.4.1 Experimental Setup

The dataset was split into 80% for training and 20% for testing. During training, there were various types of architectures attempted. The learning rate which showed high results in less than 35 epochs for *Adam* was 5e-6. Once again, *Adam* was used in order to benefit of both momentum and average weighted decay. The experiments were performed on a NVidia K80 GPU in order to increase the speed of the learning process.

After reaching a loss below 0.1, it could be noticed that the gradient was making steps that were too big. Therefore, a learning rate decay procedure was introduced in order to lower the steps when the model is converging. While training, a scheduler was monitoring the loss of each epoch. If the model could not obtain a lower loss in 3 epochs, the scheduler was reducing the learning by half so that the model could take lower steps each time it was calculating the gradient. A final test included a weighted loss to balance the model when it attempts to predict less common samples. The weights were 0.7 for *angry*, *fear*, *neutral* and *sad*, 0.6 for *happy*, 0.8 for *surprise* and 1 for *disgusted*.

Depending on their performance in the first 30 epochs, some architectures received additional training while others were dropped due to either *underfitting* or *overfitting* performance. The results of the training can be seen in Table 3.1.

Architecture	**Epochs**	**Acc**	**Prec**	**Recall**	**Interpolation Size**	**Equalized**	**Learning Rate Decay**	**Weighted Loss**
VGGNet	40	81	86	76.6	48	No	Yes	No
VGG19	35	93	94	92	48	No	No	No
DenseNet152	30	86	84	84	48	No	No	No
Xception	60	99	99	99.8	128	Yes	No	No
VGG19	25	96.4	96.8	96.32	144	Yes	No	No
DenseNet121	60	99	99	99	144	Yes	Yes	No
ResNetV2	70	99.81	99.81	99.81	144	Yes	Yes	No
DenseNet121	70	99.81	99.83	99.8	192	Yes	Yes	No
DenseNet121	30	94.28	94.28	99.83	192	Yes	Yes	Yes

Table 3.1: Training performance for each architecture

3.4.2 Experimental Results

Table 3.2 show the results obtained after each experiment in report with the current state of the art. We calculated the same metrics that were used in Chapter 2.4.2, beside the *F1-score*.

Architecture	**Acc**	**Prec**	**Recall**	**Interpolation Size**	**Equalized**	**Learning Rate Decay**	**Weighted Loss**
VGGNet	67	71	62	48	No	Yes	No
VGG19	65.25	66	64	48	No	No	No
DenseNet152	65.6	67	63	48	No	No	No
Xception	68.91	69.16	68.75	128	Yes	No	No
VGG19	68.24	69.17	67.46	144	Yes	No	No
DenseNet121	70.05	70.4	69.79	144	Yes	Yes	No
ResNetV2	69.37	69.71	69.13	144	Yes	Yes	No
DenseNet121	**70.6 $\mp$ 0.07**	**71.07 $\mp$ 0.1**	**70.69 $\mp$ 0.09**	**192**	**Yes**	**Yes**	**No**
DenseNet121	65.49	70.14	62.12	192	Yes	Yes	Yes
LHC-Net [PBBG21]	74.42	-	-	-	-	-	-
VGGNet [KC21]	73.28	-	-	-	-	-	-
Res-Net [PK16]	72.4	-	-	-	-	-	-

Table 3.2: Testing performance for each architecture in report with state of the art and other architectures

Due to the fact that the process of training took a lot of hours, k-fold cross validation could not be performed. However, we performed bootstrapping on the best performing architecture which was *DenseNet121*. There is not enough time to reproduce the experiment all over again in order to achieve a distribution. Bootstrapping offers a window for the respective constraint. If there are a number of samples related to the desired metric, a normal distribution can be generated by performing draws with replacement [Bro20]. Algorithm 6 illustrates the process of bootstrapping.

```
Input: samples - the given samples
Output: score - the average score of the desired metric

$scores \leftarrow []$

for i in 1000 do
    /* repeat the experiment for a huge number of times */

    $selected \leftarrow []$
    for i in length(samples) do
        /* Draw with replacement a number of samples uniformly */

        $selection \leftarrow random.uniform(samples)$
        $selected \leftarrow append(selected, selection)$    /* Add the samples in a list */
    end

    $scores \leftarrow append(scores, mean(selected))$    /* Add the mean of the values to the scores */

end
$score \leftarrow mean(scores)$    /* Get the desire observation.  In this example, the average of the normal distribution is being calculated */
```

Algorithm 6: Bootstrapping method

The concept of bootstrapping is simple. However, it is important to remember that it is not foolproof. Assumptions are involved in the method and it contains its limitations. For instance, it is less likely to obtain a very accurate result, if people use just a few observations. On the other hand, it is useful when we want to calculate the mean average and the standard deviation from a normal distribution.

As it can be seen in Table 3.2, the image interpolation, histogram equalization and the learning rate decay improved considerably the performance of the model. The fact that the output of each layer is being concatenated in the next layer, for the *DenseNet121* model, also improved the final result. Unfortunately, the loss with weights could not enhance the performance of the model, in spite of the fact that the obtained results contain a high variance. *ResNet* showed promising results. A possible downside compared to *DenseNet121* was the fact that it trained on images which had a smaller size and the fact that it does not concatenate the output of each hidden layer, resulting into a lack of information during the process of forward propagation.

3.5 Future Enhancement

To conclude, in this chapter we covered some possible solutions for the task of facial expression recognition and compared the performance of the proposed deep convolutional neural network architecture in report with the current state of the art on *FER2013*. For future work, we are looking to improve the performance by using additional data which comes from similar distributions. We also consider the possibility of combining multiple models in a late fusion manner to increase the chances of an accurate prediction.

Chapter 4

FASEL: Fast Affective System for Emotional Learning

In this chapter, we are going to propose a web application called *FASEL* which stands for *Fast Affective System for Emotional Learning* and which relies on the *affective* algorithms that were presented in the previous chapters. It is a platform that can provide students a way to prepare for exams, quizzes and oral tests as well as to help them monitor their own emotional states. The application is important as it can shift the current education system from one that relies solely on performance to a system which focuses on students' feelings and needs and which promotes *emotional intelligence*.

The application is an original work, as to the best of our knowledge, there is no other system which tried to analyze the emotions from people's voices straightly through spectrogram images. Beside that, there is no application on the market which offers people the possibility to monitor both their cognitive performance and their emotional states, that being one of the strongest points of this software.

There are many frameworks which were used in the development of this application. For *backend*, Python was the preferred language as it provides a wide range of artificial intelligence libraries. Concerning the emotional detectors on *face* and *voice*, TensorFlow and Keras were the preferred frameworks for training and using artificial neural networks [Cho16], meanwhile OpenCV was the main choice for preprocessing images [BK08]. The Rest API was built with FastAPI to provide agile implementation meanwhile the layout and front side of the application was built with the help of Vue.js. Each one of these frameworks will be discussed in more detail in the future chapters.

4.1 Software Development

According to a blog of IBM, software development includes all the activities which imply creation, testing and deployment of an application [IBM14]. Therefore, when an application is being developed, there is a series of engineering activities that are taken into account in order to reduce the amount of bugs that might appear.

There are different reasons which sustain such an approach. From an economical perspective, by the developing an application which does not respect some guidelines of quality insurance, there are high chances to find out a lot of bugs in the deployment phase. Deploying an application with bugs will result into additional money and time invested. From a safety perspective, bugs can escalate into major disasters. A good example is the explosion of Ariane 5 rocket [Dou00] which happened 40 seconds after its launch, resulting into a firework show of almost 8 billion dollars. The explosion's cause was a bug which turned a 64 bit floating point into a 16 bit signed integer.

In our case, having bugs that could escalate are even more problematic as the application works with sensitive data which are people's faces and voices [DPD20]. Therefore, we are going to analyze and design the system to reduce the possible bugs that may occur.

4.1.1 System requirements

In the requirements phase, all use-cases are being planned in order to see the functional requirements that our application is supposed to meet. All functionalities can be observed in Figure 4.1. There is only one actor in the application and the required functionalities are the following:

- Login - It authenticates the user with the given username and password and opens the menu page if the credentials are correct.
- Handle Invalid Credentials - it checks the user's credentials and notifies the user if there is any problem
- Open Test List - opens the page where the list of possible tests are displayed.
- Start Test - opens the test page where user's camera and test questions are going to be displayed.
- Open Results List - opens the page where the scores of the previous tests are displayed for the given user.

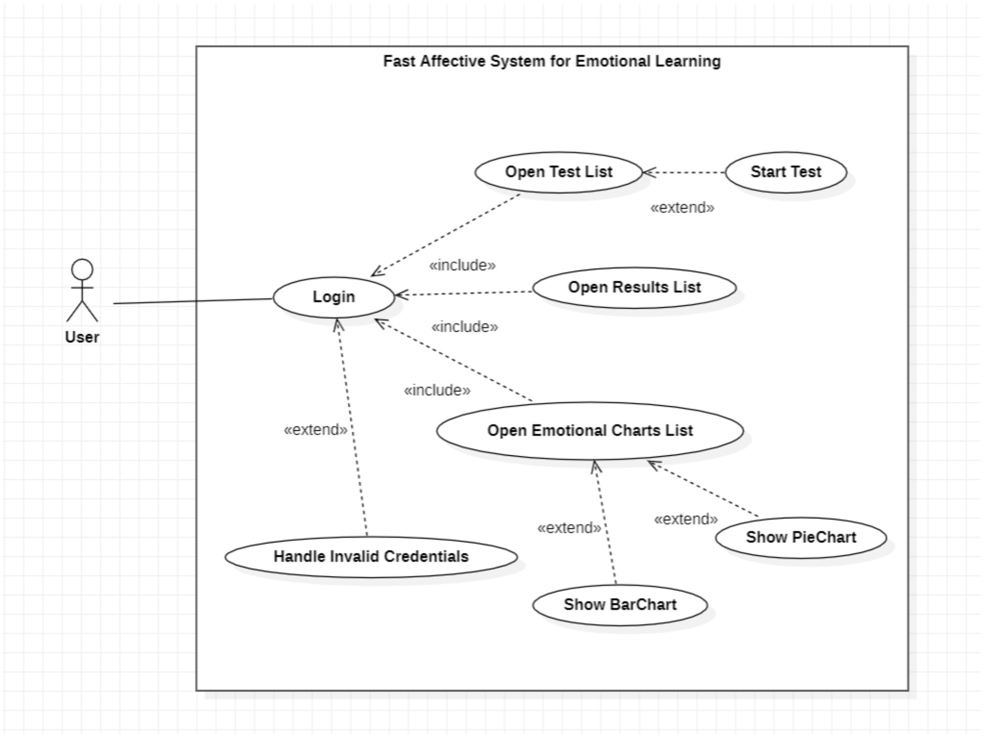


Figure 4.1: Use case diagram for *FASEL* web application

- Open Emotional Charts List - opens the page where the facial and speech charts for the previous tests are being displayed .
- Show BarChart - opens a dialog box and creates a barchart which displays the emotional states of the user during the previous tests.
- Show PieChart - opens a dialog box and creates a piechart which displays the emotional states of the user during the previous tests.

4.1.2 Conceptual Model

The conceptual model provides a bright view related to the entities that are part of the application which is being developed as well as the relationship between them. Figure 4.2 shows the conceptual model of *FASEL* application. The principal entities are *the Test*, *the Result* and *the Chart* as they are the ones which are going to interact with each other the most and will be displayed on the user's screen.

It is important to remember that the *system requirements* diagram and the *conceptual model* are used mainly in the exploratory phase of the development. It offers a map regarding the way entities and their functionalities are supposed to interact between each other and it can eliminate possible confusions before implementation.

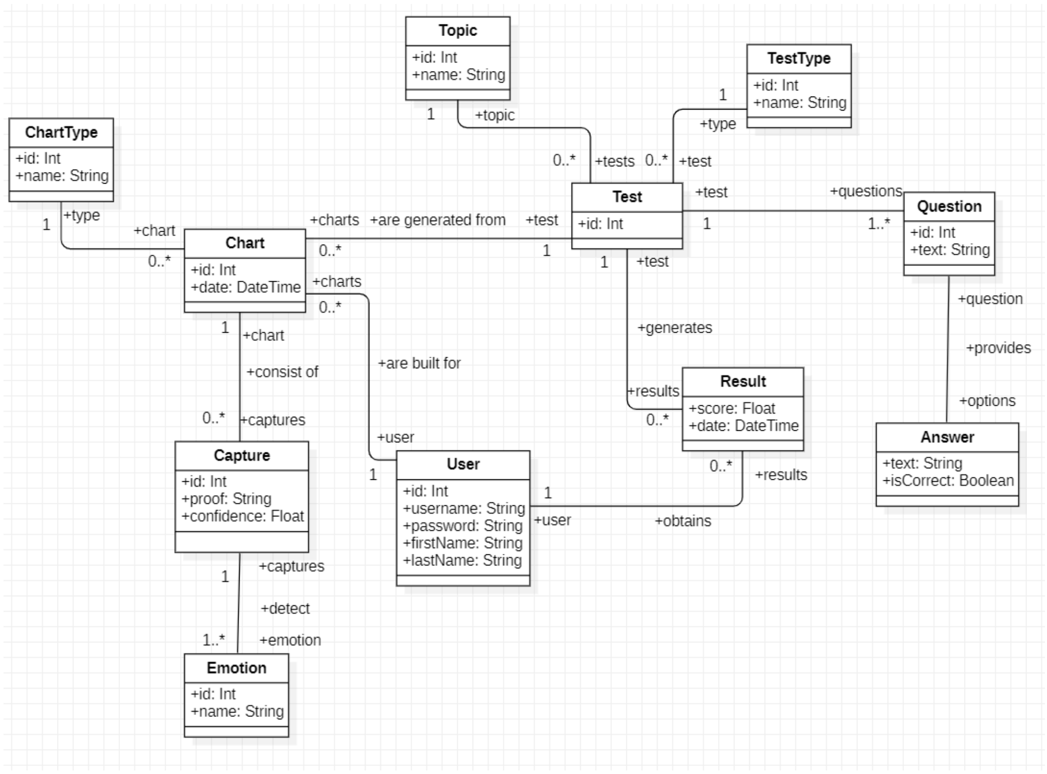


Figure 4.2: Conceptual model for *FASEL* web application

4.1.3 Deployment Diagram

The next step is to represent the deployment diagram. The final approach of an application is to deploy a final product. Therefore, *FASEL* will be implemented with a structure that can sustain a future deployment into a cloud service.

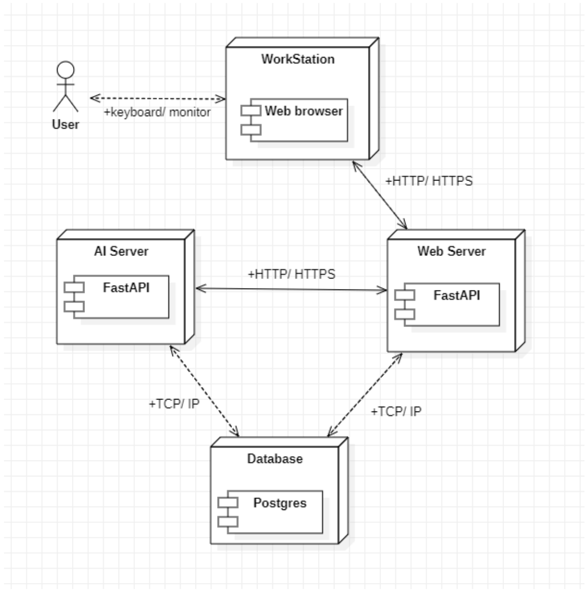


Figure 4.3: Deployment diagram for *FASEL* web application

Figure 4.3 show the relationship between each API and the client of the users.

There is an Web Server that will provide the REST services for the basic entities with whom the user will interact. For identifying emotions, the Web Server will send requests to an API that contains the artificial intelligence models which can predict emotions from faces and voices. Both servers will be able to communicate with the database.

4.1.4 API Diagrams

API stands for *Application Programming Interface*. It provides a method of communication between different products and services. REST APIs are much simpler regarding their design, being the preferred choice when people want to provide services through endpoints [Red18]. Figure 4.4 and Figure 4.5 show the API diagrams for the Web and the AI server.

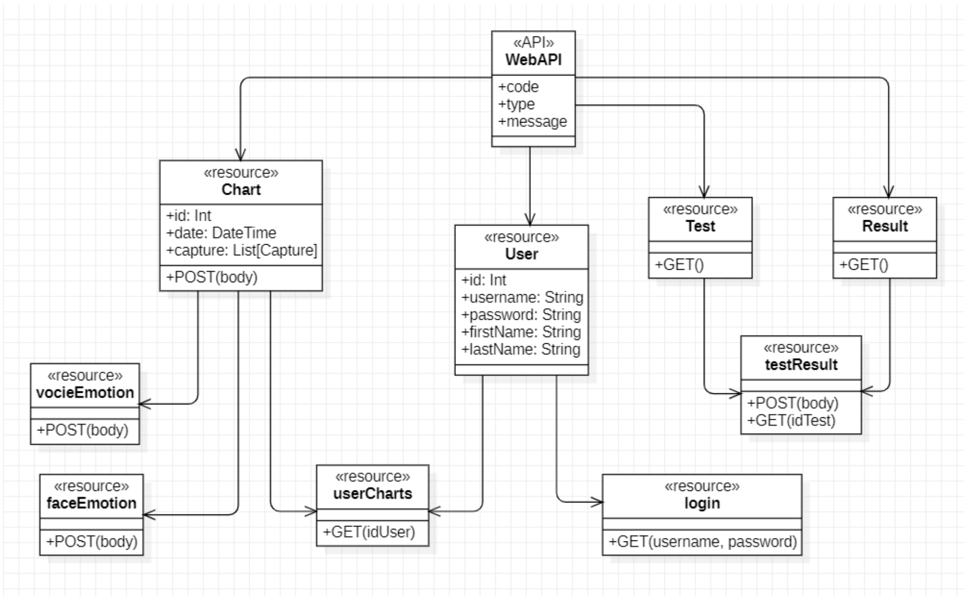


Figure 4.4: API Diagram for Web Server

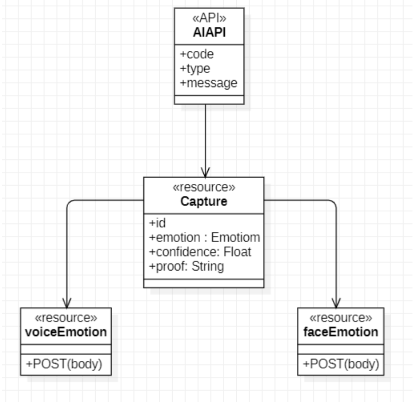


Figure 4.5: API Diagram for AI Server

4.1.5 Sequence Diagrams

Sequence diagrams illustrate the way objects interact between each other in report with the collaboration context. One of the most important step which *FASEL* accomplishes is to detect the emotions from video and audio captures, as well as generating the emotional charts. Figure 4.6 and 4.7 represents the sequence diagram for detecting an emotion and requesting the emotional chart.

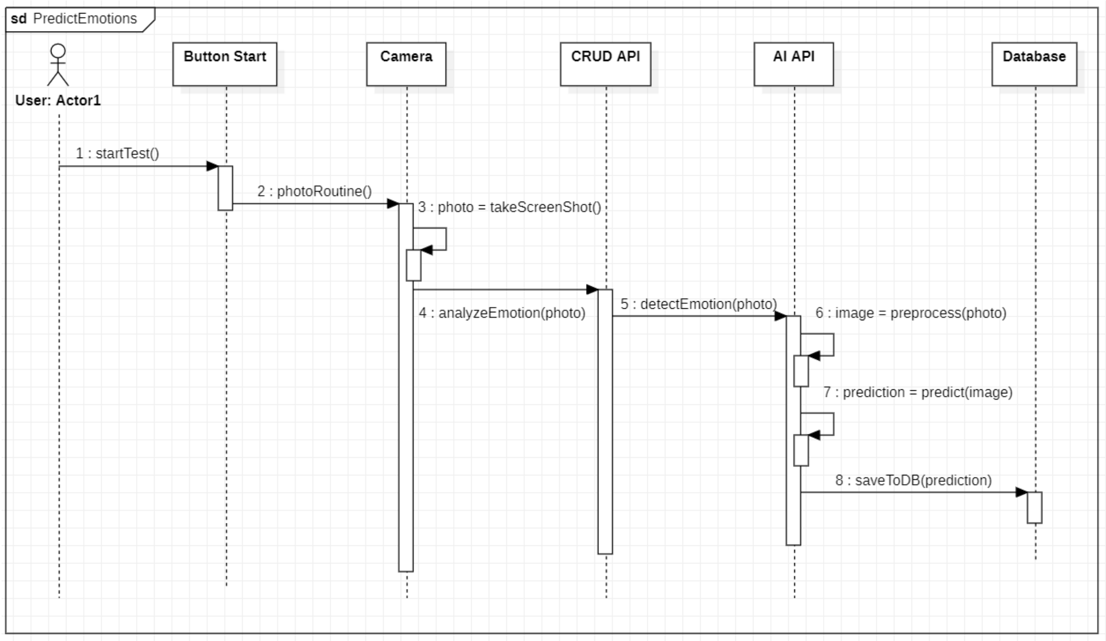


Figure 4.6: Sequence diagram for emotion detection

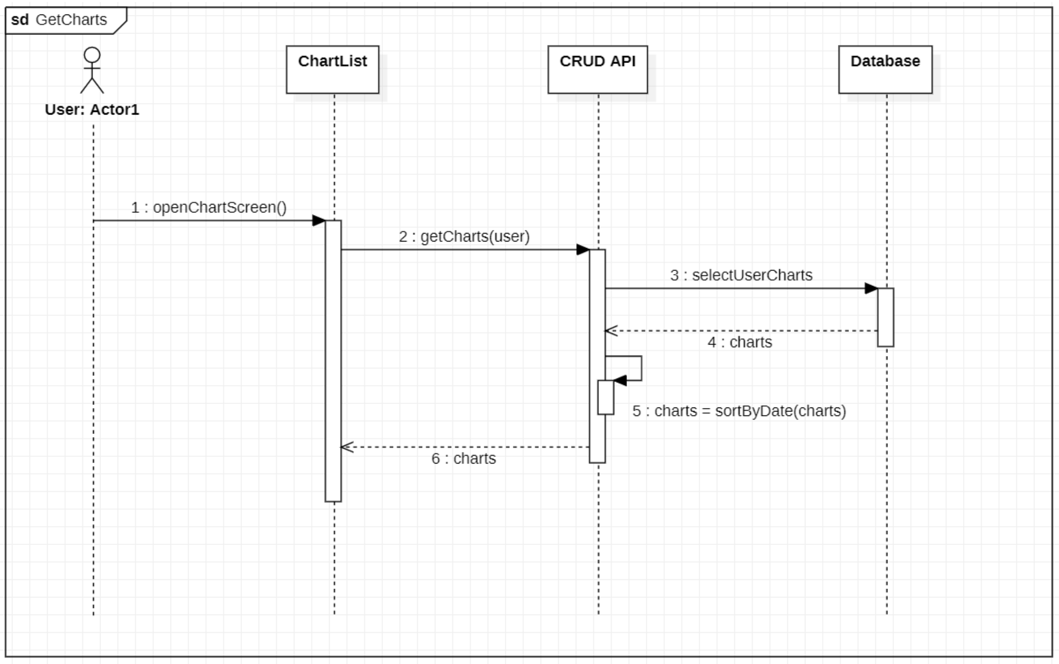


Figure 4.7: Sequence diagram for obtaining charts

4.2 Artificial Intelligence Frameworks

This chapter will present the frameworks which were used to develop the artificial intelligence part of the application.

4.2.1 TensorFlow and Keras

TensorFlow and *Keras* are two of the most popular frameworks for developing artificial intelligence. The advantage of *Keras* is the fact that it is very simple to learn and it is sustained by a solid documentation written by Francois Chollet [Cho16]. They also provide a wide range of deep neural networks, including the ones that were discussed in the previous chapters. Therefore, Keras can ease the process of implementation. In addition, *Keras* offers three paradigms for the way somebody decides to implement their neural networks: *sequential*, *functional* and *subclassed*. Both the *sequential* and the *functional* models were implemented during the experiments. One of the main advantages related to the *functional* paradigm is that it can provide access to the layer connectivity. In other words, we can decide the way layers are connected between each other. It is also close to the way researches perceive neural networks: a graph of connected layers. Figure 4.8 shows an example of the way layers can be connected between each other in the *functional* paradigm.

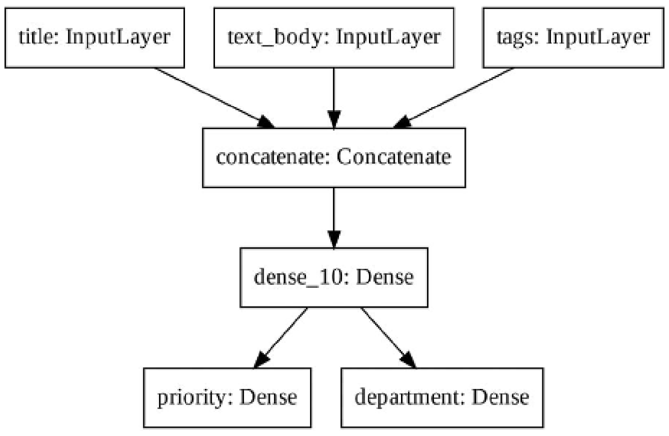


Figure 4.8: An example of multi-input multi-output neural network in Keras [Cho16]

Due to the fact that PyTorch is very often used in research, it was also taken into consideration. However, PyTorch provides a very low level API which in a future deployment is quite problematic. That is one of the reasons why TensorFlow is mostly preferred in the industry. On the other hand, Tensorflow and Keras offer a higher level API which is also user-friendly. Even if the Keras used to be quite slow compared to PyTorch, now that it is a part of Tensorflow, it works like a wrapper over the TensorFlow backend. Therefore, every optimization that TensorFlow offers can be accessed by Keras as well. Beside that, there are more resources and open source code for Keras compared to PyTorch, which offers an advantage when people need support. Figure 4.9 illustrates the search interests regarding TensorFlow, Keras, PyTorch and fast.ai on Google between 2018 and 2019. Even if PyTorch receives more and more popularity, TensorFlow and Keras remain at the top when people search for artificial intelligence frameworks.

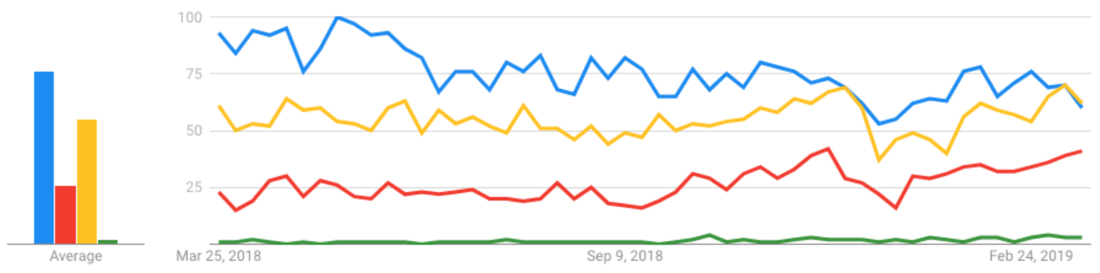


Figure 4.9: Tensorflow in blue, Keras in yellow, PyTorch in red and fast.ai in green. Google search interest between 2017 and 2018 [Jef19]

4.2.2 OpenCV

OpenCV provides a wide range of tools that can be used to preprocess images. It contains functions that can ease the process of interpolation, having both *linear* and *bicubic* interpolations implemented. Histogram equalization can also be performed and there are procedures which can turn grayscale images from one channel to three channels. Inside *FASEL*, *OpenCV* takes care of the preprocessing phase of each image and spectrogram which is intercepted through the endpoint.

It is important to be mentioned that each person will have their video camera opened and screenshots will be taken during the period of a testing. Therefore, many pictures that are sent to the AI Server contain more than just the face of the respective user. Before commencing the interpolation and histogram equalization, it is important to detect the face in that image. OpenCV provides the cascade classifier which is relying on the Viola-Jones algorithm. It is trying to detect a face by locating simple shapes of the face called *the Haar features* [VJ01]. *The Haar features* can be observed in Figure 4.10.

4.2.3 Librosa

Librosa is a framework destined for music and audio analysis. It contains a wide range of blocks which are useful for extracting audio features like mffc, chroma-

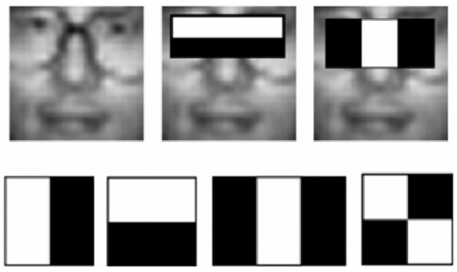


Figure 4.10: Haar features used for detecting a face [NRBI09]

grams and mel spectrograms [MRL+15]. It also provides additional preprocessing methods, such as shifting the audio files to a single channel. Librosa was used for the preprocessing phase of the audio models. In the case of *FASEL*, once the audio files reach the API and they get decoded from base64 back to mp3, librosa converts the audio file into single channel audio and extracts the mel spectrogram. Compared to other audio frameworks, librosa provides a simple way of obtaining the desired features. It still faces difficulties with less known file formats such as the browser format of webm. However, this problem was solved with the help of the *ffmpeg* call through the *Windows Linux Subsystem*(WSL).We used *ffmpeg* to switch the file from webm to mp3, before using *librosa*.

4.3 Implementation

The implementation stage plays a crucial part in the life-cycle of an application as factors like development frameworks and communication protocols are taken into consideration. This section will cover the implementation aspects of *FASEL*.

4.3.1 Backend

Backend represents the part of the application which is allowed to operate without being accessed by the user. In general, it plays the role of a server and the vast majority of data is stored and requested from it. It also contains the operating syntax necessary to manage the requests and interpret the data. Backends are built with a programming language which can be enhanced with the help of a framework that can provide either WebSocket connections or REST services.

4.3.2 FastAPI vs Flask

In spite of the fact that Flask and Django were the dominant Python web development frameworks, FastAPI has its own advantages. One of the major reasons people continue supporting the use of Flask and Django is because both frameworks have a lot of community support and they are battle-tested frameworks due to the fact that they have been around for a long time. Figure 4.11 shows the JetBrains Python Developers Survey for the most popular web framework libraries that are used with Python. Although it was the first year in which FastAPI got introduced in the survey, it was the third most popular web framework. Its constant rise in popularity will be justified.

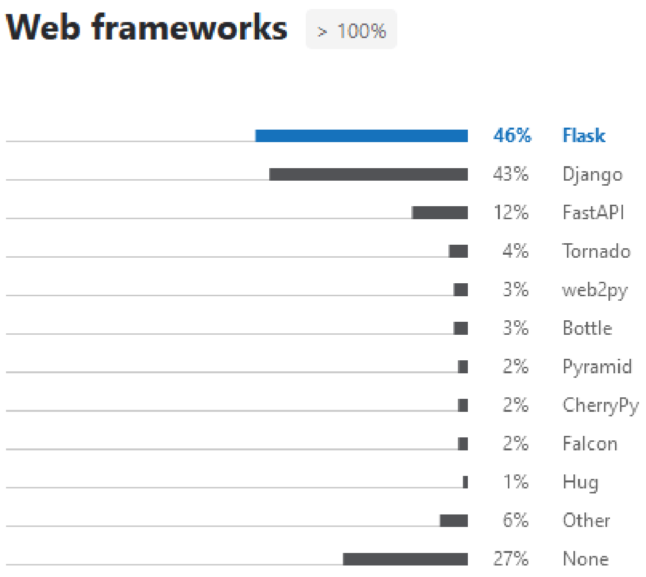
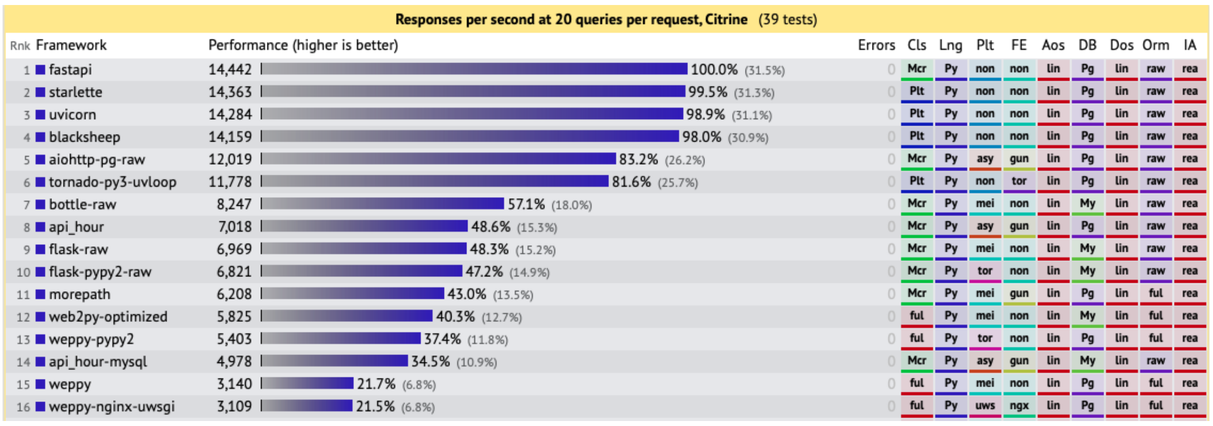


Figure 4.11: JetBrains Python developers survey 2020 [Jet20b]

One of the main reasons why people start using FastAPI is its speed performance. Unlike Flask, FastAPI is an asynchronous server gateway interface (ASGI) which makes it have higher performance than many of its competitors. Flask is relying on the Web Server Gateway Interface (WSGI) which has been considered by many people as the standard for many years. The disadvantage of WSGI is the fact that it is synchronous. Therefore, if a huge number of requests is made at the same time, they will be needed to wait in a queue in order to be completed. One the other side, FastAPI solved the problem by handling the requests in an asynchronous way. Likewise, no request needs to wait to be accomplished.

Figure 4.12 illustrates the benchmark performance for the API frameworks which use Python. It could be noticed that FastAPI is surpassing everyone regarding response speed for requests which demand a big number of queries. The other two best performing frameworks are *Starlette* and *Uvicorn* which are used internally by FastAPI. On the other side, *Flask-raw* has less than a half performance of the three mentioned frameworks. By taking into account the fact that a huge flow of requests will be initialized in order to deliver face captures and audio files, the capacity of handling multiple requests in a short amount of time is important for the perfor-



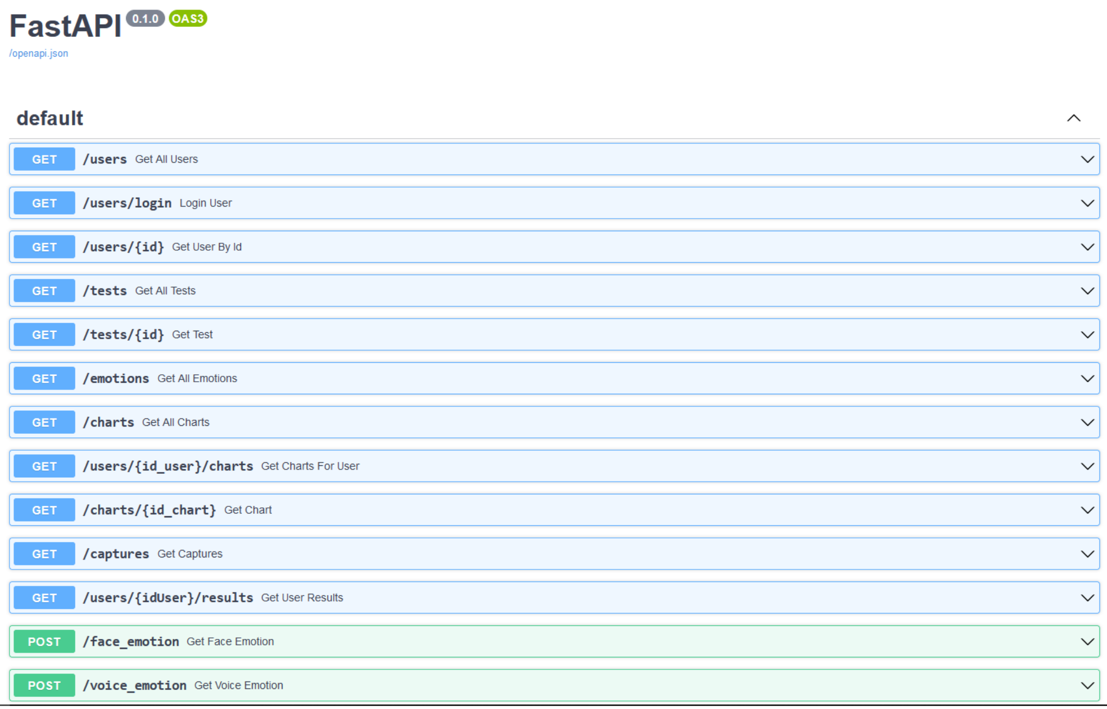
Responses per second at 20 queries per request, Citrine (39 tests)

Rnk	Framework	Performance (higher is better)			Errors	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA
1	fastapi	14,442	100.0%	(31.5%)	0	Mcr	Py	non	non	lin	Pg	lin	raw	rea
2	starlette	14,363	99.5%	(31.3%)	0	Plt	Py	non	non	lin	Pg	lin	raw	rea
3	uvicorn	14,284	98.9%	(31.1%)	0	Plt	Py	non	non	lin	Pg	lin	raw	rea
4	blacksheep	14,159	98.0%	(30.9%)	0	Plt	Py	non	non	lin	Pg	lin	raw	rea
5	aiohttp-pg-raw	12,019	83.2%	(26.2%)	0	Mcr	Py	asy	gun	lin	Pg	lin	raw	rea
6	tornado-py3-uvloop	11,778	81.6%	(25.7%)	0	Plt	Py	non	tor	lin	Pg	lin	raw	rea
7	bottle-raw	8,247	57.1%	(18.0%)	0	Mcr	Py	mei	non	lin	My	lin	raw	rea
8	api_hour	7,018	48.6%	(15.3%)	0	Mcr	Py	asy	gun	lin	Pg	lin	raw	rea
9	flask-raw	6,969	48.3%	(15.2%)	0	Mcr	Py	mei	non	lin	My	lin	raw	rea
10	flask-pypy2-raw	6,821	47.2%	(14.9%)	0	Mcr	Py	tor	non	lin	My	lin	raw	rea
11	morepath	6,208	43.0%	(13.5%)	0	Mcr	Py	mei	gun	lin	Pg	lin	ful	rea
12	web2py-optimized	5,825	40.3%	(12.7%)	0	ful	Py	mei	non	lin	My	lin	ful	rea
13	weppy-pypy2	5,403	37.4%	(11.8%)	0	ful	Py	tor	non	lin	Pg	lin	ful	rea
14	api_hour-mysql	4,978	34.5%	(10.9%)	0	Mcr	Py	asy	gun	lin	My	lin	raw	rea
15	weppy	3,140	21.7%	(6.8%)	0	ful	Py	mei	non	lin	Pg	lin	ful	rea
16	weppy-nginx-uwsgi	3,109	21.5%	(6.8%)	0	ful	Py	uws	ngx	lin	Pg	lin	ful	rea

Figure 4.12: Benchmark performance [Jet20a]

mance of our application.

Another reason which determines people to move from Flask to FastAPI is the fact that it contains automatic documentation. FastAPI relies on Pydantic to model and validate objects with ease. Pydantic models your objects under the shape of a json which is useful when you either receive or answer requests at the endpoints. Beside that, it also takes care of the constructor by default. FastAPI also comes implemented with Swagger UI, which allows you to test your endpoints without being needed to build a client. By considering these two features, FastAPI can speed up your development time. Figure 4.13 shows the endpoints from *FASEL* which can be tested straightly from Swagger UI.



FastAPI 0.1.0 OAS3
/openapi.json

default

GET `/users` Get All Users
GET `/users/login` Login User
GET `/users/{id}` Get User By Id
GET `/tests` Get All Tests
GET `/tests/{id}` Get Test
GET `/emotions` Get All Emotions
GET `/charts` Get All Charts
GET `/users/{id_user}/charts` Get Charts For User
GET `/charts/{id_chart}` Get Chart
GET `/captures` Get Captures
GET `/users/{idUser}/results` Get User Results
POST `/face_emotion` Get Face Emotion
POST `/voice_emotion` Get Voice Emotion

Figure 4.13: *FASEL* endpoints in Swagger

To summarise, FastAPI is the preferred choice because it can speed up the development time and performance. It can also decrease the number of errors and bugs due to its high level nature. These two factors are making the process easier when APIs are built from scracth.

4.3.3 FastAPI and SQLModel

While FastAPI can be sufficient to develop APIs that can be scalable, it struggles when object relational models (ORMs) are implemented in order to access data from a database. If SQLAlchemy is implemented, each entity will have both a Pydantic entity that can map the object to a json, as well as an SQLAlchemy entity that can communicate with the database. Considering the fact that an entity could have a number of classes for reading and updating purposes, it can result into multiple duplicates.

Tiangolo, the creators of FastAPI, answered to this problem by creating SQLModel. It was designed to simplify the process of interaction between FastAPI and the SQL databases. Instead of duplicating entities, it combines both SQLAlchemy and Pydantic in order to simplify the code and reduce the code duplication to minimum. It works like a layer above Pydantic and SQLAlchemy which makes sure that an entity is compatible with both of them. The Python code from below shows how entity *User* is represented in both ORM and Pydantic with the help of SQLModel.

```python
from sqlmodel import SQLModel, Field, Relationship
from typing import Optional, List

class UserBase(SQLModel):
    firstName: Optional[str] = None
    lastName: Optional[str] = None
    username: str
    password: str
    face: Optional[bool] = Field(default=False)


class User(UserBase, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    results: List["Result"] = Relationship(back_populates="user")
    charts: List["Chart"] = Relationship(back_populates="user")


class UserRead(UserBase):
    id: int
```

The base class contains the attributes of the respective entity. The *User* class represents the signature of the ORM model and it will be used to define the additional relationships between other objects. An additional entity is created by inheriting the base model and it is used with the purpose of sending information through the endpoints.

4.3.4 Database

In spite of the fact that NoSQL databases gain more and more popularity, our database is heavily relational oriented. Therefore, we chose PostgreSQL because it is suitable for a wide range of applications. Considering that multiple requests will be performed at the same time and the fact that we work with personal data like images and people's voices, its speed and security are suitable for the application that we build. Figure 4.14 illustrates the database diagram for *FASEL*.

4.3.5 Frontend

The frontend is the part of the web application with whom the user interacts the most. There are three frameworks which dominate the market: React, Angular and Vue.js. Our application will use Vue.js to build the client of our server due to its simplicity of the learning curve.

Vue

Vue works on the principle of Single File Component (SFC). The file format permits the encapsulation of the template, logic and styling in one place. The code below is an example of SFC:

```
<template>
    <div>
        <p class="greeting">{{ greeting }}</p>
    </div>
</template>
<script>
    props: {},
    data: () => ({
        greeting: 'Hello World!'
    }),
</script>
<style scoped>
    .greeting {
      color: red;
      font-weight: bold;
    }
</style>
```

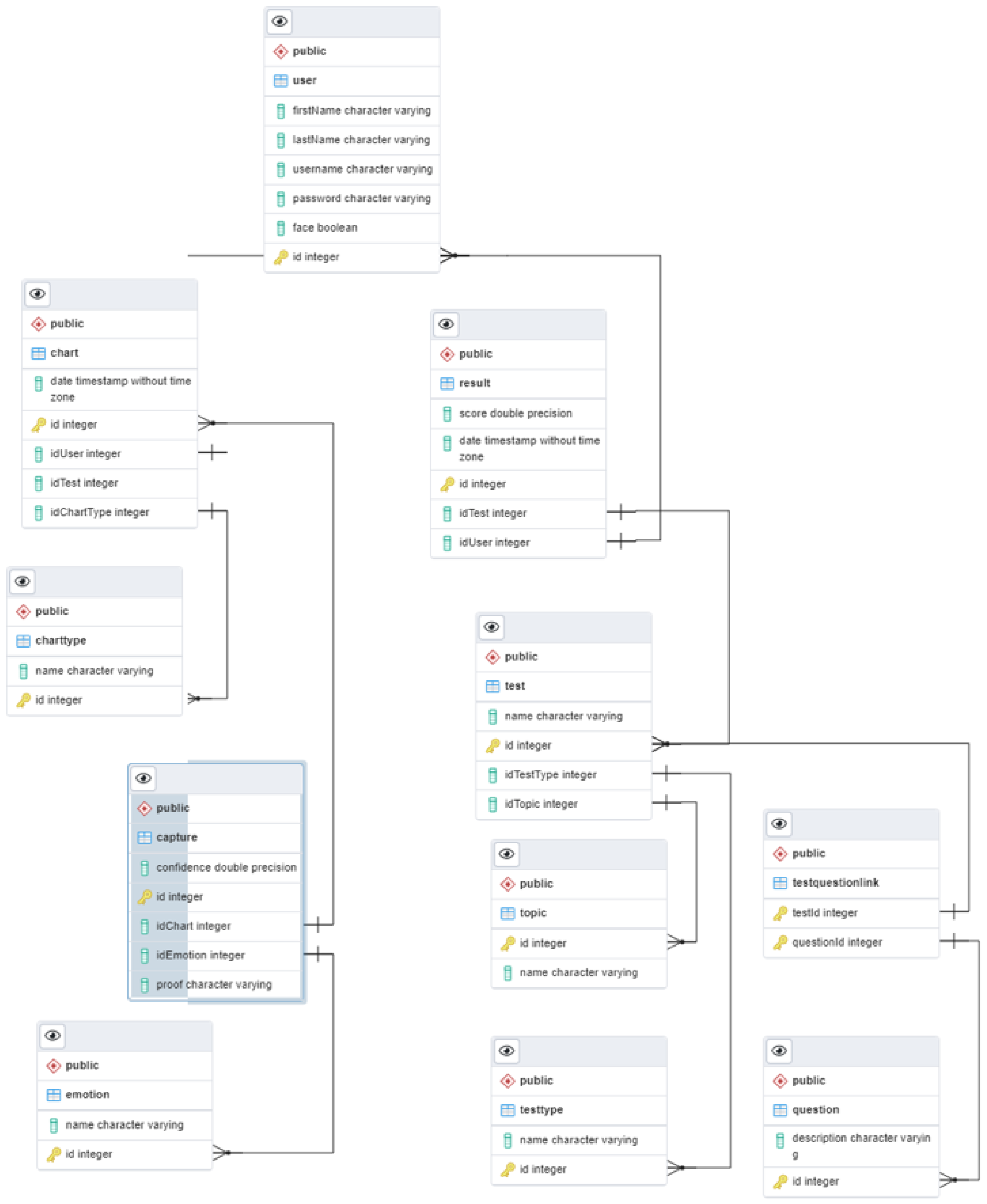


Figure 4.14: Database diagram for *FASEL*

While there are some doubts regarding the separation of concerns between HyperText Markup Language, Cascade Style Sheets and Javascript, the author of Vue claims that the separation of concerns should not be confused with the separation of file types. That is because the main goal of software engineering is to increase the maintainability of the code, meanwhile, the separation of file types does not help reaching that goal. Therefore, SFC are a suitable choice for structuring the web application.

Vuetify

In order to facilitate the process of development, a number of components will be taken from **Vuetify**. It is an UI framework which stays on top of Vue.js and its goal is to yield developers the tool needed to build high quality user experience. Compared to other Vue frameworks, Vuetify is simple and easy to learn, which makes it a suitable choice to make fast user interface that can look appealing to the public. Not all components are suitable for our application. Therefore, they are going to be adjusted based on our needs.

Chart.js

We want to monitor the emotional states which a user had while completing the tasks. Thus, there will be a bar chart which monitors the emotional states based on time and prediction confidence, and a pie chart which is going to be used to monitor the feelings that appeared during the tests from a macro perspective. Chart.js is an open source JavaScript library which is integrated in many popular frameworks to build beautiful charts. Once again, the components are going to be adjusted based on our needs. Figure 4.15 shows an example of bar chart and pie chart for the emotions of an user whose face was monitored during the test.

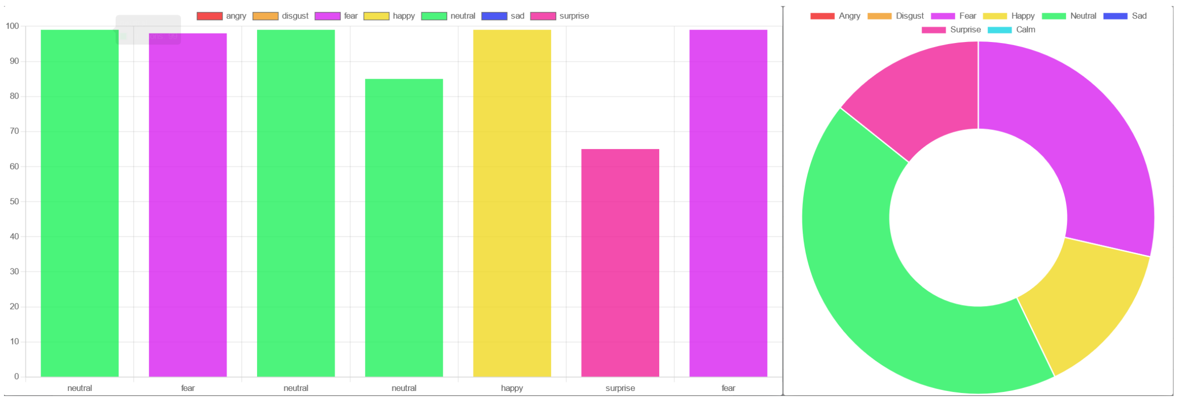


Figure 4.15: Emotional bar chart and pie chart

4.3.6 User Manual

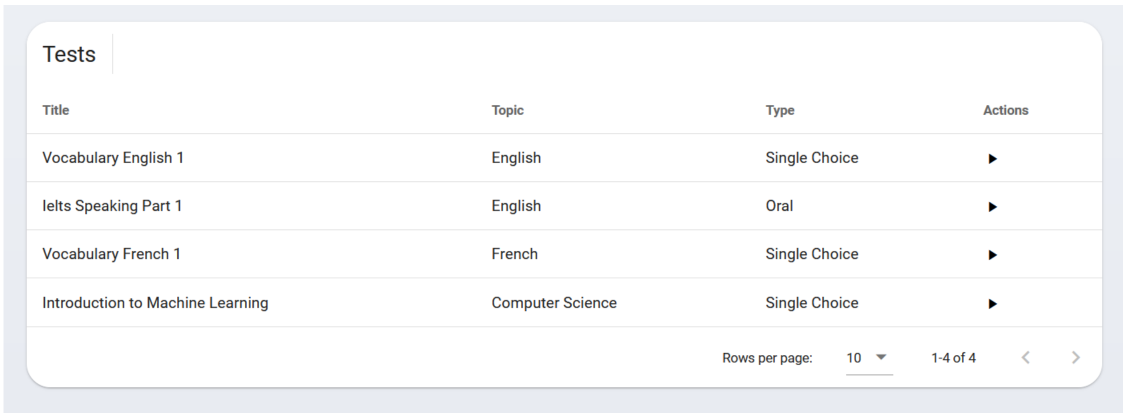
In order to ensure a good user experience, a user manual was written to facilitate the use of *FASEL*. This section will present some of the main features which are used in *FASEL*.

Login

Each user needs an account in order to login. Accounts are provided by the system engineer. In order to login, the user will introduce their own credentials. If the credentials are correct, the main menu page will be displayed. In the other case, a snack bar is displayed to notify the user that the credentials are wrong.

Taking a Test

An user can take a test by selecting one those that are provided in the test list (Figure 4.16). The tests have a topic and a type: single choice or oral. Once an user selected a test, they will be required to allow the application to use their camera and microphone. Small instructions will be displayed on the screen.



Tests

Title	Topic	Type	Actions
Vocabulary English 1	English	Single Choice	▶
Ielts Speaking Part 1	English	Oral	▶
Vocabulary French 1	French	Single Choice	▶
Introduction to Machine Learning	Computer Science	Single Choice	▶

Rows per page: 10 1-4 of 4

Figure 4.16: *FASEL* test list

Depending on the type of test the user selected, they will be required to answer by either selecting one of the provided options, or holding the microphone button to record the answer. Figure 4.17 illustrates the two types of questions that users can answer. If somebody wants to change an answer to a specific question, they can navigate between questions with the help of *next* and *back* buttons.

View Results

The main results of this application are the emotional charts which people can access by pressing on the *Charts* card in the main menu. A list will be displayed on the screen where people can see the emotional charts which are displayed in the descending order of the chronology. The title of the test, the type and the date will be displayed for each chart (Figure 4.18. In the right side of each row, there are two icons which portray a bar chart and a pie chart. If an user presses on the icons, a dialog will open and will display one of the charts which were shown in Figure 4.15.

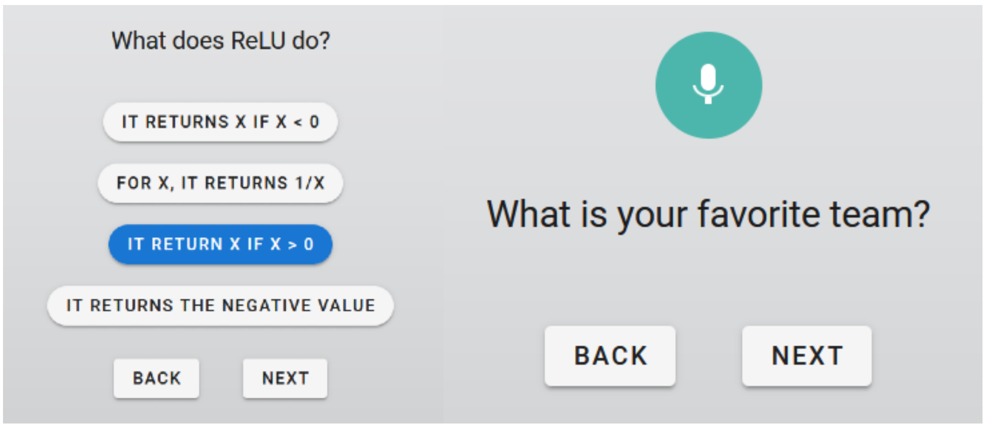
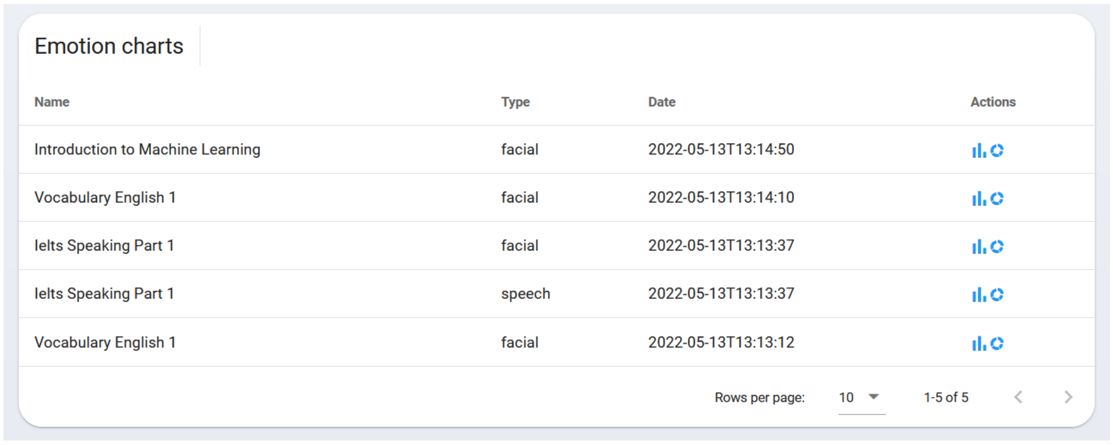


Figure 4.17: Example of single choice and oral questions



Emotion charts

Name	Type	Date	Actions
Introduction to Machine Learning	facial	2022-05-13T13:14:50	
Vocabulary English 1	facial	2022-05-13T13:14:10	
Ielts Speaking Part 1	facial	2022-05-13T13:13:37	
Ielts Speaking Part 1	speech	2022-05-13T13:13:37	
Vocabulary English 1	facial	2022-05-13T13:13:12	

Rows per page: 10 1-5 of 5

Figure 4.18: List of emotional charts

Score results are created for single choice tests. In order to see them, an user can press on the *Tests* card where a list with each score is displayed in a similar way the charts are displayed in Figure 4.16

Conclusions

This thesis proposed a spectrogram image fine-tuning approach for the speech emotion recongition task on RAVDESS dataset. The model performs much better than the current state of the art with less time for training. The goal of the proposed SER method is to encourage a fine-tuning approach for the models which work with audio files. Another objective of this research was to increase the performance of deep convolutional neural networks on FER2013 by increasing the image size, through interpolation and the contrast, through histogram equalization. The general aim of this thesis was to prove that *deep learning* approaches can be a suitable method for the development of *affective computing*.

Additional contributions were made by designing and implementing *FASEL* application which, to the best of our current knowledge, is one of the only educational applications that allows people to monitor their cognitive performance, by solving tests and quizzes, as well as monitoring the emotional states after submitting the answers. In order to develop the application, two API servers with a web client were created. In addition, multiple frameworks were utilised in the training process of the artificial intelligence models and the development of the software application, to provide a scalable architecture: Tensorflow, Keras, OpenCV, Librosa, FastAPI, SQLModel, Vue.js, Vuetify and Chart.js. I feel confident that *FASEL* will encourage more people to develop other educational affective computing applications in order to make a transition on the education centred on people's feelings and needs.

For future contributions, multimodal experiments on RAVDESS will be performed in order to combine the facial and the speech components in a single model. We will make more researches for audio on natural emotional speech databases in order to see how well the fine-tuning approach performs on noisy data. We will also test the LHC blocks in our FER approach to check if the attention mechanisms can enhance the testing performance.

Bibliography

[ACL04] Stefano Arca, Paola Campadelli, and Raffaella Lanzarotti. An automatic feature-based face recognition system. In *Proceedings of the 5th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'04)*. Citeseer, 2004.

[BK08] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[Bro20] Jason Brownlee. Statistical methods for machine learning. *Discover how to transform data into knowledge with Python. Machine Learning Mastery.*, 2020.

[Cho16] Francois Chollet. Building powerful image classification models using very little data. https://blog.keras.io/building-powerful-image-classification-models-using-very\-little-data.html, June 2016. Accessed: 2022-04-08.

[Cho17] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[Cho21] Chollet, Francois. *Deep learning with Python*. Simon and Schuster, 2021.

[Dou00] Douglas N. Arnold. The Explosion of the Ariane 5. https://www-users.cse.umn.edu/~arnold/disasters/ariane.html, 2000. Accessed: 2022-04-13.

[DPD20] Markus Dirk Dubber, Frank Pasquale, and Sunit Das. *The oxford handbook of ethics of AI*. Oxford Handbooks, 2020.

[DSL⁺18] Fatih Demir, Abdulkadir Sengur, Hao Lu, Shahin Amiriparian, Nicholas Cummins, and Björn Schuller. Compact bilinear deep features for environmental sound recognition. In *2018 International*

Conference on Artificial Intelligence and Data Processing (IDAP), pages 1–5. IEEE, 2018.

[EAKK11] Moataz El Ayadi, Mohamed S Kamel, and Fakhri Karray. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern recognition*, 44(3):572–587, 2011.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GEC+13] Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*, pages 117–124. Springer, 2013.

[Gol96] Daniel Goleman. *Emotional intelligence: Why it can matter more than IQ*. Bloomsbury Publishing, 1996.

[GPH18] P Giannopoulos, I Perikos, and I Hatzilygeroudis. *Advances in Hybridization of Intelligent Methods*. Springer, 2018.

[Gyl19] Roan Gylberth. Understanding Dropout. `https://medium.com/konvergen/understanding-dropout-ddb60c9f98aa`, July 2019. Accessed: 2022-04-03.

[HA+19] Md Hasan, Tajwar Abrar Aleef, et al. Automatic mass detection in breast using deep convolutional neural network and svm classifier. *arXiv preprint arXiv:1907.04424*, 2019.

[HFQ+21] Maria Habib, Mohammad Faris, Raneem Qaddoura, Manal Alomari, Alaa Alomari, and Hossam Faris. Toward an Automatic Quality Assessment of Voice-Based Telemedicine Consultations: A Deep Learning Approach. *Sensors*, 21(9):3279, 2021.

[HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[IBM14] IBM Research. Software Development. https://researcher.watson.ibm.com/researcher/view_group.php?id=5227, 2014. Accessed: 2022-04-13.

[IDY20] Dias Issa, M Fatih Demirci, and Adnan Yazici. Speech emotion recognition with deep convolutional neural networks. *Biomedical Signal Processing and Control*, 59:101894, 2020.

[IPG13] Radu Tudor Ionescu, Marius Popescu, and Cristian Grozea. Local learning to improve bag of visual words model for facial expression recognition. In *Workshop on challenges in representation learning, ICML*. Citeseer, 2013.

[Jef19] Jeff Hale. Which Deep Learning Framework is Growing Fastest? https://towardsdatascience.com/which-deep-learning-framework-is-growing-fastest-3f77f14\aa318, April 2019. Accessed: 2022-04-16.

[Jet20a] JetBrains. FastAPI has Ruined Flask Forever for Me. https://towardsdatascience.com/fastapi-has-ruined-flask-forever-for-me-73916127da, October 2020. Accessed: 2022-05-10.

[Jet20b] JetBrains. Python Developers Survey 2020 Results. https://www.jetbrains.com/lp/python-developers-survey-2020/, 2020. Accessed: 2022-05-10.

[Joo19] Joonatan Samuel. Most important concepts in applied machine learning. https://joonatan.medium.com/most-important-concepts-in-applied-machine-learning-11e1e5\f33ff4, October 2019. Accessed: 2022-05-06.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[KC15] Amit Konar and Aruna Chakraborty. *Emotion recognition: A pattern analysis approach*. John Wiley & Sons, 2015.

[KC21] Yousif Khaireddin and Zhuofa Chen. Facial emotion recognition: State of the art performance on FER2013. *arXiv preprint arXiv:2105.03588*, 2021.

[KK13] Sreenivasa Rao Krothapalli and Shashidhar G Koolagudi. *Emotion recognition using speech features*. Springer, New York, 2013.

[KNSS19] M Kattel, A Nepal, AK Shah, and D Shrestha. Chroma Feature Extraction. In *In Encyclopedia of GIS*, pages 1–9. Springer, 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[LN05] Chul Min Lee and Shrikanth S Narayanan. Toward detecting emotions in spoken dialogs. *IEEE transactions on speech and audio processing*, 13(2):293–303, 2005.

[LR18] 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.

[MRL$^+$15] Brian McFee, Colin Raffel, Dawen Liang, Daniel P Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25. Citeseer, 2015.

[NRBI09] Hau T Ngo, Ryan N Rakvic, Randy P Broussard, and Robert W Ives. An FPGA-based design of a modular approach for integral images in a real-time face detection system. In *Mobile Multimedia/Image Processing, Security, and Applications 2009*, volume 7351, pages 83–92. SPIE, 2009.

[PBBG21] Roberto Pecoraro, Valerio Basile, Viviana Bono, and Sara Gallo. Local Multi-Head Channel Self-Attention for Facial Expression Recognition. *arXiv preprint arXiv:2111.07224*, 2021.

[Pic00] Rosalind W Picard. *Affective computing*. MIT press, 2000.

[PK16] Christopher Pramerdorfer and Martin Kampel. Facial expression recognition using convolutional neural networks: state of the art. *arXiv preprint arXiv:1612.02903*, 2016.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[Rah21] Rahul Banerjee. Understanding Accuracy, Recall, Precision, F1 Scores and Confusion Matrices. `https://towardsdatascience.com/understanding-accuracy-recall-precision-f1-scores-and-\confusion-matrices-561e0f5e328c`, January 2021. Accessed: 2022-04-15.

[Red18] Red Hat. Understanding APIs. `https://www.redhat.com/en/topics/api#:~:text=An%20application%20programming%20interface%20(API,constantly%20build%20new%20connectivity%20infrastructure.`, June 2018. Accessed: 2022-04-15.

[Rot18] Daniel Rothmann. What's wrong with CNNs and spectrograms for audio processing? `https://towardsdatascience.com/whats-wrong-with-spectrograms-and-cnns-for-audio-\processing-311377d7ccd`, March 2018. Accessed: 2022-03-31.

[Rui18] Pablo Ruiz. Understanding and visualizing DenseNets. `https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a`, October 2018. Accessed: 2022-04-11.

[SB17] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal processing letters*, 24(3):279–283, 2017.

[SC06] M Schroder and Roddy Cowie. Issues in emotion-oriented computing toward a shared understanding. In *Workshop on emotion and computing*. Citeseer, 2006.

[She64] Roger N Shepard. Circularity in judgments of relative pitch. *The journal of the acoustical society of America*, 36(12):2346–2353, 1964.

[SPG19] Akash Saravanan, Gurudutt Perichetla, and Dr KS Gayathri. Facial emotion recognition using convolutional neural networks. *arXiv preprint arXiv:1910.05602*, 2019.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[VBV21] Sergey Verbitskiy, Vladimir Berikov, and Viacheslav Vyshegorodtsev. Eranns: Efficient residual audio neural networks for audio pattern recognition. *arXiv preprint arXiv:2106.01621*, 2021.

[VGG21] Adrian Vulpe-Grigoraşi and Ovidiu Grigore. Convolutional Neural Network Hyperparameters optimization for Facial Emotion Recognition. In *2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, pages 1–5. IEEE, 2021.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[VK03] Dimitrios Ververidis and Constantine Kotropoulos. A state of the art review on emotional speech databases. In *Proceedings of 1st Richmedia Conference*, pages 109–119. Citeseer, 2003.

[ZJL19] Mao Xia Zhao Jianfeng and Chen Lijiang. Speech emotion recognition using deep 1D & 2D CNN LSTM networks. *Biomedical signal processing and control*, 47:312–323, 2019.

[ZW13] Xiaojia Zhao and DeLiang Wang. Analyzing noise robustness of MFCC and GFCC features in speaker identification. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 7204–7208. IEEE, 2013.