FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Master's Thesis

Multi-modal emotion analysis in textual and audio data

Matěj Zeman

PILSEN, CZECH REPUBLIC

2024

FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Master's Thesis

Multi-modal emotion analysis in textual and audio data

Bc. Matěj Zeman

Thesis advisor
Ing. Ladislav Lenc, Ph.D.

PILSEN, CZECH REPUBLIC 2024

Citation in the bibliography/reference list:
ZEMAN, Matěj. *Multi-modal emotion analysis in textual and audio data*. Pilsen, Czech Republic, 2024. Master's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor Ing. Ladislav Lenc, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Matěj ZEMAN**
Osobní číslo:	**A21N0080P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Téma práce:	**Multi-modální analýza emocí z textových a zvukových dat**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte dodanou datovou sadu pro automatickou multi-modální analýzu emocí.
2. Seznamte se s relevantními metodami pro extrakci příznaků z textu a audia vhodných pro analýzu emocí.
3. Prostudujte stávající metody a algoritmy pro analýzu emocí založené na neuronových sítích.
4. Navrhněte a implementujte prototyp systému pro automatickou analýzu emocí s využitím textové a audio modality.
5. Prototyp otestujte na dodané datové množině.
6. Zhodnoťte dosažené výsledky a navrhněte další možná rozšíření.

Rozsah diplomové práce:	**doporuč. 50 s. původního textu**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce:	**Ing. Ladislav Lenc, Ph.D.** Nové technologie pro informační společnost
Datum zadání diplomové práce:	**8. září 2023**
Termín odevzdání diplomové práce:	**16. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2023

Declaration

I hereby declare that this Master's Thesis is completely my own work and that I used only the cited sources, literature, and other resources. This thesis has not been used to obtain another or the same academic degree.

I acknowledge that my thesis is subject to the rights and obligations arising from Act No. 121/2000 Coll., the Copyright Act as amended, in particular the fact that the University of West Bohemia has the right to conclude a licence agreement for the use of this thesis as a school work pursuant to Section 60(1) of the Copyright Act.

In Pilsen, on 15 May 2024

. . . . . .

Matěj Zeman

Abstract

Multimodal emotion recognition involves correctly classifying the emotion from data involving multiple modalities. There are several viable modalities when it comes to emotion recognition. Facial movements, text, voice, and video of the speaker. This thesis focuses on audio and textual modalities for emotion recognition. First, feature extraction from audio data is performed. Subsequently, these features are used for training several audio emotion recognition models, that are based on Artificial Neural Networks. These audio emotion recognition models are then used to create audio feature extraction vectors. In the multimodal deep learning models, these audio feature vectors are combined with their textual counterparts for multimodal emotion recognition. The performance of this system is evaluated on ECF, RAVDESS, and IEMOCAP datasets.

Abstrakt

Multimodální klasifikace emocí zahrnuje rozpoznávání emocí z dat, která zahrnují více modalit. Pro rozpoznání emocí se nabízí hned několik modalit. Pohyb obličeje, text, záznam hlasu, nebo videa mluvčího. Tato práce se zaměřuje především na zvukovou a textovou modalitu pro rozpoznávání emocí. Nejprve je provedena extrakce příznaků ze zvukových dat. Následně jsou tyto příznaky použity pro trénování několika modelů pro rozpoznávání emocí ze zvukových dat. Tyto modely jsou založené na umělých neuronových sítích. Modely jsou následně použity pro vytváření příznaků ze zvukových dat. V multimodálních modelech jsou tyto příznaky spojeny s jejich textovými protějšky a použity pro multimodální predikci emocí. Úspěšnost tohoto systému je vyhodnocována na ECF, RAVDESS a IEMOCAP datasetech.

Keywords

Multimodal emotion recognition • Machine learning • BERT • CNN • Python • Feature extraction

Acknowledgement

I would like to express my gratitude to Ing. Ladislav Lenc, Ph.D. for his patience, insight, guidance, and valuable advice throughout the creation of this thesis work. I would also like to express my gratitude to Ing. Jiří Martínek, Ph.D. for his guidance and insight during the implementation phase of this work.

Matěj Zeman,
(May 2024)

Contents

1 Introduction 5

2 Audio feature extraction 7

2.1 Evolution of extraction methods . . . 9

2.1.1 Time domain features . . . 10

2.2 MFCC . . . 11

2.3 Chromagram . . . 14

2.4 Zero-Crossing rate . . . 15

2.5 Additional feature extraction methods . . . 16

3 Text feature extraction 17

3.1 Text Pre-Processing . . . 17

3.2 Word2Vec . . . 20

3.2.1 Continuous Bag of Words Model . . . 21

3.2.2 Skip Gram model . . . 22

3.3 Contextual word vector representation . . . 23

3.3.1 Attention mechanism . . . 24

3.3.2 Transformer . . . 24

3.3.3 BERT . . . 25

4 Artificial Neural Networks 27

4.1 Long Short-Term Memory Network . . . 30

4.2 Convolutional Neural Networks . . . 33

4.2.1 Convolutional layer . . . 34

4.2.2 Pooling layer . . . 36

4.2.3 Fully-connected layer . . . 37

5 Datasets 39

5.1 ECF . . . 39

5.2 RAVDESS dataset . . . 40

5.3 IEMOCAP . . . 41

6 Implemented Models and Methods **43**
6.1 Audio feature extraction . . . 43
6.2 Text feature extraction . . . 44
6.3 Noise reduction . . . 45
6.3.1 FT2D . . . 45
6.3.2 REPET-SIM . . . 45
6.4 Audio emotion recognition models . . . 45
6.4.1 CNN1D . . . 45
6.4.2 CNN2D . . . 46
6.4.3 MLP . . . 46
6.5 Textual emotion recognition models . . . 46
6.5.1 LSTM . . . 47
6.5.2 BERT . . . 47
6.6 Multimodal emotion recognition models . . . 48
6.6.1 LSTM multimodal . . . 48
6.6.2 BERT multimodal . . . 48

7 Experiments **49**
7.1 Evaluation methods . . . 49
7.1.1 Accuracy . . . 49
7.1.2 F1 score . . . 49
7.2 Audio emotion recognition . . . 50
7.2.1 ECF . . . 51
7.2.2 RAVDESS . . . 51
7.2.3 IEMOCAP . . . 52
7.3 Text emotion recognition . . . 53
7.3.1 ECF . . . 53
7.3.2 IEMOCAP . . . 53
7.4 Multimodal emotion recognition . . . 54
7.4.1 ECF . . . 54
7.4.2 IEMOCAP . . . 55
7.5 Discussion . . . 57
7.5.1 Possible extensions . . . 58

8 Conclusion **59**

A List of Abbreviations **61**

B User Manual **63**

Bibliography **67**

List of Figures 75

List of Tables 77

Introduction 1

Emotions are a complex behavioral phenomenon. They often occur as a response to a situation that we are involved in but can occur as an indirect response to a wide range of external sources. It is difficult to define emotions since they are largely subjective. Humans express emotions in various ways, but the most common ones involve speech characteristics and facial expressions. Speech is a quite complex signal, that contains a lot of information. We are able to identify the speaker, content, language, and even emotion from this signal. A large amount of effort and research is put into the speech-to-text task, where the model transcribes speech signals into their textual representations. However, the performance of these models is quite low in the case of emotion recognition. This might be due to the difficulty in modeling and characterization of emotions present in speech. Speaking comes more naturally when emotions are present. Nonverbal cues in a conversation can convey essential information, such as the speaker's intention. Apart from the textual content, the way the words are pronounced also transmits important non-linguistic information.

The way a sentence is said can change its meaning drastically and the incorporated emotion takes a big part in the way a sentence is said. For example, the word *'okay'* can express admiration, disinterest, consent, or even disbelief depending on the emotion the word is expressed with. Therefore understanding the text alone is insufficient if we want to understand the whole semantics of a sentence. It is thus beneficial to incorporate non-linguistic information such as emotion with the content of a sentence together. Humans do this subconsciously by perceiving the underlying emotions in addition to phonetic information by using multimodal cues. Humans process nearly everything from their surroundings multimodally without really thinking about it. Every sense that we have adds one modality to our overall perception of the world and human interactions. This is why the multimodal approach to emotion recognition might be beneficial since humans also use multimodal perception to recognize emotions.

From the standpoint of a machine, interpreting emotions can be seen as the process of categorizing or distinguishing between different emotions. Sophisticated speech systems should not be limited to only message processing, but they should

understand the underlying intentions of the speaker by detecting the expressions in speech through speech emotion recognition. Speech emotion recognition could have several applications in normal day-to-day life as well as some applications in more specialized fields. It could be quite useful for enhancing the naturalness of speech based human-machine interactions. Emotion recognition could prove to be useful for example in car driving systems, where the underlying information about emotion could identify the mental state of the driver and therefore provide better guidance. Call center conversations could also benefit heavily where more natural human-machine conversation could lead to better quality of service for their customers. Interactive films or E-learning could be more practical if they can adapt themselves to the listeners' or students' emotional state. It could generally improve the overall naturalness and effectiveness of a human-machine interactive systems, therefore achieving better results and widening their potential applications.

The aim of this thesis is to analyze relevant methods for emotion feature extraction from speech signals and implement a system for multimodal emotion recognition from textual and audio data.

Audio feature extraction

2

The listening range of a human auditory system stretches between 20 Hz and 20 kHz. Figure 2.1 represents the behavior of our auditory system in this listening range. This graph shows the absolute threshold of the sound pressure level (SPL) depending on different frequencies. The lowest level of pure tone sound pressure that a normal ear can detect in silence is known as the absolute threshold of hearing. It is noteworthy that the most suitable frequency range for the human auditory system's sensitivity is between 2 kHz and 5 kHz, with a threshold as low as approximately 9 dB SPL [JJ04]. Our ears are generally responsive to sounds between 50 Hz and 15 kHz for music and between 100 Hz and 4.5 kHz for speaking. Without exerting additional effort, humans are able to distinguish between a wide range of sounds. For instance, we are able to distinguish between speech and music, vehicle and truck sounds, baby and adult speech quality, different speakers, noise etc. We want machines to be able to distinguish between different noises just as easily as people can.

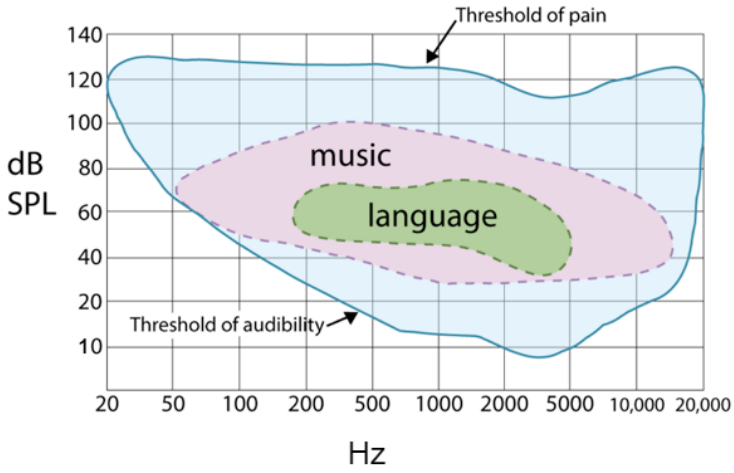

Figure 2.1: Audio hearing threshold for human auditory system [Jon17].

Regardless of the specific goal, a machine learning system needs strong and

discriminating features to enable fast and accurate machine learning. A smaller approximation of the signals is typically utilized to train the machine instead of feeding it the entire dataset in order to learn its attributes. A feature is a condensed representation of a signal. The difficulty lies in extracting the features so that the machine learning algorithm is able to learn and behave in the way it was designed to. The features need to be small in size but nevertheless need to draw attention to the signal's qualities. The signal's reduced version enhances the ML algorithms' time and computational complexity, making it more appropriate for real-time applications. Thus, we can state that feature extraction is the process of reducing a signal's dimensions to make it more appropriate for machine learning algorithms while still maintaining the signal's qualities [GK11].

Methods mentioned in this thesis are focused on feature extraction from speech audio data specifically. The speech audio signal has a vast application area. For example, speech recognition for speech-to-text translation, speaker recognition, emotion recognition, or human-computer interaction like Amazon's Alexa.

The typical pipeline for any machine learning audio system is shown in Figure 2.2. Pre-processing of the audio signal is carried out in the first step. Pre-processing techniques include normalization, noise reduction, and silence reduction. The signal windowing stage, which comes next, aids in our analysis of the potential non-stationary signal as a quasi-stationary signal. Sliding the window over the entire signal allows us to study and analyze the entire audio signal. Modern windowing techniques allow the window's size to be adjusted based on the signal's characteristics [SUK20]. The processes of feature extraction and feature selection are then carried out. The classifier takes the chosen features for testing and training as input, and a decision is made based on the classifier's prediction.

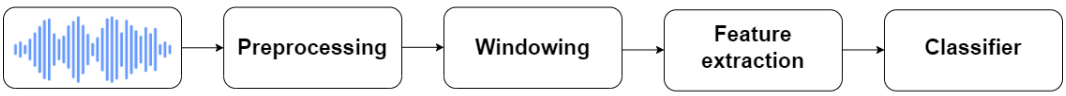

Figure 2.2: Traditional audio signal classification pipeline

The main focus of this thesis revolves around speech audio signals. Humans use a variety of organs, including the brain, mouth, nose, belly, and lungs, to make speech. When producing speech, the vocal tract and vocal cords are crucial. Starting at a frequency of 100 Hz, voice production can reach up to 17 kHz. Furthermore, speech is continuous in nature and has a smooth envelope as can be seen in Figure 2.3.

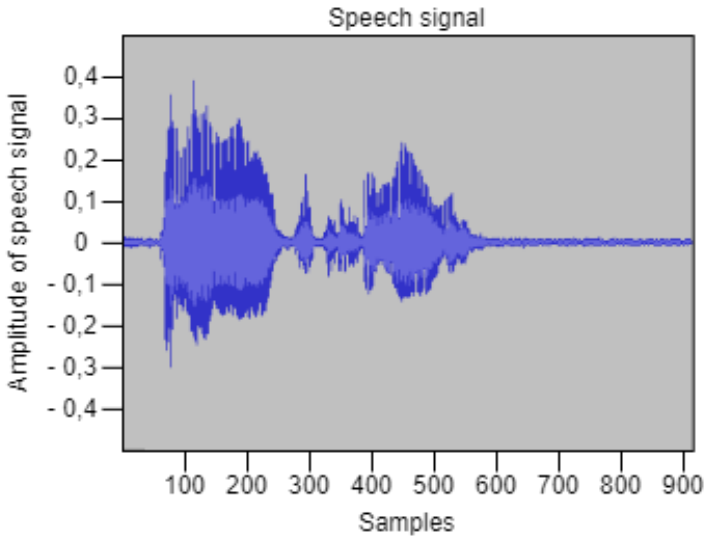

Figure 2.3: Waveform of a short speech

2.1 Evolution of extraction methods

To put it simply, feature extraction is the process of emphasizing a signal's most distinctive and dominant qualities. A suitable feature more closely resembles a signal's characteristics. The evolution of audio features can be sub-categorized into frequency domain, time domain, joint time-frequency domain, and deep features. The oldest one is the time domain feature extraction method. The time domain properties continue to be crucial for audio analysis and categorization. Many features, such as pitch and formants, were developed from the frequency domain to examine the spectrum of an audio signal and are still in use today in a variety of applications. The period of the vocal cord's output for vowels is referred to as the pitch or the fundamental frequency. The pitch of the speech signal can range from 50Hz for low-pitched male voices to 500Hz for children or high-pitched female voices. The glottal excitation signal, when introduced as a quasi-periodic impulse into the sound channel, has resonance characteristics resulting in a set of resonant frequencies known as formant frequencies or formats [HGG19]. Later the joint time-frequency feature extraction algorithms were developed. Since then, methods for audio signal processing have made use of these qualities. Deep features have been widely used in many different applications since deep learning's inception. In audio signal processing, for example, deep features have been applied since 2010 in the areas of speaker recognition, audio-video analysis, and acoustic scene classification [TGV17] [Li+18].

2.1.1 Time domain features

It is crucial to talk about the idea of windowing in the time domain before moving on to the features of the time domain. Analyzing a signal in its original form is the easiest method of analysis. Every sound signal mentioned in this thesis is a time series signal, meaning that it changes over time. A signal's key properties can be analyzed by viewing it in the time domain, and this knowledge can be applied to the analysis and prediction of similar signals. This time domain analysis is straightforward up until the signal is short or exhibits features that remain constant across time. Audio signals in real-time are not stationary. The windowing technique is used to examine such nonstationary signals, and the long non-stationary signal is broken down into smaller quasistationary signal segments for analysis. When a signal is windowed, it is multiplied by a window function that is zero outside of the region of interest. The resultant windowed signal is the subset of the original signal that is passed through the window, for the rest of the time the signal is zero.

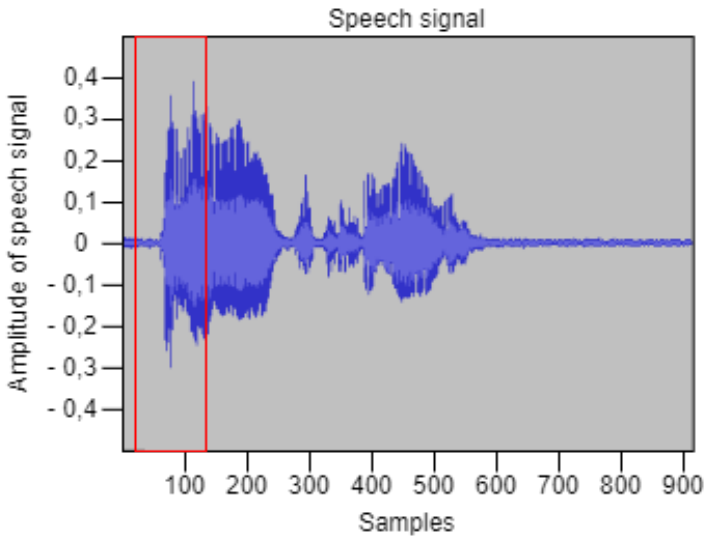

Figure 2.4: Windowing in Time Domain

Figure 2.4 explains the concept of windowing a signal using a rectangular window as a function. The window is slid over time, moving from the leftmost corner of the plot to the rightmost corner, in order to assess the entire signal. To transform the lengthy non-stationary signal into a short quasi-stationary signal, the size of the window is adjusted adaptively based on the properties of the original source signal. Figure 2.5 shows the sliding process of an adaptive rectangular window over a signal.

The rapid form shift at the boundaries of the rectangular window is one of its drawbacks since it might lead to distortion when the signal is being processed.

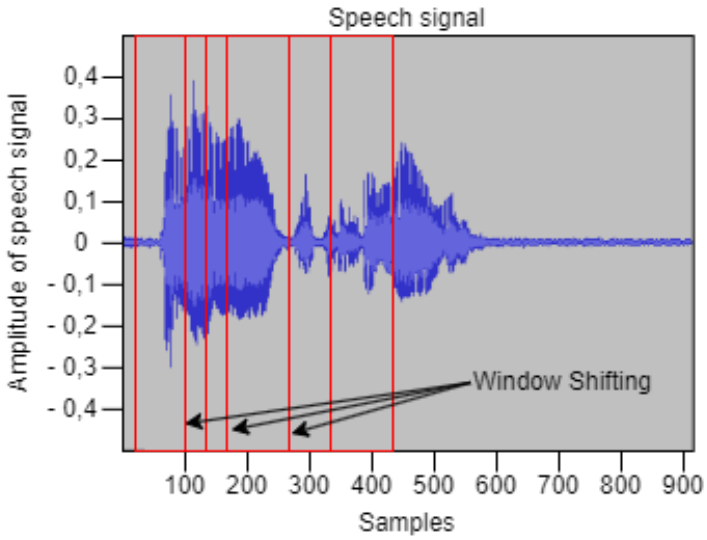

Figure 2.5: Adaptive Windowing in Time Domain

The Gibbs phenomenon [Kel96] is what causes the distortion. We might employ a window function with smooth curves, such as the Hanning or Hamming window [SM15], to solve this issue. In the middle of the window form, these window functions progressively grow from zero at the edges to one. These window functions lower the signal's edges and lessen the edge impact caused by the Gibbs phenomenon.

These days, a wide range of feature extraction methods are accessible depending on the properties of the raw data in many different fields. Any pattern recognition system must be able to identify the signals' sidebands and harmonics in both the frequency and time domains in the majority of fields. The Fast Fourier Transform (FFT) power spectrum is used to record the signal's sidebands and harmonics in the time domain. While cepstrum, such as Gamma Tone Cepstrum Coefficient (GTCC) and Mel Frequency Cepstrum Coefficient (MFCC), may extract sidebands and harmonics from the signal's spectrum version [LIZ13]

2.2 MFCC

One of the frequently employed features, Mel Frequency Cepstral Coefficient (MFCC) has been used in many different areas, especially in audio signal processing, where it is utilized for speaker identification, voice recognition, and gender identification [LIZ13]. One way to compute the MFCC is by carrying out five sequential procedures: signal framing, power spectrum computation, Mel filter bank application, logarithm calculation for each filter bank, and DCT application. Figure 2.6 represents the computational pipeline of MFCC.

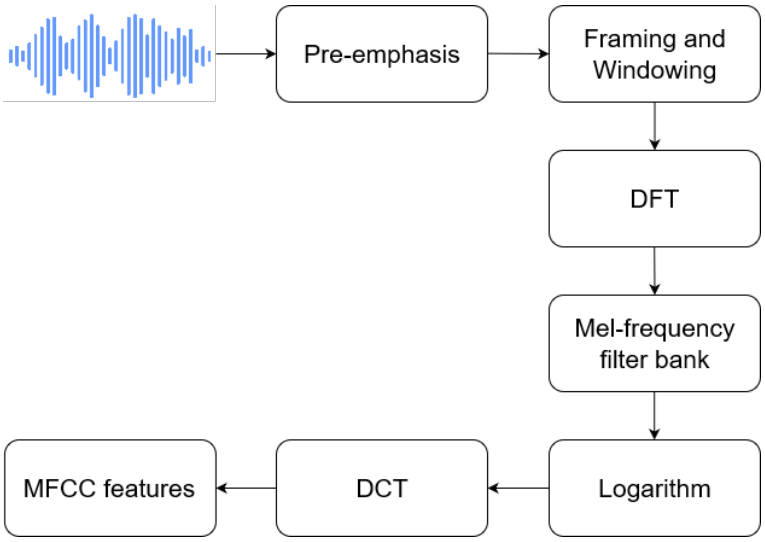

Figure 2.6: MFCC computation pipeline

- **Pre-emphasis** - Very common pre-processing practices in the signal processing area. Its purpose is to make up for the high frequency of the signal that was suppressed during signal generation. Pre-emphasis is the initial stage of the MFCC adaptation, and it may be implemented by simply setting a high-pass filter at [1, 0.97]. The energy distribution across frequencies and the total energy level are changed by the filtering process [ZW13].

- **Signal Framing and windowing** - Windowing was already introduced in 2.1.1. For stable acoustic characteristics, speech needs to be examined over a short enough time frame. Given that the time interval between two glottal closures is demonstrated to be around 20 ms, the speech signal's 20–30 ms time period is classified as a quasi-stationary segment. Vowel voices, however, are said to be recorded between 40 and 80 milliseconds [Ben+07]. As a result, short-term spectrum measurements are usually carried out over a 20-ms window, with a 10-ms overlap between each frame. 10-millisecond frame overlaps enable tracking of the temporal features of the voice signal. When speech frames overlap, the depiction of the sound would roughly center at one of the frames.

  Generally speaking, among the most well-known nominees are Hamming and Hanning windows [Re17]. When doing a DFT on the signal, these windows can reduce edge effect, smooth edges, and increase harmonics. Figure 2.7 illustrates the rectangular Hamming and Hanning windows in both time and frequency domains.

- **Power Spectrum** - The distribution of the power of the frequency compo-

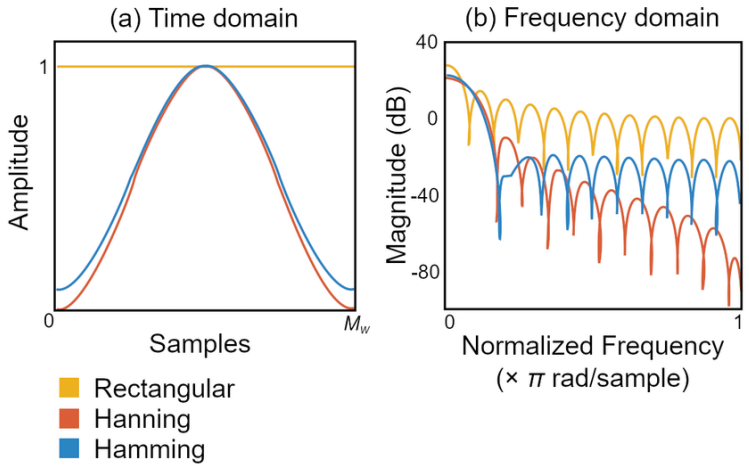

Figure 2.7: Rectangular Hamming and Hanning window in both time and frequency domain [Tim20]

nents that make up the signal is known as a power spectrum [FKP94]. Traditionally, the power spectrum is calculated using the Discrete Fourier Transform (DFT). The power spectrum of each of the obtained frames must be determined based on the below equation 2.1

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi jnK}{N}} \qquad k = 1, 2...N-1 \tag{2.1}$$

where x(n) is audio signal that is being processed and N is the length of the signal.

- **Mel-frequency Filter Bank** - The Mel band-pass filter is a bank of filters, which is constructed based on pitch perception. The Mel filter was first created for speech analysis, and it aims to extract non-linear representations of the voice signal, much like how human ears perceive speech. The conventional Mel filter-bank is constructed of 40 triangular filters [YHN11]. The transfer function of each of the n-th filter can be computed via equation 2.2 [AA22].

$$H_n(k) = \begin{cases} 0 & k < f(n-1) \\ \frac{k - f(n-1)}{f(n) - f(n-1)} & f(n-1) \le k < f(n) \\ 1 & k = f(n) \\ \frac{f(n+1) - k}{f(n+1) - f(n)} & f(n) < k \le f(n+1) \\ 0 & k > f(n+1) \end{cases} \tag{2.2}$$

where f(n) is the center frequency of the triangular filter. The Mel scale to the response frequency and vice versa is computed by equations 2.3 and 2.4

[Mol+01].

$$m = 2595 log_{10}\left(1 + \frac{f}{700}\right) \tag{2.3}$$

$$f = 700\left(10\frac{m}{2595} - 1\right) \tag{2.4}$$

- **Discrete Cosine Transform** - Discrete Cosine Transform (DCT) expresses a finite sequence of data points regarding a summation of cosine functions pulsating at various frequencies. In 1972, Nasir Ahmed unveiled the DCT. To choose the most accelerative coefficients or to isolate the relationship in the log spectral magnitudes from the filter bank, the DCT is applied to the Mel filter bank in the MFCC process [Str99]. The following formula 2.5 is used to calculate the DCT.

$$X(k) = \sum_{n=0}^{N-1} x_n * cos(\frac{2\pi jnk}{N}) \qquad k = 1, 2...N-1 \tag{2.5}$$

  where xn is the discrete signal and N is the length of the signal.

2.3 Chromagram

The chromagram, also known as the Harmonic Pitch Class Profile, represents the distribution of energy across different pitches or pitch classes [LT07]. First, we establish the chroma spectrum $X(n)$ as a metric for quantifying the intensity of a signal in relation to a specific chroma value: c. According to Shepard's pitch perception helix, chroma c can be defined as the fractional component of the logarithm of frequency, using a base-2 logarithm as can be seen in formula 2.6.

$$c = log_2 f - \lfloor log_2 f \rfloor \tag{2.6}$$

In the formula 2.6 $\lfloor\rfloor$ symbolizes the greatest integer function, which is also known as a floor function and rounds-off the real number to the integer less than the number. For example, 1.15 would be 1, and -3.1 would be -4. The chroma spectrum is similar to the conventional Fourier power spectrum. If we perform frame segmentation prior to it, we can generate a time-frequency distribution called $X(t, f)$ and establish a "timechroma" distribution called X (t, c). The distribution created from these previous two distributions is called the "chromagram". It is a modified version of a conventional time-frequency distribution, achieved by applying an aggregation function F as can be seen in formula 2.7.

$$X(t, c) = F(X(t, f)) \tag{2.7}$$

Where $f = 2^{c+h}$ which can be obtained from formula 2.6 and the function used is the summation function [Yu+10]. The elements of the chroma feature vector of the t frame $v(t,k)$ can be calculated with the formula 2.8:

$$v(t,k) = \sum_{n \in S_k} \frac{X_t(n)}{N_k} \quad k \in (0, 1...11) \tag{2.8}$$

Where $X_t(n)$ is the logarithmic magnitude of the Discrete Fourier Transform (DCT) for the t frame, N_k is the number of elements in S_k and S_k specifies a subset of the discrete frequency range for every pitch class. We take the arithmetic mean of all log magnitude DFT bins within a given set S_k. Then, we normalize each feature vector by subtracting the average value of the 12 features in that vector. The 12 sets S_k are formed by associating each Discrete Fourier Transform (DFT) bin with one of the 12 pitch classes. We determine the frequency associated with a DFT bin, and subsequently compute the chroma value using formula 2.6. The bin is linked to the pitch class that has the closest chroma value. Simply put, we need to reallocate chroma values in such a way that the pitch class C is positioned at the chroma value of 0 and the pitch class B is positioned at the chroma value of 1. The remaining pitch classes should then be positioned at chroma values of $k/12$. Ultimately, the range of the spectrum is limited. The lower limit is set at 20 Hz, while the highest limit is set at 2000 Hz [BW05].

2.4 Zero-Crossing rate

Zero-Crossing rate (ZCR) is defined as the number of times a signal crosses the zero line in the time domain, inside a specific region of the signal. This rate is calculated by dividing the number of zero-crossings by the number of samples in that region. The ZCR method was designed with a focus on effectively managing two types of additional sounds. The signal we are working with has a very short duration, often less than 100 milliseconds. As a result, a low-frequency note, performed by an instrument that overlaps with the signal, such as a bass, has a disruptive effect on the overall volume. The second form of noise we aim to address is related to the high-frequency components of other instruments. These components have amplitudes that are lower than the amplitude of the percussive sound around the onset. Examples of such components include voices and cymbals. The presence of these two attributes in signals is regarded as interference when calculating the Zero Crossing Rate (ZCR) of percussive sounds [GPD02].

2.5 Additional feature extraction methods

There is a large amount of feature extraction methods that are viable for audio feature extraction. Other methods that are used in the implementation part of this thesis are:

- **Spectral centroid** - The spectral centroid (SC) is the weighted average frequency of a specific subband. The weights used in this formula 2.9 are the normalized energy of each frequency component within that subband. By capturing the center of gravity of each subband, this measure is able to identify the rough position of formants, which are prominent peaks in a subband. Nevertheless, the subband's center of gravity is influenced by the harmonic structure and pitch frequencies generated by the vocal source. Therefore, alterations in pitch and harmonic structure have an impact on the SC trait [HK07].

$$SC_{i,b} = \frac{\sum_{f=l_b}^{u_b} f\,|S_i\,[f]|^2}{\sum_{f=l_b}^{u_b} |S_i\,[f]|^2} \tag{2.9}$$

- **Mel spectrogram** - Mel spectrogram is a transformation that provides a detailed representation of the frequency distribution of a signal as it changes over time. The Mel spectrogram is a visual representation of the audio data, it serves as the input for machine learning models such as Convolutional Neural Networks (CNNs) [ZLT19].

- **Spectral flatness** - Spectral Flatness shows how much the frequency is evenly distributed in a power spectrum. It is determined by calculating the ratio between the geometric mean and the arithmetic mean of a subband [ASS16].

- **Spectral Contrast** - Also referred to as Octave-based Spectral Contrast (OSC). OSC is defined as the difference between peaks, which generally corresponds to harmonic content in music, and valleys, where non-harmonic or noise components are more dominant, measured in subbands by octave-scale filters and using neighborhood criteria in its computation [Jia+02]. The spectral contrast features of the entire music piece are determined by calculating the mean and standard deviation of the spectral contrast and spectral peak values for all frames.

Text feature extraction

3

Text feature extraction is a procedure that captures relevant information from a text message. It serves as the foundation for many text-processing tasks. It is the process of selecting a group of features using many effective methods to decrease the size of the feature space. During the process of feature extraction, any features that are uncorrelated or unnecessary will be removed. Feature extraction is a data preprocessing technique that can enhance the accuracy and reduce the time required for learning algorithms [Lia+17]. Similarly to the audio feature extraction, we are trying to extract the most important information from our raw data in a form that can be used in our learning algorithm. Feature extraction comes hand in hand with data pre-processing.

3.1 Text Pre-Processing

Most text and document data sets often include extraneous words, such as stopwords, misspellings, and slang terms. Noise and superfluous characteristics can significantly impair the performance of various algorithms, particularly those related to statistical and probabilistic learning. This section presents techniques for cleaning text datasets, thereby eliminating inherent noise and enabling the extraction of meaningful features.

- **Stop words** - Text and document categorization sometimes involves words that lack significant relevance for use in classification methods. Words that would fall into this category are for example "*a*", "*the*", "*an*", and many others. The most common technique to deal with these kinds of words would be to simply remove them from the text.

- **Capitalization** - Sentences can be formed using text and document data points with varying capitalization. Given that documents are comprised of numerous phrases, the presence of varied capitalization might pose significant challenges when categorizing extensive texts. A widely used method

for addressing irregular capitalization is to convert all letters to lowercase. This method maps all words in the text to identical feature space. However, it presents a notable challenge in interpreting certain terms, such as the distinction between *"US"* (United States of America) and *"us"* (pronoun). Utilizing slang and abbreviation converters can assist in accommodating these deviations [DZ11].

- **Slang and Abbreviations** - Slang and abbreviations are additional types of text abnormalities that are addressed during the pre-processing stage. An abbreviation is a condensed version of a word or phrase that mostly consists of the initial letters of the phrases, such as **UN**, which is short for United Nations. Slang refers to a certain category of vocabulary that is used in casual conversations or written messages and has alternative meanings. For example, the phrase "lost their marbles" is slang and it signifies that someone has become mentally unstable or irrational. An effective approach to address these terms is to transform them into formal language [DKB16].

- **Spelling Correction** - Spelling correction is a voluntary pre-processing measure. Typographical errors are frequently found in texts and documents, particularly in social media text data sets, such as the dataset created from the Twitter platform. Researchers have access to several strategies and methods, such as hashing-based and context-sensitive spelling correction techniques, as well as spelling correction utilizing Trie and Damerau–Levenshtein distance bigram [DHD17] [MRN18].

- **Stemming** - In the field of natural language processing (NLP), a single word may occur in several forms, such as singular and plural noun forms, while maintaining an identical semantic meaning. Stemming is a technique used to merge various variants of a word into a unified feature space. Text stemming modifies words to obtain variant word forms using different linguistic processes such as affixation (addition of affixes). For instance, the base form of the word "studying" is "study" [SG16] [To007].

- **Lemmatization** - It is a natural language processing technique that modifies the suffix of a word by either replacing it with a new suffix or removing it entirely, in order to obtain the fundamental form of the word, known as the lemma [PLM04].

- **Tokenization** - Tokenization is a preprocessing technique that divides a sequence of text into individual words, phrases, symbols, or other significant units known as tokens. The primary objective of this stage is to examine the words within a sentence. Both text categorization and text mining necessitate

a parser that handles the tokenization of the documents, for example, the sentence "*Today is Monday*" would be separated into tokens: {"*Today*","*is*","*Monday*"}.

Pre-processed text then needs to be converted into a numerical representation. There are several techniques that can accomplish this task, but they differ in their appropriate usage. We can for example represent words with their syntactic meaning using *Bag of Words* model.

- **Bag of Words** - The Bag of Words model (BOW) is a concise and simplified depiction of a text document that focuses on specific aspects of the text, such as word frequency. In the BoW model, a body of text, such as a document or a sentence, is conceptualized as a collection of individual words without considering their order. The BoW procedure involves the creation of a list of words and their occurrences in the processed data.

However, the bag-of-words model does not respect the semantics of the word. For instance, the words "airplane", "aeroplane", "plane", and "aircraft" are frequently employed interchangeably. Nevertheless, the vectors associated with these words are perpendicular in the bag-of-words model. This issue poses a significant challenge to comprehending sentences within the model. Another issue with the bag-of-words approach is that it does not consider the order of words in a phrase.

Another approach that is frequently deployed is the *TF-IDF* measure

- **TF-IDF** - The *Term frequency-inverse document frequency* (TF-IDF) approach is commonly used to assign a weight to each word in our data based on the uniqueness of the said word among other words in the data. Assuming that a document d is represented by a set of words $(t_1, t_2, ...t_n)$ where each word t_i has its assigned weight calculated by the statistics $TF(w_i, di)$ and $IDF(w_i)$, document d can be therefore represented by a n-dimensional vector $d = (w_1, w_2, ...w_n)$ where n is the total number of various words in the document. Weight is a metric that signifies the statistical significance of related words. The weight of a word t_i, denoted as w_i, can be calculated by multiplying the term frequency (TF) of the word in a document d with the inverse document frequency (IDF) of the word. The term frequency (TF) value is directly proportional to the frequency of a word in a document, while the inverse document frequency (IDF) value is inversely proportional to its frequency in the entire document corpus [YLY05].

TF-IDF usually performs better in machine learning models. But similarly to the BOW model, TF-IDF is also missing the semantic value of the words and is therefore not suited for a lot of natural language processing tasks.

Since the semantics of our dataset is quite important, we use a feature learning technique called *Word embeddings*. Word embedding is a method of learning features where each word or phrase in the vocabulary is associated with a vector of real numbers in N dimensions. Several word embedding techniques have been suggested to convert individual words into interpretable input for machine learning algorithms. The most commonly used deep learning algorithms for word embeddings include Word2Vec, GloVe, and FastText.

3.2 Word2Vec

Word2vec is a method used in natural language processing (NLP) to acquire vector representations of words. These vectors encode semantic information by analyzing the context in which the word appears. The word2vec method produces these representations by analyzing text in a vast corpus. Once the model is trained, it has the ability to identify words that have the same meaning or propose alternative words for an incomplete sentence. Tomáš Mikolov and his colleagues at Google developed Word2vec, which was published in 2013 [Mik+13].

Word2vec encodes a word as a multi-dimensional vector of numerical values that capture the connections between words. Specifically, words that occur in comparable contexts are assigned vectors that are close to each other in terms of cosine similarity. This demonstrates the degree of semantic similarity between words. For instance, the vectors representing "cat" and "dog" are close to each other, as are the vectors for "furthermore" and "moreover," and "Paris" and "France."

Every word in the corpus is initially encoded as a high-dimensional vector with randomly assigned values. These vectors act as the first reference for the training process. The size of these vectors normally ranges from 100 to 300, and occasionally reaches up to a thousand, depending on the size of the corpus and the specific requirements of the work at hand. The process of randomly initializing the model's parameters serves the purpose of breaking symmetry and ensuring that the model acquires meaningful information throughout the training phase. During the training process, these vectors are modified according to the objective function of the Word2Vec model. This objective function aims to place vectors of words that appear in comparable situations closer together in the vector space.

After obtaining the first word vectors, the subsequent step is to optimize these vectors in order to more effectively represent the linguistic contexts of words. Optimization algorithms, such as gradient descent and its variants, are employed to accomplish this task. The primary concept is to progressively modify the word vectors in order to enhance the alignment between the model's predictions and the actual context words. The alignment is quantified by the objective function, and the modification is performed using a technique called backpropagation. Backprop-

agation is an algorithm employed in neural networks to compute the gradient of the loss function in relation to the network's weights. Within the framework of Word2Vec, backpropagation modifies the word vectors by taking into account the discrepancies in forecasting context words. With each iteration, the model improves its accuracy in making predictions, resulting in optimized word vectors.

The selection of window size is another crucial factor in training Word2Vec embeddings. The window size refers to a movable window that scans the text and identifies the words that are examined in relation to a specific target word. Context words are defined as the words that are within the window, whereas words beyond the window are disregarded. The selection of window size directly affects the quality of the acquired word vectors. A reduced window size facilitates the acquisition of knowledge regarding the word's syntactic functions, whereas an increased window size enhances the model's comprehension of the wider semantic context. Nevertheless, there is a compromise to take into account. The computational cost is increased as the window size is higher, as it requires processing more context words for each target word. Hence, the selection of window size must be conducted with caution, considering both the computational resources and the specific demands of the activity.

Another important part of the learning that improves overall performance is the Negative Sampling. Negative sampling solves the problem of computing efficiency by updating only a fraction of the model's weights at each step, rather than updating all of them. This is accomplished by selecting a limited number of "negative" words (words that are not part of the context) to update for every target word. Conversely, when we choose sample frequent words, it enhances the quality of word vectors. The fundamental concept is to mitigate the influence of high-frequency words during the training process, as they often convey less significant information in comparison to infrequent words. By selectively rejecting certain instances of commonly occurring words, the model is compelled to prioritize the less common terms, resulting in word vectors that are more evenly distributed and meaningful.

The Word2vec model can be implemented through either the Continuous Bag of Words model (CBOW) or the Continuous Skip Gram model. Both of these models can create dense word vectors while reducing the dimensionality of the input data.

3.2.1 Continuous Bag of Words Model

Input to the CBOW model is a context of words surrounding the word we want to predict and output is a single word. Depending on the context window size, we take multiple surrounding words and feed them as input into the hidden layer of this model. Matrice W is the weight matrices between the input layer and the hidden layer and matrice W' is the weight matrices between the hidden layer and output.

CBOW model architecture is depicted in Figure 3.1.

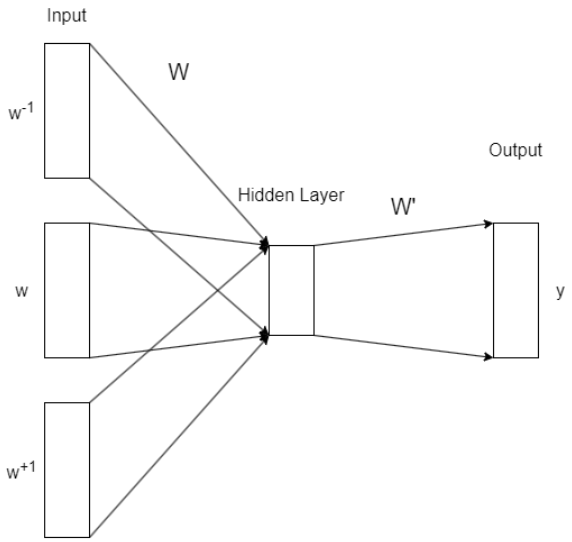

Figure 3.1: CBOW architecture

3.2.2 Skip Gram model

Similar to the CBOW model, the Skip Gram model has one hidden layer and two weight matrices *W* and *W'*. A big difference between these models is the input and output approach. Skip Gram model takes one word as input and predicts the context of the given word. Context size that has to be predicted can vary, but the principle remains the same. Simple Skip Gram model architecture can be seen in Figure 3.2.

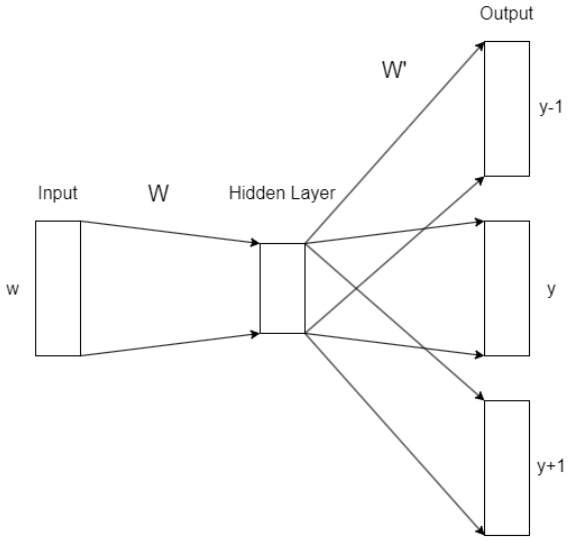

Figure 3.2: Simple Skip Gram model architecture

3.3 Contextual word vector representation

While word2vec has satisfactory accuracy in pre-training, its CBOW and Skip-Gram models focus on extracting contextual information, but they are limited and do not consider generic information. In 2014, Pennington [PSM14] introduced the *GloVe* model, which utilized the co-occurrence matrix to simultaneously incorporate both local and global information. Despite enhancements, both word2vec and GloVe produce a static encoding for trained word vectors [PSM14]. Following the completion of training, the word vector remains constant and does not undergo any changes. However, the meaning of a word can vary in different settings. In 2018, Devlin [Dev+19] introduced a BERT pre-training model that utilizes a multilayer two-way Transformer encoder. The model utilizes the contextual information from both the left and right sides of all layers to extract a comprehensive two-way representation of the text [Dev+18]. The pre-training model comprehensively extracts words, sentences, and situations, and acquires dynamically encoded word vectors. That is to say, the meaning of a term can vary depending on the context in which it is used. Hence, this study presents a text sentiment categorization model that relies on BERT. The network model initially acquires word vectors that encompass contextual semantic information by utilizing the BERT pre-training model. Subsequently, it employs the bi-directional long and short-term memory network to extract context-related features for the purpose of deep learning. The attention technique is implemented to allocate weights to the retrieved information, emphasize the crucial details, and determine the sentiment polarity of the text.

3.3.1 Attention mechanism

Attention is a method that enables the model to focus on and assimilate significant information. In the conventional sequence-to-sequence model, a text sequence is often encoded using mechanisms like Convolutional Neural Network (CNN) or Long Short-Term Memory Network (LSTM). This encoding process involves reducing the dimensionality of the sequence and transforming it into a fixed-length vector. The resulting vector is then fed as input to the Fully connected layer. Nevertheless, this gives rise to an issue. Traditional coding methods are unable to accurately represent the level of focus on various morphemes in a sequence of sentences. Within the realm of natural language, the components of a sentence possess distinct meanings and varying levels of significance. The incorporation of the Attention mechanism effectively resolves this issue. The conventional Seq2Seq model has an encoder layer and a decoder layer. During the encoder phase, each node's input consists of the concealed state of the preceding node combined with the next word. Ultimately, the encoder will generate a context and transmit it to the decoder layer. Within the attention mechanism, the decoder layer will receive the hidden state from all encoder levels and determine which hidden state is more closely associated with the text. This will incorporate a weight parameter into the hidden state and subsequently apply SoftMax computations to each weight value. As the value of the hidden state increases, the correlation also increases, and conversely, as the value of the hidden state decreases, the correlation decreases. The popular weight calculation functions are multilayer perceptron, Bilinear, dot product, and scaled-dot product.

3.3.2 Transformer

The transformer model shares the same encoder-decoder architecture as the Attention model. Nevertheless, the architecture in the Transformer model is more intricate than that in Attention. The article describes the encoder layer and decoder layer as consisting of six stacks. Each Encoder in the set of Encoders has the same structure, but they do not have shared weights. Encoders consist of two layers: a self-attention layer and a feedforward neural network. By utilizing the self-attention layer, the encoder examines the meaning of other words in the input sequence and the surrounding context while encoding words. Following the entry of each word into the Self-Attention layer, there will be a corresponding output. The Self-Attention layer relies on the input and output being interdependent. Each word in every location is initially processed by a self-attention layer, and then each word is individually processed by a feedforward neural network that has the same structure [Vas+17].

3.3.3 BERT

The BERT pre-training model, developed in 2018, is a large-scale multi-task language processing model that utilizes the attention mechanism algorithm [Dev+18]. The utilization of Word2Vec for word vector representation is fundamentally flawed. The words it acquires are inadequate to convey the shift in context, nor can they resolve the issue of terms with multiple meanings. BERT is a neural network architecture that consists of several bidirectional Transformer encoders. BERT is able to capture both the preceding and following contextual information simultaneously. BERT is a type of learning model that uses a multilayer Transformer as its primary architecture to extract linguistic information [Xu+19]. It operates in a semi-supervised manner.

The BERT model exhibits robust compatibility, enabling its adaptation and utilization across several domains, such as sentiment analysis. Akbar Karimi and his team introduced a BERT Adversarial Training approach to simplify the sentiment analysis work, which typically involves labeling words [KRP21]. They utilized BAT (BERT Adversarial Training) to implement adversarial training for the two primary objectives of sentiment analysis: Aspect Extraction and Aspect Sentiment Classification, resulting in high accuracy.

The BERT model necessitates the utilization of token embedding, segment embedding, and position embedding. Token embedding is the process of embedding word vectors. Segment embedding refers to the specific application scenario of the input sentence. Position embedding involves creating a position code based on the position information of each added word. These enhance the text's integrity by providing the relative position of words and application situations for the model. BERT uses MLM (Masked Language Model) to conceal certain parts of the language model, enhancing its efficiency and brevity. A mask is employed to conceal some variables throughout the calculation process, preventing them from being utilized for parameter adjustments. Figure 3.3 shows BERT model architecture. BERT is bidirectional, meaning it processes text in both directions simultaneously, as opposed to for example OpenAIs GPT model.

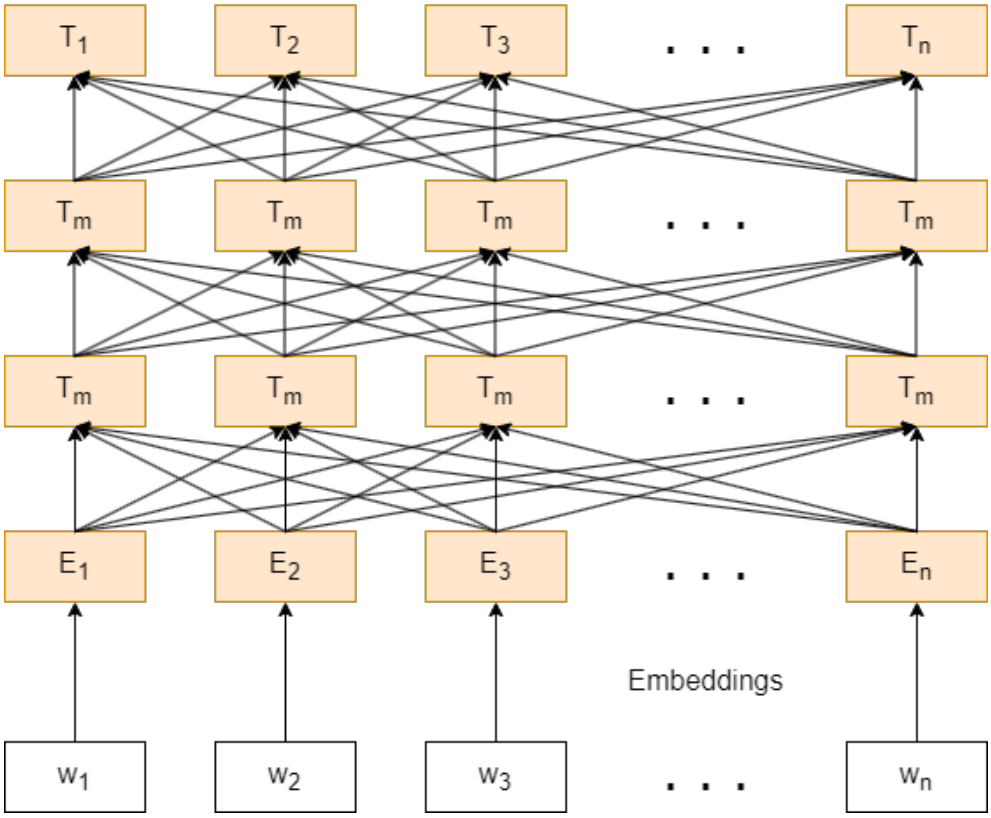

Figure 3.3: BERT model architecture

Artificial Neural Networks

4

Artificial Neural Networks (ANNs) are essentially massively parallel computational models designed to mimic the way the human brain works. A vast number of basic processors connected via weighted connections make up an ANN. The processing nodes could be referred to as "neurons" in an analogy. Each node output depends only on the information that is locally available at the node, whether it comes from internal storage or weighted connections. Every unit gets input from numerous other nodes and sends its output to further nodes. A single processing element is not very strong on its own, it produces a scalar output, which is a simple non-linear function of its inputs, with a single numerical value. The power of the system emerges from the combination of many units in an appropriate way. The ANN does not really solve the problem in a strictly mathematical sense. Instead, it shows information processing properties. ANNs are extensively utilized in image processing, pattern identification and classification, complex nonlinear function mapping, and other fields. One popular kind of neural network is a feed-forward network. A feed-forward network consists of an input layer that receives the problem's inputs, hidden layers that determine and reflect the relationship between the inputs and outputs using synaptic weights, and an output layer that outputs the problem's outputs [Kuro4]. A feed-forward neural network is modeled with three main elements:

1. *A set of synapses* - Each synapsis is characterized by its synaptic weight.
2. *Linear combiner* - Used for summing the input weights of a signal.
3. *Activation function* - Limits the amplitude of the output of certain neurons to some finite number. By utilizing a bias term, the activation function's input can be enhanced.

A typical artificial neuron can be seen in Figure 4.1.

The input values are determined by the synaptic connections leading into the neuron (x1, x2 ... xn) and their according weights. Another input into the neuron is the bias term. The following formula 4.1 is used to calculate the output value of this

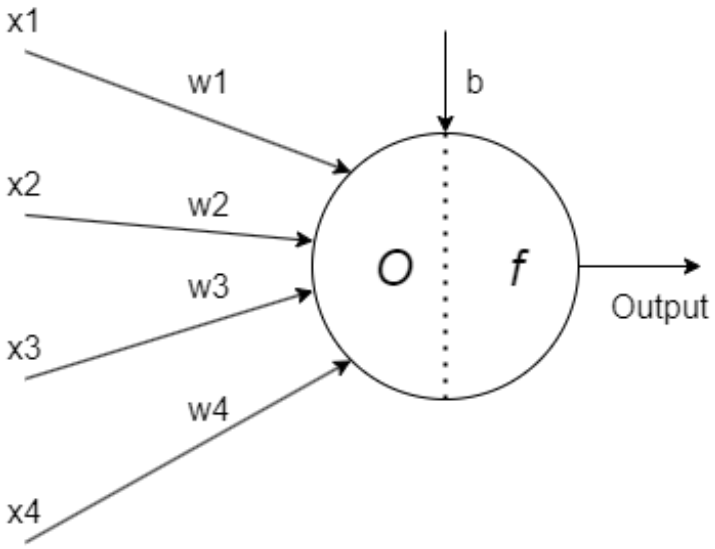

Figure 4.1: Typical artificial neuron

neuron.

$$O = f(X) = f(\sum_{i=1}^{n} w_i x_i + b) \tag{4.1}$$

Where w is the weight factor of each synaptic connection leading into the neuron, x refers to the actual value given by the connection from the previous layer, b is the bias term corresponding to this neuron and f is the activation function of this neuron. There are several viable options for the activation function for example, *Sigmoid*, *RELU*, and so on.

Three different kinds of neuron layers make up the fundamental architecture: input, hidden, and output layers. The signal flows from input to output units in feed-forward networks, strictly in a feed-forward direction. There are no feedback links, but the data processing can span several (layers of) units. There are feedback links in recurrent networks. In contrast to feed-forward networks, the network's dynamic characteristics are significant. A typical Multilayered feed-forward neural network can be seen in Figure 4.2.

Values from the input layer are fed into a hidden layer. Each input is forwarded to every neuron in the hidden layer. Depending on the network's architecture, there can be more than one hidden layer. Outputs from the network are then interpreted depending on the application.

Activation values of the units can relax in some instances, leading to the network evolving toward a stable state where these activations remain constant. In other applications, the network's output is determined by the dynamic behavior caused by large variations in the activation values of the output neurons [Bis95]. A neural net-

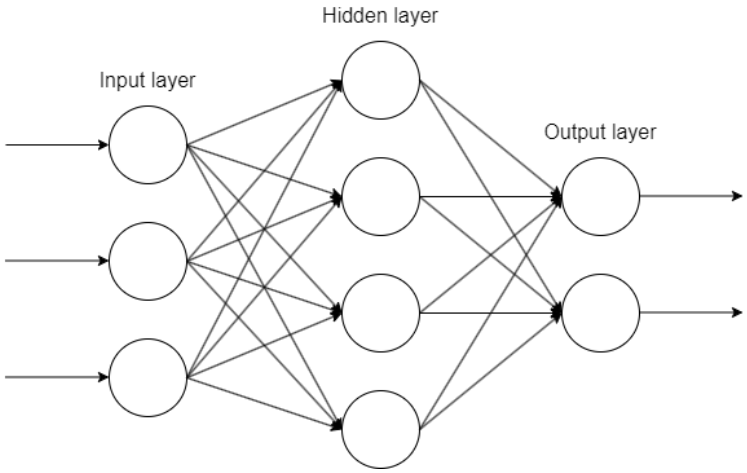

Figure 4.2: Multilayered ANN

work must be set up so that a given collection of inputs can be applied to generate a desired set of outputs. There are several ways to determine the links' strengths. One approach is to use a priori information to explicitly set the weights. An alternative method involves training the neural network by providing it with instructional patterns and allowing it to adjust its weights based on a specific learning algorithm. The learning scenarios in neural networks can be categorized into three distinct types. The three types of learning are supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, an input vector is provided at the inputs along with a corresponding set of intended replies, one for each node, at the output layer. During the forward pass, the disparities between the desired and actual response for each node in the output layer are identified. These are subsequently utilized to calculate alterations in the weight of the network based on the current learning rule. The word *supervised* is derived from the fact that the intended signals on each individual output node are given by an external teacher. The most widely recognized instances of this methodology are found in the backpropagation algorithm, the delta rule, and the perceptron rule. In the field of unsupervised learning, often known as self-organization, a unit is trained to recognize and respond to groups of patterns in the incoming data. In this paradigm, the system is expected to identify statistically significant characteristics of the input population. In contrast to the supervised learning approach, the system does not have a predefined set of categories for classifying patterns. Instead, it must create its own representation of the incoming stimuli. Reinforcement learning involves acquiring knowledge on how to effectively associate different events with appropriate behaviors in order to optimize a numerical reward signal. In this sort of machine learning, the learner is not provided with instructions on which actions to do. Instead, the learner must

determine which activities result in the highest reward through trial and error. In complex and stimulating scenarios, actions might have an impact not only on the immediate outcome, but also on the subsequent situation and, consequently, on all future rewards. The differentiating features of reinforcement learning include trial-and-error search and delayed reward.

4.1 Long Short-Term Memory Network

The LSTM model is a robust recurrent neural system specifically created to address the issues of exploding or vanishing gradients that often occur when learning long-term dependencies, even when the time lags are extremely long. In general, this issue can be avoided by implementing a constant error carousel (CEC) that keeps the error signal contained within each unit's cell. In reality, these cells are actually recurrent networks that have a unique architecture. This architecture involves extending the CEC with extra features, including the input gate and output gate, which together comprise the memory cell. The self-recurrent connections signify the presence of feedback with a delay of one-time step [HS97] [HS96]. A basic LSTM unit consists of a cell, an input gate, an output gate, and a forget gate. The forget gate was not originally included in the LSTM network, but it was suggested by Gers as a means for the network to reset its state [GSC00]. The cell retains values for indefinite time periods, and the three gates control the transmission of information within the cell. In the next section, the term LSTM will be used to refer to the basic form, which is the most widely used LSTM architecture [Gre+17]. However, this does not necessarily mean that it is also the preferable option under all circumstances.

Essentially, the LSTM architecture comprises a collection of interconnected sub-networks called memory blocks. The purpose of the memory block is to preserve its state over time and control the flow of information through nonlinear gating units. Figure 4.3 illustrates the structure of a basic LSTM block, which includes gates, the input signal x^t, the hidden state h^t, activation functions, and peephole connections [GS00]. The output of the block (new hidden state) is connected in a recurring manner to both the block input and all of the gates.

To provide a clearer understanding of the functioning of the LSTM model, let us consider a network consisting of N processing blocks and M inputs. The process of transmitting information in this recurrent neural system is outlined below.

1. **Block input** - This step focuses on updating the block input component, which merges the current input x^t, with the output of the LSTM unit h^{t-1} from the previous iteration. This can be calculated with equation 4.2.

$$z^t = g(W_z x^t + R_z h^{t-1} + b_z) \tag{4.2}$$

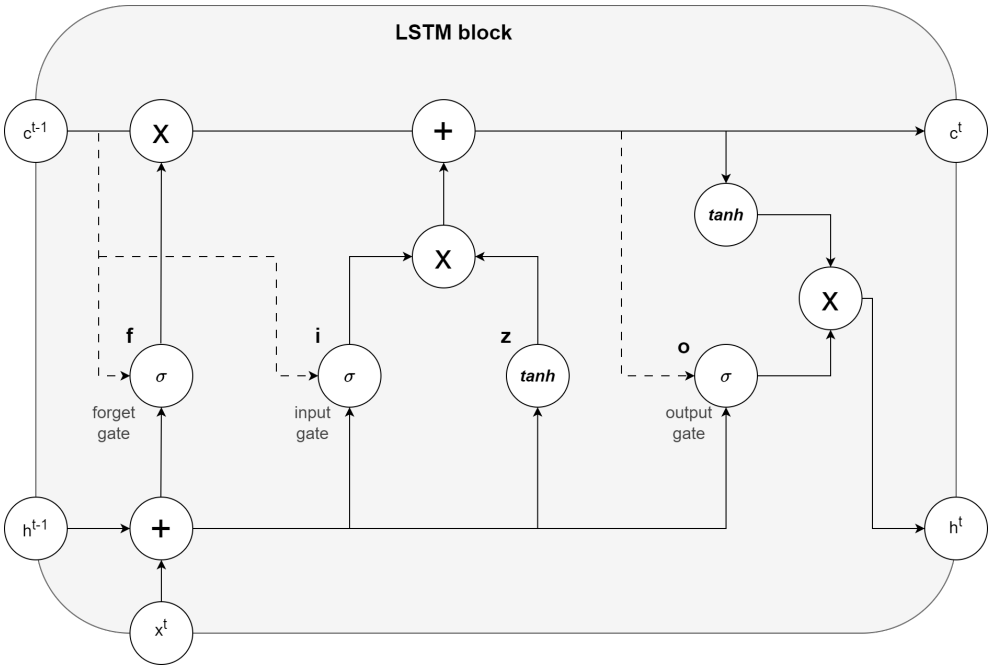

Figure 4.3: An architecture of LSTM block

Where W_z and R_z are weights associated with x^t and h^{t-1}, respectively, while b_z is the bias weight vector.

2. **Input gate** - During this phase, we modify the input gate, which merges the current input x^t, the output of the LSTM unit h^{t-1}, and the cell value c^{t-1} from the previous iteration. This can be calculated with the following equation 4.3.

$$i^t = \sigma(W_i x^t + R_i h^{t-1} + p_i \otimes c^{t-1} + b_i) \tag{4.3}$$

Where W_i, R_i and p_i are weights associated with x^t, h^{t-1} and c^{t-1}, respectively. b_i represents the bias vector and lastly the $\otimes$ denotes the point-wise multiplication of the two vectors. During the preceding stages, the LSTM layer identifies the specific information that has to be preserved in the network's cell states, denoted as c^t. This involves the identification of the candidate values z^t that may potentially be incorporated into the cell states, as well as the determination of the activation values i^t for the input gates.

3. **Forget gate** - During this stage, the LSTM unit identifies the specific information that has to be eliminated from its prior cell states c^{t-1}. Hence, the activation values f^t of the forget gates at time step t are determined by considering the current input x^t, the outputs h^{t-1}, the state c^{t-1} of the memory cells at the previous time step *(t - 1)*, the peephole connections, and the bias terms

b_f of the forget gates. This can be calculated with the following equation 4.4.

$$f^t = \sigma(W_f x^t + R_f h^{t-1} + p_f \otimes c^{t-1} + b_f) \tag{4.4}$$

Where W_f, R_f and p_f are the weights associated with x^t, h^{t-1} and c^{t-1}, respectively, while b_f denotes for the bias weight vector.

4. **Cell** - This step calculates the cell value by combining the block input z^t, the input gate i^t, and the forget gate f^t values with the previous cell value. This can be calculated with equation 4.5.

$$c^t = z^t \otimes i^t + c^{t-1} \otimes f^t \tag{4.5}$$

5. **Output gate** - This phase computes the output gate by combining the current input x^t, the output of the previous LSTM unit h^{t-1}, and the cell value c^{t-1} from the last iteration. This can be calculated with the following equation 4.6.

$$o^t = \sigma(W_o x^t + R_o h^{t-1} + p_o \otimes c^t + b_o) \tag{4.6}$$

Where W_o, R_o and p_o are the weights associated with x^t, h^{t-1} and c^{t-1}, respectively, while b_o denotes for the bias weight vector.

6. **Block output** - Lastly, we can calculate the output from this LSTM block, which combines the current cell state c^t with output gate value through following equation 4.7.

$$y^t = g(c^t) \otimes o^t \tag{4.7}$$

In the previous stages, σrepresents the point-wise non-linear activation function. The logistic sigmoid function $\sigma(x) = \frac{1}{1+e^{1-x}}$ is commonly employed as a gate activation function. On the other hand, the hyperbolic tangent function *tanh* is frequently utilized as the activation function for both the input and output of a block. Despite the already impressive performance of vanilla LSTM, multiple studies have explored ways to enhance its capabilities. For instance, Su and Kuo devised the Extended LSTM model, which enhanced the accuracy of predictions in several application domains by augmenting the memory capacity [SK19]. This demonstrates that there is still room for theoretical enhancements to be made to an architecture that is already performing at a high level. In the study conducted by Bayer, efforts to enhance the model were made. The authors sought an architectural alternative to LSTM in order to enhance the capabilities of sequence learning. They achieved the development of memory cell structures that can learn context-sensitive formal languages by gradient descent, exhibiting similar performance to LSTM models [Bay+09]. The researchers in reference [Bel+18] expanded upon recurrent networks

composed of spiking neurons by creating Long short-term memory Spiking Neural Networks (LSNN) that incorporate adaptive neurons. In experiments when the LSNN size was similar to that of LSTM, it was demonstrated that the performance of LSNN is highly comparable to that of LSTM. This serves as another demonstration of the precision and consistency of LSTM.

4.2 Convolutional Neural Networks

CNNs typically operate under the assumption that the input will consist predominantly of the images. This directs the design of the architecture to be arranged in a manner that is most suitable for handling the particular kind of data. A significant distinction lies in the composition of neurons inside the layers of the CNN, which are organized into three dimensions: the spatial dimensions of the input (height and width) and the depth. The term "depth" in this context does not pertain to the overall number of layers in the ANN, but rather to the third dimension of an activation volume. In contrast to typical ANNS, the neurons in each layer of this network only establish connections with a limited area of the preceding layer.

CNNs consist of three types of layers. The three types of layers used in this context are convolutional layers, pooling layers, and fully-connected layers. Once these layers are arranged in a specific order, a Convolutional Neural Network (CNN) architecture is created. Figure 4.4 depicts a simple Convolutional Neural Network (CNN) structure designed for the purpose of classifying MNIST data.

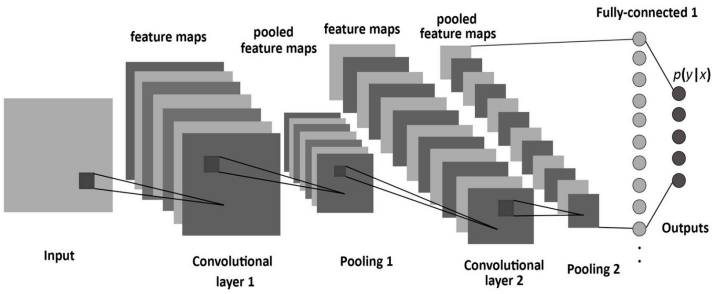

Figure 4.4: A CNN architecture [AM17]

The main functionality of CNN can be broken down into four key layers.

- **Input layer** - Same as in typical ANN, this layer hold the real values that are passed down into the architecture. In the MNIST example, it would be the actual pixel values of an input image.

- **Convolutional layer** - The output of neurons is determined by the scalar product between their weights and the local regions of the input volume to which they are connected.
- **Pooling layer** - Downsampling is applied to the input, which reduces the number of parameters in the activation by reducing its spatial dimensionality.
- **Fully-connected layer** - Performs equivalent tasks as those found in conventional ANNs and aims to generate class scores based on the activations, which are then utilized for classification.

CNNs utilize a straightforward methodology of transformation to process the original input layer by layer. This is achieved through the application of convolutional and downsampling techniques, resulting in the generation of class scores for the purposes of classification and regression.

4.2.1 Convolutional layer

The convolutional layer is essential for the functioning of CNNs, as its name suggests. The parameters of the layers revolve around the utilization of trainable kernels. Typically, these kernels have a tiny size in terms of spatial dimensions, but they extend across the entire depth of the input. When the data reaches a convolutional layer, the layer applies convolution with each filter across the spatial dimension of the input, resulting in a 2D activation map. The convolution operation can be seen in Figure 4.5. As we iterate through the input, the scalar product is computed for each value in the kernel. The network will acquire knowledge about kernels that activate when they detect a certain feature at a specified spatial location in the input. These are frequently referred to as activations.

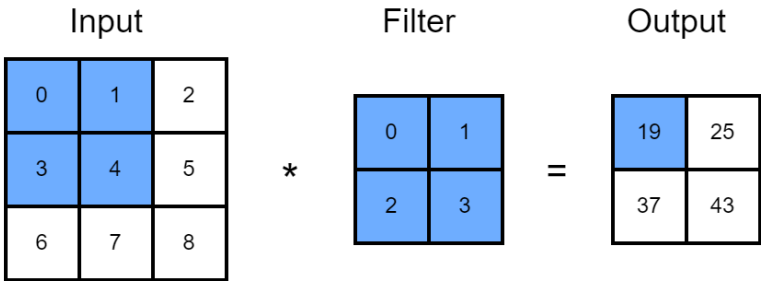

Figure 4.5: A visual representation of convolutional layer

Each kernel will generate an activation map, which will be arranged in a stack along the depth dimension to create the complete output volume of the convolutional layer. As previously said, training ANNs on inputs like photos leads to models

that are excessively large to be trained efficiently. The reason for this is the completely linked nature of standard ANN neurons. To address this issue, each neuron in a convolutional layer is only connected to a limited part of the input volume. The dimensionality of this region is sometimes denoted as the receptive field size of the neuron. The level of connection throughout the depth is consistently proportional to the depth of the input.

For instance, if the network receives an image with dimensions of 28×28×3 (representing an RGB-colored image with a size of 28×28), and we define the receptive field as 3×3, each neuron within the convolutional layer would have a total of 27 weights. The volume has dimensions of 3 units in length, 3 units in width, and 3 units in depth, with a connectivity magnitude of 3 throughout the depth. For comparison, a typical neuron seen in other types of ANN would consist of 2,352 individual weights.

Convolutional layers can effectively decrease the model's complexity by optimizing its output. Optimization of these is achieved by adjusting three hyperparameters: depth, stride, and zero-padding configuration. The output volume's depth generated by the convolutional layers can be manually adjusted by specifying the number of neurons within the layer to match the input's region. This phenomenon is seen in other types of ANNs, where each neuron in the hidden layer is directly linked to every neuron in the preceding layer. Decreasing this hyperparameter can greatly lower the overall number of neurons in the network, but it can also drastically diminish the model's ability to recognize patterns. We can also specify the stride at which we establish the depth around the spatial dimensions of the input to position the receptive field. For instance, if we were to establish a stride of 1, the receptive field would be extensively overlapped, resulting in significantly massive activations. Alternatively, increasing the stride value will decrease the amount of overlap and result in an output with smaller spatial dimensions.

An inherent limitation of the convolution stage is the potential loss of information located at the edges of the image. Since they are exclusively collected during the sliding of the filter, they never get the opportunity to be observed. An effective approach to address the problem is to employ zero-padding. Another advantage of zero padding is its ability to control the size of the output. Suppose we have an input size of 10x10 and kernel 3x3 with stride 1. The output size after the convolution would be 8x8. However, by adding the zero padding, the output size will be the original 10x10. The following formula 4.8 is used for calculating the output size of convolution

$$O = 1 + \frac{I + 2P + K}{S} \tag{4.8}$$

Where I is the input size, P is the padding, K is the filter kernel size, and S is the

stride. The concept of padding assists in maintaining the size of the network output as the depth increases, preventing it from decreasing. Consequently, it is feasible to possess an unlimited number of deep convolutional networks [ON15].

4.2.2 Pooling layer

The primary concept of pooling is to do down-sampling in order to decrease the complexity for subsequent layers. In the field of image processing, it might be regarded as analogous to decreasing the resolution. Pooling has no impact on the quantity of filters. Max-pooling is a widely used form of pooling procedure. The image is divided into sub-region rectangles, and only the highest value inside each sub-region is returned. A frequently employed dimension for max-pooling is 2×2. As depicted in Figure 4.6, when pooling is applied to the 2×2 blocks in the top-left region (indicated by the pink area), it shifts by 2 units and emphasizes the top-right portion. Pooling is performed using a stride of 2. To prevent down-sampling, an uncommon technique is to utilize a stride of 1. It is important to note that down-sampling does not maintain the positional integrity of the information. Thus, it should be utilized exclusively when the significance of information is paramount, as opposed to geographical information. Furthermore, pooling can be employed with filters and strides that are not of equal size in order to enhance efficiency. For instance, when using a 3x3 max-pooling operation with a stride of 2, there will be some overlapping regions between the areas being pooled.

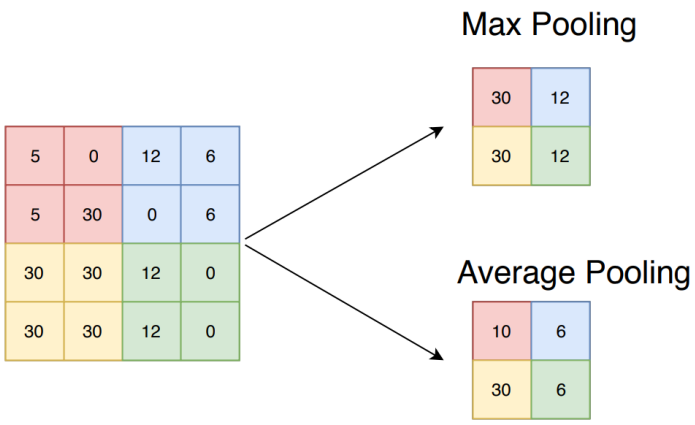

Figure 4.6: A visual representation of max and average pooling

4.2.3 Fully-connected layer

The fully-connected layer mimics the arrangement of neurons in a conventional neural network. Thus, every node in a fully connected layer is directly linked to every node in both the preceding and subsequent layers. A significant limitation of a fully connected layer is the high number of parameters it contains, which necessitates sophisticated computational operations during training. Hence, our objective is to reduce the number of nodes and connections. The eliminated nodes and connections can be addressed by implementing the dropout approach. For instance, LeNet and AlexNet developed a network that is both deep and wide, while maintaining a consistent level of computational complexity [Guo+15].

Datasets 5

5.1 ECF

ECF dataset [Wan+23; Wan+24] contains 7 classification classes for emotions including *Anger*, *Disgust*, *Fear*, *Joy*, *Sadness*, *Surprise* and *Neutral*. The distribution of these classes can be seen in table 5.1.

Emotion	Representation
Joy	16.89
Surprise	13.51
Anger	11.85
Sadness	8.42
Disgust	3.03
Fear	2.73
Neutral	43.53

Table 5.1: Distribution of emotion classification classes in ECF dataset (in %)

This Dataset contains 13619 utterances in 1374 conversations.
ECF dataset summarizes emotion causes into four main types:

- **Event:** This type of emotion-cause is evoked by something that happened in a particular situation. For example utterance *"You broke my vase?"* would evoke *Anger*.

- **Greeting:** Giving a sign of welcome. Greeting is a cause for *Joy*. So emotion in a simple utterance like *"Oh hi Joey."* would be *Joy*.

- **Opinion:** Emotion is caused by someone's feelings or thoughts about something. For example, a sentence *"I can't be mad at you"* would evoke *Joy* in its conversation counterpart.

- **Emotional Influence:** Emotion caused by the emotion of conversation counterpart. For example, when Joey is angry at Chandler, it induces sadness in Chandler

Each utterance has its text translation and corresponding video file. These files are cut from the *Friends* sitcom without any post-processing so there is a considerable amount of noise coming from background actors, music, or laughing from the studio audience. All of the video clips are provided in good quality with a 44100 Hz audio sample rate and their average length is 3.2 seconds.

5.2 RAVDESS dataset

Ryerson Audio-Visual Database of Emotional Speech and Song Dataset (RAVDESS) dataset [LR19] consists of 7356 files. These files are created with 24 actors (12 male and 12 female), vocalizing two lexically-matched statements in a neutral North American accent. Actors perform these statements with angry, fearful, calm, happy, sad, surprise, and disgust facial expressions and say them with calm, happy, sad, angry, and fearful vocal emotions. Each emotion is performed with two levels of intensity (normal and strong), with additional neutral expression. There are three possible modalities available:

- **Audio only** - files containing all actors speaking. They are further divided into speech files and song files. For our purpose, only speech files are relevant. There are 1440 speech files. 60 per actor. The audio signal is recorded in a 16-bit bitrate with 48kHz frequency.
- **Audio visual** - same as the audio only files. This part of the dataset consists of 1440 files of recorded actors speaking in certain emotions with certain facial expressions. These files have audio with the same bitrate and frequency and have 720p video quality.
- **Video only** - These files are the audiovisual files without the audio track and with the same video quality.

As mentioned before, there are 8 emotions depicted in this dataset. Their distribution in the dataset can be seen in table 5.2.

Emotion	Representation
Happy	13.33
Surprised	13.33
Angry	13.33
Sad	13.33
Disgust	13.33
Fearful	13.33
Neutral	6.66
Calm	13.33

Table 5.2: Distribution of emotion classification classes in RAVDESS dataset (in %)

As we can see, the dataset is very balanced in terms of its audio files emotion distribution. Furthermore, the length of each audio file is between 3 and 5 seconds making this dataset very uniform from the data length view.

5.3 IEMOCAP

The Interactive Emotional Dyadic Motion Capture (IEMOCAP) database [Bus+08] was created at the SAIL lab at the University of South California. It is an acting, multimodal, and multispeaker database. The dataset comprises almost 12 hours of audiovisual data, encompassing video footage, spoken language, facial motion capture, and text transcriptions. The program comprises of one-on-one sessions in which actors engage in improvisations or scripted scenarios that are carefully chosen to evoke emotional expressions. The IEMOCAP database is annotated by numerous annotators using categorical labels, including 9 emotion classification categories (angry, excited, fear, sad, surprised, frustrated, happy, disappointed and neutral), as well as dimensional labels such as valence, activation, and dominance. The dataset consists of 4490 dialogues. Audio files, same as with RAVDESS, are recorded with 48 kHz sample rate and 16-bit bitrate. Their length is also comparable to the RAVDESS with audio file lengths between 3 and 5 seconds.

Distribution of the above-mentioned emotions can be seen in table 5.3.

Emotion	Representation
Happy	7.9
Angry	14.65
Sad	14.4
Neutral	22.69
Excited	13.83
Frustrated	24.56
Fear	0.53
Surprised	1.42
Disappointed	0.02

Table 5.3: Distribution of emotion classification classes in IEMOCAP dataset (in %)

There are a total of 7527 sentences annotated with one of the emotions mentioned above. Since emotion categories *Happy, Fear, Disapointed* have a small representation in the dataset, shown in Table 5.3, I chose not to include them resulting in total of 6785 annotated sentences with a new emotion class distribution, shown in Table 5.4.

Emotion	Representation
Angry	16.25
Sad	15.97
Neutral	25.17
Excited	15.34
Frustrated	27.25

Table 5.4: Distribution of emotion classification classes in IEMOCAP dataset without *Happy, Fear and Disapointed* (in %)

Implemented Models and Methods — 6

The goal of this thesis is to implement a system for multimodal emotion recognition. To accomplish that, 3 audio emotion recognition models and 2 textual emotion recognition models have been implemented. The multimodal models combine the textual emotion recognition models with the audio emotion recognition models by taking the audio feature vector from the trained audio emotion recognition model and concatenating it with the textual representation of the sentence taken from the textual emotion recognition model. Furthermore, 2 audio feature extraction and 2 text feature extraction methods have been implemented.

6.1 Audio feature extraction

Before the training of the model itself, feature extraction from the audio signal has to be performed. There are two main feature extraction methods implemented for this system:

- **MFCC only** - This feature extraction method uses only the Mel Frequency Cepstral Coefficients. For this purpose, the **librosa** library is used. There are two variations of this method. One that takes the average of every coefficient over a time series and one that includes the time series in the output feature vector. So for example, if the audio signal is 3 seconds long, the number of MFCCs would be set to 50 and the sample rate would be 22050Hz, the time series variation of this method would output a feature vector of size (50,129). The second dimension comes from the windowing of the audio signal. By default the window size is set to 2048 and the overlap to 512. So the second dimension can be calculated with equation 6.1.

$$mfcc_samples = (seconds) * (samplerate)/(overlaplength) \tag{6.1}$$

  The variation that takes the average over time series would output a 1D vector of size (50) if the number of MFCCs was set to 50.

- **Collective features** - This method, similarly to the previous one, has two variations. One that works with the time series dimension and one that takes maxima, minima, average, and standard deviations over the time of the audio signal. This method is called collective features because there are several feature extraction methods that collectively create the feature vector. The vectors or scalars from these feature extraction methods are then concatenated and used as the feature vector, the methods used in the collective features function are:
  - *Spectral centroid*
  - *Spectral flatness*
  - *MFCC*
  - *Chromagaram*
  - *Zero Crossing rate*
  - *Spectral Contrast*
  - *Mel spectrogram*

  Similarly to the previous method the variation that takes the average, mean, maximum and standard deviations over the time series outputs a 1D vector and the one that works with time series outputs a 2D vector.

6.2 Text feature extraction

Similarly to the audio signal, the text data also has to be transformed into feature vectors. There are two text feature extraction methods used by the multimodal and text emotion recognition models.

- **w2v** - This method takes pretrained word vectors from the Gigaword 5th Edition Corpus and uses them as feature vectors for words in the above-mentioned datasets. During the loading of the dataset, each word gets a corresponding vector that is searched in the w2v pretrained file. Simply put, each word gets a vector of size 300 which is the word feature vector. These vectors are then concatenated and this creates the representation of the whole sentence.
- **BERT** - This method creates contextualized feature vectors for each word depending on their surroundings. Text is firstly tokenized through the BERT tokenizer and then fed into the BERT model. The word embeddings are retrieved from the last hidden state of the BERT model.

6.3 Noise reduction

Noise can be described as unwanted additional data. Noise can be caused by a faulty microphone, rough wind conditions when recording, bad quality camera and so on. Noise does not add any beneficial information and usually worsens the overall performance of the classifier.

As stated in the ECF dataset analysis in section 5.1, the ECF dataset contains a lot of noise from environmental sounds (for example the kitchenware in cafeteria scenes) as well as the laughter from the studio audience. There are two noise reduction methods used to separate the main speaker from the background noise.

6.3.1 FT2D

This method is based on the 2D Fourier transformation (2DFT). It takes advantage of the periodicity that often occurs in audio signals. The audio signal is represented by its spectrogram as an image. The 2DFT breaks down the image into a combination of 2D sinusoids that are weighted and have their phase changed[SPP17].

6.3.2 REPET-SIM

REPET-SIM is an extension of the REpeating Pattern Extraction Technique (REPET) that employs a similarity matrix to distinguish the recurring background from the non-recurring foreground in a combination. The method presupposes that the background, usually the musical accompaniment, is thick and subordinate, while the front, often the singing voice, is sparse and diverse. While this assumption is frequently valid for the background music and prominent vocals in musical blends, it also commonly applies to the background noise and prominent speech in noisy blends [RP12].

6.4 Audio emotion recognition models

Audio models use only the audio signal to determine the correct emotion that's being expressed in the signal.

6.4.1 CNN1D

This model is based on Convolutional Neural Networks and has only one-dimensional kernels used for its convolution operation, there are a total of 4 convolution layers with a maximum over-time pooling between each layer. Since the convolution is only one dimensional, first and second layer outputs 256 kernels, the third layer reduces this number to 128, and the last one to 64. After these convolution layers, there

are 3 linear layers. The first one takes the output from the last convolution layer and has an output size of 512. The second linear layer is also the one that outputs the audio feature vector for which are these models trained. The third and last linear layer serves as a classifier for the emotion recognition task. The activation function used in this model is *Relu*. The architecture of this model is shown in Figure 6.1

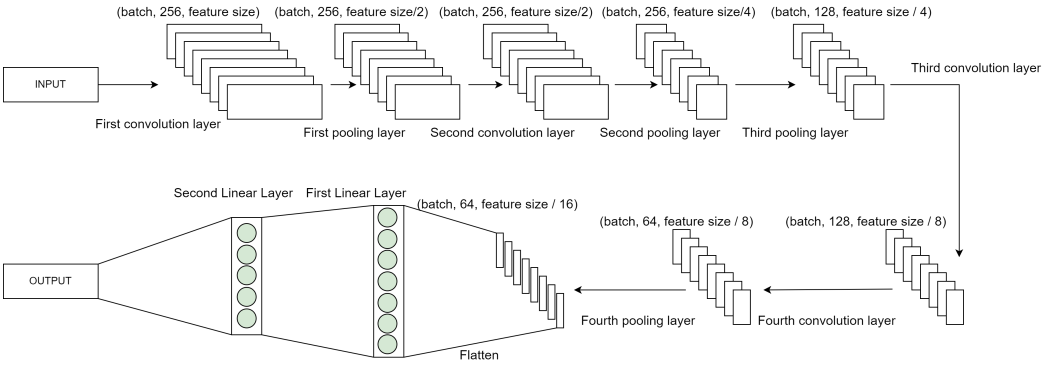

Figure 6.1: Architecture of the CNN1D audio emotion recognition model

6.4.2 CNN2D

This model resembles more conventional convolutional models. There are two convolution layers first that outputs 2 kernels and second that outputs 4. Maximum over-time pooling is deployed before the first linear layer. This layer also serves as an output of the model for the multimodal task, where the output of this layer is the audio feature vector for the multimodal model. The second and also the last linear layer of this model is used for classification in the audio emotion recognition task. The activation function used in this model is *Relu*.

6.4.3 MLP

This model is the least complicated one and consists of only 3 linear layers. The output of the first linear layer has a size of 512. The output of the second layer is used as the audio feature vector in the multimodal task. The last layer serves as the classifier for the audio emotion recognition task. The *Relu* activation function is used between the layers.

6.5 Textual emotion recognition models

There are two models implemented for the emotion recognition task from text. The first is the LSTM model that utilizes the *Attention* mechanism after the LSTM layer, and the second model is pretrained BERT model.

6.5.1 LSTM

This model consists of two LSTM layers and one linear layer for classification purposes. Both of the LSTM layers are bidirectional and the output from the first one is then passed through the *Attention* layer.

- **Attention** - Attention mechanism is explained in the section 3.3.1. There are two linear layers. The output of the first layer goes through the *tanh* function before being used as input into the second layer. The output of the second layer is then multiplied with the input into the Attention layer and the result is returned.

The output of the Attention layer is then passed into the second bidirectional LSTM layer and lastly, the output goes through the linear layer for classification.

6.5.2 BERT

The second text emotion recognition model is the pretrained transformer-based model *BERT*. For the purpose of this thesis, a pretrained *bert-base-uncased* model is finetuned on the emotion recognition task with the IEMOCAP and ECF datasets. BERT has undergone pretraining on a substantial collection of English data using a self-supervised approach. This indicates that the model was trained solely on raw texts, without any human annotation. The training procedure involves an automated method to produce inputs and labels from the texts. To be more explicit, it was pretrained with two learning objectives:

- **Masked Language Modeling** - Masked language modeling (MLM) involves randomly masking 15% of the words in a sentence and then running the entire masked text through a model. The model's task is to predict the masked words. Contrary to conventional recurrent neural networks (RNNs) that typically process words sequentially, or autoregressive models such as GPT that hide future tokens internally, this approach is different. The model is able to acquire a bidirectional representation of the sentence through this process.

- **Next Sentence Prediction** - The model combines two masked words as inputs during pretraining. Occasionally, they align with adjacent sentences in the original text, although this is not always the case. The model must then predict whether the two sentences were consecutive or not.

This way, the model acquires an internal representation of the English language, which may subsequently be employed to extract valuable characteristics for downstream tasks. For instance, if you possess a collection of labeled sentences, you can train a conventional classifier by utilizing the features generated by the BERT model

as inputs. After the BERT model, there is one linear layer used as the classifier for our specific task.

6.6 Multimodal emotion recognition models

To make use of the second modality (the audio modality) every audio emotion recognition model has one linear layer that has the same output size and is returned from the *forward* function of each model together with the classification vector. Each of the above-named audio models outputs a feature vector of size 50. This audio feature vector is then used in the text emotion recognition models to provide further information about the sentence the model is currently processing and improve the overall accuracy and F1 score of the model as can be seen in Figure 6.2.

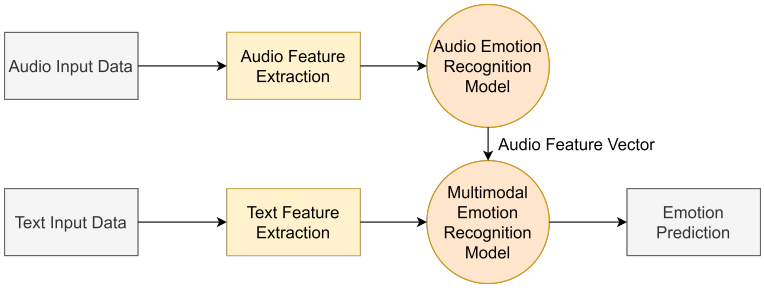

Figure 6.2: Multimodal emotion recognition pipeline

6.6.1 LSTM multimodal

The multimodal LSTM model is very similar to the textual one with the change to the input size of the second LSTM layer. The output from the Attention layer is concatenated with the audio feature vector provided by one of the audio emotion recognition models and used as input into the second LSTM layer of the model. The last linear layer stays the same.

6.6.2 BERT multimodal

The Multimodal BERT model is also very similar to the text emotion recognition model with the change in the input size of the last linear layer that serves as the classifier. When we want to use the model with the audio signals, the BERT part of the model processes the text, and the feature vector from the audio emotion recognition model is then concatenated with the output vector from the BERT model.

Experiments 7

The experiments are divided into 3 sections: emotion recognition from audio, text, and lastly emotion recognition from multimodal data. In each of these sections, the implemented models are tested on the above-mentioned datasets with relevant feature extraction methods based on the utilized data. All the results in the tables below are listed in %.

7.1 Evaluation methods

To determine if the model behaves better or worse we have to have a method to evaluate its performance. There are several evaluation methods and to evaluate models implemented for this task, **Accuracy** and **F1 score** are used.

7.1.1 Accuracy

This metric takes all correctly identified cases and divides them by the size of the dataset. It is a suitable evaluation method when working with evenly distributed datasets, but if the dataset is imbalanced it can be quite misleading. For example, if one classification class would make up 60% of the dataset, the classifier only needs to classify every input as this class and will achieve a minimal accuracy of 60%. For this reason, a second evaluation metric is used.

7.1.2 F1 score

- This metric is a harmonic mean of Precision and Recall. Both of these metrics use:

- **True Positives** - TP are instances where the classifier identifies the right class as positive.

- **True Negatives** - TN are instances where the classifier identifies wrong classes as negative.

- **False Positives** - FP are instances where the classifier identifies wrong class as positive.
- **False Negatives** - FN are instances where the classifier identifies right class as negative.

For example, if we would have 3 classification classes, the true value would be [0, 0, 1] and our classifier would classify [0, 0, 1] it would count as 1 True positive and 2 True negatives. If the true value would be [1, 0, 0] and our classifier would classify [0, 1, 0], it would count as 1 False negative, 1 False positive, and 1 True negative.

With these instances, we then calculate the Precision, Recall, and subsequently the F1 score.

7.1.2.1 Precision

Precision is a measure of the correctly identified positive cases from all the predicted positive cases. It can be calculated with equation 7.1.

$$Precision = \frac{TP}{TP + FP} \tag{7.1}$$

7.1.2.2 Recall

Recall is the measure of the correctly identified positive cases from all the actual positive cases. It can be calculated with equation 7.2.

$$Precision = \frac{TP}{TP + FN} \tag{7.2}$$

F1 score can be calculated with equation 7.3.

$$2 * \frac{Precision * Recall}{Precision + Recall} \tag{7.3}$$

The F1 score is a useful evaluation metric when the classification classes in the dataset are unbalanced.

7.2 Audio emotion recognition

There are 3 audio emotion recognition models (CNN1D, CNN2D, MLP) described in section 6.4. These models utilize 2 feature extraction methods (*mfcc_only*, *collective_features*) described in section 6.1. The noise reduction methods mentioned in section 6.3 are deployed on the **ECF** dataset. Since the datasets are very different and their results vary a lot, the results are grouped by the used dataset.

7.2.1 ECF

The results on the **ECF** dataset are in Table 7.1. The first column refers to the noise reduction methods, the second column to the audio recognition models, and the third column to the audio feature extraction methods.

Noise reduction method	Model	Feature extraction method	Train acc	Train F1	Test acc	Test F1
	CNN1D	mfcc_only	76.2	54.8	34.6	17.3
	CNN1D	collective_features	70.5	45.6	40.8	**18.7**
	CNN2D	mfcc_only	47.5	16.7	**45.8**	15.5
	CNN2D	collective_features	46.8	16.1	45.6	14.8
	MLP	mfcc_only	50.1	24.7	42.1	15.2
	MLP	collective_features	68.3	39.6	41.6	17.3
FT_2D	CNN1D	mfcc_only	82.5	55.8	32.9	16.6
	CNN1D	collective_features	60.4	34.1	40.5	18.4
	CNN2D	mfcc_only	41.3	8.4	43.6	8.6
	CNN2D	collective_features	45.1	12.9	44.4	11.2
	MLP	mfcc_only	49.9	23.7	44	13
	MLP	collective_features	66.2	34.5	37	17.9
REPETSIM	CNN1D	mfcc_only	78.1	52.8	33.1	16.2
	CNN1D	collective_features	14.6	3.6	11.6	3.2
	CNN2D	mfcc_only	46.4	13.3	44	13.7
	CNN2D	collective_features	12.5	3.1	11.1	3.1
	MLP	mfcc_only	49.4	19.9	39.9	17.2
	MLP	collective_features	14.1	3.5	11.8	3.6

Table 7.1: Comparison of different model configurations and their results on the **ECF** dataset

The best-performing configuration was the **CNN1D** audio recognition model with the **collective_features** audio feature extraction method with average F1 score **18.7%** and average accuracy **45.6%**. The noise reduction methods overall worsened the performance of all the configurations especially the **REPETSIM** method with **collective_features** extraction method.

7.2.2 RAVDESS

There are several configurations for the **RAVDESS** dataset as well, with the exception of noise reduction methods. Since the dataset was recorded in a studio with little to no noise, there is no reason to use any of the implemented noise reduction methods. The results are in Table 7.2.

Model	Feature extraction method	Train acc	Train F1	Test acc	Test F1
CNN1D	mfcc_only	83.5	74.9	**58.7**	51.2
CNN1D	collective_features	68.3	63.2	53.2	47.4
CNN2D	mfcc_only	36.1	33	25.4	19.4
CNN2D	collective_features	25	17.2	16.5	11.3
MLP	mfcc_only	56.3	48.7	41.5	38.5
MLP	collective_features	86.8	84.4	55.3	**51.6**

Table 7.2: Comparison of different model configurations and their results on the **RAVDESS** dataset

CNN1D and **MLP** audio models were quite similar from the performance standpoint and both worked much better than the **CNN2D** model. **MLP** worked better with the **collective_features** extraction method and **CNN1D** worked better with the **mfcc_only** extraction method. Both of the models had an average F1 score around **51%**. **CNN1D** reached better average accuracy of **58.7%**.

7.2.3 IEMOCAP

IEMOCAP dataset has a similar quality of audio recording as the **RAVDESS** dataset. The columns in the result table also refer to the same attributes of the used configurations. The results are in Table 7.3

Model	Feature extraction method	Train acc	Train F1	Test acc	Test F1
CNN1D	mfcc_only	97.3	97.2	45.2	45.1
CNN1D	collective_features	89.6	89.4	**51.4**	**51.5**
CNN2D	mfcc_only	42.8	38.2	42.6	38
CNN2D	collective_features	44.2	38.3	43.6	38.1
MLP	mfcc_only	58.6	56.3	44.9	44.3
MLP	collective_features	80.4	81	46.2	45.3

Table 7.3: Comparison of different model configurations and their results on the **IEMOCAP** dataset

The best-performing configuration for the **IEMOCAP** dataset proved to be **CNN1D** audio model with the **collective_features** audio feature extraction method achieving **51.5%** average F1 score and **51.4%** average accuracy.

7.3 Text emotion recognition

There are two text emotion recognition models implemented, that are described in the section 6.5. These models use two text feature extraction methods described in section 6.2. **RAVDESS** dataset does not have text transcription, so text emotion recognition models are trained on **ECF** and **IEMOCAP** datasets. The **BERT** model uses its own text feature extraction.

7.3.1 ECF

Results on the **ECF** dataset are in Table 7.4. The first column refers to the text emotion recognition model described in section 6.5 and the second column to the text feature extraction method described in section 6.2. Since the **BERT** model uses its own text feature extraction, the column is left blank.

Model	Feature extraction method	Train acc	Train F1	Test acc	Test F1
LSTM	w2v	64.2	40.8	56.8	33.8
LSTM	BERT_embeddings	95.5	94	56	40.9
BERT		81.2	71.2	**62.3**	**43.8**

Table 7.4: Comparison of different model configurations and their results on the **ECF** dataset

The pretrained **BERT** model achieved the best average F1 score of **62.3%** and average accuracy of **43.8%**. The **LSTM** model achieved worse results but worked better with the **BERT_embedding** feature extraction.

7.3.2 IEMOCAP

The result Table 7.5 of the **IEMOCAP** dataset has the same columns as the **ECF** dataset.

Model	Feature extraction method	Train acc	Train F1	Test acc	Test F1
LSTM	w2v	55.3	52.4	42.3	40.2
LSTM	BERT_embeddings	92.1	91.9	54.7	55
BERT		93.7	92.7	**60.5**	**55.8**

Table 7.5: Comparison of different model configurations and their results on the **IEMOCAP** dataset

The pretrained **BERT** model achieved better results again, with the average F1 score of **55.8%** and average accuracy of **60.5%**. The **LSTM** model achieved lower scores and worked better with the **BERT_embedding** feature extraction method.

7.4 Multimodal emotion recognition

Multimodal emotion recognition models combine the audio and text models described in sections 6.4 and 6.5. They are similar to the text emotion recognition models, but take additional information in the form of an audio feature vector. There are two methods of using the audio feature vectors. The first one uses the output of the trained audio emotion recognition model described in 6.4. The second one bypasses this extra step and uses the feature vector from one of the audio feature extraction methods (**mfcc_only** and **collective_features**) described in section 6.1.

7.4.1 ECF

The results of the multimodal models with audio feature vectors from the audio emotion recognition models are in Table 7.6. The results of multimodal models where the model uses one of the audio feature extraction methods as an input audio feature vector for its learning are in table 7.7. The multimodal **BERT** model, the same as the text emotion recognition one, has its own text feature extraction, so the column for the text feature extraction method is left blank.

Model	Audio model	Text feature extraction method	Audio feature extraction method	Train acc	Train F1	Test acc	Test F1
LSTM	CNN1D	w2v	mfcc_only	85.1	61.6	39	24.9
LSTM	CNN1D	w2v	collective_features	78.3	53.8	47.6	28.4
LSTM	CNN1D	BERT_embeddings	mfcc_only	98.7	98.1	43.3	32.7
LSTM	CNN1D	BERT_embeddings	collective_features	97.9	94.6	52.9	38.8
LSTM	CNN2D	w2v	mfcc_only	63.3	41.2	56.7	34.2
LSTM	CNN2D	w2v	collective_features	65.8	43.2	56.9	33.8
LSTM	CNN2D	BERT_embeddings	mfcc_only	96.1	94.6	54.8	40
LSTM	CNN2D	BERT_embeddings	collective_features	96	94.3	53.2	38.9
LSTM	MLP	w2v	mfcc_only	67.6	43.1	54.2	31.3
LSTM	MLP	w2v	collective_features	79.1	57	48.9	28.9
LSTM	MLP	BERT_embeddings	mfcc_only	96	91.5	53.9	32.7
LSTM	MLP	BERT_embeddings	collective_features	98	96.6	51.1	36.9
BERT	CNN1D		mfcc_only	82.8	69.7	**62.4**	44.5
BERT	CNN1D		collective_features	73.4	51.8	**62.4**	**44.8**
BERT	CNN2D		mfcc_only	79.6	65.2	61.7	45
BERT	CNN2D		collective_features	70.3	51.4	60.6	41
BERT	MLP		mfcc_only	59.3	47.8	55.8	37.5
BERT	MLP		collective_features	59.3	32.7	53.6	35.4

Table 7.6: Comparison of different model configurations and their results on the **ECF** dataset

The multimodal **LSTM** model does not have better performance than the textual **LSTM** model on the **ECF** dataset. **CNN2D** feature extraction model with

mfcc_only audio features and **BERT_embeddings** seem to be performing the best among other audio feature extraction models, but the overall performance is not better than the purely textual one. The multimodal **BERT** model achieves better results than the multimodal **LSTM** model. The best-performing configuration is with **CNN1D** audio model with **collective_features** feature extraction method. This configuration achieves an average F1 score of **44.8%** and an average accuracy of **62.4%**.

Model	Text feature extraction method	Audio feature extraction method	Train acc	Train F1	Test acc	Test F1
LSTM	w2v	mfcc_only	66.7	41.9	53.6	29.9
LSTM	w2v	collective_features	67.2	41.3	51.2	26
LSTM	BERT_embeddings	mfcc_only	92.5	86.2	55.2	**38.9**
LSTM	BERT_embeddings	collective_features	91.4	76	**55.3**	38
BERT		mfcc_only	15.6	8.9	19	11.8
BERT		collective_features	18.7	10.2	20.1	11.4

Table 7.7: Comparison of different model configurations without the audio model feature extraction and their results on the **ECF** dataset

The performance of the multimodal models with audio features from the feature extraction methods (without the audio mode) overall worsened. The multimodal **LSTM** model works with the audio feature extraction method feature vectors better than the multimodal **BERT**, but the performance with the audio models is better.

7.4.2 IEMOCAP

The results of the multimodal models with audio feature vectors from the audio emotion recognition models are in Table 7.8. The results of multimodal models where the model uses the additional audio feature vector from one of the audio feature extraction methods for its learning are in table 7.9. The text feature column for the **BERT** model is left blank since the model uses its own feature extraction.

Model	Audio model	Text feature extraction method	Audio feature extraction method	Train acc	Train F1	Test acc	Test F1
LSTM	CNN1D	w2v	mfcc_only	90.4	90.4	**79.6**	**79.5**
LSTM	CNN1D	w2v	collective_features	82.4	82.3	74.6	74.7
LSTM	CNN1D	BERT_embeddings	mfcc_only	98.7	98.9	78.6	79.2
LSTM	CNN1D	BERT_embeddings	collective_features	98.8	98.2	74.6	75.6
LSTM	CNN2D	w2v	mfcc_only	75.3	76.2	52.1	52.4
LSTM	CNN2D	w2v	collective_features	63.7	62	50.8	50.3
LSTM	CNN2D	BERT_embeddings	mfcc_only	96.5	96.5	60.4	61.3
LSTM	CNN2D	BERT_embeddings	collective_features	96	96.1	60.4	61
LSTM	MLP	w2v	mfcc_only	69.4	70.2	54.5	55.4
LSTM	MLP	w2v	collective_features	84.5	84.2	68.7	68.9
LSTM	MLP	BERT_embeddings	mfcc_only	95.6	95.8	62.4	63.2
LSTM	MLP	BERT_embeddings	collective_features	96.8	87.2	64.9	65.9
BERT	CNN1D		mfcc_only	75.9	76	66.9	61.6
BERT	CNN1D		collective_features	81.2	77.6	61.6	58.1
BERT	CNN2D		mfcc_only	71.8	70.7	55.8	52.2
BERT	CNN2D		collective_features	65.6	54.7	41.7	35.5
BERT	MLP		mfcc_only	68.7	64.6	54.4	51.1
BERT	MLP		collective_features	75	60.7	49.9	46.5

Table 7.8: Comparison of different model configurations and their results on the **IEMOCAP** dataset

The performance of the multimodal **LSTM** model on the **IEMOCAP** dataset is far better than the purely textual one. The highest average accuracy without the audio modality was **54.7%** and the average F1 score was **55.5%**. With the additional information provided by the audio feature vector from the audio emotion recognition model, the **LSTM** multimodal model achieves up to **79.5%** average F1 score and **79.5%** average accuracy. The performance of the multimodal **BERT** model on the dataset is again improved with the additional information from the audio emotion recognition models. The maximum accuracy of the text BERT model for the **IEMOCAP** dataset was **60%** average accuracy and average F1 score was **55%**. The multimodal BERT model with the **CNN1D** audio model with **mfcc_only** feature extraction achieves an average F1 score of **58.1%** and average accuracy of **66.9%**. Similarly to the **ECF** dataset, the performance on **IEMOCAP** dataset also wersened

Model	Text feature extraction method	Audio feature extraction method	Train acc	Train F1	Test acc	Test F1
LSTM	w2v	mfcc_only	68	68.4	54.6	54.4
LSTM	w2v	collective_features	72.2	71.8	52.9	53.3
LSTM	BERT_embeddings	mfcc_only	94.9	95	59.2	60.7
LSTM	BERT_embeddings	collective_features	96.7	96.7	**62.1**	**63.3**
BERT		mfcc_only	31.2	22.6	27.1	21.2
BERT		collective_features	25	16.7	25.3	18.5

Table 7.9: Comparison of different model configurations and their results on the **IEMOCAP** dataset

without the feature vectors from the audio models. The multimodal **LSTM** again

worked better than the multimodal **BERT** but both showed worse results when the feature vector is taken from the audio feature extraction methods directly.

7.5 Discussion

Emotion recognition from only audio signals proved to be more difficult than from text data. Depending on the feature extraction method the best overall results climbed up to **18%** F1 test score and around **40%** test accuracy on the **ECF** dataset. The best performing was the **CNN1D** model that uses the 1D feature vector from the audio signal, meaning that the time-specific information might not be very important when it comes to emotion recognition tasks. There were a few performance differences between the **mfcc_only** and **collective_feature_extraction** methods, but the collective features worked slightly better (around **1%** to **2%** F1 and accuracy score). The noise reduction for the **ECF** dataset overall worsened the model performance. That might be due to the fact that we lose a lot of information about pitch, which is an important part of the collective feature extraction method.

As said above, the text emotion recognition showed better results. The **LSTM** model worked better with the **BERT** word embeddings and was able to achieve a maximum of **56%** test accuracy and **40%** F1 score. Contextualized word embeddings worked better for this particular task, which might be due to the fact that a lot of emotions are bound by the context of the sentence. The **BERT** pretrained model worked better for both of the datasets, but not by a large margin (around **2%** to **3%** for both accuracy and F1 score).

Multimodal emotion recognition had a lot of different configurations in which the classification could be performed. Since there are 3 audio models, that work with 2 audio feature extraction methods, and there are 2 multimodal models that also work with 2 text feature extraction methods on 2 possible datasets, the overall number of configurations climbed up to 36 (**BERT** model does not work with Word2Vec text features). The **LSTM** multimodal model had worse performance on the **ECF** dataset than the purely textual one. Only the **BERT** model achieved a slightly better F1 score (around **1%** better) when used multimodally with **CNN1D** audio model and collective audio feature extraction. However, the **IEMOCAP** dataset benefited from the additional audio features, since both the **BERT** and **LSTM** multimodal models achieved better results than the purely textual ones. The **BERT** model performed better with the **CNN1D** audio model and both of the audio feature extraction methods. The highest test accuracy was **66%** and the F1 score was about **61%** (around **6%** more than the purely textual **BERT** model). Nearly all of the configurations of the multimodal **LSTM** model performed better than the purely textual **LSTM** model with the exception of:

- Word2Vec text features | CNN2D audio model | mfcc only audio features
- Word2Vec text features | CNN2D audio model | collective audio features
- Word2Vec text features | MLP audio model | mfcc only audio features

The best-performing configurations even surpassed the **BERT** multimodal model. The best working configurations were with the **CNN1D** audio model and **mfcc_only** audio feature extraction method. The best configuration of the multimodal **LSTM** model achieved **79%** test F1 score and **79%** test accuracy. Around **13%** better accuracy and **18%** better F1 score than the BERT model.

Furthermore, the configurations with audio feature vectors from the audio models worked better than the feature vectors from audio feature extraction methods. The multimodal **LSTM** model was able to work with these feature vectors, but its accuracy and F1 score lowered. The multimodal **BERT** model worked significantly worse. The extra step of training audio emotion recognition models to get better audio feature vectors from them proved to be beneficial for both the **BERT** and **LSTM** multimodal models. The best-performing configurations on each dataset and their results are in Table 7.10.

Model	Text feature extraction method	Audio feature extraction method	Train acc	Train F1	Test acc	Test F1
ECF	audio	CNN1D + collective_features	70.5	45.6	40.8	18.7
	text	BERT	81.2	71.2	62.3	43.8
	audio + text	BERT + CNN1D + collective_features	73.4	51.8	**62.4**	**44.8**
RAVDESS	audio	MLP + collective_features	86.8	84.4	**55.3**	**51.6**
IEMOCAP	audio	CNN1D + collective_features	89.6	89.4	51.4	51.5
	text	BERT	93.7	92.7	60.5	55.8
	audio + text	LSTM + w2v + CNN1D + mfcc_only	90.4	90.4	**79.6**	**79.5**

Table 7.10: Best performing configurations for each modality and on each dataset

7.5.1 Possible extensions

Since the audio emotion recognition models performed better on both **RAVDESS** and **IEMOCAP** datasets, further research into the noise reduction methods might be beneficial. Real audio from normal human conversation will have some noise levels so implementing a noise reduction system where the pitch information remains and the background noise is reduced might help the overall performance.

Another possible extension would be to introduce additional modalities to the system. The audio modality improved the performance so other modalities like video or facial motion information might also improve the performance. The **IEMOCAP** dataset offers these modalities.

Conclusion 8

The aim of this thesis was to analyze relevant feature extraction methods for textual and audio data and implement a system for multimodal emotion recognition.

The analyzed audio feature extraction methods focus on the time domain features as well as the spectral features of an audio signal. Based on the experiments the spectral features proved to be more important since the **CNN2D** audio emotion recognition model did not outperform the **CNN1D** or **MLP** audio emotion recognition models, that use feature vectors with time domain information reduced, since the feature vectors used for training of these models take maxima, minima and average values over the time domain. The maximum average F1 score for audio emotion recognition was achieved by the **CNN1D** model reaching up to **18.7%** on the **ECF** dataset and **51.5%** on the **IEMOCAP** dataset. The maximum average F1 score on the **RAVDESS** dataset was achieved by the **MLP** model with F1 score of **51.6%**.

The amount of noise in the dataset also proved to be very important, because the models achieved better results on **RAVDESS** and **IEMOCAP** datasets. These datasets were recorded in a studio environment with little to no additional noise. The **ECF** dataset consists of snippets taken from the sitcom Friends and contains environmental sounds. For this reason **FT2D** and **REPETSIM** noise reduction methods were deployed on the **ECF** dataset, but they only worsened the overall performance of the audio emotion recognition models. This might be due to the fact, that by using these methods the pitch information was partly lost and since both of the audio feature extraction methods rely to some extent on the pitch information, the overall performance worsened.

Contextualized word embeddings from the pretrained **BERT** model proved to work better for this particular task. But even with the **BERT** word embeddings the **LSTM** model was outperformed on the text emotion recognition tasks by the pretrained **BERT** model, where the **BERT** model achieved an average F1 score of **43.8%** on the **ECF** dataset and **55.8%** on the **IEMOCAP** dataset. The **LSTM** model with **BERT** word embeddings achieved **40.9%** average F1 score on the **ECF** dataset and **55%** on the **IEMOCAP** dataset.

The additional information added by the audio feature vector proved to be beneficial for emotion recognition since both the multimodal **BERT** and **LSTM** models achieved better results with the audio feature vectors. Especially the **LSTM** model improved by over **15%** on accuracy and over **20%** on F1 score on the **IEMOCAP** dataset. Furthermore, the audio feature vectors taken from the audio models worked better than the audio feature vectors taken from the audio feature extraction methods directly. With the audio feature vectors taken directly from the audio feature extraction methods the multimodal **BERT** model achieved an average F1 score of **11%** on the **ECF** dataset and **21%** on the **IEMOCAP** dataset, which is significantly less than with the audio feature vectors taken from the trained audio emotion recognition models. The multimodal **LSTM** model was able to achieve better results with the audio vectors taken directly from the audio feature extraction methods than the multimodal **BERT** model. The results on the **ECF** dataset were comparable with the **LSTM** model that used audio feature vectors from trained audio emotion recognition models but had around **17%** lower F1 score and accuracy on the **IEMOCAP** dataset.

List of Abbreviations

SPL - Sound Pressure Level
ML - Machine Learning
FFT - Fast Fourier Transform
GTCC - Gamma Tone Cepstrum Coefficient
MFCC - Mel Frequency Cepstrum Coefficient
DCT - Discrete Cosine Transform
DFT - Discrete Fourier Transform
ZCR - Zero Crossing Rate
SC - Spectral Centroid
OSC - Octave-based Spectral Contrast
NLP - Natural Language Processing
BOW - Bag of Words
TF-IDF - Term Frequency - Inverse Document Frequency
CBOW - Continuous Bag of Words
BERT - Bidirectional Encoder Representations from Transformer
CNN - Convolutional Neural Network
LSTM - Long Short-Term Memory Network
MLP - Multilayer Perceptron
MLM - Masked Language Model
GPT - Generative Pre-training Transformer
ANN - Artificial Neural Network
CEC - Constant Error Carousel
LSNN - Long Short-Term Memory Spiking Neural Network
ECF - Emotion Cause in Friends
RAVDESS - Ryerson Audio-Visual Database of Emotional Speech and Song
IEMOCAP - Interactive Emotional Dyadic Motion Capture
TP - True Positive
TN - True Negative
FP - False Positive
FN - False Negative

User Manual B

Link to the github repository: git repository
Before installing the requirements, there are two prerequisites. ffmpeg framework and sox sond processing tool is required for this project to run. Installation is different for Linux and Windows operation systems:

- **Windows**
  - `ffmpeg` - Detailed instructions to install `ffmpeg` on Windows can be found here: ffmpeg installation guide. You have to download the `ffmpeg` from their official webpage, extract it, move it to your root disk folder and rename it to *ffmpeg*. After that, you need to add the `bin` folder of this `ffmpeg` folder into you PATH.
  - `sox` - This processing tool can be downloaded from their forge page: sox download page. After the installation is complete, you need to add the path of the installed target folder to your PATH variable.
- **Linux**
  - `ffmpeg` - This framework can be installed on Linux with the following command

```
sudo apt install ffmpeg
```

  - `sox` - This processing tool can be installed on Linux with the following command

```
sudo apt-get install sox libsox-dev
```

The main entry point of the code is the `main.py` script. The first input argument determines which modality should be used. There are 7 input arguments. Usage of these arguments depends on the chosen modality

- `-modality` - determines which modality should be used. Is required for the script to start. There are 3 possible modalities: **audio**, **text**, **multimodal**.

- `-text_model` - determines which multimodal or textual model is supposed to be used. Either **LSTM** or **BERT**. This argument is not required if **audio** modality was chosen, but is required if **text** or **multimodal** modality was chosen in the `-modality` argument.

- `-text_feature_extraction` - determines which text feature extraction method is supposed to be used. Either **w2v** or **bert**. This argument is not required if **audio** modality was chosen, but is required if **text** or **multimodal** modality was chosen in the `-modality` argument.

- `-audio_model` - determines which audio emotion recognition model is supposed to be used. Either **CNN1D** or **CNN2D** or **MLP**. This argument is not required if **text** modality was chosen or **multimodal** modality with argument `-use_audio_model` set to **false** was chosen , but is required if **audio** modality or **multimodal** modality with argument `-use_audio_model` set to **true** was chosen in the `-modality` argument.

- `-audio_feature_extraction` - determines which audio feature extraction method is supposed to be used. Either **mfcc_only** or **collective_features**. This argument is not required if **text** , but is required if **audio** modality or **multimodal** modality was chosen in the `-modality` argument.

- `-dataset` - determines which dataset is supposed to be used. There are 3 datasets for audio emotion recognition: **ECF**, **RAVDESS** and **IEMOCAP**. When **RAVDESS** is chosen, only the audio model learning will be deployed, because it does not have text transcriptions. When **ECF** or **IEMOCAP** dataset is chosen, the whole learning process is deployed. The **ECF** dataset can be also run with noise reduction method **FT2D** or **REPETSIM**. If you want use one of these methods, the input parameter should be **ECF_FT2D** for **FT2D** noise reduction method and **ECF_REPETSIM** for the **REPETSIM** noise reduction method. This argument is always required.

- `-use_audio_model` - determines if the audio feature vectors should be taken from the indicated audio emotion recognition model from parameter `-audio_model` (if set to **true**) or if the audio feature vectors should be taken from the indicated audio feature extraction method from the parameter `-audio_feature_extraction` (if set to **false**).

So for example if the system should be used multimodally with **LSTM** multimodal model, **w2v** text feature method, **CNN1D** audio model, **mfcc_only** audio feature extraction on the **ECF** dataset with **FT2D** noise reduction method, the run command would look like this:

```
python main.py -modality multimodal -text_model LSTM
-text_feature_extraction w2v -audio_model CNN1D
-audio_feature_extraction mfcc_only -dataset ECF_FT2D
-use_audio_model true
```

If we want to run audio emotion recognition learning only with **MLP** audio model and **collective_features** audio feature extraction method on the **IEMOCAP** dataset, the run command would look like this:

```
python main.py -modality audio -audio_model MLP
-audio_feature_extraction collective_features
-dataset IEMOCAP
```

Python version used in this project is **python3.10**

There is also the `requirements.txt` file that includes all the necessary libraries to run the project. Libraries in use:

```
librosa==0.10.2
torch==2.3.0
torchmetrics==1.3.2
torcheval==0.0.7
nussl==1.1.9
transformers==4.40.1
scipy==1.11.3
numpy==1.23.1
```

In the **config** folder, there is `config.py` script with hyperparameters for each model, system path to w2v embedding file and with system paths to the **ECF**, **RAVDESS** and **IEMOCAP** datasets.

Bibliography

[AA22] ABDUL, Zrar Kh.; AL-TALABANI, Abdulbasit K. Mel Frequency Cepstral Coefficient and its Applications: A Review. *IEEE Access*. 2022, vol. 10, pp. 122136–122158. Available from DOI: 10.1109/ACCESS.2022.3223444.

[AM17] ALBELWI, Saleh; MAHMOOD, Ausif. A Framework for Designing the Architectures of Deep Convolutional Neural Networks. *Entropy*. 2017, vol. 19, no. 6. ISSN 1099-4300. Available from DOI: 10.3390/e19060242.

[ASS16] ALÍAS, Francesc; SOCORÓ, Joan Claudi; SEVILLANO, Xavier. A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds. *Applied Sciences*. 2016, vol. 6, no. 5. ISSN 2076-3417. Available from DOI: 10.3390/app6050143.

[BW05] BARTSCH, Mark; WAKEFIELD, Gregory. Audio Thumbnailing of Popular Music Using Chroma-Based Representations. *Multimedia, IEEE Transactions on*. 2005, vol. 7, pp. 96–104. Available from DOI: 10.1109/TMM.2004.840597.

[Bay+09] BAYER, Justin; WIERSTRA, Daan; TOGELIUS, Julian; SCHMIDHUBER, Jürgen. Evolving Memory Cell Structures for Sequence Learning. In: 2009, pp. 755–764. ISBN 978-3-642-04276-8. Available from DOI: 10.1007/978-3-642-04277-5_76.

[Bel+18] BELLEC, Guillaume; SALAJ, Darjan; SUBRAMONEY, Anand; LEGENSTEIN, Robert; MAASS, Wolfgang. *Long short-term memory and learning-to-learn in networks of spiking neurons*. 2018. Available from arXiv: 1803.09574 [cs.NE].

[Ben+07] BENZEGHIBA, M. et al. Automatic speech recognition and speech variability: A review. *Speech Communication*. 2007, vol. 49, no. 10, pp. 763–786. ISSN 0167-6393. Available from DOI: https://doi.org/10.1016/j.specom.2007.02.006. Intrinsic Speech Variations.

[Bis95] BISHOP, Christopher M. Neural networks for pattern recognition. In: 1995. Available also from: https://api.semanticscholar.org/CorpusID:60563397.

[Bus+08] BUSSO, Carlos et al. IEMOCAP: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation*. 2008, vol. 42, pp. 335–359. Available from DOI: 10.1007/s10579-008-9076-6.

[DZ11] DALAL, Mita; ZAVERI, Mukesh. Automatic Text Classification: A Technical Review. *International Journal of Computer Applications*. 2011, vol. 28. Available from DOI: 10.5120/3358-4633.

[Dev+18] DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

[Dev+19] DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019. Available from arXiv: 1810.04805 [cs.CL].

[DKB16] DHULIAWALA, Shehzaad; KANOJIA, Diptesh; BHATTACHARYYA, Pushpak. SlangNet: A WordNet like Resource for English Slang. In: 2016.

[DHD17] DZIADEK, Juliusz; HENRIKSSON, Aron; DUNELD, Martin. Improving Terminology Mapping in Clinical Text with Context-Sensitive Spelling Correction. *Studies in Health Technology and Informatics*. 2017, vol. 235, pp. 241–245. Available from DOI: 10.3233/978-1-61499-753-5-241.

[FKP94] FELDMAN, Hume A.; KAISER, Nick; PEACOCK, John A. Power-spectrum analysis of three-dimensional redshift surveys. *The Astrophysical Journal*. 1994, vol. 426, p. 23. ISSN 1538-4357. Available from DOI: 10.1086/174036.

[GS00] GERS, Felix; SCHMIDHUBER, Jurgen. Recurrent nets that time and count. In: 2000, vol. 3, 189–194 vol.3. ISBN 0-7695-0619-4. Available from DOI: 10.1109/IJCNN.2000.861302.

[GSC00] GERS, Felix; SCHMIDHUBER, Jürgen; CUMMINS, Fred. Learning to Forget: Continual Prediction with LSTM. *Neural computation*. 2000, vol. 12, pp. 2451–71. Available from DOI: 10.1162/089976600300015015.

[GK11] GHORAANI, Behnaz; KRISHNAN, Sridhar. *Time–Frequency Matrix Feature Extraction and Classification of Environmental Audio Signals*. Vol. 19. 2011. No. 7. Available from DOI: 10.1109/TASL.2011.2118753.

[GPD02] GOUYON, Fabien; PACHET, Francois; DELERUE, Olivier. On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds. 2002.

[Gre+17] GREFF, Klaus; SRIVASTAVA, Rupesh K.; KOUTNIK, Jan; STEUNEBRINK, Bas R.; SCHMIDHUBER, Jurgen. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*. 2017, vol. 28, no. 10, pp. 2222–2232. ISSN 2162-2388. Available from DOI: 10.1109/tnnls.2016.2582924.

[Guo+15] GUO, Yanming et al. Deep learning for visual understanding: A review. *Neurocomputing*. 2015, vol. 187. Available from DOI: 10.1016/j.neucom.2015.09.116.

[HGG19] HAMZENEJADI, Sajad; GOKI, Seyed Amir Yousef Hosseini; GHAZVINI, Mahdieh. Extraction of Speech Pitch and Formant Frequencies using Discrete Wavelet Transform. In: *2019 7th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. 2019, pp. 1–5. Available from DOI: 10.1109/CFIS.2019.8692150.

[HS96] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. LSTM can solve hard long time lag problems. In: 1996, pp. 473–479.

[HS97] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. *Neural Computation*. 1997, vol. 9, no. 8, pp. 1735–1780. ISSN 0899-7667. Available from DOI: 10.1162/neco.1997.9.8.1735.

[HK07] HOSSEINZADEH, Dan; KRISHNAN, Sridhar. On the Use of Complementary Spectral Features for Speaker Recognition. *EURASIP Journal on Advances in Signal Processing*. 2007, vol. 2008. Available from DOI: 10.1155/2008/258184.

[Jia+02] JIANG, Dan-Ning; LU, Lie; ZHANG, HongJiang; TAO, Jianhua; CAI, Lianhong. Music type classification by spectral contrast feature. *Proceedings. IEEE International Conference on Multimedia and Expo*. 2002, vol. 1, 113–116 vol.1. Available also from: https://api.semanticscholar.org/CorpusID:9237444.

[JJ04] JOHNSON, Keith; JOHNSON, Keith. *Phonetica*. 2004, vol. 61, no. 1, pp. 56–58. Available from DOI: doi:10.1159/000078663.

[Jon17] JONATHAN VAISBERG PHD/MCLSC CANDIDATE, Paula Folkeard AuD. *Comparison of Music Sound Quality Between Hearing Aids and Music Programs*. 2017. Available also from: https://www.audiologyonline.com/articles/comparison-music-sound-quality-between-20872.

[KRP21] KARIMI, Akbar; ROSSI, Leonardo; PRATI, Andrea. Adversarial Training for Aspect-Based Sentiment Analysis with BERT. 2021, pp. 8797–8803. Available from DOI: 10.1109/ICPR48806.2021.9412167.

[Kel96] KELLY, Susan E. Gibbs Phenomenon for Wavelets. *Applied and Computational Harmonic Analysis*. 1996, vol. 3, no. 1, pp. 72–81. ISSN 1063-5203. Available from DOI: https://doi.org/10.1006/acha.1996.0006.

[Kur04] KURBAN, Alishir. Analysis of shafts surface pressures using neural networks. *Industrial Lubrication and Tribology*. 2004, vol. 56, pp. 217–225. Available also from: https://api.semanticscholar.org/CorpusID:109155614.

[LT07] LARTILLOT, Olivier; TOIVIAINEN, Petri. MIR in Matlab (II): A Toolbox for Musical Feature Extraction from Audio. In: 2007, pp. 127–130.

[Li+18] LI, Yanxiong et al. Acoustic Scene Classification Using Deep Audio Feature and BLSTM Network. In: *2018 International Conference on Audio, Language and Image Processing (ICALIP)*. 2018, pp. 371–374. Available from DOI: 10.1109/ICALIP.2018.8455765.

[LIZ13] LIANG, B.; IWNICKI, S.D.; ZHAO, Y. Application of power spectrum, cepstrum, higher order spectrum and neural network analyses for induction motor fault diagnosis. *Mechanical Systems and Signal Processing*. 2013, vol. 39, no. 1, pp. 342–360. ISSN 0888-3270. Available from DOI: https://doi.org/10.1016/j.ymssp.2013.02.016.

[Lia+17] LIANG, Hong; SUN, Xiao; YUNLEI, Sun; GAO, Yuan. Text feature extraction based on deep learning: a review. *EURASIP Journal on Wireless Communications and Networking*. 2017, vol. 2017. Available from DOI: 10.1186/s13638-017-0993-1.

[LR19] LIVINGSTONE, Steven R.; RUSSO, Frank A. *RAVDESS Emotional speech audio*. Kaggle, 2019. Available from DOI: 10.34740/KAGGLE/DSV/256618.

[MRN18] MAWARDI, Viny; RUDY; NAGA, Dali. Fast and Accurate Spelling Correction Using Trie and Damerau-levenshtein Distance Bigram. *Telkomnika (Telecommunication Computing Electronics and Control)*. 2018, vol. 16, pp. 827–833. Available from DOI: 10.12928/TELKOMNIKA.v16i2.6890.

[Mik+13] MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient Estimation of Word Representations in Vector Space. 2013. Available from arXiv: 1301.3781 [cs.CL].

[Mol+01] MOLAU, S.; PITZ, M.; SCHLUTER, R.; NEY, H. Computing Mel-frequency cepstral coefficients on the power spectrum. In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. 2001, vol. 1, 73–76 vol.1. Available from DOI: 10.1109/ICASSP.2001.940770.

[ON15] O'SHEA, Keiron; NASH, Ryan. *An Introduction to Convolutional Neural Networks*. 2015. Available from arXiv: 1511.08458 [cs.NE].

[PSM14] PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher. Glove: Global Vectors for Word Representation. In: 2014, vol. 14, pp. 1532–1543. Available from DOI: 10.3115/v1/D14-1162.

[PLM04] PLISSON, Joël; LAVRA, Nada; MLADENIĆ, Dunja. A Rule based Approach to Word Lemmatization. In: 2004. Available also from: https://api.semanticscholar.org/CorpusID:15628229.

[RP12] RAFII, Zafar; PARDO, Bryan A. Music/voice separation using the similarity matrix. In: *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*. 2012, pp. 583–588. Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012. ISBN 9789727521449. 13th International Society for Music Information Retrieval Conference, ISMIR 2012 ; Conference date: 08-10-2012 Through 12-10-2012.

[Re17] RAO, K.; E, Manjunath k. *Speech Recognition Using Articulatory and Excitation Source Features*. 2017. ISBN 978-3-319-49219-3. Available from DOI: 10.1007/978-3-319-49220-9.

[SM15] SAINI, Jagriti; MEHRA, Dr. Rajesh. Power Spectral Density Analysis of Speech Signal using Window Techniques. *International Journal of Computer Applications*. 2015, vol. 131, pp. 33–36. Available from DOI: 10.5120/ijca2015907549.

[SPP17] SEETHARAMAN, Prem; PISHDADIAN, Fatemeh; PARDO, Bryan. Music/Voice separation using the 2D fourier transform. In: *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 2017, pp. 36–40. Available from DOI: 10.1109/WASPAA.2017.8169990.

[SUK20] SHARMA, Garima; UMAPATHY, Kartikeyan; KRISHNAN, Sridhar. Trends in audio signal feature extraction methods. *Applied Acoustics*. 2020, vol. 158, p. 107020. ISSN 0003-682X. Available from DOI: https://doi.org/10.1016/j.apacoust.2019.107020.

[SG16] SINGH, Jasmeet; GUPTA, Vishal. Text Stemming: Approaches, Applications, and Challenges. *ACM Computing Surveys (CSUR)*. 2016, vol. 49. Available from DOI: 10.1145/2975608.

[Str99] STRANG, Gilbert. The Discrete Cosine Transform. *SIAM Review*. 1999, vol. 41, no. 1, pp. 135–147. Available from DOI: 10.1137/S0036144598336745.

[SK19] SU, Yuanhang; KUO, C.-C. Jay. On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing*. 2019, vol. 356, pp. 151–161. ISSN 0925-2312. Available from DOI: 10.1016/j.neucom.2019.04.044.

[TGV17] TAKAHASHI, Naoya; GYGLI, Michael; VAN GOOL, Luc. AENet: Learning Deep Audio Features for Video Analysis. *IEEE Transactions on Multimedia*. 2017, vol. PP. Available from DOI: 10.1109/TMM.2017.2751969.

[Tim20] TIMOTIUS, Ivanna Kristianti. *Computational Methods for Gait Analysis in Rodents*. FAU University Press, 2020. ISBN 978-3-96147-321-2.

[Too07] TOOLAN, Michael. Geoffrey Sampson, The "language instinct" debate. Revised edition -. *Language in Society*. 2007, vol. 36, pp. 622–626. Available from DOI: 10.1017/S0047404507070510.

[Vas+17] VASWANI, Ashish et al. Attention Is All You Need. 2017.

[Wan+23] WANG, Fanfan; DING, Zixiang; XIA, Rui; LI, Zhaoyu; YU, Jianfei. Multimodal Emotion-Cause Pair Extraction in Conversations. *IEEE Transactions on Affective Computing*. 2023, vol. 14, no. 3, pp. 1832–1844. Available from DOI: 10.1109/TAFFC.2022.3226559.

[Wan+24] WANG, Fanfan; MA, Heqing; XIA, Rui; YU, Jianfei; CAMBRIA, Erik. SemEval-2024 Task 3: Multimodal Emotion Cause Analysis in Conversations. In: *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*. Mexico City, Mexico: Association for Computational Linguistics, 2024, pp. 2022–2033. Available also from: https://aclanthology.org/2024.semeval2024-1.273.

[Xu+19] XU, Hu; LIU, Bing; SHU, Lei; YU, Philip. BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis. 2019.

[YHN11] YIN, Hui; HOHMANN, Volker; NADEU, Climent. Acoustic features for speech recognition based on Gammatone filterbank and instantaneous frequency. *Speech Communication*. 2011, vol. 53, no. 5, pp. 707–715. ISSN 0167-6393. Available from DOI: https://doi.org/10.1016/j.specom.2010.04.008. Perceptual and Statistical Audition.

[Yu+10] YU, Xiaoqing; ZHANG, Jing; LIU, Junwei; WAN, Wanggen; YANG, Wei. An audio retrieval method based on chromagram and distance metrics. In: *2010 International Conference on Audio, Language and Image Processing*. 2010, pp. 425–428. Available from DOI: 10.1109/ICALIP.2010.5684543.

[YLY05] YUN-TAO, Zhang; LING, Gong; YONG-CHENG, Wang. An improved TF-IDF approach for text classification. *Journal of Zhejiang University - Science A: Applied Physics Engineering*. 2005, vol. 6, pp. 49–55. Available from DOI: 10.1007/BF02842477.

[ZLT19] ZHANG, Boyang; LEITNER, Jared; THORNTON, Samuel. Audio Recognition using Mel Spectrograms and Convolution Neural Networks. In: 2019. Available also from: https://api.semanticscholar.org/CorpusID:237274283.

[ZW13] ZHAO, Xiaojia; WANG, DeLiang. Analyzing noise robustness of MFCC and GFCC features in speaker identification. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 7204–7208. Available from DOI: 10.1109/ICASSP.2013.6639061.

List of Figures

2.1 Audio hearing threshold for human auditory system [Jon17]. 7
2.2 Traditional audio signal classification pipeline 8
2.3 Waveform of a short speech 9
2.4 Windowing in Time Domain 10
2.5 Adaptive Windowing in Time Domain 11
2.6 MFCC computation pipeline 12
2.7 Rectangular Hamming and Hanning window in both time and frequency domain [Tim20] 13

3.1 CBOW architecture 22
3.2 Simple Skip Gram model architecture 23
3.3 BERT model architecture 26

4.1 Typical artificial neuron 28
4.2 Multilayered ANN 29
4.3 An architecture of LSTM block 31
4.4 A CNN architecture [AM17] 33
4.5 A visual representation of convolutional layer 34
4.6 A visual representation of max and average pooling 36

6.1 Architecture of the CNN1D audio emotion recognition model 46
6.2 Multimodal emotion recognition pipeline 48

List of Tables

5.1 Distribution of emotion classification classes in ECF dataset (in %) . . 39
5.2 Distribution of emotion classification classes in RAVDESS dataset (in %) 41
5.3 Distribution of emotion classification classes in IEMOCAP dataset (in %) 42
5.4 Distribution of emotion classification classes in IEMOCAP dataset without *Happy, Fear and Disapointed* (in %) 42

7.1 Comparison of different model configurations and their results on the **ECF** dataset 51
7.2 Comparison of different model configurations and their results on the **RAVDESS** dataset 52
7.3 Comparison of different model configurations and their results on the **IEMOCAP** dataset 52
7.4 Comparison of different model configurations and their results on the **ECF** dataset 53
7.5 Comparison of different model configurations and their results on the **IEMOCAP** dataset 53
7.6 Comparison of different model configurations and their results on the **ECF** dataset 54
7.7 Comparison of different model configurations without the audio model feature extraction and their results on the **ECF** dataset 55
7.8 Comparison of different model configurations and their results on the **IEMOCAP** dataset 56
7.9 Comparison of different model configurations and their results on the **IEMOCAP** dataset 56
7.10 Best performing configurations for each modality and on each dataset 58