Emotion recognition from speech

Evgeniia Razumoskaia

Fourth Year Project Report
BSc Cognitive Science(Informatics)
School of Informatics
University of Edinburgh
2018

Abstract

Emotion recognition is crucial in human-computer interaction as emotions are what makes human interaction flawless and natural. In this project, the focus is on emotion recognition from speech. A review of previously used datasets, features and models is performed.

The experiments are performed using The Interactive Emotional Dyadic Motion Capture (IEMOCAP) database. Two different input feature types are extracted from the speech in the database – the eGeMAPS and spectrograms. The models used in the experiments are fully connected feedforward neural network, convolutional neural network and Long Short Term Memory networks. The eGeMAPS consistently outperform spectrograms on the task. While studying the discrepancy between the two feature types, we have shown that it appears as pitch of the voice(F_0) cannot be extracted from the spectrograms.

Acknowledgements

I would like to thank my supervisor Prof. Simon King for his support and guidance throughout the project. I am especially grateful for the advice he has given to me as a lot of it is applicable to everyday life as well as research. Also, Prof. Kings courses on Speech Processing and Text-to-Speech Synthesis were outstanding. I am positive that I would be doing a different project if I did not take the courses last year. Now that all the lectures are finished, I can definitely say that the courses were the highlight of my degree. Simon, thank you for your teaching and the effort you put in it!

I would also like to thank Zack Hodari who has helped me when, initially, nothing worked in Tensorflow and I found it hard to get my head around it.

Finally, I would like to thank my family and my friends for their immense support. Without them, I would not be anywhere where I am now. Mama, papa, Ira, thank you for the feeling that you are always there for me even though we are thousands of miles away. Dear friends who have been around throughout the year and especially the last month, without such smart and positive people as you are, my 4th year would not be the same and this past month could have driven me crazy.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Evgeniia Razumoskaia*)

Table of Contents

Chapter 1

Introduction

Emotions are part and parcel of human communication. Expressing and recognizing emotions is what makes the communication truly natural and allows humans to be empathic to each other. As psychological studies show (Sauter et al., 2010), the emotional content is expressed through various channels including facial expressions, posture, body language and emotionally inflected speech (Tracy et al., 2015). Humans are surprisingly perceptive to the emotional inflexion of speech. When asked to recognize emotions from speech, people demonstrate the exceptional accuracy of 100% when given a limited set of labels and almost 75% accuracy when they are not limited to a narrow set of labels (Johnson et al., 1986).

By improving emotion recognition of machines, we can enhance human-computer interaction significantly. For example, it would be useful for the virtual assistants such as Amazon Alexa or Apple's Siri to be able to recognise emotions from speech. They would be able to adapt their recommendations to different emotional states of the user.

Recognising emotions and using the extracted emotional information is an essential human-computer interaction challenge. The project focuses on emotion recognition from speech.

One of the issues which arises when doing emotion recognition is that we need to define what emotion is.

1.1 Defining emotions

Defining what emotion is has been a challenge for scientists studying emotions in many domains including psychology and sociology. There is increasing agreement about the"working definition" proposed by Fontaine et al. (2007). Emotions are episodic coordinated responses of, for example, neurophysiologic activation, movements and subjective feelings in response to important internal or external events.

Defining what emotion is for an automated emotion recognition task is even more challenging. Based on the working definition, Fontaine et al. (2007) propose that emotions should be described in four-dimensional space. Describing emotions as coordinates in the four-dimensional space corresponds to describing emotions using continuous labels. An alternative approach utilised further in the project is using categorical labels.

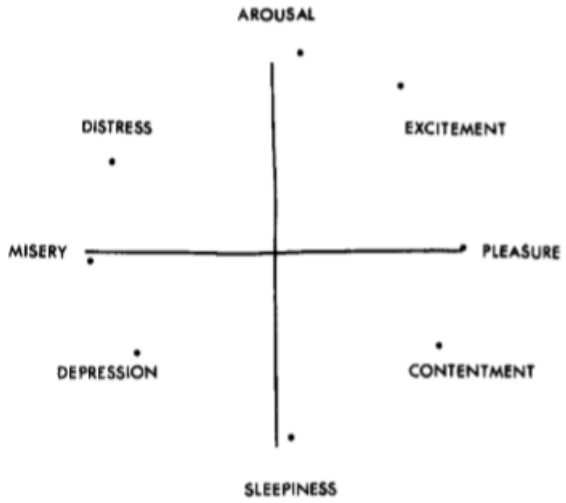


Figure 1.1: Example mapping of categorical labels onto activation-valence 2D space, from Russell (1980)

1.1.1 Continuous

Intuitively, the emotions lie on a spectrum. One way to describe emotions is in activation-evaluation space introduced by Russell (1980).The idea of the model is that all of the emotions can be mapped onto two dimensions, namely, valence and activation. An example of mapping of categorical labels to activation-valence space is shown in fig. 1.1. Valence is on the horizontal axis, and it shows to which extent the emotion is negative or positive. The vertical axis represents activation which shows the intensity of the emotion. The origin of the coordinate system is the neutral state.

Later, it was shown (Fontaine et al., 2007) that 2 dimensions are not enough to capture th full spectrum of emotions. Instead, a new approach with four dimensions was proposed. The dimensions are arousal, valence, dominance and surprise where arousal measures intensity of emotion, valence measures to which extent the emotion is positive, dominance is the measure of control and surprise measures how predictable it is.

The approach with continuous labels is more flexible that the one with the categorical labels. However, because the labels are continuous, the annotators who are to label speech or videos are given a lot of flexibility. Thus, they rarely agree. For example, in the IEMOCAP dataset (Busso et al., 2008), the inter-annotator agreement for dimensional labels is Cronbach's alpha(Cronbach, 1951) $\alpha = 0.67$ which means that the inter-annotator agreement is questionable (Cortina, 1993).

1.1.2 Categorical

An alternative approach is based on the idea introduced by Descartes who stated that the whole emotional life consists of a limited set of emotional states (Cowie et al., 2001). This idea was further developed by Plutchik (1984) who offered the idea of pure emotions – that only one emotion can be expressed at a single time.

The problem of the inter-annotator agreement is not acute as the annotators are given a set of emotions to label the speech or videos with. For the IEMOCAP dataset, the agreement on categorical labels is measured in Fleiss' kappa (Fleiss, 1971), $\kappa = 0.27$. This shows that there is a fair agreement between the annotators (Landis and Koch, 1977).

With continuous labels, emotion recognition is regression while with categorical ones it is classification. The issue is that every dataset uses its own set of categorical labels. Thus, the results of research conducted on different datasets are not directly comparable. Recently, increasing number of papers present their results on 4 emotions – happy, sad, angry and neutral (Neumann and Vu (2017); Xia and Liu (2017); Jin et al. (2015); Ghosh et al. (2016); Gideon et al. (2017); Satt et al. (2017)). Therefore, the results are more comparable. We will further refer to this set of emotions as 'basic 4'. Thus, in the project, emotion recognition will be approached as a speech classification problem.

1.2 Why is emotion recognition from speech difficult?

Speech is one of the channels through which emotions are expressed. When working with it, we need to make some choices. For example, should we only use the paralinguistic features or the words in the speech as well because the emotions might influence the word choice? Eyben et al. (2016) state that a limited number of paralinguistic features can characterise emotional state of the speaker.

However, there are several issues with speech as the signal transmitting emotions.

1.2.1 Expressed vs internal emotions

Humans are social creatures, and sometimes they prefer not to show their real inner feelings. Usually, their feelings are expressed through changes in voice, gestures and postures. However, the external signs of emotional state might not be equal to real emotional state.

In the case of this project, this does not pose an issue as we are interested in recognising the expressed emotion rather than the internal emotional state.

1.2.2 Ambiguity

Any feature which is characteristic of an emotion can have alternative meanings (Cowie et al., 2001). For instance, the prosodic characteristics of speech which are demonstrative of depressive states (Nilsonne, 1988) can also show poor reading skill (Cowie et al., 1999). Incorporating the videos od facial expressions (also available in IEMOCAP dataset (Busso et al., 2008)) can help to disambiguate in such cases.

1.3 Contributions

- Experimentation with raw and knowledge-driven features with various neural network models such as FNN, CNNs and BLSTMs

- Comparison of the results between them and with previous literature
- Study of the features within eGeMAPS to determine most important ones for emotion classification which has not been undertaken before
- Experiments showing that F_0 is an essential feature; the experiments were conducted in two ways: abolishment from the eGeMAPS and augmenting spectrograms with F_0

1.4 Thesis outline

In chapter 2, we will describe the prior work in emotion recognition and the novelty of our work in relation to the prior work. In chapter 3, we will describe the dataset which will be used throughout the project to train different machine learning models. The chapters 4 to 7 present the experiments conducted. The chapters are self-contained: they present the input features and models used in each experiment and subsequently the results of each experiment and their relation to prior work. In chapter 8, we conclude our results.

Chapter 2

Background

There has been a lot of interest in automating emotion recognition (Mustafa et al., 2018) as it would be especially useful to make the communication with virtual assistants as natural as possible. Prior work in the area will be presented in this chapter as well as how prior work relates to work in the project.

2.1 Features

Speech features which have been used for speech emotion recognition can be categorized into segmental (calculated over a small segment of sound) features, suprasegmental features (calculated over several segments of speech) and functionals which are calculated over the whole utterance usually using some statistical functions. The differences between the segmental and suprasegmental features are illustrated in fig. 2.1.

Segmental features include Mel frequency cepstral coefficients, formant amplitude and bandwidth. Suprasegmental features include, for example, fundamental frequency(F0). Functionals would include extreme values, averages and percentiles of the segmental and suprasegmental features.

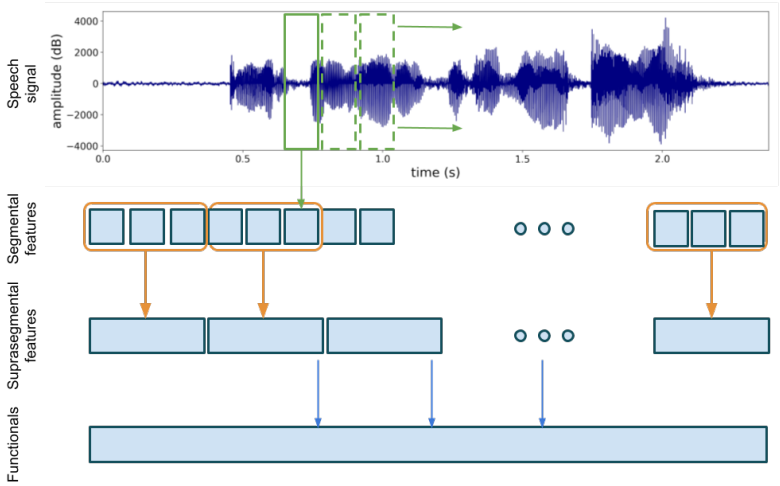


Figure 2.1: The differences between the segmental, suprasegmental features and functionals. Segmental features are calculated from every small segment of waveform and suprasegmental are calculated from several segments. Functionals are calculated over the whole utterance length.

Previously, the focus was mainly on prosodic features such as pitch or intensity with small feature sets of 10 to 100 features (Nwe et al., 2003; Wang et al., 2008). This inclination towards prosodic features was driven by previous linguistic and psychological research which had demonstrated that pitch and loudness gave the speech its emotional quality (Murray and Arnott, 1993; Lin et al., 1999).

Recently, the trend has been on increasing the number of features used in order to capture as much information as possible. For example, voice quality features such as fundamental frequency or jitter and spectral features such as Mel Frequency Cepstral Coefficients started to be used extensively in emotion recognition research (Anagnostopoulos and Iliou, 2010). A showcase example of this trend is the number of baseline features used in the Interspeech Computational Paralinguistic Challenge(ComParE) which in 8 years of its existence has grown form 384 to 6373. The number of features used by year is shown in 2.1. The data was obtained from Schuller et al. (2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017)

Table 2.1: Number of features used in the ComParE challenge

Year	2009	2010	2011	2012	2013	2014	2015	2016	2017
Number of features	384	1582	4368	6125	6373	6373	6373	6373	6373

However, using such a huge feature set as a standard in the area is implausible since this means that a vast number of model parameters need to be trained which in turn means that a lot of input data is needed. To decrease the amount of data required to train the models, emotion recognition uses smaller sets of knowledge-driven features such as eGeMAPS (Eyben et al., 2016). Although there is not one agreed standard feature set, eGeMAPS is used as a baseline feature set in the Audio/Visual Emotion Challenge(AVEC) since 2016 and increasingly used in recent research, for example, in

research published within the Interspeech conference. eGeMAPS will be described in further detail in section 4.1.

Lately, when the computing resources allowed for training more complex machine learning models, an increasing amount of research is focused on using more complex models with raw features. For automatic speech recognition, as well as for emotion recognition, the most common raw features are waveforms and spectrograms. In ASR, recent results of convolutional neural networks applied over waveforms outperform the systems trained on MFCCs (Palaz et al., 2015; Dai et al., 2017).

The same shift towards using raw features is happening in emotion recognition. Unlike in ASR, in emotion recognition, more work is done with spectrograms than with raw waveforms. The spectrogram is a frequency-domain representation of the waveform. It is more suitable for the task as humans perceive speech in the frequency domain and emotion recognition aims to replicate human emotion recognition. Satt et al. (2017) demonstrated state-of-art result on limited time-frame spectrograms(3 sec) with a combination of convolutional neural network and long short term memory layers.

2.2 Models

For emotion recognition, various classification models have been used including Gaussian Mixture Models(GMMs), hidden Markov Models(HMMs), Support Vector Machines(SVMs), k-Nearest Neighbours and various architectures of artificial neural networks.

GMMs estimate a density distribution of a combination of normal distributions(referred to as Gaussian components). GMMs are especially efficient at modelling multimodal distributions (Douglas-Cowie et al., 2007). Therefore, they are appropriate for using over suprasegmental features (Anagnostopoulos et al., 2015).

HMMs were popular due to their previous successful use in ASR (Gales et al., 2008). In an ASR system, the HMMs were used to model the temporal structure of speech. Each HMM state modelled a spectrum of a sound wave as a mixture of Gaussians. In contrast, for emotion recognition, each emotion is modelled by one HMM state. The HMMs are usually trained by maximising the margin between emotions, while the margin is scaled by a loss function (Anagnostopoulos et al., 2015).

Both HMMs and GMMs were also popular as, in the process of developing the models for ASR, a lot of useful toolkits such as HTK (HMM toolkit, Young et al. (2002)) were created which made the models easier to implement and work with.

Similarly to HMMs, Support Vector Machines are trained by margin maximisation. Unlike HMMs or GMMs, the training procedure is bound to find the optimal solution. In application to emotion classification, SVMs have demonstrated outstanding 84.84% accuracy (Schuller et al., 2005) on the Berlin Database of Emotional Speech (Burkhardt et al., 2005).

The experiments with SVMs are demonstrating results comparable with state-of-art techniques. However, recently there has been an increased interest in using neural networks to model emotion recognition. This is due to the accessibility of computing

resources, tools to create neural networks and increasing variety of neural network architectures. The first attempts of using artificial neural networks were unsuccessful in comparison to SVMs. Anagnostopoulos and Vovoli (2009) obtained 47.4% accuracy on the Berlin Database of Emotion recognition where they were using fully connected feedforward forward network. The feedforward neural network will be further discussed in 4.2.

However, in more recent work effort has been put in finding an appropriate model architecture for speech emotion recognition. For example, Satt et al. (2017) experimented with finding an optimal combination of convolutional layers and LSTM layers to do emotion recognition on spectrograms. Convolutional neural networks and LSTMs will be discussed in further detail in sections 5.3 and 5.4.1, respectively. Gideon et al. (2017) used progressive networks(ProgNets) to do emotion recognition from speech. ProgNets are a special case of transfer learning. The idea is first to train a network for the source task(gender or speaker recognition in their case) and 'freeze the network', i.e., save its parameters. Then, a new model is initialised randomly, connected with an old ('frozen') model. Parameters of the new model are trained using backpropagation. With ProgNets, Gideon et al. (2017) obtain 65% accuracy on the IEMOCAP dataset (Busso et al., 2008).

2.3 Datasets

A lot of research in emotion recognition from speech is conducted either on datasets collected specifically for a research project or on datasets which cannot be disclosed due to privacy issues.

The privacy issues arise from the source of the speech. Because collecting a dataset of natural emotional speech is extremely difficult, spontaneous emotional speech is usually taken from the recordings from call centers, TV or radio programs. The difficulty with speech recorded in real-world conditions is that such recordings could reveal the speaker's identity which gives rise to ethical concerns.

Because of that, most databases use acted speech. Collecting acted speech is simple in comparison to natural speech - the actors are asked to simulate the emotions in the recording studio. For example, the Danish Emotional Database (Engberg and Hansen, 1996) consists of recordings of scripts acted out by 4 actors. One half of IEMOCAP (Busso et al., 2008) is recordings of dialogues acted out by 10 actors.

It is important to note that there are drastic differences between the recordings of acted speech and natural speech. One of the risks is that the actors might not act well, i.e., their speech might lack speech traits which are characteristic of each emotion. Also, the actors might exaggerate the emotions as they know that this is what the researchers are looking for (Anagnostopoulos et al., 2015).

The differences and issues which arise from the differences were outlined in Cowie et al. (2001). They suggest that the future datasets should be natural rather than read by actors. In line with the suggestion, firstly, Douglas-Cowie et al. (2003) outline main issues with acted databases including evident lack of naturalness and limited scope(number of speakers, dialects and emotional states considered). Secondly, they create Dublin Natural Database (Douglas-Cowie et al., 2000) which is a rare example of a dataset which includes the natural speech from TV programs.

Table 2.2: Overview of the standard emotion recognition datasets.
Abbreviations: A – acted; N– natural

Name	MSP-IMPROV	RECOLA	eNTERFACE	EMO-DB	IEMOCAP
A / N	A	N	A	A	A
Language	English	French	English	German	English
Duration	9h	3h 50 mins	1h	22 mins	12 hours
# part-nts	12	46	42	10	10
# categories	6	2 (continuous)	6	7	11

Another direction is creating a standard speech emotion recognition dataset. As a result, several standardised datasets have appeared, for example, MSP-IMPROV (Busso et al., 2017), RECOLA dataset (Ringeval et al., 2013), eNTERFACE (Martin et al., 2006), Berlin Database of Emotional Speech(Burkhardt et al., 2005), and IEMOCAP (Busso et al., 2008). Summarised information about the datasets is presented in table 2.2. IEMOCAP was chosen for work in this project because it has the most data and the speech in the dataset is most varied. The dataset will be described in further detail in chapter 3.

2.4 How my work differs

The most recent research has focused on obtaining best model performance from one given feature set. For example, Satt et al. (2017) have focused on obtaining best performance on spectrograms by applying different convolutional(described in section 5.3) and long short term memory neural network(described in section 5.4.1) architectures. Gideon et al. (2017) were interested in obtaining the best performance on the eGeMAPS features.

Unlike previous research, this project aims to study features for emotion recognition in greater detail. The focus will be on obtaining good results with both knowledge-driven and raw features, comparing them and, if one feature set performs consistently better than the other one, find the source of this discrepancy.

Here, we will present dataset, model and feature choices based on previous research and introduce the experiments undertaken in the project.

2.4.1 Dataset

As stated in 2.3, IEMOCAP dataset was chosen to train the machine learning algorithms in the project. It was chosen due to several reasons.

Firstly, the dataset has the most data out of the standard emotion recognition datasets as shown in table 2.2.

Secondly, the recordings in the dataset are varied which allows testing how well the model generalises. The recordings in the dataset are both acted and improvised. Acted recordings are those where the participants were given a script to read out. The improvised recordings were those where the participants were described a situation (without specified text) and were supposed to act out the situation. The improvised

recordings sound more natural than the acted ones because the participants are free to choose the words which they usually use to express their emotions. As well as being variable in naturalness, the dataset is varied in terms of emotions included in it. It has 11 different categorical labels which is more than in any other standard dataset(see table 2.2).

Thirdly, a lot of recent research uses IEMOCAP as a dataset. Thus, we will be able to compare our results to the results in the previous literature. Only last year Satt et al. (2017); Zhang et al. (2017); Gideon et al. (2017); Parthasarathy and Busso (2017); Kim et al. (2017) and Neumann and Vu (2017) have all presented their results on IEMOCAP dataset.

2.4.2 Models

Following the recent trend outlined in section 2.2, we will use neural networks for emotion recognition.

For the eGeMAPS features, we will experiment with different feedforward neural network architectures. We have chosen the feedforward networks as they are the simplest neural networks. Similarly to Gideon et al. (2017), we will use the performance of the feedforward neural network as a reference. Another advantage of using feedforward network is that its weights can be analyzed. Also, when an utterance is represented using eGeMAPS features, there is one set of features per utterance. Thus, we do not need to train a model that can capture the sequence data or spatial patterns.

On the contrary, when working with spectrograms, we need to take into account the fact that the spectrograms can include the spatial features and that they are time series. To capture those, we will follow Satt et al. (2017) and experiment with convolutional neural networks and long short term memory networks. However, Satt et al. (2017) apply the networks to 3-second slices of the audio. In the project, the models will be applied to whole utterances as we assume that if the model has access to all the information that is available when extracting the eGeMAPS, it should be able to perform similarly to the model learning from eGeMAPS by extracting optimal intermediate representations. Thus, access to the whole utterance should make the task easier, on the one hand. On the other, we need to take into account that emotional inflexion of speech changes over time, so, in this respect the task becomes more complex.

2.4.3 Features

Regarding features, as shown in section 2.1, there are two major trends – using knowledge driven features with simpler models (Gideon et al., 2017) and raw features with more complex models (Satt et al., 2017).

The project aims to bring the two approaches together by developing a simple network for classification based on eGeMAPS(e.g., feedforward neural network) and experimenting with more complex networks with raw features as explained in 2.4.2. The results of the models will be compared and, if there is a discrepancy between them, we aim to find the reason behind it.

The eGeMAPS were created such that they represent as much emotional information as possible in as few features as possible (Busso et al., 2008). However, the fea-

tures were optimized separately from the models. We assume that the models which will be trained on raw features, i.e., doing emotion classification in the end-to-end fashion, will extract intermediate representations of the raw signal which are most useful for emotion recognition for a given neural network.

The project will be divided into three experiments presented below.

2.5 Experiments

2.5.1 Knowledge-driven features

The first set of experiments was concerned with working out a reference performance of a comparatively simple machine learning model over the utterance level acoustic features. As the input, eGeMAPS (Eyben et al., 2016) were used. The features will be further explained in section 4.1. As the features are heavily knowledge driven, we start with applying simpler models to them, i.e., feedforward networks explained in section 4.2. If further experiments beat the results of feedforward network, more complex models will be explored.

2.5.2 Raw features

The second set of experiments was concerned with raw features. As raw features, we will use spectrograms(explained in 5.6). We will obtain the baseline performance using optimal feedforward model from the first experiment. This is done for consistency. Because the nature of the features is different to eGeMAPS as discussed in section 2.2, we will experiment with Convolutional Neural Networks and Long Short Term memory networks. We hypothesise that the models trained on raw features will perform similarly or better than those trained on eGeMAPS. The model is expected to extract optimal intermediate representations for the task.

2.5.3 Feature comparison: why does one feature set work better than the other one?

If our hypothesis is supported by the experimental evidence(model trained on raw features performs better than that trained on eGeMAPS), we will study the features extracted by the neural networks from the spectrogram and examine how close they are to eGeMAPS.

If the results disprove our hypothesis, we will examine what hinders the model from performing at least as well as that trained on raw features. There are two possible reasons. The first one is the amount of data available, i.e., there is not enough data for the model to learn useful representations. The second one is the content of features, i.e., whether the raw features we are passing into the model does not contain some information which is crucial for the model's performance on raw features.

Chapter 3

Dataset

To train machine learning algorithms, data is required. Because throughout the project emotion recognition will be approached as a supervised learning problem, we will need labelled data to train the algorithms. As input data, we will use IEMOCAP dataset described in this chapter.

3.1 IEMOCAP

The Interactive Emotional dyadic Motion Capture database (IEMOCAP) is used to train the models for emotion recognition (Busso et al., 2008). The database contains 12 hours of speech recorded with 10 actors 5 female and 5 male recorded in 5 sessions. Each session includes scripted and improvised dialogues. When the scripted dialogues were recorded, the actors were required to act out pre-written dialogues. When the improvised dialogues were recorded, the actors were instructed to act out a described emotionally loaded situation.

The scenarios and dialogues were aimed to capture 4 most common emotional descriptors (Picard, 1995) happy, angry, sad and neutral and frustration. Each speech recording is accompanied with text transcription. Also, in each session, the facial expressions on one of the actors were recorded in videos. In the project, only the audio recordings will be used.

The recordings were annotated with emotional labels by six students using the ANVIL annotation tool (Kipp, 2001). Each utterance was assessed by three annotators. The annotators were instructed to take into account surrounding context when picking suitable emotional label.

Although the scenarios and dialogues were aimed to capture the 4 emotions above, the annotators were allowed to choose a label from a longer list. The longer list included anger, sadness, happiness, disgust, fear, surprise, frustration, excitement, and neutral state resulting in 9 categories. If neither of the options was adequate, the annotators were instructed to choose the option 'other' and write their comments. The label was assigned to each utterance by majority voting. When there was no agreement between the annotators, the utterance was labelled with 'xxx'. The annotators reached agreement in 74.6%. The resulting distribution of labels is shown in fig. 3.1.

Following the most recent work in the area (Neumann and Vu, 2017; Xia and Liu, 2017; Gideon et al., 2017; Satt et al., 2017), the experiments in the project will be

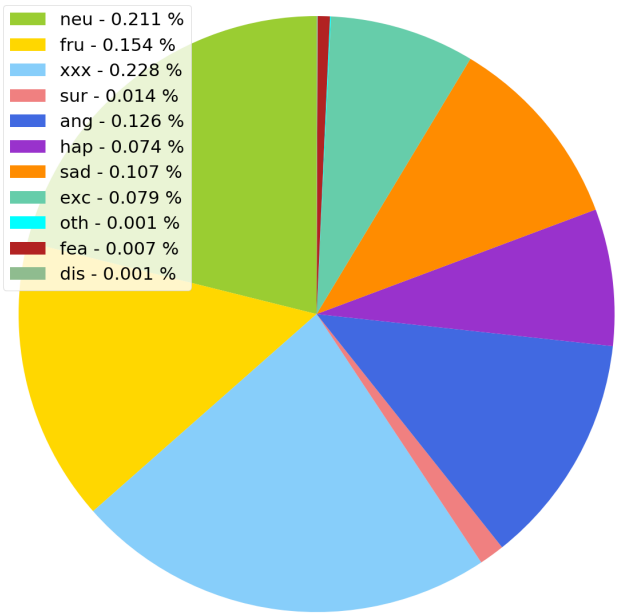


Figure 3.1: Distribution of categorical emotion labels in IEMOCAP dataset

conducted on the basic 4 emotions to make the results comparable to the previous literature. Out of 10039 utterances in the dataset, 4490 fall into one of the four categories. The distribution of the 4 emotions is shown in fig. 3.2. In the distribution, most of the utterances are neutral. In real life, the distribution would be even more skewed towards neutral emotion. Thus, we are not concerned with the lack of balance.

As well as categorical labels, each utterance was also annotated in a continuous space along the dimensions of activation (arousal), valence and dominance. In the project, only the categorical labels are used.

To train the machine learning algorithms, 5-fold cross-validation was performed. In each fold, the training data consists of 4 pairs of speakers, the validation and test data consist of the utterances of each of the 2 left speakers in the remaining session.

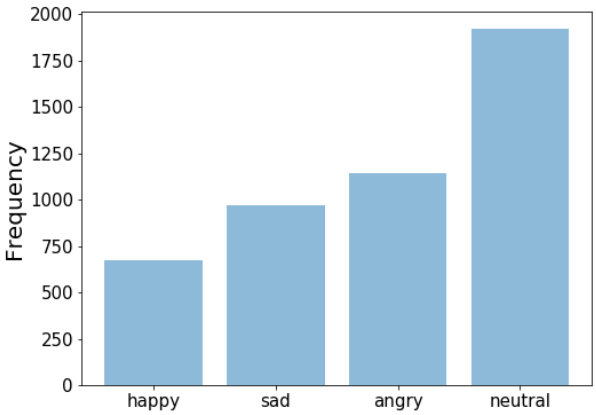


Figure 3.2: Distribution of basic 4 emotions in IEMOCAP dataset

Chapter 4

Knowledge-driven features

We present investigation of the variations of the feedforward neural network architectures in application to emotion recognition. Throughout our experiments, we will formulate the problem of emotion recognition as supervised learning, i.e., when a label is provided for each example.

The input to all the models in the experiment was the eGeMAPS features (Eyben et al., 2016) labelled with one-hot encoded emotional labels. In the experiment, we use basic 4 emotional categories (happy, sad, angry and neutral).

In this chapter, we, firstly, describe the eGeMAPS input features, then describe the feedforward network and regularization methods which will be tested and, subsequently, present the results of the experiments.

4.1 (e)GeMAPS

The (extended) Geneva Minimalistic Acoustic Parameter Set ((e)GeMAPS) (Eyben et al., 2016) feature set was devised in order to standardize the research in the area of affective computing by creating the best set of engineered features. To create it, the researchers have chosen the most effective features based on three principles: a) if a feature is able to demonstrate changes in voice production, b) how useful a feature was in previous work, and c) its theoretical significance.

Eyben et al. (2016) propose 2 versions of the feature set: minimalistic and extended. The minimalistic set includes 18 voice quality, spectral and amplitude low level descriptors (LLDs) which were shown to be most important by previous research. LLDs are extracted every 10 ms of speech. The features together with an explanation and group to which they belong are presented in table 4.1.

In order to obtain utterance level functionals, mean and standard deviation of the 18 LLDs are calculated making 36 features. Further, from fundamental frequency and loudness, the following statistics are calculated: 20^{th}, 50^{th} and 80^{th} percentiles, range of 20^{th} to 80^{th} percentiles and mean and standard deviation of the slope of rising and falling parts of signal (Eyben et al., 2016). As a result, we get 52 functionals. As well as that, the means of Alpha Ratio, Hammarberg Index and spectral slopes are included. As a result, we obtain 56 parameters. In addition to the features in table 4.1, the minimalistic set includes rate of loudness peaks (number of loudness peaks per second), mean and standard deviation of the length of continuous voiced speech (speech pro-

duced when the vocal folds vibrate, e.g., when producing vowels or consonants like [z] or [b]), mean and standard deviation of the length of continuous unvoiced speech (speech produced when the vocal folds don't vibrate, e.g., when producing consonants like [s] or [p]), number of continuous voiced regions per second. As a result, we get 62 parameters which constitute the minimalistic set.

The disadvantage of the minimalistic feature set is that it does not contain cepstral coefficients which have demonstrated to be useful for emotion recognition (Schuller and Rigoll, 2009). Further, it does not contain dynamic features capturing how the spectrum changes over time. Thus, the extended set was proposed which, in addition to 18 LLDs in table 4.1, extracts 7 more LLDs presented in table 4.2.

Table 4.2: The LLDs from which additional features to create eGeMAPS are extracted.

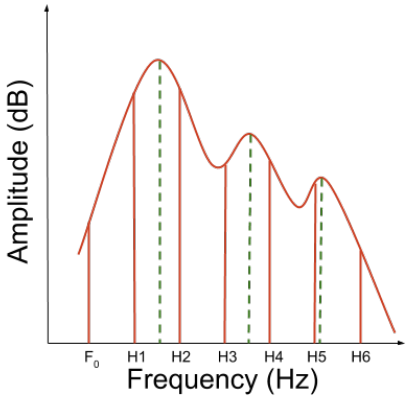
Feature	Description	Group
First 4 Mel Frequency Cepstral Coefficients (MFCCs 1 – 4)	Decorrelated features capturing the spectrum of a 10 ms window of speech; they are described in detail in section 5.6	Spectral
Formants 2 and 3 bandwidth	Frequency region around the F_2 and F_3 frequencies which is amplified	Voice quality
Spectral flux	Quadratic normalised distance between adjacent spectra; used as one of the ways to capture the dynamics of the spectrum	Spectral

To obtain the utterance level features, mean and standard deviation of each feature in table 4.2 were calculated. The mean of spectral flux in unvoiced regions only and the mean and standard deviation of the spectral flux and MFCCs 1-4 in voiced regions only are also included. As a result, we obtain 25 additional functionals. Also, the equivalent sound level is added. It is a feature to measure the average amount of background noise during the recording. Consequently, there are 88 functionals and they constitute the extended Geneva Minimalistic Acoustic Parameter Set (eGeMAPS).

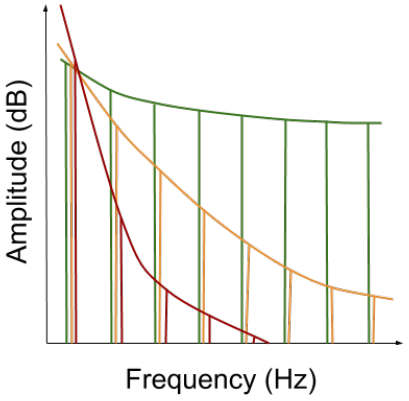
In the experiments in the project, only the eGeMAPS features will be used.

Table 4.1: The LLDs from which the Geneva Minimalistic Acoustic Parameter Set is constructed.

Feature	Description	Group
Fundamental frequency (F_0)	The frequency with which vocal folds snap; it is perceived as pitch. Illustrated in fig. 4.1(a).	Voice source features
Jitter	How much F_0 varies from one period of speech to the next	Voice source features
Shimmer	How much amplitude varies from one period of speech to the next	Amplitude / energy
Loudness	Estimate of perceived signal intensity computed as a sum of the spectrum mimicking human auditory perception	Amplitude / energy
Harmonics to Noise Ratio (HNR)	Degree of periodicity of speech; HNR=10 $\times log_{10} \frac{percentageofperiodicwave}{percentageofnoise}$ where a period is estimated from each 10 ms slice	Amplitude / energy
Spectral Slope (0 – 500 Hz and 500 – 1500 Hz)	How rapidly the amplitudes of successive frequencies in spectrum decrease as they become higher in frequency; perceived as the timbre of the voice. Illustrated in fig. 4.1(b).	Spectral
Alpha Ratio	Ratio between the sum of energy in low frequency region (50 to 1000 Hz) and in high frequency region (1 to 5 kHz);	Spectral
Hammarberg Index	Ratio between the energy maximum in 0 to 2 kHz and 2 to 5 kHz;	Spectral
Formants 1 – 3 (F_1 to F_3) frequency	Frequencies at which F_1 to F_3 are located (Formants are resonating frequencies of the vocal tract). In fig. 4.1(a), it is the frequencies at which the green dashed lines are located.	Voice source features
Formant 1 bandwidth	Frequency region around the F_1 frequency (as described above) which is amplified	Voice source features
Formants 1 – 3 relative energy	Ratio of closest harmonic of F_0 to the frequency of the formant. For Formant 1, it will be the ratio of the first formant(i.e., frequency of the first dashed line) to the frequency of H1.	Spectral
Harmonic difference (H1 – H2)	Ratio of energy of the first harmonic of F_0 to the second one (Harmonics are frequency bands at the multiples of F_0)	Spectral
Harmonic difference (H1 – A3)	Ratio of energy of the first harmonic of F_0 to the highest harmonic in F_3 range	Spectral



(a) A spectrum of a vowel where the red vertical solid lines demonstrate the fundamental frequency and the harmonics(H1 – H6) and green dashed lines show the formants (F_1 – F_3).



(b) Spectra with different spectral slopes. The green one has the smallest slope. The yellow one has a higher slope and the red one has the highest slope.

Figure 4.1: Example of a spectrum and illustration of spectral slopes.

4.2 Feedforward neural network

Fully-connected feedforward neural networks (multilayer perceptrons, further referred to as feedforward neural networks) are the simplest neural network architecture. In the network, every node in layer l is connected to every node in layer $l+1$.

The output of each layer is calculated as:

$$\mathbf{a}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$$
$$\mathbf{h}^{(l)} = f(\mathbf{a}^{(l)})$$

where $\mathbf{W}^{(l)}$ is the weight matrix, $\mathbf{h}^{(l-1)}$ is the input to layer (l), $\mathbf{b}^{(l)}$ is the bias vector which allows us to shift the activation function to the right or to the left by increasing or decreasing the value of $\mathbf{a}^{(l)}$, $\mathbf{a}^{(l)}$ is the activation of layer (l) and f is the activation function.

The first layer takes the feature vector $\mathbf{x}$ as input.

The output layer performs in the same way as previous hidden layers but, usually, a different activation function is used.

A graph of a simple feed forward network is shown in fig. 4.2.

g is usually chosen such that the output of the function corresponds to the expected output. For classification problems, two most popular ones are softmax and sigmoid functions.

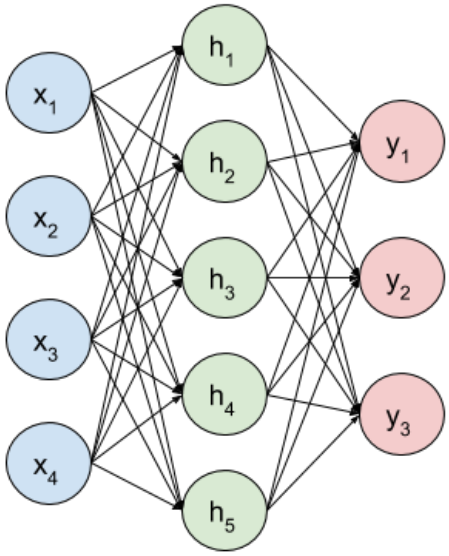


Figure 4.2: An example feedforward neural network with 4 dimensional input **x**, 1 hidden layer with 5 units and 3 dimensional output. The nodes in hidden layer are not marked with the layer number(as $h_1^{(1)}$) for clarity.

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}}$$

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

The plots of the functions are shown in fig. 4.3. It can be observed from the graph that big positive input to sigmoid function will lead to output close to 1 while big negative values will lead to output close to 0. The output of the softmax function is dependent on the input vector – a lot of probability mass will be accumulated in the biggest values in the vector and for a lot of the rest it will be close to 0.

Sigmoid function is usually used to represent the output of binary classification while softmax is thought of as output of probabilities for multiclass classification. Softmax converts the input vector **x** into a vector of probabilities that sums to one. This is achieved by normalization with a constant.

The sigmoid function is often used as the activation function in hidden layers of a neural network while softmax is usually used in the output layer of the network.

4.3 Fitting the parameters of the model

A feedforward network model must have a way to learn its parameters for the given task given the data. The trainable parameters, in this case, are weights $\mathbf{W}^{(l)}$ and biases $\mathbf{b}^{(l)}$. These are usually trained by updating at every training step. The updates are performed in a gradient fashion(using the derivatives of a loss function to minimise it)

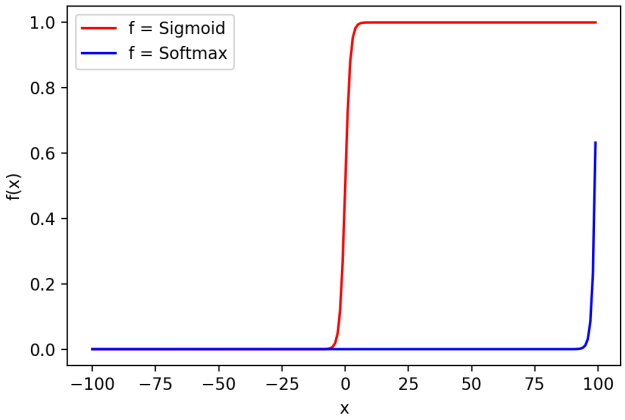


Figure 4.3: Sigmoid and softmax functions plotted on the range of values from -100 to 100.

when there is no analytical solution for optimal weights, e.g., when using non-linear activation functions.

We will explain the loss function used throughout the project. Further, we will explain simple Gradient Descent and Adam optimizers which are used to update the weights such that the loss function is minimized.

4.3.1 Loss function

In the project, cross-entropy is used as the loss function. It computes the 'distance' between 2 probability distributions – the true one hot encoded label and the prediction from an algorithm where the prediction is probability distribution across labels. Minimising cross-entropy means maximising the likelihoods of the training data. Cross entropy between a prediction $q(x)$ and true distribution $p(x)$ is computed as:

$$H(p(x),q(x)) = -\sum_{x} p(x)log(q(x))$$

For example, if the label is 'angry' one-hot encoded as [1, 0, 0, 0] and the prediction of the model is [0.4, 0.2, 0.3, 0.1] (after softmax function which turns raw predictions into probability distribution), the cross entropy is

$$H(p(x),q(x)) = -(1 \times log(0.4) + 0 \times log(0.2) + 0 \times log(0.3) + 0 \times log(0.1)) = -log(0.4)$$

It is important to note that in the project the performance of the algorithms when they have been trained will be measured using accuracy. Accuracy is calculated as the ratio of the number of samples classified correctly over the total number of samples.

4.3.2 Gradient descent

If there is an error function E which measures the difference between the predictions of the model and the true labels $\mathbf{y}$, we can calculate partial derivatives of E with respect to the weights $\mathbf{W}$ $\left(\frac{\partial E}{\partial \mathbf{W}}\right)$ and biases $\mathbf{b}$ $\left(\frac{\partial E}{\partial \mathbf{b}}\right)$ and update the parameters accordingly, in the direction of the minimum of the error function. The weights are updated as follows:

$$\mathbf{W} = \mathbf{W} - \eta \frac{\partial E}{\partial \mathbf{W}}$$

where η is the learning rate which defines how big of a step is taken at every update.

Choosing suitable η is essential for successful training. If η is large, the model can miss the point of optimum by "stepping over". If η is too small, training can take a long time because at every training step only a small step in the right direction is taken. The differences are illustrated in fig. 4.4

This leads to the problem that not the same learning rate is appropriate at all stages of training. In the beginning, when the model is far from the right predictions, we need to take big steps to get to the area where the optimum might be. Once we are close to the potential minimum, we need to take small steps to reach the optimal point. Also, the same learning rate might not be appropriate for different weights.

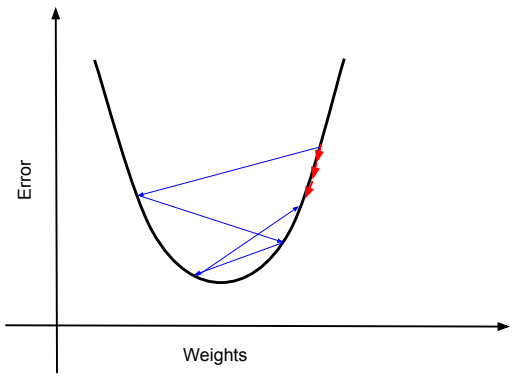


Figure 4.4: The training steps taken during gradient descent with too large and too small learning rates. Blue arrows show the training process with too large η and red arrows with too small.

4.3.3 Adam

In contrast to gradient descent, Adam(Kingma and Ba, 2014) maintains a learning rate for each weight and adaptively changes them during training.

It has inherited the properties of two of its predecessors - RMSProp and AdaGrad. AdaGrad (Duchi et al., 2011) has a different learning rate for each parameter which helps with problems with a lot of sparse weights which frequently happens in speech processing problems. RMSProp also has per parameter learning rates and adapts them based on the average of gradients of the previous steps.

Unlike RMSPRop, Adam updates the weights using not only the mean of derivatives of the error by weights(grade of change) but also uses the variance of the derivatives.

In other words, it computes a moving average of the gradient and second gradient and two parameters control the decay rate – β_1 and β_2. The values are usually set close to 1, so that they draw the momentum towards 0. In Tensorflow implementation, the default values are 0.9 and 0.999 respectively.

The update process can be described in equations as follows:

$$M_i(t) = \beta_1 M_i(t-1) + (1-\beta_1) g_i(t)$$
$$S_i(t) = \beta_2 S_i(t-1) + (1-\beta_2) g_i(t)^2$$
$$\mathbf{W} = \mathbf{W} - \frac{\eta}{\sqrt{S_i(t)} + \varepsilon} M_i(t)$$

4.3.4 Regularization

A machine learning algorithm aims to make the algorithm generalise to new, unseen data. To improve the generalisation capabilities of the model, we can prevent it from fitting to the noise of the training data –overfitting – by applying regularisation techniques. A classic example of overfitting is shown in fig. 4.5 where the training loss keeps decreasing when the loss on the validation set increases.

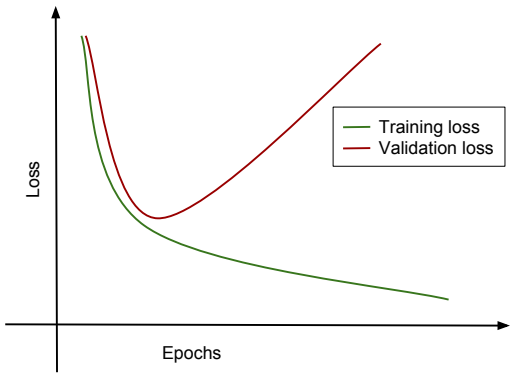


Figure 4.5: Overfitting to the training set

4.3.4.1 Early stopping

The idea of early stopping is that we should stop training the model when the validation cost stops improving. When training the algorithm, we iteratively improve its performance on the training set. Up to a certain step, it also improves its performance on the validation set (the approximation of generalisation error). However, past a certain point, the error on validation set will start increasing. This shows that the model has started overfitting to the training set as shown in fig. 4.5.

4.3.4.2 L1- and L2-regularisation

Another way to stop the model from overfitting is to force the model to find smaller weights which represent the data well. Intuitively, it prevents overfitting because with

larger weights a small change in the inputs can cause a big change in the output although the small change might be just the noise. If the weights are forced to be smaller, the network is resistant to capturing local noise in training data. Thus, it will learn existing patterns in the data rather than the noise.

There are two techniques to force the weights to be smaller – L2- and L1-regularization. To apply L2 regularisation, we modify the error function as follows:

$$E_{L2}(X,Y,\mathbf{W}) = E(X,Y) + \lambda \mathbf{W}^2$$

By adding $\lambda \mathbf{W}^2$, the error function is forced to trade off the error and the magnitude of the weights. Which component is more important is defined by the hyperparameter λ which needs to be set separately. When updating the weights, the derivative of the error function forces all the weights to be small.

In contrast, L1 regularisation draws some weights to zero and others to large values, thus, often producing sparse weight vectors. For L-1 regularisation the error function is modified as follows:

$$E_{L1}(X,Y,\mathbf{W}) = E(X,Y) + \lambda |W|$$

The main difference between the two is that L1-regularisation the weights are drawn towards zero at a constant rate while in L2-regularisation the weights are decreased by the portion proportional to $\mathbf{W}$ at each step. So, if a particular weight is big, L2 regularisation will decrease its magnitude much more than L1. If the weight is small, L1 regularisation will decrease its magnitude much more than L2.

4.3.4.3 Dropout

The idea behind the dropout (Srivastava et al., 2014) is to train many simpler neural networks hoping that every network will pick up slightly different patterns in the training data (fig. 4.6). However, training multiple neural networks is time and computationally costly. Therefore, the smaller networks are obtained by randomly dropping different nodes of the network at each training step forcing the network not to rely heavily on any one node. Although a different network is trained at each iteration, all of them share the same weights.

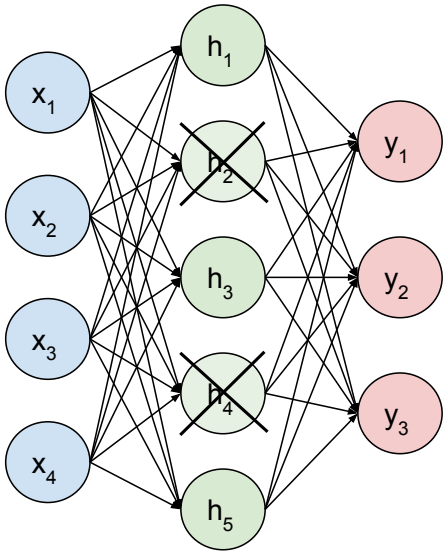


Figure 4.6: Example of one iteration of training with dropout with dropout rate 0.4

The randomised dropping is obtained by creating a binary mask using Bernoulli sampling. The mask is then multiplied with the units in the hidden layers. The units which are multiplied by 0 are dropped out.

The forward pass through the feed forward network with dropout is:

$$
\begin{aligned}
m^l &\sim Bernoulli(p) \\
h^l &= m^l * y^l \\
a^{(l+1)} &= \mathbf{W}_i^{(l+1)} h^l + \mathbf{b}^{(l+1)} \\
y^{(l+1)} &= f(a^{(l+1)})
\end{aligned}
$$

where p is the dropout rate.

Dropout has shown to be effective in many practical circumstances (Srivastava et al., 2014).

4.4 Experimental setup and implementation

Throughout the experiments, the Adam optimizer was used (Kingma and Ba, 2014) with its default parameters for β_1 and β_2 of 0.9 and 0.999, respectively. The batch size was set to 100. For all experiments, early stopping was applied while separate experiments with different other regularizations (L1, L2 and dropout) were conducted. The activation function used in the hidden layers of the model was sigmoid function as pilot experiments have demonstrated that the activation function does not have much effect on the model's performance and sigmoid was the fastest in training.

In this experiment, the optimal model architecture was found using grid search through the number of hidden layers(2 - 6) and the number of units(16 - 2048) in each

layer. The best learning rate was obtained through grid search through 20 values in the logspace between 10^{-6} and 10^{-4}. Further experiments were concerned with applying different regularisation techniques and tuning the hyperparameters for them.

We have trained the models using 5-fold cross-validation. 4 sessions(8 speakers) were used as the training data and the 5th one was divided into the validation and test set. We have switched whether the male speaker was in validation and the female one in the test set or vice versa not to introduce biases.

The eGeMAPS features were extracted using openSMILE (Eyben et al., 2013) package which enables standardized feature extraction, thus, making the results more comparable. The models were implemented in Python (Van Rossum and Drake, 2011) using NumPy (Walt et al., 2011), Tensorflow (Abadi et al., 2016), Matplotlib (Hunter, 2007) and Scikit-Learn (Pedregosa et al., 2011) packages with the aid of Jupyter Notebooks (Kluyver et al., 2016).

4.5 Results

To check that the model learns, we compare it to two simple models – model that predicts the label randomly and model that always predicts most common label. Their accuracy is 24% and 33%, respectively.

The results of the experiments with different number of layers are illustrated with fig. 4.7. Although the graph shows the results for the models with 128 nodes in each layer, the experiments were conducted with 16-2048 nodes per layer. Irrespective of the number of nodes in the layer, the trend is similar.

As we can observe, the performance of the model deteriorated when the number of layers was increased. When we increase the number of layers, we allow the model to learn more complex representation of input features. More subtle representations should improve model's performance. However, many of the eGeMAPS features are different statistics applied to the values of the same characteristic of speech, for example, 20^{th}, 50^{th} and 80^{th} percentile of logarithmic F_0. Intuitively, the statistics are the result of computations which are useful for mapping the features to labels but not for extracting complex representations from them. Another possible explanation is that there is not enough data to train increasing number of parameters of the model.

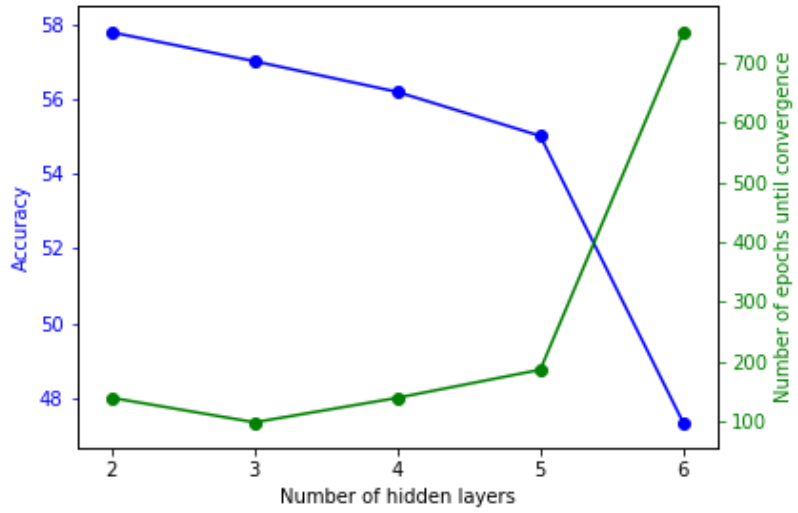


Figure 4.7: Experiment with different number of layers. For the graph, the models with 128 nodes in each layer were used.

As the experiment above has demonstrated, irrespectively of the number of nodes in the layers, models with 2 layers perform best and take least time to train. Further experiments will be concerned with studying the 2 layer networks in further detail.

Aiming to obtain the optimal architecture, the experiments with the different number of hidden units in each layer were conducted. The results are presented on the heatmap in fig. 4.8. The trend is that the models with wide first layer and narrow second one perform best. As we can observe, the optimal performance was obtained with the 2048-16 architecture. The accuracy on validation set was 61.8%. The performance of this model will be used as a reference in future experiments.

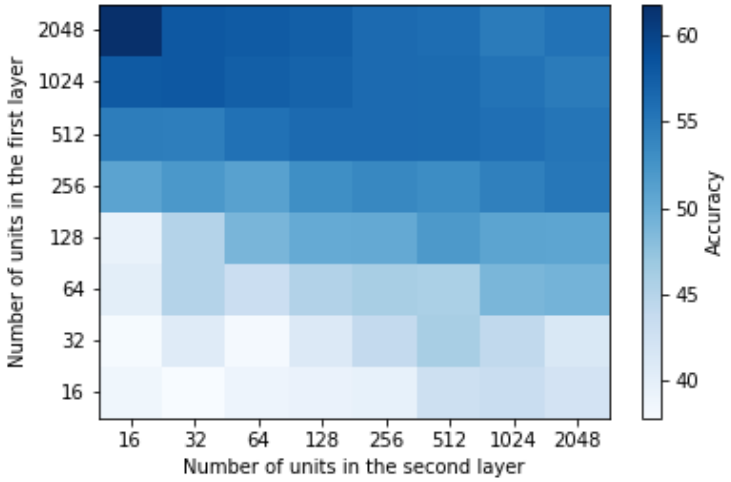


Figure 4.8: Experiment with different number of units in 2 layer networks. The darker blue colour is, the better the performance of the network is. The best performance was obtained with 2048-16 architecture.

When experimenting with different regularization techniques, the model (2048x16)'s performance decreased which means that the model was not overfitting. Moreover, when looking at the performance of the model with different dropout rates, both accuracy and loss deteriorate as more units are randomly dropped out. Moreover, even when dropping out only from the input layer the performance decreased significantly. This shows that the eGeMAPS feature set is minimal, i.e., that if we use a subset of the parameters the performance of the model will drop.

The experiments with different learning rates demonstrated that the models trained better when using smaller values than the default 10^{-3}. The optimal learning rate value was found to be 10^{-4}. This value will be used in further experiments.

We have also experimented with different learning rate schedules (exponential decay, time based decay and step decay). The model is robust to changes in them. We assume that this is due to using Adam optimizer which applies a different learning rate to each weight.

4.6 Summary

The best feed-forward architecture trained on eGeMAPS features obtained 61.8 % accuracy on the test set. This result will be used as the reference in further experiments.

The architecture of the model was 2 layer network with 2048 nodes in the first layer and 16 nodes in the second trained with Adam optimizer with the learning rate of 10^{-4} and no regularisation.

Chapter 5

Raw features

In this chapter, we investigate optimal model which could learn to recognize emotions from raw features. As in chapter 4, we treat emotion recognition as a supervised learning problem where each utterance is labeled with one of the basic 4 emotions.

The input to the models in this experiment is spectrograms extracted from the waveforms of the utterances in IEMOCAP dataset.

In this chapter, we will describe what the spectrogram is and how it is extracted. Further, the new models which will be used in the experiments, namely, the convolutional neural networks and Long Short Term Memory networks – a type of recurrent neural networks – will be described. Then, we will present the results of the experiments.

5.1 Waveform

A waveform is a two dimensional time-domain representation of sound where the first dimension is time (sec) and the second one is amplitude (in dB). An example of a waveform is shown in fig. 5.1.

Waveforms are the simplest features to obtain as they are the form in which the sounds are stored. One of the limitations is in sampling. In order to record the waveform, we need to measure (sample) the amplitude f_s times per second. f_s is known as sampling rate. When sampling the speech at a constant sampling rate, the frequencies which can be represented are limited.

A disadvantage of waveforms for any speech processing task is that the waveforms do not represent the features which humans perceive. Humans perceive sounds as a combination of frequencies (Jurafsky and Martin, 2014). A possible solution would be to use the features which represent the waveforms in the frequency domain, i.e., spectrograms.

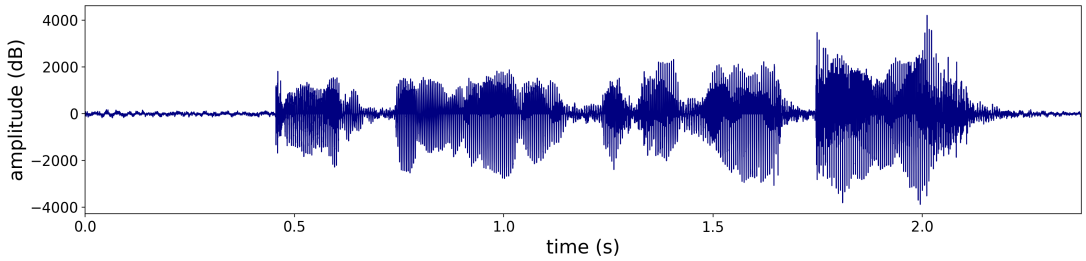


Figure 5.1: Waveform of the recording Ses04F_impro01_F011 from IEMOCAP (Busso et al., 2008) dataset: "I have been waiting for over an hour."

5.2 Spectrogram

A spectrogram is a representation of a waveform in the frequency domain. Every sine wave contains one frequency, thus, if we want to represent speech in the frequency domain, we need to represent it as a sum of sine waves. Fast Fourier transform(FFT) is used for this aim. FFT is based on the Fourier principle which states that any periodic signal can be represented as a sum of sine waves. FFT shows which sine waves constitute the given signal. The amplitudes of the sine waves define the frequency content of the signal.

The process of converting a waveform into a spectrogram consists of several steps which are depicted in fig. 5.2. The waveform is passed through pre-emphasis filter in order to balance the frequency spectrum. Usually, high frequencies have smaller magnitudes and low frequencies have high magnitudes. Pre-emphasis amplifies the high frequencies to balance that.

The next step is framing, i.e., dividing the waveform into short time frames of 20-40 ms. The idea behind it is that, to apply FFT, we need a periodic signal. Speech is not periodic over am extended period of time. However, we assume that the it is periodic over a short period.

After framing, we apply windowing to smooth the edges between the rectangular windows.

Further, the fast Fourier Transform is applied on each windowed frame. The Fourier transform gives a frequency domain view of a time-domain window. The Fourier Transform decomposes the signal in time-domain into frequencies that make it up. In other words, it measures how much energy there is in each frequency band. As a result, we obtain power spectrum.

To obtain frequency bands on Mel Scale, triangular filters of increasing width are applied to the power spectrum. The Mel scale is more detailed at lower bands and less detailed at higher bands as it aims to mimic human perception. Each filter responds with 1 at its center frequency and decreases linearly to 0 as the frequency approaches to the centers of adjacent filters. As a result, we obtain Mel-Frequency Filterbanks.

The filterbanks are highly correlated, especially, those in the adjacent frames. For the algorithms such as HMMs previously used for automatic speech recognition, it is problematic. Thus, the decorrelated Mel Frequency Cepstral Coefficients(MFCCs) were developed. Discrete Cosine Transform is applied to decorrelate the filterbanks.

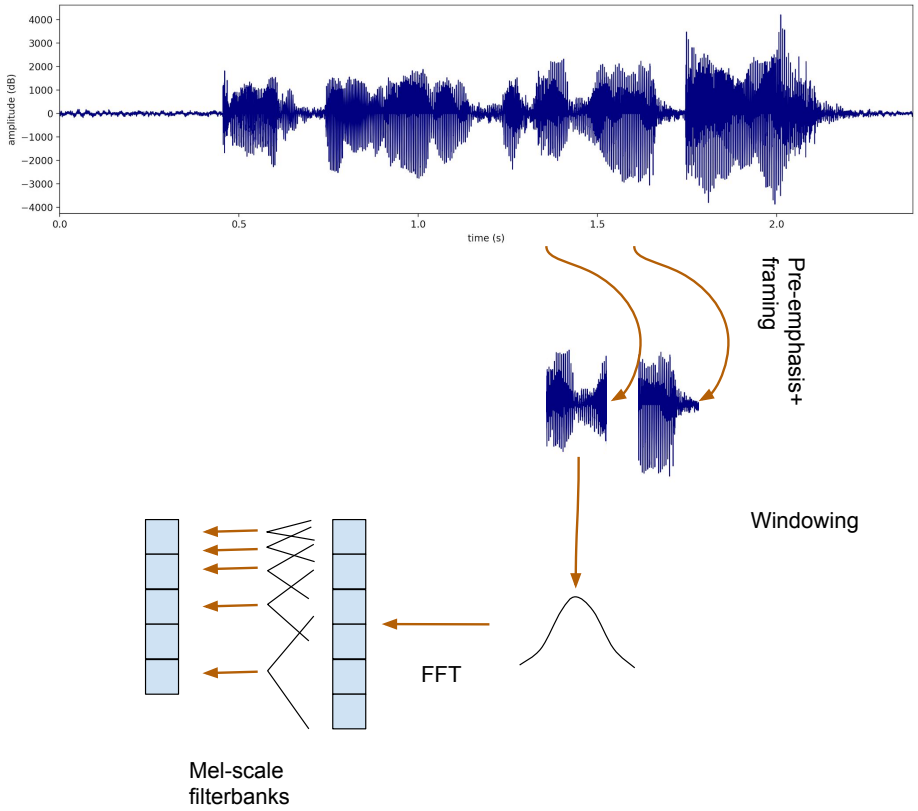
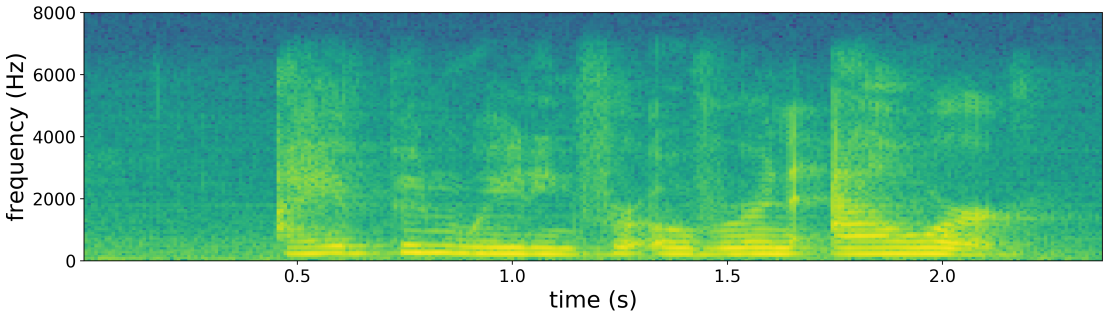


Figure 5.2: Step by step spectrogram extraction

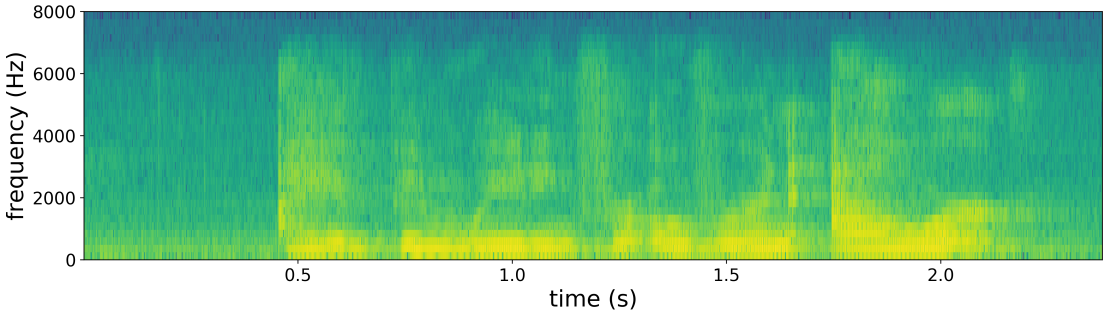
An example of a spectrogram is shown in fig. 5.3 where the degrees of energy are represented as yellow-to-blue (yellow shows a lot of energy at a frequency and blue shows no energy) at different frequencies(on the vertical axis) by time(on the horizontal axis). In fig. 5.3, a vertical slice represents a Fourier transform of a time window, and a row shows how the signal varies across time at a given frequency band.

Depending on the size of the time window, we will obtain spectrograms of different time and frequency resolution. With a wide window, we get narrow band spectrogram. In this case, the frequency will be resolved at the expense of time which means that the individual harmonics are revealed but adjacent spectra are mixed together. An example of a narrowband spectrogram is shown in fig. 5.3(a). In fig. 5.3(a), harmonics are the horizontal lines one over the other. The source of harmonics is vocal cord vibration. The first of harmonics(the lowest in fig. 5.3(a)) is fundamental frequency F_0. It is the lowest frequency produced by vocal folds perceived as pitch. The shape of vocal folds influence F_0. It can be observed from fig. 5.3(a) that harmonics and F_0 are discontinuous across time. That is because the vocal folds do not vibrate when producing voiceless sounds (such as [f] or [s]).

With a short window, we get wideband spectrogram. In this case, we get better time resolution, but adjacent harmonics are blurred. We can capture pitch periods appear as vertical lines. Pitch periods are the moments when the vocal folds slap. Because in wideband spectrogram the energy is measured across a small period of time, we can capture rapid changes in amplitude as the one observed when the vocal folds slap. An example of a wideband spectrogram is shown in 5.3(b).



(a) Narrowband spectrogram



(b) Wideband spectrogram

Figure 5.3: Spectrogram of the recording Ses04F_impro01_F011 from IEMOCAP (Busso et al., 2008) dataset: "I have been waiting for over an hour."

5.3 Convolutional Neural Networks

If we treat the spectrogram as an image which has some local patterns which are relevant to the emotional inflection of speech, we can use the convolutional neural networks (CNNs) frequently applied to image classification task (Goodfellow et al., 2016). The advantage of convolutional neural networks is that they learn local patterns from the images and, then, they can recognize the patterns anywhere in the picture. The weight matrix in the convolutional layers is called a kernel.

Usually, the convolutional neural networks consist of several convolutional layers each followed by pooling layer and one or several fully connected layers applied at the end to obtain final probability distributions across classes as shown in fig. 5.4. Each type of layer will be explained below.

The input layer holds the raw pixel representation of an image. In our case, it will be a spectrogram. The convolutional layer will compute the response of each kernel by sliding it across the input. The output is called a feature map. In a sense, each feature map represents the presence of a feature across the image. Each element in the feature map is connected to a small area in the previous feature map. Every convolutional layer contains several kernels which allow it to recognise several patterns at every layer. Number of kernels in each layer is usually referred to as number of channels.

After that, a max pooling layer is usually applied which is used to reduce the dimensionality of the input. Max pooling layer steps across the image and outputs the maximum value of the values in the kernel. In this case, the kernel is not a weight matrix but a rectangular area slid across the image. Max pooling layer does not have

any trainable parameters.

Although it is conventional to use max pooling layer on top of the convolutional one, Springenberg et al. (2014) state that it can be replaced by convolutional layer with a bigger step when convolving and it will learn to do max-pooling.

The final layers in the CNNs are usually fully connected ones although Springenberg et al. (2014) doubt that this is necessary either.

The parameters in fully connected and convolutional layers are trained using gradient descent described in 4.2.

Thus, the CNNs gradually transform the input image into the probability distribution across classes.

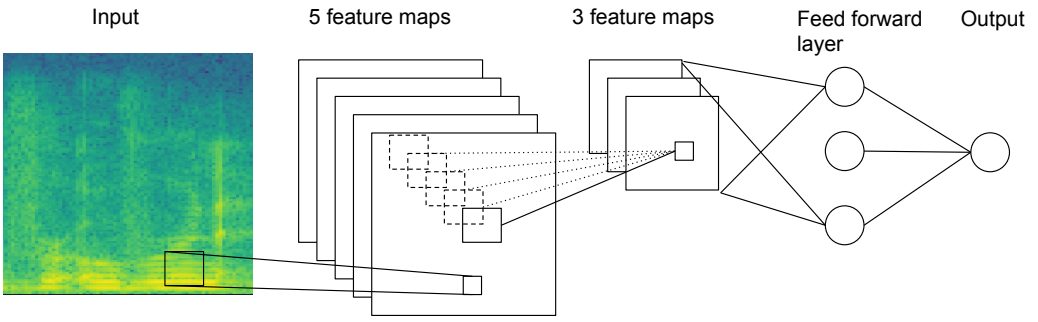


Figure 5.4: Convolutional neural network. A new number in a feature map in the next layer is obtained by multiplying a region of the input with the weight matrix called kernel.

It is important to note that the convolutional layers have far less trainable parameters then the fully connected ones. For example, if the input image has the shape of 28 $\times$ 28 $\times$ 3 and a fully connected layer has 100 nodes, we have 28 $\times$ 28 $\times$ 3 $\times$ 100 = 235200 weights. With the same size of the input image and the kernel of the size 3 $\times$ 3 with 10 output channels, the convolutional layer would have 3 $\times$ 3 $\times$ 3 $\times$ 10 = 270 weights to learn. If we increased the number of output channels to 100, we would still have an order of magnitude less trainable parameters.

In our experiments, the convolutional neural network is used because the model capable of learning spatial patterns will learn the patterns in the spectrograms which are characteristic of different emotional states. Williams and Stevens (1972) have demonstrated that the patterns in the spectrograms vary across different emotional states, especially, the basic 4.

5.4 Recurrent neural networks

Recurrent neural network (RNN) is the network used to model sequential data. The network receives every entry as an input and it remembers the information from previous state by getting the previous hidden state as the input as shown in fig. 5.5.

$$
\begin{aligned}
net_{in}^{(t)} &= \mathbf{V}x^{(t)} + \mathbf{U}s^{(t-1)} \\
s^{(t)} &= f(net_{in}^{(t)}) \\
net_{out}^{(t)} &= \mathbf{W}s^{(t)} \\
y^{(t)} &= g(net_{out}^{(t)})
\end{aligned}
$$

where $x^{(t)}$ is the input at time t, $net_{in}^{(t)}$ is the activation for hidden state, $s^{(t)}$ is the hidden state at time t, $net_{out}^{(t)}$ is the activation of the output layer, $y^{(t)}$ is the output vector at time t, and g and f are nonlinearities.

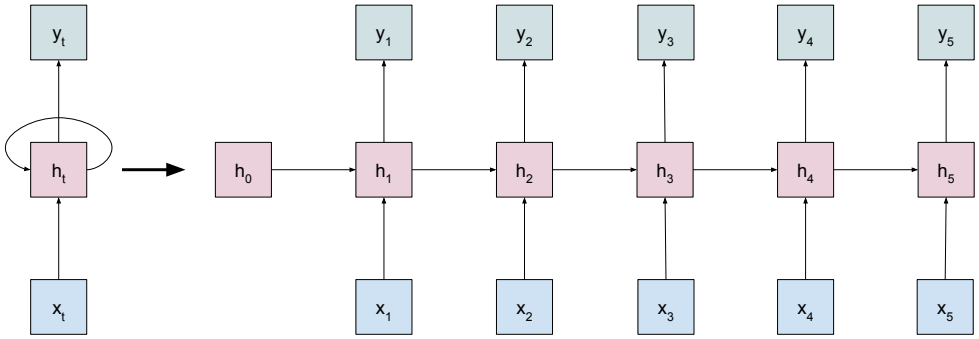


Figure 5.5: Recurrent sequence to sequence neural network. For each input x, there is an output y. The hidden state is passed on across time so that for each next output the output from the previous state is taken into account.

RNNs can be used either for sequence to sequence prediction or for sequence classification.

As the project will use RNNs for classification, more details on sequence classification will be presented further, in section 5.5. The main idea is that only the output of the last element is taken to predict the class of the sequence as by that point full sequence has already been seen.

Because the network is applied to inputs which constitute a sequence, the weight update should be applied 'across time', i.e., accounting for all of the previous input states. The algorithm by which the network is trained is called backpropagation through time (Mozer, 1995).

As shown in fig. 5.6, backpropagation through time works by unrolling the network in time q times where q is the length of the input sequence. Each timestep has one input, one copy of the network and one output. Every copy of the network has the same parameters. Then, the weights are updated through time using simple backpropagation as described in section 4.3.2.

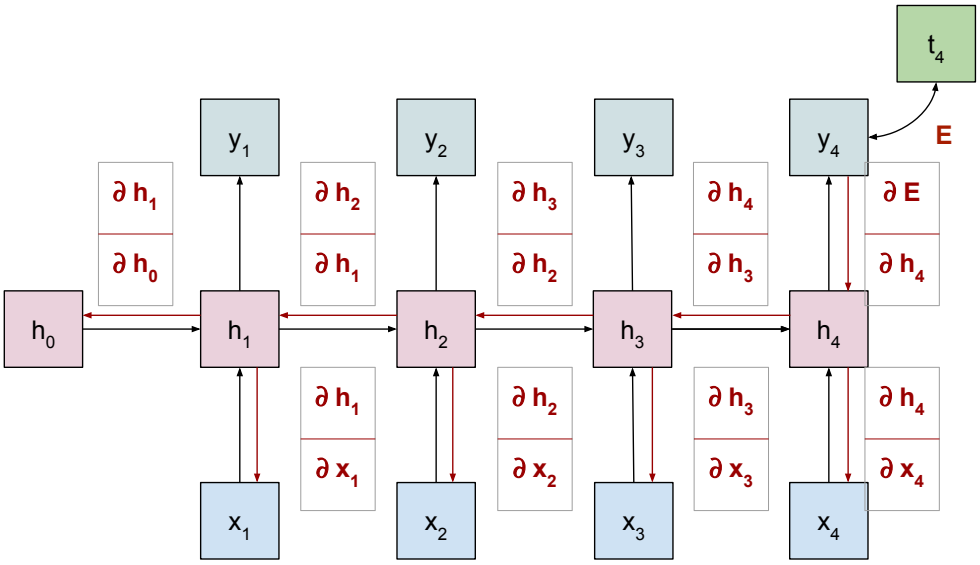


Figure 5.6: Backpropagation through time: the network is unrolled, and the gradient of the error is backpropagated across time to update weights.

The disadvantage of the RNN with BPTT as described here is the problem of the exploding or vanishing gradients. This happens because, with increasing lengths of the sequences, the number of derivatives required to be taken for a single weight update will also increase. Thus, the weights might vanish or explode. Therefore, the network might not be able to learn long-term dependencies past a few time-steps because the weights will not be updated reasonably.

The model which deals with the issue which will be used in the project is Long-Short Term Memory network.

5.4.1 Long Short Term Memory

LSTMs (Hochreiter and Schmidhuber, 1997) are a particular case of RNNs which can learn long-term dependencies. The difference between the Long Short Term Memory (LSTM) and simple RNNs is that non-linearity is not applied to the internal state. The derivative will be one, so, the backpropagated error is kept constant.

The special trait of LSTMs is their cell state and the way it is regulated. Every cell state has 3 gates which 'decide' by how much the information should be let through. Every gate has a sigmoid function which (as shown in fig. 5.7) outputs the values between 0 and 1. A value of zero means that no information will be let through and value of one means that all the information will be let through.

There are three gates like this in each cell state. The first gate('forget gate') looks at the current input and the hidden state from the previous state. This gate decides how much of information will be forgotten by the cell state. The second gate('input gate') decides which values will be updated in the cell state. The values are then updated.

Next, the output needs to be generated. It will be based on the cell state. However, it will not include the full output. It will be a filtered version. To filter, the third('output

gate') will be used. Then, the values will be pushed through the tanh layer to make the values between -1 and 1 and then multiplied by the output of output gate to filter the values.

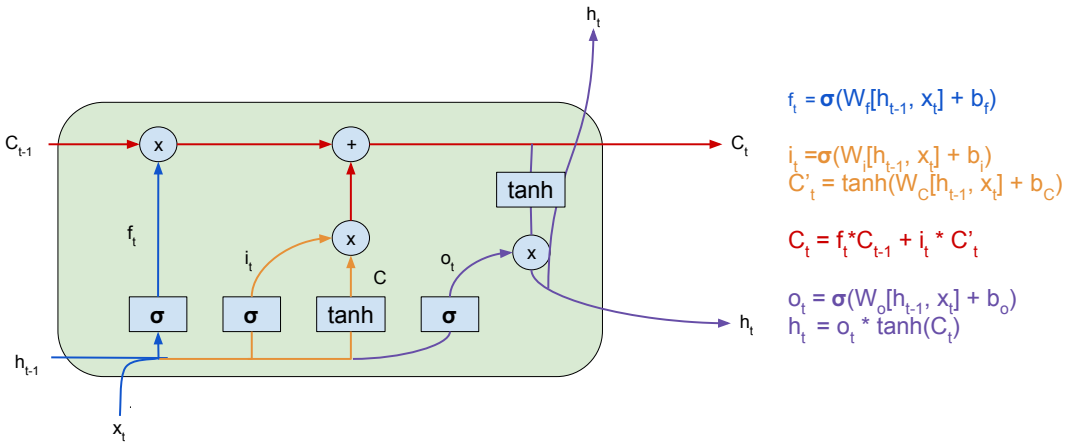


Figure 5.7: Long short term memory unit with equations for each step. Inspired by Ch. Olah's 'Understanding LSTM Networks' blog

5.4.2 Bidirectional LSTMs

The idea behind the bidirectional LSTMs (Schuster and Paliwal, 1997) is to have two of the same LSTM layers one of which reads the input in forward order and the other one in reversed. In this way, the model will be trained on all the sequence both before and after the current input. In our case, it means that the model has seen the spectrogram of the full utterance.

The reasoning behind using the bidirectional LSTMs in this project lies in the fact that emotional features of the current frame of speech may depend on both preceding and following frames. This was shown to be the case in speech recognition (Graves and Schmidhuber, 2005). As well as that, we would like to compare the performance of the network on raw features to the network trained on the knowledge-driven features. The eGeMAPS features (described in section 4.1) are extracted from the full utterance. Using bidirectional LSTMs allows us to make the results more comparable. We hypothesise that if the eGeMAPS features are optimal for emotion recognition, then the network will learn them and will perform at least similarly to the network trained on the eGeMAPS features.

5.4.3 Dropout in RNNs

Dropout as a regularisation technique was created for fully connected feedforward neural networks as described in section 4.3.4.3. When we remove the nodes randomly from the fully connected network, we know (because of the architecture of the network) that removing the nodes will not disrupt the network's learning. However, it is not the case for RNNs. If some recurrent connections were randomly removed from the network, the network could lose its ability to learn long term dependencies. There

are two major approaches proposed to solve the problem. The first way is to apply the dropout only to the output layer (Pham et al., 2014). Another approach is to apply dropout between layers but not across recurrent connections (Zaremba et al., 2014). In the project, the dropout will be applied following the Zaremba et al. (2014)'s method as shown in fig. 5.8.

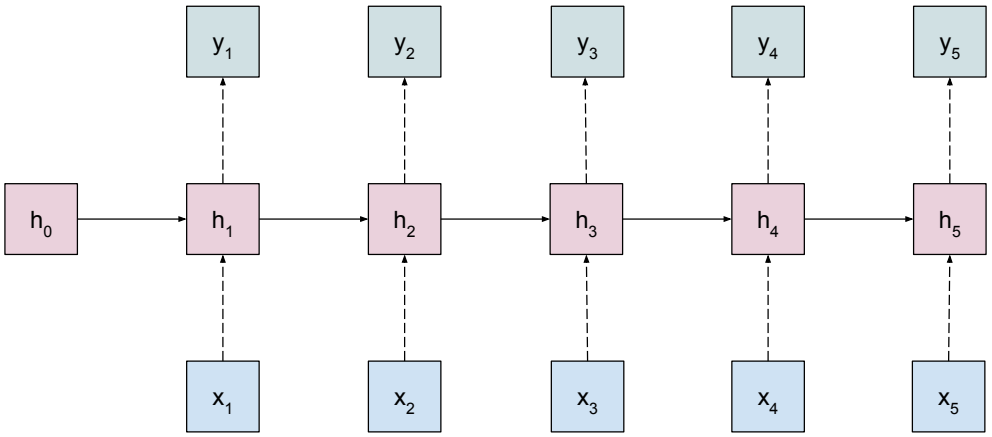


Figure 5.8: Regularized RNN following the Zaremba et al. (2014)'s method. The dashed arrows indicate the connections where the dropout is applied and solid arrows indicate the connections where the dropout is not applied.

5.5 Sequence classification

The networks which process sequences such as RNNs and LSTMs have been successful in language-related tasks such as speech recognition (Graves et al., 2013) and language modelling (Sundermeyer et al., 2012).

The success of the recurrent networks lies in the sequential nature of language. The text is a sequence of words. Speech is a continuous sound wave. It can be divided into frames using framing as described in section 5.6. As our task is essentially speech classification and we are going to treat speech as a sequence of frames, we will further discuss how sequence classification using RNNs is done.

Usually RNNs output values at every step. When doing the classification, all the sequence is passed in and the output of the last state is used as the final prediction as shown in fig. 5.9.

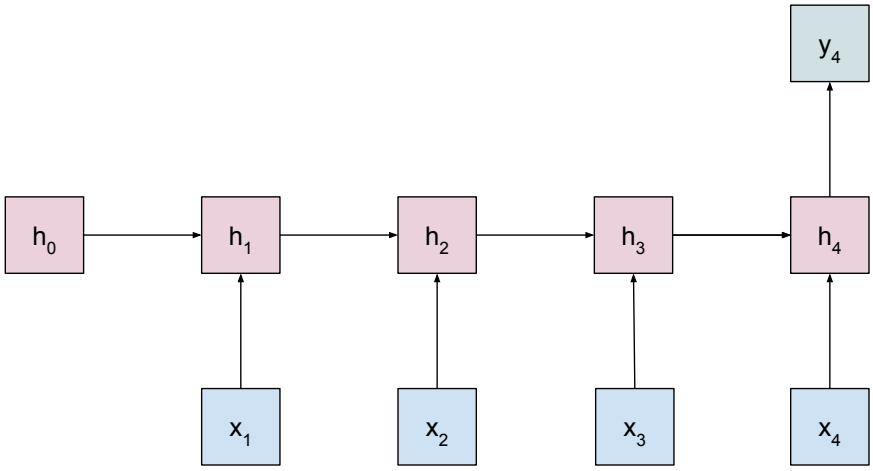


Figure 5.9: RNN network used for classification. There is one output for the whole sequence, the output from the last state which predicts the class of the sequence.

5.6 Experimental setup and implementation

The shape of the spectrograms varies with the length of the utterance. Thus, it is impossible to feed the raw spectrograms(as is) into the CNNs. To solve the issue, we can either clip all spectrograms to the length of the shortest utterance and label each of them using the label of the utterance, divide the spectrograms into smaller pieces and label each of them using the label of the whole utterance or pad it to the length of the longest utterance. Using the third option due to memory limitations was impossible. With the second approach, during prediction, we need an approach to accumulate the labels across the pieces of the same utterance. In the experiments, we have chosen the second approach and accumulated the label for each utterance using the majority voting.

Throughout the experiments in this part, we have used Adam optimizer (Kingma and Ba, 2014) with its default parameters for β_1 and β_2 of 0.9 and 0.999, respectively. The learning rate η was set to 10^{-4}. The batch size was set to 100. The early stopping was applied in all of the experiments and dropout will be applied in separate experiments.

The activation function used in the hidden layers was sigmoid to make the results comparable with those in section 4.5.

For comparison, an experiment was conducted with the feed-forward model with the same architecture as the optimal one in section 4.5. The result of this network will be used as the baseline in this experiment.

For CNNs, the optimal model architecture was found using grid search through the number of layers (2 - 5), size of the kernel (2×2 to 15×15) and number of channels(16 - 64).

For bidirectional LSTMs, the optimal model was found using grid search with the number of hidden nodes in each unit.

Further, we have experimented with combinations of the two models with 1 to 3 convolutional layers preceding the LSTM layer. The motivation behind this experiment lies in the fact that the convolutional layers can create a new representation of the spectrogram from which it would be easier for the LSTMs to extract emotional information.

We have trained the models using 5-fold cross-validation. 4 sessions(8 speakers) were used as the training data and the 5th one was divided into the validation and test set. We have switched whether the male speaker was in validation and the female one in the test set or vice versa not to introduce biases.

The spectrograms were extracted using librosa (McFee et al., 2015) package which was developed especially for processing the audio signal in Python. The initial waveform was sliced into 1 second long pieces. From the pieces, the spectrograms were extracted with the window size of 25 ms and 40 frequency bands. This is a wideband spectrogram giving us a good time resolution as explained in section . The wideband spectrograms have previously demonstrated state-of-art results with LSTMs in speech recognition (Hannun et al., 2014), thus, we have chosen wide- rather than narrowband spectrograms.

The models were implemented in Python (Van Rossum and Drake, 2011) using NumPy (Walt et al., 2011), Tensorflow (Abadi et al., 2016), Matplotlib (Hunter, 2007) and Scikit-Learn (Pedregosa et al., 2011) packages with the aid of Jupyter Notebooks (Kluyver et al., 2016).

5.7 Results

As the baseline, we will use the accuracy of the feedforward neural network with best architecture from section 4.5. On the spectrograms, its performance was 49.6%.

As a result of the grid search, we have found that the best convolutional neural network was the one with 2 layers, the kernel size of 10 and 32 channels in each layer. It has achieved the accuracy of 56.01%. Because this result is considerably above the baseline with random or most frequent predictors(24% and 33%, respectively) as well as the feedforward network, we can say that the model has learnt useful patterns from the spectrogram. The patterns assist in determining the emotional state. This proves the idea proposed by Williams and Stevens (1972). However, even the best convolutional neural network is considerably worse than the feed forward neural network trained on the eGeMAPS features.

Our first assumption was that the model cannot achieve the same results as the one trained on the eGeMAPS features because the CNN does not have access to full utterance while the knowledge-driven features were extracted on utterance level.

Thus, we have further implemented the bidirectional LSTMs. The best accuracy of 57.07% was obtained with 32 hidden nodes. The performance of the model has improved slightly in comparison to the CNN above, so, the access to more information might have helped. However, the difference is only marginal. Therefore, it could be the case that learning to create any utterance level features similar to eGeMAPS is more complex than expected.

In the experiment with different hidden nodes(see fig. 5.11), we can see that as the number of hidden nodes in the LSTM layer increases the performance of the network degrades. We assume that this tendency is observed because the training set has a small number of samples. Because of the size of the dataset, the network with 512 nodes might be heavily overfitting to the training set because it has enough parameters to capture the noise of the training set. To verify this hypothesis, we have applied dropout. The model's performance stayed as low as it was for 512 units without dropout(for detailed results see fig. A.1). Thus, the model must be undertrained, i.e., there is not enough data for the model to learn useful weights.

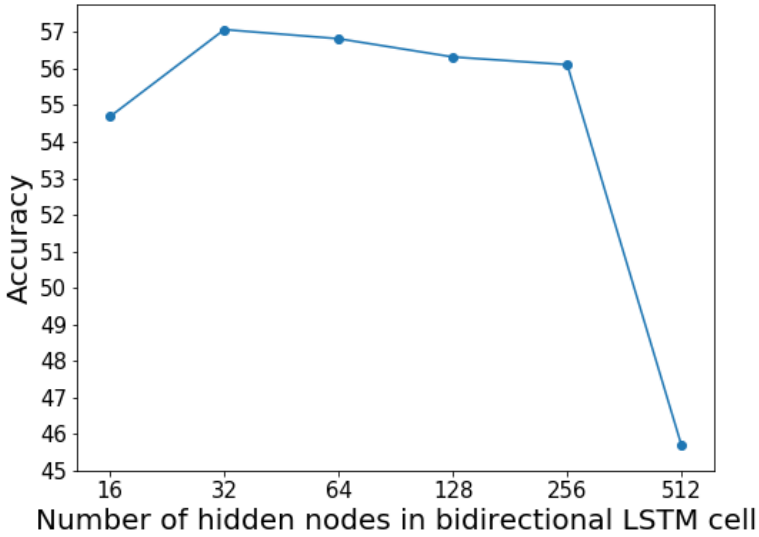


Figure 5.10: Accuracy of the bidirectional LSTM depending on the number of hidden nodes

In experimenting further with the models, we have tried combining the best performing LSTM layer with 32 hidden nodes with the convolutional layers. However, as more convolutional layers were added, performance dropped linearly. The underlying cause must be the same as in the case above – the network might not be able to learn useful weights.

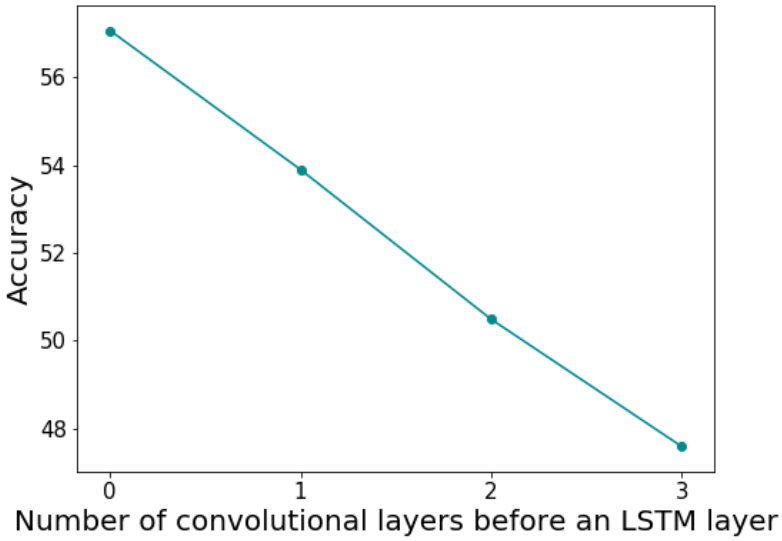


Figure 5.11: Accuracy of the bidirectional LSTM depending on the number of convolutional layers preceding it.

5.8 Summary

The best model trained on raw features obtained the accuracy of 57.07% on the test set which is 4% less than the model trained on the knowledge-driven features (section 4.5).The bidirectional LSTM with 32 hidden nodes is the best performing model which shows that being able to access full length of the utterance helps improves the performance in emotion recognition, similarly to speech recognition(Graves and Schmidhuber, 2005).

From the results presented above, we have noticed that, although we have experimented with different network architectures, the networks trained on raw features cannot obtain the results close to those with knowledge-driven features. After we compare our results to the results in previous research, we will explore what could be the underlying reason for it.

Chapter 6

Comparison

The results presented in chapters 4 and 5 are summarised in table 6.1. We can observe from the table that the performance on the spectrograms is dropping as the number of trainable parameters increases. We assume that it is due to undertraining.

Table 6.1: Comparison of the results presented in chapters 4 and 5

Features	Model	Accuracy
eGeMAPS	FNN	61.80%
Spectrogram	LSTM	57.07%
Spectrogram	CNN	56.01%
Spectrogram	1 layer CNN + LSTM	53.90%
Spectrogram	2 layers CNN + LSTM	50.49%
Spectrogram	3 layers CNN + LSTM	47.59%

Table 6.2 presents the comparison of the best results in experiments to the results in prior work. It is important to notice that not all results are directly comparable due to several reasons. The main one is that there is no standard separation of IEMOCAP dataset into training, validation and test sets. Some papers train their models using cross-validation (as in this project) while others do not use it. Moreover, in some papers (e.g., (Gideon et al., 2017)) the utterances labelled with 'excited' are included into the 'happy' emotion changing the balance of the dataset and amount of training data available.

Moreover, Lee and Tashev (2015) use only improvised sentences in their experiments which reduces the variability of the dataset. As Neumann and Vu (2017) have demonstrated, when using only one type of recording(only scripted or improvised), the models demonstrate better performance than if it is trained and evaluated on the full dataset.

Gideon et al. (2017) only use the unweighted average recall(UAR) as a metric of their performance which makes it not directly comparable to other results in the table. Because Gideon et al. (2017) do not explain how they calculate UAR for the multiclass problem, it is impossible to compare our result to theirs directly.

Table 6.2: Comparison of best results in the experiments (in bold) with previous results in the literature
Abbreviations:
Δ – delta features showing how a feature changes across time; BLSTM – bidirectional LSTM; ELM – extreme learning machine; FNN – feedforward neural network; RNN - recurrent neural network; ABS2S – attention-based sequence to sequence model.

Paper	Input features	Models	Accuracy
Ghosh et al. (2016)	Spectrogram	Autoencoder, BLSTM	49.75%
	Spectrogram	**BLSTM**	**57.07%**
Han et al. (2014)	F_0 (pitch), voice probability, zero-crossing rate, 12 MFCCs with log energy, and their Δ	FNN + ELM	57.91%
Latif et al. (2017)	80 log Mel frequency filterbanks	Variational autoencoders + LSTM	58.1%
Tao and Liu (2017)	ComParE10 + sequential acoustic features	RNN + FNN	58.7%
Satt et al. (2017)	Spectrogram(3 seconds)	5 convolutional layers and LSTM	59.4%
Zhang et al. (2017)	F_0 (pitch), voice probability, zero-crossing rate, 12 MFCCs with log energy, and their Δ + previous emotion	ABS2S + LSTM	60.8%
	eGeMAPS	**FNN**	**61.82%**
Lee and Tashev (2015)	F_0 (pitch), voice probability, zero-crossing rate, 12 MFCCs with log energy, and their Δ	BLSTM + ELM	62.85%
Gideon et al. (2017)	eGeMAPS	ProgNets	65.7% (UAR)

Chapter 7

Why do the raw features perform worse than the knowledge driven ones?

In this chapter, we attempt to answer the following question: why cant the networks trained on raw features perform on par with the networks trained on knowledge-driven features?

The first hypothesis is that available data is not sufficient for the algorithm to learn to extract optimal features. To test this, we will experiment with varied amount of training data and observe whether the reduction of training set size degrades the performance of the model significantly.

If the results of the first experiment are inconclusive, the difference should lie in the content of the features.

To study that, we will, firstly, check whether the eGeMAPS features are good enough that any, even the simplest classifier such as the K-Nearest Neighbors, would perform well with them or if the neural network is necessary to obtain the high performance. If so, we will study the weights of the network to determine which features are most important. Subsequently, we will attempt an abolishment experiment where different features will be taken out when training the feedforward network and we will observe lack of which of them degrades the models performance most. Finally, we will try to augment the raw features in such a way that the raw features have access to the same information as the network with the knowledge-driven features.

7.1 Amount of training data

The first reason which we will study is whether the available training data is not sufficient for the model to extract optimal features. When there is insufficient data, carefully chosen features have to be extracted while, with sufficient or excessive training data, we can use raw features and let the model learn optimal representations. If in our case the amount of data is insufficient, we should see the following trend when the amount of training data is reduced: the models performance degrades significantly on the raw features and only slightly for the knowledge-driven features. If so, we will be able to extrapolate that and predict how much data is needed for the model to learn optimal

features from raw data and perform similarly to the one trained on knowledge-driven features. We will train the models on the datasets including a quarter, a half and three quarters of the whole training set.

7.1.1 Experimental setup and implementation

For the experiment, we have used the eGeMAPS (as described in section 4.1) and spectrograms (as described in section 5.6). Similarly to preceding experiments, we have used 5-fold cross-validation where 4 sessions were used as the training set and the 5th was divided into validation and test set. In order to conduct the experiments, we had to create subsets of the training data containing 25%, 50% and 75% of the data. To do so, we have randomised the order of the utterances in training set and picked first 25%, 50% and 75%, respectively. As a result, the training sets consisted of 887, 1773, and 2660 utterances respectively.

The models used in this experiment are those which demonstrated the best performances in chapters 4 and 5. They are 2 layer FNN with 2048 x 16 architecture and 1 layer LSTM with 32 hidden nodes for eGeMAPS and raw features, respectively.

The models were implemented in Python (Van Rossum and Drake, 2011) using NumPy (Walt et al., 2011), Tensorflow (Abadi et al., 2016), Matplotlib (Hunter, 2007) and Scikit-Learn (Pedregosa et al., 2011) packages with the aid of Jupyter Notebooks (Kluyver et al., 2016).

7.1.2 Results

The fig. 7.1 presents the results of the experiment. As the fig. 7.1 shows, the performance of both models improves as more training data is used. The overall trend contradicts our hypothesis, i.e., as the size of training set decreases, the performance of the model trained on knowledge-driven features degrades faster than that of the model trained on raw features. We cannot extrapolate the results of the experiment and predict how much more data would be needed because the trends are not well pronounced. What can be concluded from the experiment is that if we had more training data the performance of the models trained on both feature sets could improve.

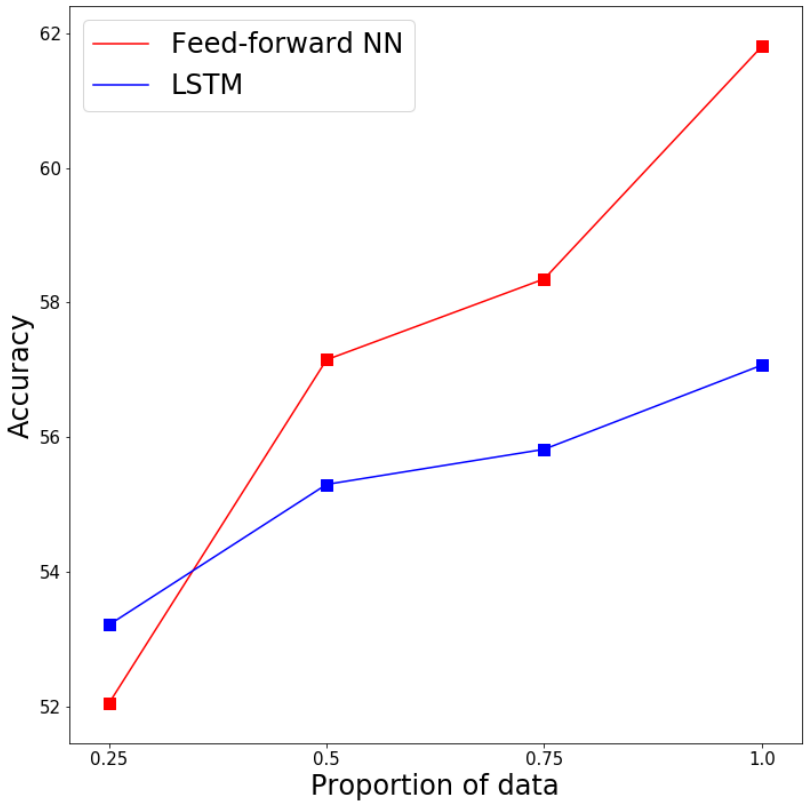


Figure 7.1: Performance of the FNN model trained on eGeMAPS and LSTM model trained on raw spectrograms depending on the proportion of available training data included in the training set

7.2 Classification by similarity of eGeMAPS

The assumption behind this experiment is that the core of the success of the model trained on eGeMAPS lies in the features themselves. eGeMAPS (described in section 4.1) were chosen by their potential to capture physiological state, established success in previous research and their significance in theory (Eyben et al., 2016). Thus, they are heavily knowledge driven. To check that, we have trained the K-Nearest neighbours classifiers (kNN) which are even simpler than the feed-forward networks as kNNs are non-parametric classifiers, i.e., they do not have weights to be learnt.

7.2.1 k Nearest Neighbours

The idea behind the K-nearest neighbours is to create a majority voting between K most similar cases in the training set to the unseen instance from the test set. The similarity is defined using a distance metric. In our experiments, we have used the Euclidean distance:

$$d(x,x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_n - x'_n)^2}$$

To classify an unseen instance x, given a positive integer K and similarity metric d, the K-Nearest Neighbors classifier acts as follows. Firstly, it computes the distance

between the instance x and every instance in the training set. We will call K points in the training set closest to x $\mathbb{A}$. Then, the conditional probability of each class is estimated. It is estimated as the fraction of the points in $\mathbb{A}$ which are elements of the class:

$$P(j|x) = \frac{1}{K} \sum_{i \in \mathbb{A}} I(y^{(i)} = j)$$

where I returns 1 its argument is true and 0 if false.

Then, the input x is assigned to the class with the largest probability.

7.2.2 Experimental setup and implementation

The eGeMAPS features were used as the input into the classifier. We have trained the classifier using 5-fold cross-validation. 4 sessions(8 speakers) were used as the training data and the 5th one was divided into the validation and test set. We have switched whether the male speaker was in validation and the female one in the test set or vice versa not to introduce biases.

The best model was chosen using the grid search to find optimal K. We have searched through the values between 4 and 512.

The models were implemented in Python (Van Rossum and Drake, 2011) using NumPy (Walt et al., 2011), Matplotlib (Hunter, 2007) and Scikit-Learn (Pedregosa et al., 2011) packages with the aid of Jupyter Notebooks (Kluyver et al., 2016).

7.2.3 Results

As shown in 7.2, the best accuracy that the system achieves is 49%. It is better than both random predictor and most probable predictor, which means that the similarity in eGeMAPS aids to determine the class of each utterance. However, the best model was 10 percent worse than the FNN trained on eGeMAPS. Therefore, the network must be learning some more useful weights than just the similarity of the features. In the next experiment, we will investigate the weights of the feedforward network.

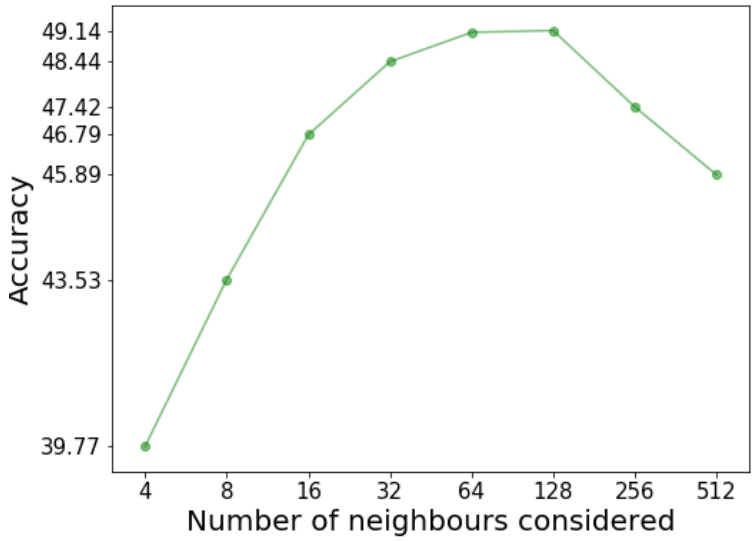


Figure 7.2: Accuracy of the kNN classifiers depending on the number of neighbours considered during classification

7.3 Importance of features

The eGeMAPS features are devised for voice analysis such as paralinguistic or clinical analysis (Eyben et al., 2016). Thus, not all of them are equally important for emotion recognition. Here, we aim to investigate the importance of different features for emotion classification. To find out how important a feature is, we will use Olden's algorithm (Olden and Jackson, 2002) which was created to devise the importance of each feature from the weights of the network.

7.3.1 Olden's algorithm

Olden's algorithm (Olden and Jackson, 2002) is a generalisation of Garson's algorithm (Garson, 1991) in which the neural network's weights determine the importance of each feature.

In Olden's algorithm, the importance of each feature is determined as the product of connection weights between each input and output neuron summed across all the hidden units.

Olden's algorithm (unlike the Garson's algorithm) can be used to evaluate feature importance in networks with several hidden layers. Moreover, the contribution of each weight is maintained in terms of magnitude and sign in contrast with Garson's algorithm which only considers the absolute magnitude.

7.3.2 Experimental setup and implementation

As the input to the algorithm, we have used the weights of the optimal network from 4.5. Using those, we have explored the importance of 88 eGeMAPS features.

Table 7.1: Most important features by Olden's algorithm(Olden and Jackson, 2002)

Feature	Group
Spectral Flux (mean, standard deviation, mean and standard deviation in voiced regions)	Spectral
Loudness (standard deviation, mean and standard deviation of rising and falling slopes)	Voice source (computed from spectral features)
F_0 (mean, standard deviation, standard deviation of rising slopes)	F_0 (voice source)
4th MFCC	Spectral
Alpha ratio (standard deviation)	Spectral
Jitter (standard deviation)	F_0 (voice source)

The algorithm was implemented in Python (Van Rossum and Drake, 2011) using NumPy (Walt et al., 2011) package with the aid of Jupyter Notebooks (Kluyver et al., 2016).

7.3.3 Results

The output of Olden's algorithm shows that in 20 most important features, the majority are spectral features and features characterising voice quality (see table 7.1).

These results support our results in section 5.7. The models trained on spectrograms demonstrated performance well above the baseline. Thus, useful spectral features for emotion classification were learnt. Therefore, the question is: which important features cannot be extracted from the spectrograms? Out of the features with high importance in table 7.1, the information which lacks from the spectrograms in 5.7 is F_0 because we used the wideband spectrograms where the information about formants (and fundamental frequency) is lost.

Further, we will explore how the removal of features extracted from F_0 from the eGeMAPS features influences the performance of the network and whether the addition of F_0 features to spectrograms improves the performance of the network.

7.4 Feature selection experiments

We assume that the models trained on spectrograms perform worse than feedforward neural network trained on the eGeMAPS features because the former cannot access F_0. To test this hypothesis, we will remove the three groups of features one by one, namely, the spectral, voice source and energy features. We expect the model's performance to drop to approximately 57% when the voice source features are removed.

7.4.1 Experimental setup

In this experiment, we will use the model with the 2048-16 architecture with optimal settings (as determined in section 4.5). It is trained with three feature sets. The first

feature set excluded the voice source features (F_0, jitter and formants). The second one excluded the energy features (shimmer, harmonics to noise ratio and loudness). The third excluded the spectral features (first 4 MFCCs and spectral flux).

The models were implemented in Python (Van Rossum and Drake, 2011) using NumPy (Walt et al., 2011), Tensorflow (Abadi et al., 2016), Matplotlib (Hunter, 2007) and Scikit-Learn (Pedregosa et al., 2011) packages with the aid of Jupyter Notebooks (Kluyver et al., 2016).

7.4.2 Results

The results are presented in table 7.2. As we can see, the results support our hypothesis that the performance of the network with the voice source feature removed will drop to approximately 57% (the level of models trained on raw spectrograms). This means that having access to F_0 is crucial for accurate emotion recognition.

However, in the experiment, there is a confounding factor. Each group of features is represented by a different number of features, therefore, different number of weights needs to be trained. Thus, another approach will be attempted – training the models on narrower-band spectrogram and augmenting the wideband spectrogram with the F_0 extracted at every 10 ms in the process of constructing the eGeMAPS by openSMILE (Eyben et al., 2013).

Table 7.2: Performance of the model depending on the features removed

Type of features	Voice source	Energy	Spectral features
Number of features taken out	34	15	35
Accuracy	56.9 %	59.19 %	58 %

7.5 Wideband spectrogram

To leverage the influence of a different number of features in the experiment above, we have experimented with the spectrograms extracted differently – with bigger windows to obtain better formant information.

7.5.1 Experimental setup

In this case, the best convolutional network architecture was trained on narrower band spectrograms. In this case, the window size was 50 ms(twice as big as in section 5.6).

The convolutional network had 2 hidden layers, the kernel size of 10 and 32 channels in each layer.

The models were implemented in Python (Van Rossum and Drake, 2011) using NumPy (Walt et al., 2011), Tensorflow (Abadi et al., 2016), Matplotlib (Hunter, 2007) and Scikit-Learn (Pedregosa et al., 2011) packages with the aid of Jupyter Notebooks (Kluyver et al., 2016).

7.5.2 Results

When the wideband spectrograms were used as input, the performance of the network has has gone from 56.1% reported in section 5.7 to 58%.

The result is an improvement. Thus, the spectrograms with better frequency resolution improve the performance of the emotion recognition models. We assume that the improvement is observed since from spectrograms with better frequency resolution information about F_0 can be extracted. However, the improvement is not as large as expected. This might be caused by the noise which is stored in a separate feature in eGeMAPS features. To explore this, we have experimented with augmenting raw spectrograms with F_0 extracted to obtain eGeMAPS features.

7.6 Augmentation of spectrogram with F_0 feature

As shown in 7.5, when the model has access to raw features from which the F_0 can be extracted, its performance improves. The spectrogram on which the model was trained above was extracted with a slightly wider window (50 ms vs. 25 ms in 5). In order to verify that as the window gets wider the performance of the model improves, we would need to do a grid search across the window sizes and all the model parameters which is not feasible given the time constraints. Thus, as an alternative, we will try augmenting the raw spectrogram with the F_0 extracted for eGeMAPS.

In tables 4.1 and 4.2 describe the low level descriptors used in eGeMAPS. In the experiment, only F_0 extracted every 10 ms will be used.

We expect that when we add F_0, the performance will improve as F_0 (pitch) was shown to be indicative of the emotional state in human perception (Murray and Arnott, 1993).

7.6.1 Experimental setup

As the input, we have used the spectrograms with 25 ms window and 40 frequency bands (as used in 5). To incorporate F0, we have taken the log of F_0 which was extracted every 10 ms. Both of the features had varying length depending on the length of each utterance. For example, for the utterance which is 5 seconds long, the vector of F_0s has length 500 and the spectrogram has shape 331×40 where 331 corresponds to time steps and 40 corresponds to frequency bands. This difference between the input vector lengths as well as them depending on the utterance length presented a difficulty in using both features as input.

To capture that, we have implemented the LSTM-based model as shown in fig. 7.3. The bidirectional LSTM network 1 processes the F_0 while LSTM 2 processes spectrogram. Their outputs (64 dimensional vector for each) are concatenated. On top of the concatenation, a fully connected layer with 16 units is applied the output of which is fed into another one to provide probability distribution across the 4 classes.

Each bidirectional layer has 32 hidden units. We have used Adam optimizer with the learning rate of 10^{-4}.

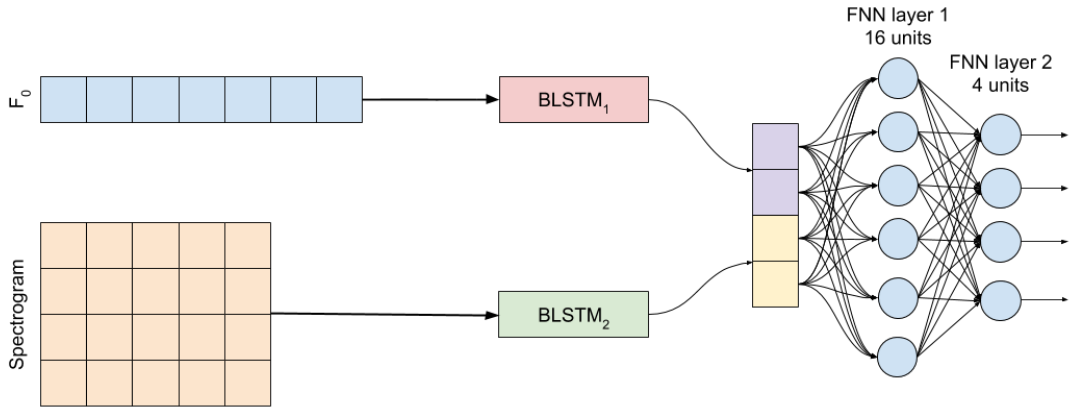


Figure 7.3: Architecture of the model used to augment spectrograms with F_0 as input features

7.6.2 Results

The model where the spectrogram was augmented with F_0 performed better than the model without it with 60.2% accuracy. It has demonstrated the best performance out of the models trained on the spectrograms. Moreover, as shown in fig. 7.4, the gap between performance on the raw and knowledge-driven features was reduced more than twice. This result supports our hypothesis that the comparatively bad result of raw features was grounded in the fact that it was impossible for the models to extract F_0.

7.7 Summary

The experiments presented in this chapter were concerned with exploring why knowledge driven features performed better than the raw ones. Firstly, the study of the amount of training data available gave inconclusive results showing that performance of both models improves with a similar rate as the size of training set grows. Secondly, we went into studying the features themselves. Having checked that, even using eGeMAPS, the similarity is not sufficient to produce good classification results, we investigated the weights of each feature. Out of the most important features, the ones which cannot be directly calculated from spectrogram were those calculated from F_0. Then, we did abolishment experiments where we excluded features by the group (spectral, voice source, energy). The biggest drop in performance was observed when the voice source features were removed. Thus, we experimented with spectrograms with better frequency resolution and tried augmenting the spectrograms with F_0. In both cases, the performance of the model improved supporting the idea that F_0 is required for improved emotion recognition results. This means that we have proved the importance of F_0 in two ways, by removing from knowledge-driven features and by adding to the raw features.

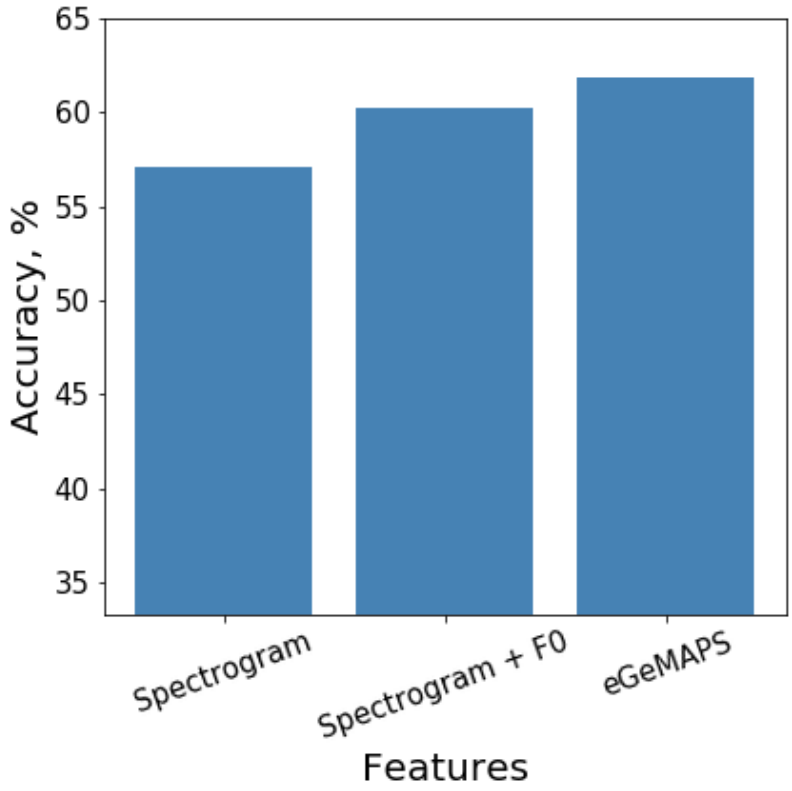


Figure 7.4: Barchart comparing the best results with eGeMAPS, spectrograms and spectrograms augmented with F_0

Chapter 8

Conclusions and future work

8.1 Conclusions

In the project, the work was devoted to emotion recognition using different input features and studying the difference between them. The summary of the best results in the project is presented in table 8.1. The overall best result was obtained with eGeMAPS features and feedforward neural network while the best result on raw features was obtained when F_0 was added to spectrograms explicitly. This result, together with the results of the abolishment experiments on eGeMAPS and the improved result on wideband spectrograms, has demonstrated that F0 is a crucial feature for emotion recognition. Interestingly, this result corresponds to the result of the psychological study(Lin et al., 1999; Murray and Arnott, 1993) which has shown that pitch is a key factor for emotion recognition in humans.

Table 8.1: Comparison of the results presented in throughout the project

Features	Model	Accuracy
eGeMAPS	FNN	61.80%
Spectrogram + F_0	LSTM	60.20%
Wideband spectrogram	LSTM	58.11%
Spectrogram	LSTM	57.07%
Spectrogram	CNN	56.01%
Spectrogram	1 layer CNN + LSTM	53.90%
Spectrogram	2 layers CNN + LSTM	50.49%
Spectrogram	3 layers CNN + LSTM	47.59%

8.2 Future work

Further work will include improving the results of each of the networks by further tuning their parameters which were not yet experimented with due to limitations of time and computational resources.

To compare the performance of the model presented in chapter 4 to current state-of-art performance, further experiments would include replicating the results presented

in Gideon et al. (2017) and computing accuracy rather than unweighted average recall or specifying with the authors the formula they used for unweighted average recall.

In order to improve the performance of the models presented in chapter 5, more detailed grid search would need to be conducted. For convolutional neural networks, the experiments could include different learning rate schedules or more extensive grid search through number of channels in each layer. For LSTM, the experiment with different number of hidden layers was not undertaken due to time constraints, so, it would be the subsequent step.

The results presented in chapter 7 could be improved dramatically through more careful choice of parameters. The result with wideband spectrograms (section 7.5) could be greatly improved by searching through different widths of the window with which the spectrograms are extracted. As well as that, the experiments could be conducted to determine the optimal number of layers and units and best learning rate. It is important to note that the experiment presented in section 7.5 was a proof of concept, so, the model was not well tuned.

When optimal window size and network architecture are determined, we can build a network which can take both optimal wide- and narrowband spectrograms as input. As shown in Mao et al. (2014), when passing the 2 kinds of spectrograms in two separate networks and combining their predictions, emotion recognition improves. In Mao et al. (2014), CNNs were used. In our case, we would use bidirectional LSTMs as they have demonstrated best results. Thus, we would have one bidirectional LSTM processing wideband spectrogram and the other one processing narrowband spectrogram.

Appendix A

Appendix

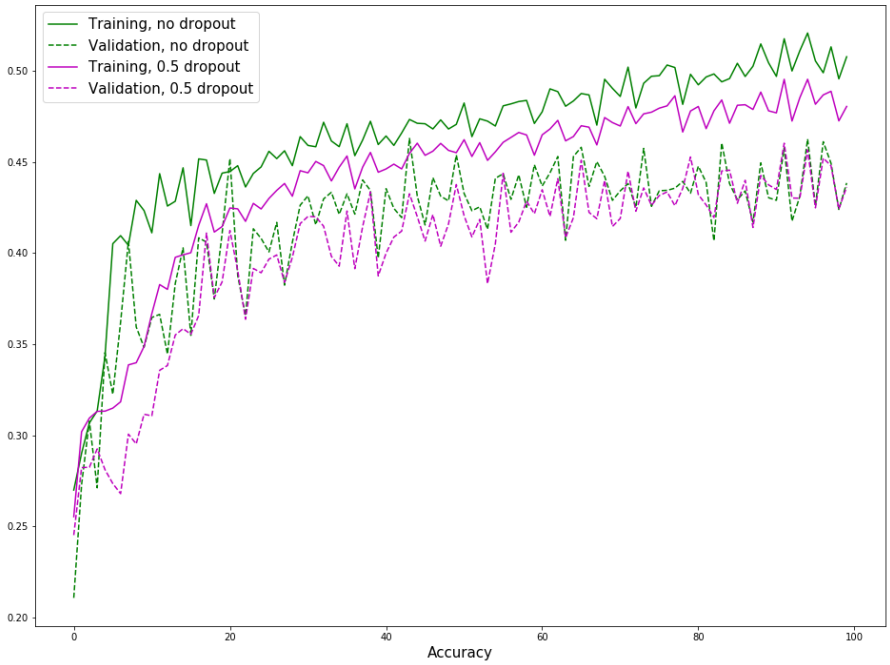


Figure A.1: Performance of the bidirectinal LSTM with 512 hidden units with and without dropout

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Anagnostopoulos, C.-N. and Iliou, T. (2010). Towards emotion recognition from speech: definition, problems and the materials of research. In *Semantics in Adaptive and Personalized Services*, pages 127–143. Springer.

Anagnostopoulos, C.-N., Iliou, T., and Giannoukos, I. (2015). Features and classifiers for emotion recognition from speech: a survey from 2000 to 2011. *Artificial Intelligence Review*, 43(2):155–177.

Anagnostopoulos, C. N. and Vovoli, E. (2009). Sound processing features for speaker-dependent and phrase-independent emotion recognition in berlin database. In *Information systems development*, pages 413–421. Springer.

Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W. F., and Weiss, B. (2005). A database of german emotional speech. In *Ninth European Conference on Speech Communication and Technology*.

Busso, C., Bulut, M., Lee, C.-C., Kazemzadeh, A., Mower, E., Kim, S., Chang, J. N., Lee, S., and Narayanan, S. S. (2008). Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335.

Busso, C., Parthasarathy, S., Burmania, A., AbdelWahab, M., Sadoughi, N., and Provost, E. M. (2017). Msp-improv: An acted corpus of dyadic interactions to study emotion perception. *IEEE Transactions on Affective Computing*, 8(1):67–80.

Cortina, J. M. (1993). What is coefficient alpha? an examination of theory and applications. *Journal of applied psychology*, 78(1):98.

Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., and Taylor, J. G. (2001). Emotion recognition in human-computer interaction. *IEEE Signal processing magazine*, 18(1):32–80.

Cowie, R., Wichmann, A., Douglas-Cowie, E., Hartley, P., and Smith, C. (1999). The prosodic correlates of expressive reading. In *Proceedings of the 14th ICPhS, San Francisco*, pages 1–7.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *psychometrika*, 16(3):297–334.

Dai, W., Dai, C., Qu, S., Li, J., and Das, S. (2017). Very deep convolutional neural networks for raw waveforms. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 421–425. IEEE.

Douglas-Cowie, E., Campbell, N., Cowie, R., and Roach, P. (2003). Emotional speech: Towards a new generation of databases. 40:33–60.

Douglas-Cowie, E., Cowie, R., and Schröder, M. (2000). A new emotion database: considerations, sources and scope. In *ISCA tutorial and research workshop (ITRW) on speech and emotion*.

Douglas-Cowie, E., Cowie, R., Sneddon, I., Cox, C., Lowry, O., Mcrorie, M., Martin, J.-C., Devillers, L., Abrilian, S., Batliner, A., et al. (2007). The humaine database: addressing the collection and annotation of naturalistic and induced emotional data. In *International conference on affective computing and intelligent interaction*, pages 488–500. Springer.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Engberg, I. S. and Hansen, A. V. (1996). Documentation of the danish emotional speech database des. *Internal AAU report, Center for Person Kommunikation, Denmark*, page 22.

Eyben, F., Scherer, K. R., Schuller, B. W., Sundberg, J., André, E., Busso, C., Devillers, L. Y., Epps, J., Laukka, P., Narayanan, S. S., et al. (2016). The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing. *IEEE Transactions on Affective Computing*, 7(2):190–202.

Eyben, F., Weninger, F., Gross, F., and Schuller, B. (2013). Recent developments in opensmile, the munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 835–838. ACM.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Fontaine, J. R. J., Scherer, K. R., Roesch, E. B., and Ellsworth, P. C. (2007). The world of emotions is not two-dimensional. *Psychological Science*, 18(12):1050–1057.

Gales, M., Young, S., et al. (2008). The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304.

Garson, G. D. (1991). Interpreting neural-network connection weights. *AI expert*, 6(4):46–51.

Ghosh, S., Laksana, E., Morency, L.-P., and Scherer, S. (2016). Representation learning for speech emotion recognition. In *INTERSPEECH*, pages 3603–3607.

Gideon, J., Khorram, S., Aldeneh, Z., Dimitriadis, D., and Provost, E. M. (2017). Progressive neural networks for transfer learning in emotion recognition. *arXiv preprint arXiv:1706.03256*.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Han, K., Yu, D., and Tashev, I. (2014). Speech emotion recognition using deep neural network and extreme learning machine. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95.

Jin, Q., Li, C., Chen, S., and Wu, H. (2015). Speech emotion recognition with acoustic and lexical features. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4749–4753. IEEE.

Johnson, W. F., Emde, M., and Klinnert, M. D. (1986). Recognition of emotion from vocal cues. *Arch Gen Psychiatry*, 43:280–283.

Jurafsky, D. and Martin, J. H. (2014). *Speech and language processing*, volume 3. Pearson London.

Kim, J., Englebienne, G., Truong, K. P., and Evers, V. (2017). Towards speech emotion recognition" in the wild" using aggregated corpora and deep multi-task learning. *arXiv preprint arXiv:1708.03920*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipp, M. (2001). Anvil-a generic annotation tool for multimodal dialogue. In *Seventh European Conference on Speech Communication and Technology*.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., et al. (2016). Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Latif, S., Rana, R., Qadir, J., and Epps, J. (2017). Variational autoencoders for learning latent representations of speech emotion. *arXiv preprint arXiv:1712.08708*.

Lee, J. and Tashev, I. (2015). High-level feature representation using recurrent neural network for speech emotion recognition.

Lin, X., Chen, Y., Lim, S., and Lim, C. (1999). Recognition of emotional state from spoken sentences. In *Multimedia Signal Processing, 1999 IEEE 3rd Workshop on*, pages 469–473. IEEE.

Mao, Q., Dong, M., Huang, Z., and Zhan, Y. (2014). Learning salient features for speech emotion recognition using convolutional neural networks. *IEEE Transactions on Multimedia*, 16(8):2203–2213.

Martin, O., Kotsia, I., Macq, B., and Pitas, I. (2006). The enterface05 audio-visual emotion database. In *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pages 8–8. IEEE.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25.

Mozer, M. C. (1995). A focused backpropagation algorithm for temporal. *Backpropagation: Theory, architectures, and applications*, page 137.

Murray, I. R. and Arnott, J. L. (1993). Toward the simulation of emotion in synthetic speech: A review of the literature on human vocal emotion. *The Journal of the Acoustical Society of America*, 93(2):1097–1108.

Mustafa, M. B., Yusoof, M. A., Don, Z. M., and Malekzadeh, M. (2018). Speech emotion recognition research: an analysis of research focus. *International Journal of Speech Technology*, pages 1–20.

Neumann, M. and Vu, N. T. (2017). Attentive convolutional neural network based speech emotion recognition: A study on the impact of input features, signal length, and acted speech. *arXiv preprint arXiv:1706.00612*.

Nilsonne, A. (1988). Speech characteristics as indicators of depressive illness. *Acta Psychiatrica Scandinavica*, 77(3):253–263.

Nwe, T. L., Foo, S. W., and De Silva, L. C. (2003). Classification of stress in speech using linear and nonlinear features. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–9. IEEE.

Olden, J. D. and Jackson, D. A. (2002). Illuminating the black box: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling*, 154(1-2):135–150.

Palaz, D., Magimai.-Doss, M., and Collobert, R. (2015). Analysis of cnn-based speech recognition system using raw speech as input. Technical report, Idiap.

Parthasarathy, S. and Busso, C. (2017). Jointly predicting arousal, valence and dominance with multi-task learning. *INTERSPEECH, Stockholm, Sweden*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Pham, V., Bluche, T., Kermorvant, C., and Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 285–290. IEEE.

Picard, R. W. (1995). Affective computing.

Plutchik, R. (1984). Emotions: A general psychoevolutionary theory. *Approaches to emotion*, 1984:197–219.

Ringeval, F., Sonderegger, A., Sauer, J., and Lalanne, D. (2013). Introducing the recola multimodal corpus of remote collaborative and affective interactions. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–8. IEEE.

Russell, J. (1980). A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161–1178.

Satt, A., Rozenberg, S., and Hoory, R. (2017). Efficient emotion recognition from speech using deep learning on spectrograms. *Proc. Interspeech 2017*, pages 1089–1093.

Sauter, D. A., Eisner, F., Calder, A. J., and Scott, S. K. (2010). Perceptual cues in nonverbal vocal expressions of emotion. *Quarterly Journal of Experimental Psychology*, 63(11):2251–2272.

Schuller, B., Müller, R., Lang, M., and Rigoll, G. (2005). Speaker independent emotion recognition by early fusion of acoustic and linguistic features within ensembles. In *Ninth European Conference on Speech Communication and Technology*.

Schuller, B. and Rigoll, G. (2009). Recognising interest in conversational speech-comparing bag of frames and supra-segmental features. In *Tenth Annual Conference of the International Speech Communication Association*.

Schuller, B., Steidl, S., and Batliner, A. (2009). The interspeech 2009 emotion challenge. In *Tenth Annual Conference of the International Speech Communication Association*.

Schuller, B., Steidl, S., Batliner, A., Bergelson, E., Krajewski, J., Janott, C., Amatuni, A., Casillas, M., Seidl, A., Soderstrom, M., et al. (2017). The interspeech 2017 computational paralinguistics challenge: Addressee, cold & snoring. In *Computational Paralinguistics Challenge (ComParE), Interspeech 2017*, pages 3442–3446.

Schuller, B., Steidl, S., Batliner, A., Burkhardt, F., Devillers, L., Müller, C., and Narayanan, S. (2010). The interspeech 2010 paralinguistic challenge. In *Proc. INTERSPEECH 2010, Makuhari, Japan*, pages 2794–2797.

Schuller, B., Steidl, S., Batliner, A., Epps, J., Eyben, F., Ringeval, F., Marchi, E., and Zhang, Y. (2014). The interspeech 2014 computational paralinguistics challenge: Cognitive & physical load. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Schuller, B., Steidl, S., Batliner, A., Hantke, S., Hönig, F., Orozco-Arroyave, J. R., Nöth, E., Zhang, Y., and Weninger, F. (2015). The interspeech 2015 computational paralinguistics challenge: nativeness, parkinson's & eating condition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Schuller, B., Steidl, S., Batliner, A., Nöth, E., Vinciarelli, A., Burkhardt, F., Son, R. v., Weninger, F., Eyben, F., Bocklet, T., et al. (2012). The interspeech 2012 speaker trait challenge. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Schuller, B., Steidl, S., Batliner, A., Schiel, F., and Krajewski, J. (2011). The interspeech 2011 speaker state challenge. In *Twelfth Annual Conference of the International Speech Communication Association*.

Schuller, B., Steidl, S., Batliner, A., Vinciarelli, A., Scherer, K., Ringeval, F., Chetouani, M., Weninger, F., Eyben, F., Marchi, E., et al. (2013). The interspeech 2013 computational paralinguistics challenge: social signals, conflict, emotion, autism. In *Proceedings INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France*.

Schuller, B. W., Steidl, S., Batliner, A., Hirschberg, J., Burgoon, J. K., Baird, A., Elkins, A. C., Zhang, Y., Coutinho, E., and Evanini, K. (2016). The interspeech 2016 computational paralinguistics challenge: Deception, sincerity & native language. In *Interspeech*, pages 2001–2005.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). Lstm neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Tao, F. and Liu, G. (2017). Advanced lstm: A study about better time dependency modeling in emotion recognition. *arXiv preprint arXiv:1710.10197*.

Tracy, J. L., Randles, D., and Steckler, C. M. (2015). The nonverbal communication of emotions. *Current opinion in behavioral sciences*, 3:25–30.

Van Rossum, G. and Drake, F. L. (2011). *An introduction to Python*. Network Theory Ltd.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Wang, Y., Du, S., and Zhan, Y. (2008). Adaptive and optimal classification of speech emotion recognition. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 5, pages 407–411. IEEE.

Williams, C. E. and Stevens, K. N. (1972). Emotions and speech: Some acoustical correlates. *The Journal of the Acoustical Society of America*, 52(4B):1238–1250.

Xia, R. and Liu, Y. (2017). A multi-task learning framework for emotion recognition using 2d continuous space. *IEEE Transactions on Affective Computing*, 8(1):3–14.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (2002). The htk book. *Cambridge university engineering department*, 3:175.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zhang, R., Atsushi, A., Kobashikawa, S., and Aono, Y. (2017). Interaction and transition model for speech emotion recognition in dialogue. *Proc. Interspeech 2017*, pages 1094–1097.