TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Mensch-Maschine-Kommunikation

Application of Deep Learning Methods in Computational Paralinguistics

Raymond Christian Brückner

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. sc. techn. Andreas Herkersdorf
Prüfer der Dissertation:	1. Prof. Dr.-Ing. habil. Björn W. Schuller
	2. Prof. Dr. rer. nat. Jakob Macke

Die Dissertation wurde am 11.04.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 20.01.2020 angenommen.

Zusammenfassung

In dieser Arbeit wird das Potenzial aktueller *Deep Learning*-Methoden zur Lösung anspruchsvoller Aufgaben in der Computerlinguistik (CL) untersucht, u.a. die Extraktion und Prädiktion von Sprecherzuständen aus dem akustischen Sprachsignal. Anhand von vier verschiedenen paralinguistischen Problemstellungen werden verschiedene Ansätze exemplarisch behandelt, wobei eine Reihe unterschiedlicher Architekturen tiefer neuronaler Netzwerke verwendet wird und deren Wirksamkeit nachgewiesen wird. Die vorstellten Ansätze erzielen Ergebnisse nach dem neuesten Stand der Technik und übertreffen diese teilweise. Zunächst wird die erste bekannte Anwendung tiefer neuronaler Netzwerke auf das Problem der *Sympathieerkennung* in Sprache vorgestellt, wobei flache Feed-Forward-Netzwerken und unüberwachtes RBM-Vortraining angewandt werden. Danach wird das Problem der Detektion *sozialer Signale* in Sprache durch die Kombination konventioneller Feed-Forward-Netzwerke mit rekurrenten Netzwerkvarianten behandelt. Desweiteren wird ein neuer und hocheffektiver Ansatz zur Glättung der a-posteriori Wahrscheinlichkeiten durch Verwendung von tiefen, gestapelten Netzwerken eingeführt und auf mono- und crosslingualen Aufgaben evaluiert. Zusätzlich wird ein erster Vergleich ressourcenschonender rekurrenter Modelle mit konventionellen rekurrenten Modellen präsentiert. Im Anschluss wird die *Detektion von Konflikten* in spontansprachlicher Unterhaltung zwischen mehreren Teilnehmern untersucht. Es wird ein neuartiges Verfahren zur Prädiktion eines hochinformativen Merkmals für dieses Problem vorgeschlagen. Kombiniert mit einem prosodischen Merkmalssatz übertrifft man damit alle Ergebnisse der Beiträge, die zu diesem Teilproblem der Interspeech 2013 Computational Paralinguistics Challenge eingereicht wurden. Zuletzt wird ein rekurrentes Faltungsnetzwerk zur *Prädiktion von Emotion* in menschlicher Spontansprache eingeführt, welches Ende-zu-Ende direkt auf Sprachsignalen trainiert wird und zeitkontinuierliche Vorhersagen über das Niveau von Erregung und Wertigkeit trifft. Weiterhin wird vorgeschlagen, die Optimierung direkt mittels des Konkordanz-Korrelations-Koeffizienten durchzuführen. Es wird gezeigt, dass die resultierenden Modelle Zellen enthalten, deren Aktivierungen stark mit bekannten prosodischen Merkmalen korreliert sind.

Abstract

This thesis investigates the potential of deep learning methods for the challenging task of computational paralinguistics (CP), i. e. the extraction and prediction of speaker states and traits from acoustic manifestations of speech. On the basis of four different paralinguistic tasks various approaches are exemplified, leveraging numerous deep neural network architectures and demonstrating their effectiveness applied to acoustic-based CP. The presented approaches are demonstrated to achieve or even outperform several existing state-of-the-art results. To start with, the first known application of deep neural networks to the problem of *likability classification* from speech is presented, using shallow feed-forward networks and unsupervised RBM pre-training. Second, the problem of detecting *social signals* from speech is addressed by combining conventional feed-forward with recurrent neural network variants. Further, a novel and highly effective posterior smoothing technique via deep, stacked networks is introduced and evaluated on mono-lingual and cross-lingual tasks. Finally, a first comparison of resource-efficient recurrent models to conventional recurrent models is presented. Next, *conflict detection* in spontaneous, multi-party conversations is investigated. In particular, a novel technique to predict a highly informative feature for this problem is presented, which, when combined with a conversational-prosodic feature set, outperforms all contributions to the Interspeech 2013 Computational Paralinguistics Challenge on this task. Finally, for the task of *predicting emotion* in human spontaneous speech a convolutional, recurrent neural network model is introduced, trained in a fully end-to-end fashion on unprocessed, raw speech signals. This model predicts the level of arousal and valence in a time-continuous manner. Further, the direct optimization of the proposed system on the concordance correlation coefficient (CCC) is suggested. It is found that the resulting models contain cells whose activations are highly correlated with well-known prosodic features.

Preface

This thesis is based on selected pre-publications made during my time as a PhD student in the Machine Intelligence & Signal Processing Group, Institute for Human-Machine Communication (MMK), Technische Universität München (TUM) in Munich, Germany. The selection is based on scientific relevance, covering a range of applications of deep learning methods to the field of computational paralinguistics. The first aim of this thesis is to make these pre-publications more accessible to the generally knowledgeable reader, who should be familiar with the basics of digital signal processing, neural networks, and machine learning. The second aim is to provide a broader view on the concepts of deep learning in computational paralinguistics than is present in the pre-publications. For this purpose, the pre-publications have been restructured into methodology and results, cast into a general research framework, and augmented by an introductory chapter and concluding remarks. Furthermore, the chapters on the mathematical background and methodology have been extended to provide a better understanding of the complexity of the problems and the provided solutions. Hopefully, this results in a publication that is informative and enjoyable to read. The interested reader can find references to the pre-publications and related literature throughout this thesis.

Munich,
Spring 2019 *Raymond C. Brückner*

Acknowledgment

This research was carried out in the role of an external PhD student at the Technical University Munich while I was a full-time employee at Nuance Communications Deutschland GmbH. Hence, I would like to thank my supervisor Univ.-Prof. Dr.-Ing. habil. Björn W. Schuller for giving me the opportunity to pursue my aspirations as an external PhD student, for his inspiration and motivation, and for his many helpful comments on scientific writing. Particular thanks also go to my friend and colleague Felix Weninger and his wonderful wife Yue Zhang for their exceptional support, their friendship, and their delicious hot pot. Furthermore, I want to express my gratitude to Maximilian Schmitt, Florian Pokorny, Erik Marchi, and Simone Hantke for their continuous support on allocating data and providing me with valuable input while writing this thesis. Besides, I would like to thank the co-authors of the pre-publications being collected in this thesis – in addition to the people named above and in alphabetical order, these are Elisabeth André, Fabien Ringeval, Jun Deng, Florian Lingenfelser, Mihalis Nicolaou, George Trigeorgis, Johannes Wagner, and Stefanos Zafeiriou.

I further want to thank my friend, colleague, and former supervisor at Nuance, Daniel Willett, who has always been a source of inspiration and support.

Finally, my warmest thanks go to my parents, who always believed in me and my abilities, and always supported me in pursuing my aspirations.

First and foremost, however, my deepest gratitude goes to my wonderful wife Claudia and my fantastic children Christian and Laura, for their unconditional love, support, patience, encouragement, and understanding during all these years. Without you, I would not have made it!

Publications

Journal Papers

- Florian Lingenfelser, Johannes Wagner, Jun Deng, **Raymond Brueckner**, Björn Schuller, and Elisabeth André, *Asynchronous and Event-Based Fusion Systems for Affect Recognition on Naturalistic Data in Comparison to Conventional Approaches*, IEEE Transactions on Affective Computing, vol. 9, no. 4, pp. 410–423, 2016.

Book Chapters

- **Raymond Brueckner** and Björn Schuller, *Be at Odds? Deep and Hierarchical Neural Networks for Classification and Regression of Conflict in Speech.* Springer International Publishing, 2015, ch. Conflict and Multimodal Communication: Social Research and Machine Intelligence, pp. 403–429.

Conference Papers

- **Raymond Brueckner**, Maximilian Schmitt, Maja Pantic, and Björn Schuller, *Spotting Social Signals in Conversational Speech over IP: A Deep Learning Perspective*, in Proceedings of the 18th Annual Conference of the International Speech Communication Association (INTERSPEECH). Stockholm, Sweden: ISCA, 2017, pp. 2371–2375.

- Florian Pokorny, Björn Schuller, Peter Marschik, **Raymond Brueckner**, Pär Nyström, Nicholas Cummins, Sven Bölte, Christa Einspieler, and Terje Falck-Ytter, *Earlier Identification of Children with Autism Spectrum Disorder: An Automatic Vocalisation-Based Approach*, in Proceedings of the 18th Annual Conference of the International Speech Communication Association (INTERSPEECH). Stockholm, Sweden: ISCA, 2017, pp. 309–313.

- George Trigeorgis, Fabien Ringeval, **Raymond Brueckner**, Erik Marchi, Mihalis Nicolaou, Björn Schuller, and Stefanos Zafeiriou, *Adieu Features? End-to-End Speech Emotion Recognition using a Deep Convolutional Recurrent Network*, in Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Shanghai, China: IEEE, 2016, pp. 5200–5204.
- **Raymond Brueckner** and Björn Schuller, *Social Signal Classification Using Deep BLSTM Recurrent Neural Networks*, in Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Florence, Italy: IEEE, 2014, pp. 4856–4860.
- **Raymond Brueckner** and Björn Schuller, *Hierarchical Neural Networks and Enhanced Class Posteriors for Social Signal Classification*, in Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). Olomouc, Czech Republic: IEEE, 2013, pp. 361–364.
- **Raymond Brueckner** and Björn Schuller, *Likability Classification - A Not So Deep Neural Network Approach*, in Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH). Portland, OR, USA: ISCA, 2012.

Earlier Conference Papers

- Martin Raab, Tobias Herbig, **Raymond Brueckner**, Rainer Gruhn, and Elmar Nöth, *Adaptation of Frequency Band Influence for Non-Native Speech Recognition*, in 19. Konferenz Elektronische Sprachsignalverarbeitung (ESSV), Frankfurt am Main, Germany, 2008, pp. 149–156.
- Jochen Schwenninger, **Raymond Brueckner**, Daniel Willett, and Marcus E. Hennecke, *Language Identification in Vocal Music*, in Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR), Victory, Canada, 2006, pp. 377–379.
- Ying Sun, Daniel Willett, **Raymond Brueckner**, Rainer Gruhn, and Dirk Bühler, *Experiments on Chinese Speech Recognition with Tonal Models and Pitch Estimation Using the Mandarin Speecon Data*, in Proceedings of the 7th Annual Conference of the International Speech Communication Association (INTERSPEECH). Pittsburgh, PA, USA: ISCA, 2006, pp. 1245–1248.
- Ronald Römer and **Raymond Brueckner**, *Vergleichende Untersuchungen zur Zuverlässigkeit von Pitch-kohärenten Merkmalen bei verschiedenen Störgeräuschen unter Verwendung der Aurora-2 Datenbasis*, in 17. Konferenz Elektronische Sprachsignalverarbeitung (ESSV), Freiberg, Germany, 2006.

- Daniel Vásquez, Rainer Gruhn, **Raymond Brueckner**, and Wolfgang Minker, *Comparing Linear Feature Space Transformations for Correlated Features*, in Proceedings of the 4th International Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems (PIT). Irsee, Germany: Springer, 2008, pp. 176–187.

- Daniel Willett, Franz Gerl, and **Raymond Brueckner**, *Discriminatively Trained Context-Dependent Duration-Bigram Models for Korean Digit Recognition*, in Proceedings of the 31st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Toulouse, France: IEEE, 2006, pp. 25–28.

Contents

Part I

Introduction

Chapter 1

Introduction

Imagination is the beginning of creation.
You imagine what you desire,
you will what you imagine
and at last you create what you will.

George Bernard Shaw

1.1 Motivation

The last decade has experienced a transformation in the way technology pervades each individual's life and society. This transformation has been caused in large part by the advancement of machine learning in general and deep learning in particular. Many technological areas have witnessed a paradigm shift from working solutions that had been developed over decades by skilled engineers and researchers to systems which are able to learn (at least part of) a problem in an automated fashion purely from data. This change has occurred ubiquitously, for example in automatic speech recognition (ASR), which until the advent of deep learning used to work with Hidden Markov Models (HMM), or object recognition, which employed carefully hand-crafted features to detect and extract, e. g. faces from pictures. Many of these previous approaches have now been superseded by modern machine learning algorithms, which learn the underlying structure from data. This paradigm shift has led to an explosion of ideas and approaches published in the research literature and it is foreseen that this trend will continue.

With the recent advancements of machine learning, many related applications have become realizable and have penetrated large markets in various industries, such as security, marketing, healthcare, automotive, law, retail, agriculture, and manufac-

turing. Furthermore, machine learning underlies the success of industry giants, such as Google, Apple, Facebook, and Amazon, and many emerging start-up companies. With the permeation of society by these technologies the public perception has also changed considerably, leading to a hype around and even a fear for *artificial intelligence* (AI). This term is exceptionally wide in scope and according to Moore [230] can be defined as "*... the science and engineering of making computers behave in ways that, until recently, we thought required human intelligence.*" This concept of *general AI* is expected to solve generic problems and make judgments under uncertainty. It is further supposed to reason, plan, learn, and to even be innovative and creative. However, this is in stark contrast to *narrow* or *weak AI*, which leverages the power of machine learning algorithms in rather limited, well-defined, technical areas. Machine learning currently can be seen as a branch of the latter, i. e. weak AI, and it relies on working with (ideally large) datasets, examining the data in order to find common underlying patterns. Deep learning in turn is a sub-branch of machine learning, mostly adopting deep neural network structures.

In recent years, deep learning has lead to unprecedented developments in ASR, Natural Language Understanding (NLU), and dialog user interfaces, and these improvements have resulted amongst other in an increasingly widespread dissemination of conversational virtual agents (VA), both consumer-level (e. g. Google Assistant, Apple Siri, or Amazon Echo) and industry-level (e. g. call center and automotive business sectors). And despite the fact that these solutions cover an increasing number of domains, they still lack many aspects of natural human-to-human communication. One major deficiency of these systems currently is that they fail to cope with non-verbal communication appropriately – a characteristic humans instead handle with ease and the subject of the study of paralinguistics. In a sense, virtual agents today are able to understand *what* is said, but not *how* things are said. The way semantics are embedded in humans speech, however, often carries important information: it may modify or reinforce the meaning of a sentence and hence lead to misinterpretations of ambiguous expressions. Moreover, paralingustic embeddings convey a multitude of information about the speaker to a potential listener (be it human or a machine), e. g. his/her emotional state, health condition, or biological traits, such as age, gender, or personality. Finally, social signal, e. g. laughters or fillers, mediate the semantic transmission in human communication. Therefore, if the proliferation and acceptance of intelligent virtual assistants and similar technology is to be enhanced in the future, the paralinguistic facets of automated machinery must be strengthened and reinforced, which is exactly what *Computational Paralinguistics* is concerned with.

Paralinguistic events manifest in human speech in two modalities, lexical and acoustic. The former overlaps with semantics to some degree, i. e. it is based on *what* a speaker says. This is usually handled by ASR and NLU systems, but in

the context of paralinguistics the semantics (i. e. content) of speech is used to infer some underlying characteristic or state of the speaker. On the other hand, the acoustic modality captures all auditory elements of speech, such as prosodic, spectral, voice quality, etc. To give an example a frustrated user might say: "Stupid system!" - the acoustic component hopefully gives some indication of the user's state of frustration in terms of, for example, the energy level of the acoustic signal. Additionally, however, the lexical content by all means indicates the speaker's discontent about the systems (non-)performance. In this thesis, the lexical constituent is ignored and instead all focus is directed towards the purely acoustic aspects of speech.

While working on this study, research on deep learning has enormously gained impetus and has spawned an increasing interest in applying this technology to aspects of computational paralinguistics. The great number of paralinguistic phenomena and the particular constraints they entail require dedicated investigations for each of the manifold paralinguistic tasks. As will be expounded in Chapter 2, these tasks strongly differ in both their subjectiveness of perception and their temporal localization. This might necessitate to adapt the employed feature set, network architecture, training algorithm, evaluation measure, etc., to the problem at hand. Hence, it is a central concern of this thesis to shed light on some of the yet unsolved issues in applying deep learning methods to computational paralinguistic prediction problems.

1.2 Contributions

To address the challenges listed above, this thesis contributes in the following aspects:

1. For the *likability classification* of speech recordings, a comparison of feed-forward neural networks is presented on the Speaker Likability Database (SLD), which is the official dataset of the Likability Sub-Challenge of the Interspeech 2012 Speaker Trait Challenge [291]. Furthermore, unsupervised Restricted Boltzmann Machine pre-training is proposed and demonstrated to yield substantial gains over the challenge baseline performance. The suggested approaches are the first known application of deep neural networks to this task.

2. Several effective methods for *detecting social signals* (i. e. 'laughter' and 'filler') in speech signals are introduced and evaluated on the SSPNet Vocalisation Corpus, the official dataset of the Social Signals Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge [292]. Unsupervised pre-training via a stacked autoencoder approach is proposed to significantly improve the performance of feed-forward neural network architectures. Moreover, a novel posterior smoothing technique by stacking multiple networks is introduced, which effectively smoothes the highly variable posterior trajectories of a base

network. This approach is extended to recurrent, BLSTM and hierarchical DNN-BLSTM network topologies and achieves state-of-the-art results on the SSPNet Vocalization Corpus. These findings are verified and extended to obtain the first results on the recent SEWA database [191] in the literature, for both mono-lingual and cross-lingual scenarios. Eventually, this study shares a first comparison of resource-efficient models to more conventional recurrent neural models on this task and demonstrates that the proposed models and approaches are highly effective.

3. For the paralinguistic task of *conflict detection* in spontaneous, multi-party conversations, several different feature sets are investigated using deep feed-forward DNNs. Classification and regression experiments are evaluated on the SSPNet Conflict Corpus, which was used in the Conflict Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge [292]. The application of pre-training is once more shown to be highly beneficial in a paralinguistic task. Moreover, a method for predicting the *overlap ratio* via a BLSTM is proposed and its performance is demonstrated to yield substantial gains. Finally, it is shown that a combination of this feature with a conversational-prosodic feature set outperforms the challenge baseline and the best challenge contributions on both the classification task and the regression task.

4. On the task of *predicting spontaneous emotion* in human speech a convolutional, recurrent neural network model (CNN-BLSTM) trained on unprocessed, raw speech signals in a fully end-to-end approach is proposed. This approach automatically learns intermediate representations that are utilized to predict the level of arousal and valence in a time-continuous manner on the audio part of the RECOLA database. The study shows that the proposed method achieves significantly higher performance compared to two baseline systems, evaluated on two paralinguistic frame-level feature sets, commonly used in paralinguistic tasks. Further, the direct optimization of the proposed system on the concordance correlation coefficient (CCC) is introduced, contrary to the commonly adopted MSE criterion, which leads to further performance improvements. As a final contribution, internal activations of the recurrent output BLSTM network are compared to acoustic-prosodic features, which are known to affect arousal and which are commonly used in computational paralinguistic tasks. It is found that certain cell activations are highly correlated with those prosodic features.

1.3 Thesis Outline

This thesis is structured as follows. Part II gives an extensive overview over the methodology and theoretical background underlying the experiments performed in

this work. Chapter 2 provides a structural definition of paralinguistics and discusses aspects of annotating and rating paralinguistic phenomena. Subsequently, Chapter 3 describes the mathematical background and practical aspects of common acoustic features used in paralinguistic studies. Chapter 4 expounds the theory and methods required for objectively evaluating the proposed algorithms. In particular, it covers evaluation metrics for classification and regression problems, touches upon some aspects of cross-validation and testing, and defines some common significance tests. Chapter 5 attempts to give a broad overview of the wide field of deep neural networks. After a short background on their biological origins and a brief taxonomy, the basic structure of neural networks is described, as well as more recent, advanced neural network models. Finally, it touches upon important aspects of training such models, both in a supervised and unsupervised fashion. Part III then presents a number of selected applications of deep neural networks to the problem of predicting and classifying paralinguistic events. It starts with the classification of likability in Chapter 6, a difficult task, given its subjective nature, and continues with Chapter 7, which presents a number of novel approaches and results regarding the detection of social signals embedded in the speech signal. In Chapter 8 the detection and prediction of conflict in speech is addressed, whereas Chapter 9 completes the series of experimental applications by presenting results of some first attempts of full end-to-end learning in computational paralinguistics. The thesis concludes with a summary of achievements and an outlook for promising research directions towards further improving the applicability and performance of deep learning in the paralinguistic cosmos.

Part II

Methodology And Theoretical Background

Chapter 2

Computational Paralinguistics

The most important thing in communication is hearing what isn't said.

Peter F. Drucker

This thesis focuses on the application of deep learning methods to the realm of *computational paralinguistics* and while the following chapters will focus on how to build, learn, and evaluate such methods, this chapter will outline what the term computational paralinguistics actually refers to. It will present a short definition of the term *paralanguage*, as relevant for this dissertation, and sketch a brief taxonomy regarding the structure and types of paralinguistic manifestations. Furthermore, it will address the computational aspect of paralinguistics and its applications. Finally, the chapter concludes with thoughts and solutions to the problems of annotation and rating, i. e. creating the "reference" for objective evaluation required in the following chapters.

2.1 Paralinguistics - A Structural Definition

Paralanguage connotes "alongside language" (from the Greek preposition $\pi\alpha\rho\alpha$) [303] and generally describes the non-verbal elements of human communication, i. e. it comprises all meta-information which accompanies and complements language. The paralinguistic properties of speech often play a very important role, since they can modify the meaning of what is being communicated, how that is perceived by others, and transmit many other characteristics of the speaker, such as gender, age, mood, etc. Loosely speaking, the study of paralanguage, called *paralinguistic*, is rather concerned about *how* something is said, instead of *what* is said.

The terms "paralanguage" and "paralinguistics" first appeared in works by Hill [143] and Trager [345] and were later extended by Crystal [62, 63]. In a broad sense [131], paralanguage comprises all non-verbal manifestations in conversation, including facial expressions, gestures, body posture. In a narrow sense, as argued by Crystal [64], the term paralinguistics excludes the visual components of non-verbal communication such as kinesics, proxemics, and haptics, and instead restricts the scope of the term to "vocal factors involved in paralanguage" [293]. Yet, even in this narrow sense there are two components of paralanguage: one is concerned with *linguistic* aspects of speech, including the structure and grammar of language, the phonetics, and the semantics of speech. The other evolves around the acoustic signal and the involved phenomena. This thesis will *exclusively* focus on the latter, acoustic-based aspect of paralinguistics. Furthermore, it will only treat aspects of analysis and detection and ignore paralinguistic synthesis, since the latter requires a completely different approach to the problem than the former.

The paralinguistic realm can be structured in several different ways, as described by Schuller and Batliner [293]. One common way to classify paralinguistic events is according to their time-scale: long-term events are often referred to as *traits*, also defined as a "distinguishing quality or a genetically determined characteristic, typically one belonging to a person" [293], whereas short-term events usually are termed *states*, which describe "a particular condition that someone is in at a specific time" [327]. Schuller and colleagues [303] further make a distinction between long-term, medium-term, and short-term events as follows:

- **Long term traits** include *biological trait primitives* (e. g. height, weight, age, or gender) or *personality traits* (likability, personality).
- **Medium term events**, between traits and states, embrace *structural signals*, such as non-verbal social signals or positive/negative attitude, and *self-induced states*, e. g. sleepiness, intoxication, mood, or health state.
- **Short term states** instead comprise *emotions*, *emotion-related states* (e. g. stress, interest, uncertainty, deception, politeness, or frustration), and *mode* (speaking style, voice quality).

Often correlations co-exist between speaker traits and states. Byrd [44], for example, showed that a speaker's pronunciation is affected by both his/her gender and dialect region. Furthermore, all non-binary traits and states can exhibit different intensity levels. These temporal characteristics affect the computational approaches to a particular problem. As pointed out by Zhang [380], other taxonomic dimensions such as the labeling scheme (objectivity vs. subjectivity) have an immediate impact on data annotation, feature extraction, and statistical modeling. These characteristics can be visualized in a two-dimensional *time-label* space, as depicted in Figure 2.1.

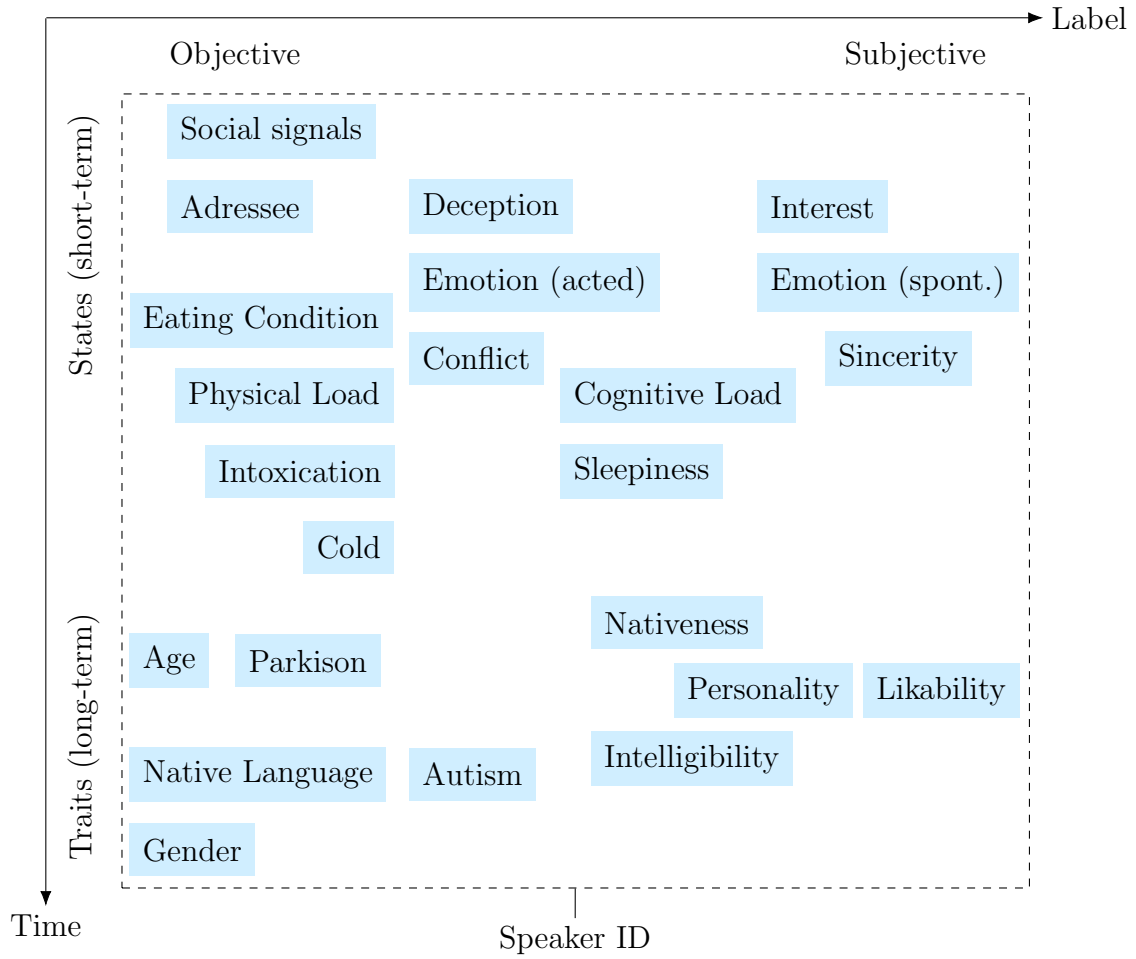


Figure 2.1: Two-dimensional paralinguistic trait/state space based on the Computational Paralinguistics Challenge tasks from 2009–2017. [380]

From the figure one can observe that, for example, social signals (in the upper-left corner), such as laughters and fillers (cf. Chapter 7), are short-term paralinguistic events which often occur in fractions of a second, and hence require frame-wise annotation and localization. Gender on the other hand is a (usually) persistent, biological attribute, which can objectively be labeled. To the right one can find increasingly subjective characteristics, ranging from short-term states, e. g. emotions, to long-term traits, such as likability. All these tasks depend on subjective, individual judgments [380].

Based on this taxonomic characterization the study of *computational paralinguistics* deals with the computer-based or computer-assisted analysis (and synthesis) of the paralinguistic types described above [293]. While the first endeavors in this

direction date back to the early works by Picard on affect recognition [248], computational paralinguistics has become a very active field of research especially in recent years. This progress has been substantially fostered by the series of Interspeech challenges held since 2009 [291, 292, 294, 295, 296, 297, 298, 300, 304, 305, 306], with the intention to support research in the many aspects of paralanguage and to improve human-machine communication. This interaction allows for a plethora of possible applications and today an increasing number of deployments integrate automatic paralinguistic approaches. These include, but are not limited to, the areas of healthcare, assistive robotics, surveillance, gaming, call center quality assurance, or target-group specific advertising [303]. Further paralinguistic cues can be used to support ASR systems or to detect frustrated users in automated call centers. From this almost endless pool of use cases, this thesis selects and describes four different domains, namely the detection and classification of *likability*, *conflict*, *social signals*, and *emotion/affect*, as prototypical examples of trait-based and state-based tasks and to demonstrate the feasibility of deep learning methods in these areas.

2.2 Annotation and Rating

In order to train models (in a supervised fashion) and evaluate their performance paralinguistic data must obviously be annotated or rated. In this respect one needs to differentiate between the *ground truth*, which is defined to be the 'actual truth' of a paralinguistic phenomenon and which can validated in an objective manner, and the *gold standard*, which is the result of the annotation process [289]. Ideally the gold standard is identical to the ground truth, but in reality it might be (slightly) erroneous. This is highly task-dependent: While in certain situations the ground truth is easy to infer, e. g. the gender or age of the speakers, it can be highly subjective in other cases, for example when annotating personality or emotions. Erroneous or *noisy* labels, however, lead to error-prone models and possible misinterpretations of the test results. A common way to mitigate these negative effects is to have several annotators to rate or label a paralinguistic instance (e. g. an audio recording).

In this respect it is important to reflect and decide about the label space, i. e. how the labels, and hence the outputs to be predicted by a model, shall be represented. Biological trait primitives, such as age or gender, are typically categorical or even binary. In certain other domains, such as likability, personality, or emotion research, the target labels can either be categorical, e. g.'likable' vs. 'non-likable' or 'angry' vs. 'neutral', or they can be represented in a continuous space. In emotion recognition, for example, it is very common to represent the different emotions in the two-dimensional valence-arousal plane, as shown in Figure 2.2, sometimes even complemented by dominance as a third dimension. Furthermore, the annotation can occur on different time-resolutions and time-scales: Audio recordings can be marked

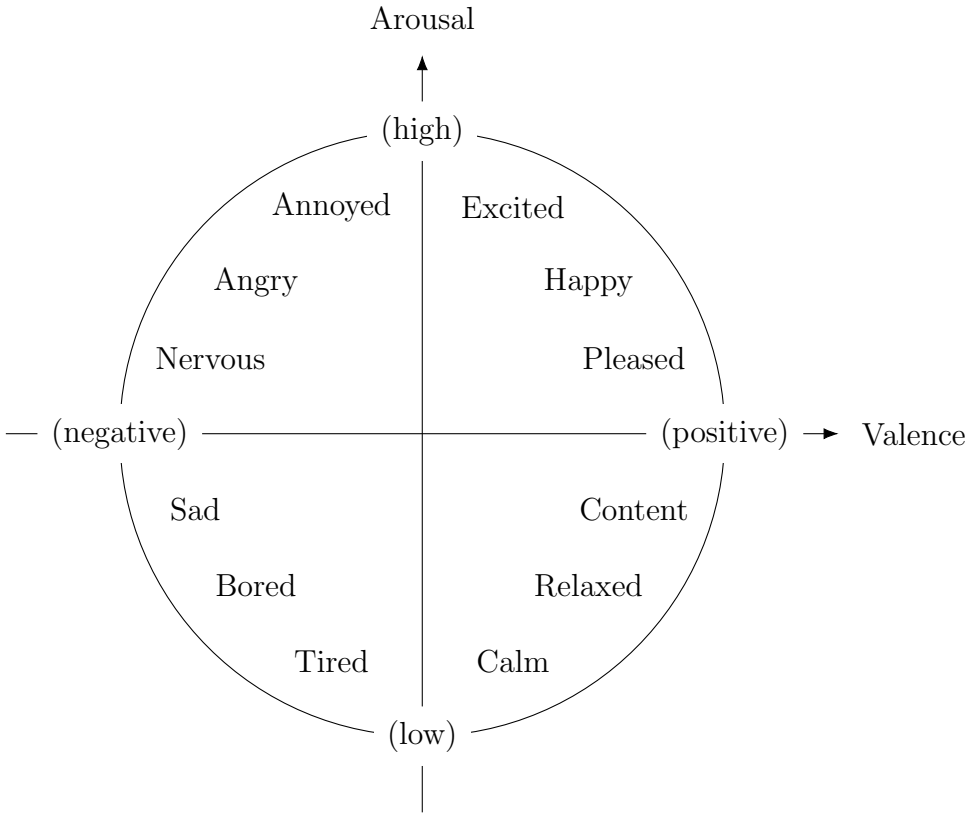


Figure 2.2: Two-dimensional valence-arousal plane for emotion-related paralinguistic states.

on an utterance-level (one label per recording), on a segment-level (e. g. one label every second), on a frame-level (as common in social signals research), or generally in a continuous manner (marking the start and end of certain paralinguistic events).

When annotations are provided by multiple raters the natural question arises how to obtain the gold standard from them. If the labels are of continuous nature often correlation or mean linear/absolute error measures among labeled are used [289]. In the categorical case, the simplest approach one can follow is to use majority voting to obtain a single value, i. e. to use the value on which most annotators agree. However, this is often problematic, since first, there might not be any agreement at all, and second, majority voting does not take into account any inter-rater and intra-rater agreement, which captures the difference between the raters (e. g. how emotions are perceived) or any variation over time (e. g. due to fatigue during the annotation process). In order to obtain a more consistent gold standard Grimm and

Kroschel [118] proposed the *evaluator weighted estimator (EWE)* as follows [289]:

$$y_{n,EWE} = \frac{1}{\sum_{k=1}^{K} r_k} \sum_{k=1}^{K} r_k \, y_{n,k}, \tag{2.1}$$

where $k = 1, \ldots, K$ denotes the rater, $y_{n,k}$ is the annotation given by rater k to instance n, and r_k is the weight corresponding to the 'reliability' of rater k:

$$r_k = \frac{\sum_{n=1}^{N} \left(y_{n,k} - \frac{1}{N} \sum_{n'=1}^{N} y_{n',k} \right) \left(\overline{y}_n - \frac{1}{N} \sum_{n'=1}^{N} \overline{y}_{n'} \right)}{\sqrt{\sum_{n=1}^{N} \left(y_{n,k} - \frac{1}{N} \sum_{n'=1}^{N} y_{n',k} \right)^2} \cdot \sqrt{\sum_{n=1}^{N} \left(\overline{y}_n - \frac{1}{N} \sum_{n'=1}^{N} \overline{y}_{n'} \right)^2}}, \tag{2.2}$$

where $\overline{y}_n$ is the mean over the ratings of all evaluators:

$$\overline{y}_n = \frac{1}{K} \sum_{k=1}^{K} y_{n,k}. \tag{2.3}$$

Equation (2.2) defines r_k as the cross-correlation between the estimations $y_{n,k}$ and the mean ratings of all evaluators. The first term in the numerator takes any intra-rater variations into account, which might occur during the evaluation session of rater k. Similarly, the second term in the numerator measures the average deviation of all evaluators from the overall mean for instance n. From this follows that the inter-evaluator agreement can be described by the correlation coefficients r_k using equation 2.2 and the standard deviations σ_n of the assessments [293],

$$\sigma_n = \sqrt{\frac{1}{K-1} \sum_{k=1}^{K} (y_{n,k} - y_{n,EWE})^2}, \tag{2.4}$$

which indicates how similarly a paralinguistic instance is perceived by the annotator in terms of the target problem.

Chapter 3

Acoustic Features

Colors, like features, follow the changes of the emotions.

Pablo Picasso

In machine learning and pattern recognition, a feature is defined to be an individual, measurable property or characteristic of a phenomenon being observed [26]. The underlying data, from which features are extracted, manifest the manifold modalities which represent the reality surrounding us: pictures and videos are visual representations, heart rate or skin conductance measurements reflect manifestations of physiological and physical conditions, while audio recordings retain the information of human and non-human sounds. This latter modality is the sole focus of interest in this thesis.

The audio wave form is the starting point of our considerations and since most machine learning algorithms require a numerical representation as input one seeks to extract a set of *features* from the sound signal, normally in the form of an D-dimensional feature vector $\mathbf{x} \in \mathbb{R}^D$. Audio analysis and signal processing have a long history and there exist many different fields of audio-based research [289]. As a consequence, many different types of feature vectors have been devised over time. Historically, audio features have been inspired by psychoacoustics [391], usually mimicking the auditory processing steps of the human ear (up to the auditory cortex). And while there exist a number of physiological models, which attempt to model the auditory processes on a very detailed level, in the machine learning community it is more common to find phenomenological models, i. e. models which mathematically describe empirical relationships of the operations performed by the auditory system.

Over the last decades a lot of effort has been spent on designing good acoustic features, leveraging expert domain knowledge from many different areas of research –

an approach often termed *feature engineering*. And for a long time these features were often strongly influenced by their successful adoption in areas such as speech recognition or speaker recognition, one prominent example being the use of Mel-frequency cepstral coefficients [147, 151]. However, since the advent of deep learning in the last decade or so, researchers have attempted to learn internal representations which are informative for a particular task at hand automatically from the underlying audio signal. This process termed *feature learning*, in contrast to feature engineering, reduces or even eliminates the need for expert domain knowledge and enables machine learning algorithms to not only make use of features for learning, but also to learn the required features themselves.

The following sections provide a general description of the extraction of acoustic features from the audio signal. The most important frame-based, *low-level descriptors* (LLD) will be shortly discussed, followed by a delineation of higher-level features, i. e. feature statistics derived from these LLDs. Subsequently, a short account of feature normalization is given, which is relevant for successful training of neural network models based on these features.

3.1 Feature Extraction

An illustration of the general feature extraction processing chain is given in Figure 3.1. The starting point is the audio database, which contains the digitized waveforms of the speech recordings. For the purpose of the work presented in this thesis it is assumed that these recordings have undergone some preprocessing, such as resampling, noise reduction, or energy normalization, wherever necessary. Further it is presumed that the recordings contain speech segments suitable for the task under consideration; this could be one utterance of an individual speaker per audio file, e. g. in the case of likability classification (cf. Chapter 6), or segments of overlapping and non-overlapping speech (cf. Chapter 8).

As indicated above, the traditional feature engineering approach to extracting features from each instance in the audio database is to perform a sequence of processing steps, where each step is a manually constructed component with no or a few parameters, which allow to modify the exact effect the component has on its input. Given a predictor function $\hat{y}$ and a model h, which generates predictions based on some input features, the complete processing chain can be defined as [361]

$$\hat{y} = h \circ g_K \circ \ldots \circ g_1(x) \tag{3.1}$$

where g_k denotes the k-th feature extraction step and x is the input signal. The model h is either a classification or regression model and typically is trained on the output of g_K to minimize the prediction error of $\hat{y}$. The goal of feature engineering is to find

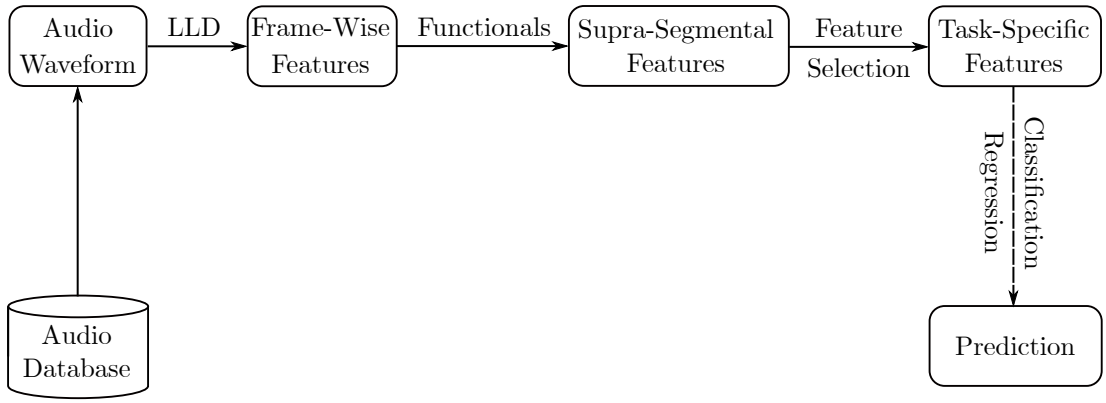


Figure 3.1: Illustration of the general feature extraction processing chain.

a set of feature extraction steps $g_1, \ldots, g_K$ so that the model h can unfold its full potential. Finding a suitable set of g_k can actually be viewed as a learning problem itself and the model h can attempt to subsume parts of the feature extraction steps $g_k, \ldots, g_K$, reformulating (3.1) to yield

$$\hat{y} = h' \circ g_{k-1} \circ \ldots \circ g_1(x) \tag{3.2}$$

where h' represents the new predictor model. In the extreme case, all feature extraction steps g_k can be merged into h', an approach often called *end-to-end learning*. The underlying idea is that h' can automatically find the relevant relationships between the task of interest and the input features by learning a non-linear, optimized mapping that short-cuts the processing steps $g_K \circ \ldots \circ g_k$. Feature learning is an active field of research and some important work indicates that it is at least partially responsible for the power and success of deep learning models [22, 227].

The following two sections give a brief overview of frame-based LLDs and higher-level supra-segmental features derived from them. Furthermore, the *ComParE acoustic feature set* will be described, since it serves as the standard feature set for many paralinguistic tasks in the literature [85] and in this thesis. Its development and availability has been fundamental for the objective evaluation and comparison of many different algorithms in the area of paralinguistic research [301]. Describing every single feature is well beyond the scope of this thesis and the interested reader is referred to the detailed literature on the openSMILE toolkit [84, 85], which was used to extract all features in this thesis, unless noted otherwise.

3.1.1 Low-Level Descriptors

As stated before, low-level descriptors (LLD) are frame-level features, extracted directly from the (pre-processed) audio waveform. The ComParE acoustic feature set contains 65 low-level descriptors [301], which are listed in Table 3.1.

Table 3.1: ComParE acoustic feature set: 65 low-level descriptors (LLD) [301].

4 Energy-related LLD	**Group**
Sum of Auditory Spectrum (Loudness)	Prosodic
Sum of RASTA-Style Filtered Auditory Spectrum	Prosodic
RMS Energy	Prosodic
Zero-Crossing Rate	Prosodic
55 Spectral/Cepstral LLD	**Group**
RASTA-Style Auditory Spectrum, Bands 1-26 (0–8kHz)	Spectral
MFCC 1–14	Cepstral
Spectral Energy 250–650Hz, 1kHz–4kHz	Spectral
Spectral Roll Off Point 0.25, 0.50, 0.75, 0.90	Spectral
Spectral Flux, Centroid, Entropy, Slope, Harmonicity	Spectral
Spectral Psychoacoustic Sharpness	Spectral
Spectral Variance, Skewness, Kurtosis	Spectral
6 Voicing-related LLD	**Group**
F_0 (SHS & Viterbi smoothing)	Prosodic
Probability of Voicing	Sound Quality
Logarithmic HNR, Jitter (Local, Delta), Shimmer (Local)	Sound Quality

The LLDs can be divided into four different feature groups: spectral, cepstral, prosodic, and voice/sound quality features. Each of these feature sets are described briefly in the following.

Spectral Features

First, the audio signal is sub-divided into overlapping windows[1] of fixed size N, i. e. $x(n)$ with $n = 0, 1, \ldots, N-1$. In the ComParE feature set a window size of 20 ms and a frame shift of 10 ms is used. However, this truncation of the original time-signal also alters its frequency characteristics, leading to an effect called *spectral leakage* [210]. This effect essentially leads to a smeared version of the spectrum

[1] The terms *window* and *frame* will be used interchangeably in this thesis.

derived from the original signal with potentially negative side-effects for further processing. In order to combat spectral leakage a tapered window function w is applied[2] to each audio frame x

$$x^{(w)}(n) = x(n) \cdot w(n) \qquad \text{for } n = 0, \ldots, N-1 \tag{3.3}$$

i. e. the frame $x(n)$ undergoes a sample-wise weighting of its values. There are numerous window functions and each one of them attempts to control the trade-off between the width of the main lobe and the level of the side lobes of the spectral leakage pattern in a different way [10]. For the spectral ComParE LLDs a Hamming window [132] is employed:

$$w(n) = a_0 - (1 - a_0) \cdot \cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \leq n \leq N-1, \tag{3.4}$$

with $a_0 = 25/46 \approx 0.54$. Each individual windowed frame is then transformed into the spectral domain by the Discrete Fourier Transform (DFT) [314], denoted by $\mathcal{F}$,

$$\begin{aligned} X(k) &= \mathcal{F}\{x^{(w)}(n)\} && \text{(3.5a)} \\ &= \sum_{n=0}^{N-1} x^{(w)}(n) \cdot e^{-\frac{j \cdot 2\pi k n}{N}}, \qquad k = 0, 1, \ldots, N-1 && \text{(3.5b)} \\ &= \sum_{n=0}^{N-1} x^{(w)}(n) \cdot \left[\cos\left(\frac{2\pi/N}{kn}\right) - j \sin\left(\frac{2\pi/N}{kn}\right)\right], && \text{(3.5c)} \end{aligned}$$

where (3.5c) follows from (3.5b) by Euler's identity [6]. $X(k) \in \mathbb{C}^N$ is the (complex) spectrum of $\mathbf{x}^{(w)}$ at the discrete frequency k and its phase is commonly neglected in paralinguistic applications. Hence, only the magnitude spectrum $|X(k)| \in \mathbb{R}_+^M$ or the power spectrum $|X(k)|^2 \in \mathbb{R}_+^M$ is kept, with $M = \lfloor N/2 + 1 \rfloor$ frequency samples. This is due to the fact that the audio samples $x(n)$ are real-valued and therefore the DFT symmetry $X(N-k) = X^*(k)$ holds, where X^* is the complex conjugate of X.

To account for the non-linear properties of human perception, two non-linear warping mechanisms are commonly applied to the spectrum. One addresses the non-linear perception of frequencies by warping the linear frequency scale (in Hz) to an approximately logarithmic scale, such as the Mel scale [326], which is given by

$$m(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.6}$$

where f denotes the frequency in Hz and m is the corresponding Mel frequency. One proceeds to select M bins equidistantly distributed over the Mel scale and

[2]Interestingly, the division into frames itself is equivalent to windowing with a rectangular or Dirichlet window function.

then to filter the DFT magnitude (or power) spectrum by applying a filter bank of overlapping triangular filters. One thus obtains an M-dimensional vector per time step, which in analogy to the standard frequency spectrum is called *critical band spectrum* [83]. A desirable side effect of the application of the filter bank is the reduction of the numbers of bands, $M < k$, which effectively serves as a feature reduction mechanism.

In early studies it was found that the human perception of loudness approximately follows a logarithmic scale [391] and for this reason a logarithmic mapping $|X(k)| \mapsto log(|X(k)|)$ is often applied to the magnitude spectrum. Alternatively, a cubic root amplitude compression, as proposed by Hermansky [141], can be applied to the magnitude spectrum to obtain the *auditory spectrum* as

$$X_{aud}(d) = \sqrt[3]{X(d)} \tag{3.7}$$

where d are the individual filter values of the filter bank.

All of the spectral LLD features shown in Table 3.1 are directly or indirectly derived from the spectra described above and are specified in the MPEG-7 multimedia content description standard [215]. Details on the RASTA-style auditory spectrum can be found in [83].

Cepstral Features

The derivation of cepstral coefficients is based on the idea of separating the filter from the excitation signal in the speech source-filter model [129] by using a homomorphic transformation, e. g. the cepstrum $c(n)$, which is defined as [257]

$$c(n) = \mathcal{F}^{-1}\{\log|\mathcal{F}\{x(n)\}|\}, \tag{3.8}$$

i. e. it is the inverse Fourier transform of the log-magnitude[3] Fourier spectrum (3.5a)[4]. Cepstral analysis shows that for the time-domain convolution of the excitation or source signal $x_s(n)$ and the filter $x_f(n)$ (representing the vocal tract) the following conditions hold:

$$x(n) = x_s(n) * x_f(n) \Leftrightarrow X(k) = X_s(k) \cdot X_f(k) \tag{3.9}$$

$$\log(X(k)) = \log(X_s(k)) + \log(X_f(k)) \tag{3.10}$$

$$c(n) = c_s(n) + c_f(n) \tag{3.11}$$

[3]Sometimes the power spectrum $|\mathcal{F}\{x\}|^2$ is used instead of the magnitude spectrum $|\mathcal{F}\{x\}|$.

[4]For notational simplicity the superscript '(w)' from (3.3) is dropped in (3.8), but all derivations apply to the windowed time signal.

Hence, the convolved time-domain signals are additive in the cepstral domain. If the logarithm is taken on the mel-frequency spectrum X_{mel}, as described in the previous paragraph on spectral features, the *Mel-Frequency Cepstral Coefficients (MFCC)* [67] are obtained. The inverse Fourier transform $\mathcal{F}^{-1}$ is usually replaced by the (type-II) discrete cosine transform (DCT):

$$X_{mfcc}(k) = \sum_{m=0}^{M-1} X_{mel}(m) \cdot \cos\left(\frac{\pi k}{M}(m+0.5)\right), \qquad k = 0, 1, \ldots, M' - 1 \quad (3.12)$$

where $X_{mfcc}(k)$ is the k-th MFCC, M' is the total number of MFCCs, and X_{mel} is the Mel-frequency critical band spectrum. The DCT approximately decorrelates the MFCCs and by choosing the number of MFCCs to be smaller than the number of critical bands, $M' < M$, one obtains some sort of feature reduction. In fact, for the ComParE LLD feature subset, $M' = 14$. This is common, since most of the information is contained in the lower-order coefficients. For example, $X_{mfcc}(0)$, which accounts for the distribution of high vs. low-frequency components in the Mel-frequency spectrum X_{mel}, is highly informative for the prediction of arousal in a number of audio applications [359]. As a final step the ComParE cepstral coefficients $X_{mfcc}(k)$ are filtered in order to emphasize low-order coefficients, a process referred to as *liftering*:

$$X'_{mfcc}(k) = X_{mfcc}(k)\left(1 + \frac{L}{2}\sin\frac{\pi k}{L}\right) \quad (3.13)$$

with L being the liftering coefficient[5]. The MFCCs listed in Table 3.1 correspond to X'_{mfcc}.

Prosodic Features

Prosody refers to linguistic characteristics such as intonation, rhythm, stress, or tone, and hence to the elements of speech not being individual phonetic segments, but rather properties of larger units of speech [180, 238]. Similar to the spectral features the prosodic features are extracted from a windowed audio frame. For determining the fundamental frequency F_0, often also loosely called *pitch*, a 60-ms Gaussian window is applied to each audio frame:

$$w(n) = e^{-\frac{1}{2}\left(\frac{n-(N-1)/2}{\sigma(N-1)/2}\right)^2} \quad (3.14)$$

with $\sigma = 0.4$. The choice of the Gaussian windows offers the advantage of not distorting the subharmonic structure, while the larger windows size, compared to the

[5] A typical value for the liftering coefficient is $L = 22$

spectral features, avoids distorting low, male pitch frequencies down to approximately 50 Hz. Based on the magnitude spectrum $|X(k)|$ the SHS algorithm is used to extract an estimate of F_0 [83]. The resulting F_0 contour is smoothed by a Viterbi-based algorithm, in order to eliminate artifacts such as spurious pitch detections and halving/doubling errors [208].

Besides the F_0 contour, another prosodic LLD is the sum over the auditory spectrum (cf. (3.7)),

$$\sum_{d=0}^{D-1} X_{aud}(d). \tag{3.15}$$

It is a perceptual measure of loudness, approximating the Zwicker loudness [392]. This feature is complemented by the sum of the RASTA-style filtered auditory spectrum. RASTA-style filtering is a temporal filtering in analogy to temporal properties of the human auditory system, in particular the fact that speech is primarily composed of modulations around 4 Hz [141, 393]. The specific design and use of the RASTA bandpass filters is described in [83].

Further prosodic features included in the ComParE LLD feature subset are the *zero-crossing rate (ZCR)*, which is the number of sign changes per second in the audio signal $x(n)$, and the *root-mean square (RMS) energy* of the audio signal:

$$E_{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x^2(n)} \tag{3.16}$$

Sound Quality Features

In contrast to the prosodic features described in the previous paragraph, sound or voice quality features are referred to as micro-prosodic descriptors [289]. *Jitter* describes the variability of the length of the fundamental period, $T_0 = 1/F_0$, over time. The ComParE LLDs contain both the *local* jitter J_{pp} defined as

$$J_{pp}(n) = T_0(n) - T_0(n-1) \tag{3.17}$$

as well as the cycle or *delta* jitter, given by

$$J_p(n) = T_0(n) - \overline{T}_0 \tag{3.18}$$

where $\overline{T}_0$ is the average of T_0 over a short-time frame. Analogously, *shimmer* describes amplitude variations over consecutive voice signal periods [83], expressed as

$$S_{pp}(n) = |A(n) - A(n-1)| \tag{3.19}$$

where $A(n) = \max(x(n)) - \min(x(n))$ is the peak-to-peak amplitude.

The *harmonic-to-noise ratio (HNR)* is computed as the ratio of the energies of harmonic and noise-like components in a speech signal [260] and therefore can be used as a measure to describe how harmonic or noise-like it is. Further, since defined as a ratio of energies, it is preferably expressed on a logarithmic scale. The HNR can be computed, for example, in the time-domain via the auto-correlation function (ACF):

$$HNR = 10 \cdot \log \frac{ACF(T_0)}{ACF(0) - ACF(T_0)} \tag{3.20}$$

where $ACF(0)$ is the value of the ACF at the origin and $ACF(T_0)$ is the ACF peak at the fundamental period T_0. Similarly, the *probability of voicing* p_v is estimated as

$$p_v = \frac{ACF_{max}}{ACF(0)}. \tag{3.21}$$

Here ACF_{max} is the maximum value in the range $T_{0,min}, \ldots, T_{0,max}$, where this range corresponds to the range of expected F_0 values [83].

3.1.2 Supra-Segmental Features

Contrary to ASR, which is usually based on short-term speech units such as phonemes, the problems dealt with in this thesis often require to consider longer time spans. The emotional state of a person, for example, might require several seconds of speech to be modeled effectively. Hence, either the model class itself must be able to handle temporal dependencies over long ranges, as e. g. achieved by the adoption of recurrent neural networks (cf. section 5.5), or one must construct features which incorporate relevant information over longer time spans[6]. One approach to do so, and one which is adopted in the ComParE feature set, is to apply a number of *functionals* to a window of stacked LLDs, where the window can be chosen to include all LLD vectors into a single super-vector. The resulting features are called *supra-segmental* features. This latter approach has the pleasant effect of reducing the variable-length audio recordings into a fixed-length representation, which is very helpful for models which are unable to handle time-information, e. g. Support Vector Machines (SVM) or Random Forests (RF). A functional $\mathscr{F}$ represents a mapping of a series of values $x(n)$ to a single value per functional $X_{\mathscr{F}}$ [289]:

$$x(n) \xrightarrow{\mathscr{F}} X_{\mathscr{F}} \tag{3.22}$$

Examples of functionals are the minimum, maximum, mean, standard deviation, or higher-order moments such as skew and kurtosis. Further, percentiles, temporal

[6]The necessary length, however, depends on the problem under consideration.

centroids, regression coefficients, peaks and valleys are commonly used. For a detailed description the interested reader is referred to [83]. Different functionals are applied to different features or feature groups. The used functionals in the ComParE acoustic feature set are listed in Table 3.2. In total the ComParE feature set consists of 6.373 features.

3.2 Normalization

The value range of features can vary quite drastically, depending on the type of feature or recording environment as well as the speaker subject. For example, the fundamental frequency F_0 can be expected to vary approximately between 50-300 Hz, while the probability of voicing will be between 0 and 1, since it is a probability estimate. In many machine learning algorithms features with a-priori larger value ranges might be given a higher importance and hence dominate and skew the training process. In particular, for neural networks it is very important to properly normalize the input features, because when using saturating activation functions (cf. section 5.3.2) care must be taken to avoid ending up in the saturating tails of the activation functions. Failing to do so will lead to near-zero gradients and hence to slowed-down or even non-convergent training [318]. Therefore, one of the two following methods are commonly applied to normalize the input features.

Min-max normalization scales the feature vectors so that their values end up in the predefined interval $[a, b]$. Let $\mathbf{X}$ be the feature matrix, defined by the sequence of N D-dimensional feature vectors, i. e. $\mathbf{x}(n) \in \mathbb{R}^D$ for $i = 1, 2, \ldots, N$. Further, let $\overline{\mathbf{x}}$ be the mean over all un-normalized feature vectors and $\mathbf{x}^{min}$ and $\mathbf{x}^{max}$ be the vectors containing the minimum and maximum values for each feature dimension over the complete matrix $\mathbf{X}$. Then the min-max normalized vector $\mathbf{x}'$ is given by

$$x'_i = \frac{x_i - \overline{x}_i}{x_i^{max} - x_i^{min}} \cdot (b - a) + a, \qquad i = 1, 2, \ldots, D \tag{3.23}$$

i. e. each feature dimension i is scaled individually so that $x'_i \in [a, b]$. The most common values for the interval endpoints are $[0, 1]$ and $[-1, +1]$. Eyben [83] pointed out that min-max normalization is vulnerable to single outliers and hence its usefulness in real conditions is limited. As an alternative, standardization or *z-score normalization* is commonly used. It is defined as

$$x'_i = \frac{x_i - \overline{x}_i}{\sigma_i}, \qquad i = 1, 2, \ldots, D \tag{3.24}$$

where σ is the standard deviation computed over $\mathbf{X}$. The normalized values x'_i are zero-mean and unit-variance variables and as was shown by Zhang [381] they are

more robust to outliers than min-max normalized features.

As explained in section 4.2 the available data is usually split into a training set $\mathbf{X}_{train}$ and a test set $\mathbf{X}_{test}$ (plus optionally a development set $\mathbf{X}_{valid}$). In this context it is important to remember that all statistics must only be computed on the training set and need to be kept fixed and applied to each data sub-set separately. Therefore, it would be an error to perform feature normalization before performing the data split.

Table 3.2: ComParE acoustic feature set: Functionals are applied to the LLDs defined in Table 3.1 [301].

Mean Values
Arithmetic Mean$^{A!\Delta,B}$, Arithmetic Mean of Positive Values$^{A\Delta,B}$
Root-Quadratic Mean, Flatness
Moments: Standard Deviation, Skewness, Kurtosis
Temporal Centroid$^{A!\Delta,B}$
Percentiles
Quartiles 1–3, Inter-Quartile Ranges 1–2, 2–3, 1–3
1%-tile, 99%-tile, Range 1–99%
Extrema
Relative Position of Maximum and Minimum, FullRange (Maximum–Minimum)
Peaks and Valleys^{A}
Mean of Peak Amplitudes
Difference of Mean of Peak Amplitudes to Arithmetic Mean
Mean of Peak Amplitudes Relative to Arithmetic Mean
Peak to Peak Distances: Mean and Standard Deviation
Peak Range Relative to Arithmetic Mean
Range of Peak Amplitude Values
Range of Valley Amplitude Values Relative to Arithmetic Mean
Valley-Peak (Rising) Slopes: Mean and Standard Deviation
Peak-Valley (Falling) Slopes: Mean and Standard Deviation
Up-Level Times: 25%, 50%, 75%, 90%
Rise and Curvature Time
Relative Time in which Signal is Rising
Relative Time in which Signal has Left Curvature
Segment Lengths^{A}
Mean, Standard Deviation ,Minimum, Maximum
Regression$^{A!\Delta,B}$
Linear Regression: Slope, Offset, Quadratic Error
Quadratic Regression: Coefficients a and b, Offset c, Quadratic Error
Linear Prediction
LP Analysis Gain (Amplitude Error), LP Coefficients 1–5

^{A} Functionals applied only to energy-related and spectral LLDs (group A)
^{B} Functionals applied only to voicing-related LLDs (group B)
$^{\Delta}$ Functionals applied only to LLDs
$^{!\Delta}$ Functionals not applied to LLDs

Chapter 4

Measures of Success

Success is simple. Do what's right, the right way, at the right time.

Arnold H. Glasgow

As can be seen throughout this thesis a multitude of steps is required to successfully construct a machine learning model: data needs to be selected and conditioned, features must be designed or selected, the model is required to be constructed and trained, etc. Eventually, in order to assess the quality and effectiveness of a model one needs to evaluate it in an objective manner. To this end science resorts to *evaluation metrics*, which play an important role in machine learning, because they are not only used to compare different learning algorithms, but often also as goals to be optimized during learning the models. Hence, they represent the quality of a model or algorithm evaluated on some data set, preferably as a single value, which simplifies ranking and comparison.

The first part of this chapter describes some of the established objective metrics for binary (also termed *binomial*) and multinomial classification, i. e. problems with more than two output classes. *Classification* refers to the problem of identifying to which of a set of categories or classes an observation belongs. This is in contrast to *regression*, which is the task of predicting a continuous value. This situation arises, for example, when estimating the level of arousal or valence (cf. Chapter 2), and evaluation metrics will also be discussed for this case. This first part is followed by a characterization of general guidelines and procedures for making adequate use of the available data during the training and test phases of model development and selection. This chapter concludes with a short digression on significance testing as relevant for the work covered in this thesis.

4.1 Evaluation Metrics

As mentioned above, evaluation metrics play a very important role in machine learning mainly for two reasons: First, they are used as optimization criteria during the training phase, usually by minimizing some form of loss function. Second, in supervised and semi-supervised training, models are commonly evaluated on cross-validation data during the training process, e. g. in order to detect convergence, and on test data once training has completed. Moreover, evaluation metrics also serve as the basis for comparing and selecting learning algorithms.

Different performance metrics are used to evaluate different machine learning problems and there exists a plethora of proposed measures for assessing the performance of models and algorithms, each with its advantages and disadvantages. In the following, the underlying concepts and metrics relevant for the work presented in this thesis are introduced and described.

4.1.1 Evaluation of Classifiers

Given a sequence of M-dimensional data elements, $\mathbf{x}_1, ..., \mathbf{x}_N$, and their associated discrete class labels, $y_1, ..., y_N$, then $\mathbf{D} = \{\mathbf{x}_i, y_i\}_1^N$ defines a *data set* with $\mathbf{x_i} \in \mathbb{R}^M$ and $y_i \in \{0, ..., C-1\}$, where C is the number of classes. Further, let $\hat{y}_i$ be a discrete estimate or *prediction* of the true label y_i:

$$\hat{y}_i = f(\mathbf{x}_i) \tag{4.1}$$

In machine learning the function $f(\mathbf{x}_i)$ is usually represented by a model trained on some training set and this model ideally produces a small number of errors, $\hat{y}_i \neq y_i$, on unseen test data, i. e. it possesses good *generalization* properties. To this end, evaluating a classifier consists in finding a measure that represents the classification (or misclassification) performance on some specific data set and hence the quality of the model or algorithm under consideration.

The simplest case, binary classification, can be viewed as a detection problem, with one class being the class of interest, or *positive* class ($y_i = 1$), and the other class the one to discriminate against, the *negative* class ($y_i = 0$). One can then count the frequency of correct and incorrect predictions $\hat{y}_i$ and compute

$$\begin{aligned}
\text{True positives (TP):} &\quad y_i = 1 \wedge \delta(\hat{y}_i, y_i) = 1 \\
\text{True negatives (TN):} &\quad y_i = 0 \wedge \delta(\hat{y}_i, y_i) = 1 \\
\text{False positives (FP):} &\quad y_i = 0 \wedge \delta(\hat{y}_i, y_i) = 0 \\
\text{False negatives (FN):} &\quad y_i = 1 \wedge \delta(\hat{y}_i, y_i) = 0
\end{aligned}$$

where δ is the Kronecker delta.

Let N_{TP} be the number of true positives, N_{TN} the number of true negatives, N_{FP} the number of false positives, and N_{FN} the number of true negatives. Then these frequency counts can be arranged into a 2x2 contingency table [317], as shown in Table 4.1, where columns correspond to the truth value, y_i, and rows correspond to the classification result, $\hat{y}_i$: Based on the Table 4.1 a number of measures can be

		Truth (y)	
		True	False
Classification ($\hat{y}$)	True	N_{TP}	N_{FP}
	False	N_{FN}	N_{TN}

Table 4.1: Contingency table for binary classification

defined. One important example is *precision*, defined as

$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{4.2}$$

measuring the number of correct positive instances divided by the number positive instances predicted by the classifier, and *recall*, given by

$$Recall = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{4.3}$$

which is the fraction of correct positive results over the number of all relevant samples, i. e. all instances that should have been identified as positive [342]. While precision and recall are metrics inherently suited for binary classification, problems with more than two classes can be reduced to a one-versus-all problem by selecting one class to be the positive class and merging the other classes into the negative class [7, 221]. Thus, multi-class situations can be approached by computing the *unweighted average recall (UAR)*, defined as

$$UAR = \frac{1}{C} \sum_{c=1}^{C} Recall(c) \tag{4.4}$$

$$Recall(c) = \frac{\sum_{i:y=c} \delta(y_i, \hat{y}_i)}{N_c} \tag{4.5}$$

where $Recall(c)$ is the recall specific of class c and $N_c = |\{i : y_i = c\}|$ is the number of instances belonging to class c. Note that the UAR is sometimes also referred to as unweighted accuracy (UA) [293].

Arguably the most intuitive and straightforward objective measure of a classifier's performance is the probability of correct classification or *accuracy* (Acc):

$$Acc = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FP} + N_{TN} + N_{FN}} = \frac{\sum_{i=1}^{N} \delta(\hat{y}_i, y_i)}{N} \tag{4.6}$$

Accuracy is defined as the number of correct classifications divided by the total number of all test instances, N, and it is a good measure when the distribution of class labels is approximately uniform. However, if this distribution is highly non-uniform, i. e. in problems exhibiting high *class imbalance*, accuracy is less appropriate [94], since deciding for the class having the highest prior N_c/N can result in high accuracy, while leading to chance level UAR.

At evaluation time, one can often define one or more free parameters, e. g. a threshold, which can be used to tune the classifier's sensitivity towards one class or the other. This selected value (or set of values) is then referred to as an *operating point*. For a specific classifier, changing this operating point can result in increasing the precision, while decreasing recall, and vice versa. However, having two conflicting measures in order to evaluate or optimize a classifier is cumbersome. For this reason, precision and recall are often commonly combined into a single evaluation metric, e. g. via the harmonic mean of both values, which results in the F_1 *score* defined as

$$F_1 = 2 \cdot \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{4.7}$$

The F_1 score gives equal weight to both measures and $F_1 \in [0; 1]$, as many of the evaluation metrics defined above. The use of the harmonic mean instead of the arithmetic mean has the advantage that it punishes extreme values. For example, a classifier with a precision of 1.0 and a recall of 0.0 has a F_1 score of 0, but an arithmetic mean of 0.5 [342].

The ROC curve and AUC

For binary classification, the *receiver operating characteristic (ROC)* graph is a technique for visualizing, organizing and selecting classifiers based on their performance [88, 255]. Originating from radar signal detection in World War II, they have long been used to depict the tradeoff between hit rates and false alarm rates of classifiers [80], especially in psychology and medical diagnostics [335]. ROC curves have been used in machine learning for almost three decades [223], but in recent years have been increasingly adopted, because researchers realized that the classification accuracy (4.6) often is a poor metric for performance measurements [258].

An ROC curve is a two-dimensional representation which graphically depicts two values deducible from Table 4.1: The *true positive rate (TPR)*, which is identical to

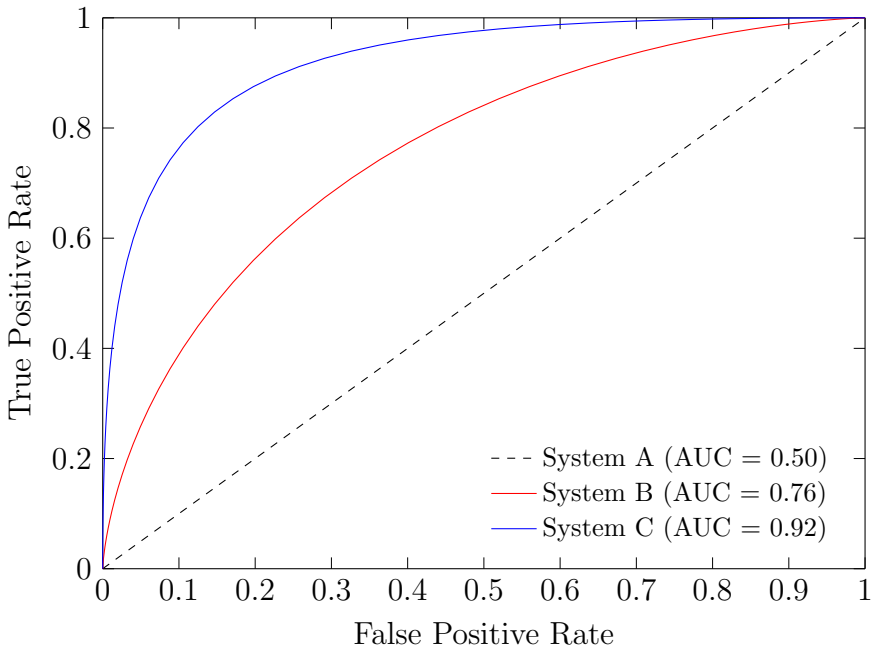


Figure 4.1: ROC curves and respective AUC values for random guessing (System A), medium performance (System B), and high performance (System C).

recall (4.3) of the positive class, and the *false positive rate (FPR)* (also called *false alarm rate*), which is defined as

$$FPR = \frac{N_{FP}}{N_{FP} + N_{TN}} = 1 - Recall(0). \tag{4.8}$$

As mentioned above, one can change the operating point by defining a threshold value $0 \leq \theta \leq 1$ and sweeping θ through this range. Assigning all instances i with $\hat{y}_i \geq \theta$ to the positive class, TPR and FPR result as a function of θ. The ROC curve thus results from interpolating different (TPF, FPR) points in the area $[0, 1] \times [0, 1]$ [380] and depicts the relative tradeoff between benefits (true positives) and costs (false positives) [88]. Figure 4.1 shows an example of three different ROC graphs.

The diagonal line between $(0, 0)$ and $(1, 1)$ (System A) represents random guessing, also termed *chance level*, and classifiers should result in curves above this line to be of any value. The closer the ROC curve gets to the upper left corner $(0, 1)$ the higher its ratio of TPR/FPR, and hence the better the classifier's performance. Therefore, system C (blue curve) shows better classification performance than system B (red curve). To reduce the ROC performance to a single scalar value, one can numerically integrate over the ROC curve on the interval $[0; 1]$ and compute the *area under the curve (AUC)* [34, 128]. The AUC reveals some interesting properties, which can easily be deduced from Figure 4.1: First, its value will always be between 0 and 1, i. e. $0 \leq AUC \leq 1$, since it is a portion of the area of the unit square. Second, random

guessing results in a value of $AUC = 0.5$. Finally, higher AUC values reflect better classification performance independent of any threshold parameter; hence the AUC is a preferred measure for situations where positive instances are rare, e. g. social signal detection and other related paralinguistic tasks.

One can generalize the AUC measure to the multi-class scenario by creating a *one-versus-all* binary classification setup for each class $c \in [0, ..., C-1]$, compute the class-wise $AUC(c)$, and eventually average over all these individual AUCs, resulting in the *unweighted average area under the curve (UAAUC)* [127]:

$$UAAUC = \frac{1}{C} \sum_{c=1}^{C} AUC(c) \tag{4.9}$$

The DET curve
While the AUC is of great value for comparing classifiers, the underlying ROC plot does not make good use of its graphical real estate, since for reasonably good classifiers the curves tend to end up in upper left corner, leaving empty most of the unit square. As in improved graphical representation the *detection error tradeoff (DET)* curve is sometimes used in the research community [218]. The DET curve plots the false rejection rate (FRR)

$$FRR = \frac{N_{FN}}{N_{FN} + N_{TP}} = 1 - Recall(1) \tag{4.10}$$

versus the false positive rate (4.8), which in the context of DET is often termed *false acceptance rate (FAR)*. The DET curve commonly adopts a non-linear scaling of the x- and y-axes in order to yield approximately linear trade-off curves. Hence, it uses most of the image area to highlight the differences of importance in the critical operating region, in particular around the the *equal error rate (EER)*, which is defined as the point where $FPR = FRR$. Figure 4.2 shows DET curves for System B and System C from Figure 4.1.

4.1.2 Evaluation of Regression Problems

Contrary to classification, in regression problems the target y_i is not categorical, but some scalar, continuous value. Hence, the prediction $\hat{y}_i$ is also continuous. A canonical evaluation of regressors can be achieved by measuring the mean error, which is given in its generic form by

$$\sqrt[\psi]{\frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|^{\psi}} \tag{4.11}$$

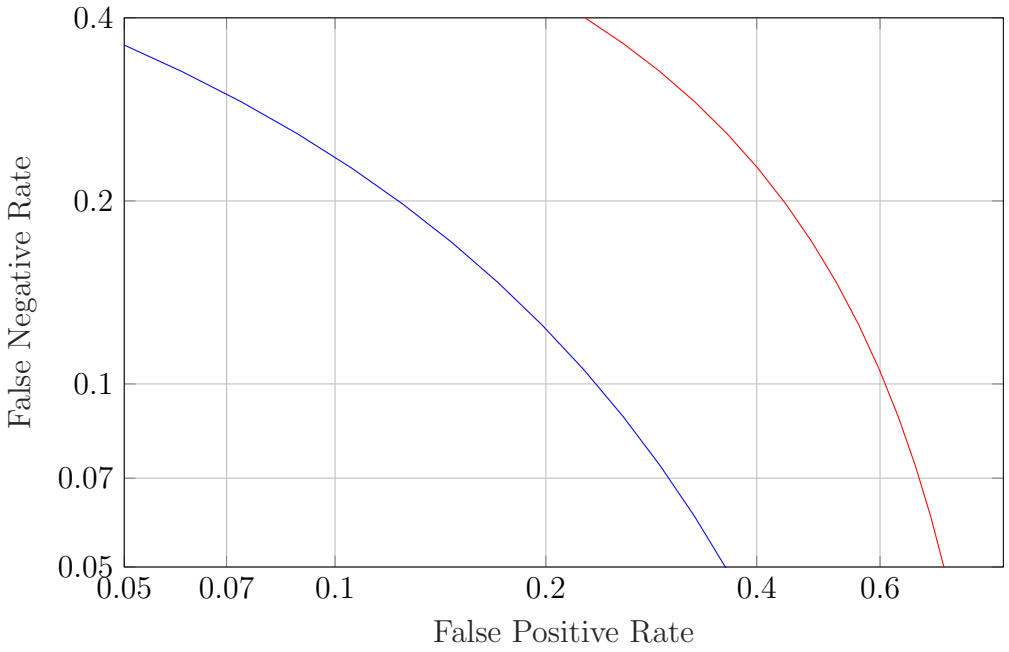


Figure 4.2: Detection error trade-off curves for systems B (red, medium performance) and C (blue, high performance) from Figure 4.1.

A commonly used value is $\psi = 1$, which yields the *mean absolute error (MAE)*:

$$MAE(\hat{y}, y) = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{4.12}$$

A more widespread choice is $\psi = 2$, however, which results in the *root mean squared error (RMSE)*. When comparing regressors the square root is often omitted, since it is a monotonic function unnecessary for comparison, and the resulting metric is known as the *mean squared error (MSE)*:

$$MSE(\hat{y}, y) = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|^2 \tag{4.13}$$

Other standard measures for the evaluation of regressors are (non-parametric) correlation coefficients. Of particular interest is Pearson's ρ, which for a data sample is defined as:

$$\rho(\hat{y}, y) = \frac{\sum_{i=1}^{N} (\hat{y}_i - \mu_{\hat{y}})(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{N} (\hat{y}_i - \mu_{\hat{y}})^2 \sum_{i=1}^{N} (y_i - \mu_y)^2}}, \tag{4.14}$$

with $\mu_{\hat{y}}$ denoting the mean over all predictions and μ_y the mean over the truth target values. It can be shown that $\rho \in [-1; +1]$ [202]. Spearman's rank correlation

coefficient or *Spearman's* ρ_s instead is a measure of rank correlation, i. e. it assesses the statistical dependence between the rankings of two variables. It is computed by first ranking the predictions $\hat{y}_i$ and truth targets y_i by their value and then computing Pearson's ρ (4.14) on the ranks. Since Spearman's ρ_s is robust to the underlying data distribution, it is sometime preferred over Pearson's ρ, particularly in the presence of outliers [75].

In the field of computational paralinguistics the gold standard targets are usually obtained by manual annotation of human raters. Notwithstanding careful design procedures to eliminate as many subjective differences as possible, these annotations are always prone to a certain degree of variation across different raters. In order to quantify the inter-rater agreement and to obtain a single target value, e. g. for training and evaluation, the *concordance correlation coefficient (CCC)*, ρ_c, is often relied upon in the research community [198, 224, 324, 346], defined as

$$\rho_c(v_1, v_2) = \frac{2\rho(v_1, v_2)\sigma_{v_1}\sigma_{v_2}}{\sigma_{v_1}^2 + \sigma_{v_2}^2 + (\mu_{v_1} - \mu_{v_2})^2}, \tag{4.15}$$

where v_1, v_2 are arbitrary variables (e. g. labels), μ_{v_1} and μ_{v_2} are the means over all instances of v_1 and v_2, respectively, and σ_{v_1} and σ_{v_2} are the corresponding standard deviations.

4.2 Crossvalidation and Testing Considerations

The paramount goal of any machine learning approach is to create a model which generalizes well to unseen test data. While methods and models might perform well on the data they are trained on, this is meaningless both from an academic and practical standpoint. It is well known that, given a specific training data set, neural networks (and many other models of machine learning) with a sufficient number of learnable parameters can approximate a target function, e. g. the distribution of training targets given the input, arbitrarily close. In other words, such a model can learn the training data distribution by heart, but fail to extract the underlying structure, thus eliminating irrelevant variation and noise from the inherent, relevant information, which is necessary to perform well on unseen data. In other words, for a reliable and objective evaluation of any classifier, regressor, or learning algorithm, it is necessary to obtain an unbiased estimate of the accuracy of a learned model.

For this purpose, available data should be divided into three disjunct data sets: the *training set*, used to learn the model parameters and any other related parameters helping to derive the underlying structure of the data distribution; a development or *validation* set, which can be regarded as an independent test set during the training phase, e. g. to determine when to stop training or to perform model selection; and

finally the *test set*, which should be completely held out during training and only be used for final evaluation and reporting. If the test data influences the learned model in any way, accuracy estimates will be biased. This might also imply that speakers in the test or CV sets should not be included in the training set, if possible, so that the model cannot learn specific characteristics of the test speakers.

Data sets in the area of computational paralinguistics are often relatively small, e. g. compared to the thousands of hours of available training data in the area of automatic speech recognition. This is mainly due to the necessity of manual annotation by multiple human raters, which typically is a time-consuming and expensive process. However, if there isn't enough data to create sufficiently large training and test sets, a trade-off must be found to divide the available data into those data splits and the following dilemma arises: the larger the test and validation sets the lower the variance estimate of accuracy will be. On the other hand, a larger training set will generally allow for larger and more powerful models to be trained. One commonly adopted approach to alleviate this situation is to resample a common data set into train, validation, and test splits in a round-robin fashion, an approach which is called *k-fold cross-validation (CV)*: A certain percentage of the data is held out for testing (or validation) and the rest is used for training. Then a model is trained on the training set and evaluated on the validation and/or test sets. The evaluation scores are stored and the trained model is discarded. This process then proceeds by selecting a different (and ideally disjunct) data split until all data instances have been used once for testing. In the end of k-fold CV all results are merged. A typical value for k is $k = 10$, but smaller values are used when training is resource-intensive [11].

There exist a number of aspects one might want to consider when setting up and performing k-fold cross-validation. One is *stratification* when partitioning the data, which means that in each of the training, validation, and test sets one strives for approximately the same percentage of speakers of the same age range, gender, cultural background, dialect region, etc., depending on the task at hand. A related idea to stratification is *balancing* the class label distributions [293]. The goal here is to achieve an approximately uniform distribution of label classes of the same paralinguistic content, for example emotion classes, in each of the data splits.

Yet another way to structure k-fold cross-validation is by the amount of left-out data for testing. One extreme case is *leave-one-out* CV, where only one data instance is used for testing, while the remaining data is used for training. An alternative, less extreme approach is called *leave-one-speaker-out (LOSO)*, where in each of the folds one speaker is split off for testing. A modification of this idea is *leave-one-speaker-group-out (LOSGO)*, where a set of speakers is reserved for testing, while all other speakers are used during training [189]. These are just a few examples of commonly adopted approaches in setting up and running k-fold experiments.

4.3 Significance Tests

As alluded to in the introduction of this chapter it is of paramount interest to compare the performance of different systems, be if for the purpose of model selection or algorithm evaluation. In this regard it is necessary to determine whether observed performance differences are caused by structural differences or if they are just due to random fluctuations. This can be accomplished by applying significance tests, which represent a formal process to evaluate the reliability of the evaluation. The quality of a test is typically judged on Type I error (how often the test indicates a difference when it should not) and Type II error (how often it indicates no difference when it should). The subsequent section gives a brief summary of some methods of significance testing relevant for the work in this thesis.

4.3.1 Comparison of Two Classification Accuracies

The McNemar test [73, 102, 220] is a statistical test for paired nominal data and hence can be used to test if the classification accuracies of two systems A and B are significantly different. It is assumed that both systems have been trained on the same training set $\mathbf{D}_{train}$ and are tested on the same test set $\mathbf{D}_{test}$. Then one can count and record the number of misclassifications of both systems in a contingency table as follows:

n_{00}: Number of examples misclassified by both A and B	n_{01}: Number of examples misclassified by A but not by B
n_{10}: Number of examples misclassified by B but not by A	n_{11}: Number of examples misclassified by neither A nor B

Table 4.2: Contingency table for McNemar error counts.

It follows that $n = n_{00} + n_{01} + n_{10} + n_{11}$ equals the total number of examples in the test set, $|\mathbf{D}_{test}|$. As laid out in [73], under the null hypothesis H_0, systems A and B have the same probability of error, i. e.

$$H_0 : \quad p_{n_{01}} = p_{n_{10}} \tag{4.16}$$

$$H_1 : \quad p_{n_{01}} \neq p_{n_{10}} \tag{4.17}$$

where H_1 is the alternative hypothesis, and $p_{n_{01}}$ and $p_{n_{10}}$ are the probability of the respective error counts from Table 4.2. The McNemar test statistic is defined as a chi-square random variable:

$$X^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \tag{4.18}$$

where the "-1" in the numerator is a continuity correction term as suggested by [79], which accounts for the fact that the test statistic is discrete while the χ^2 distribution is continuous. Under the null hypothesis H_0 and with a sufficiently large number of counts n_{01} and n_{10}, X^2 has an approximately chi-squared distribution with 1 degree of freedom and therefore it is possible to perform a χ^2 goodness-of-fit test to determine whether observed sample frequencies differ significantly from expected frequencies specified in the null hypothesis. First, one computes the p-value, which corresponds to the probability of wrongly rejecting H_0, i. e. assuming a significance difference when none is given:

$$p = P(X^2 \geq \chi^2_{1;1-\alpha}) \tag{4.19}$$

for a specific significance level α. A typical value for the significance level is $\alpha = 0.05$ and this value is used in this thesis unless stated otherwise. The criterion for rejecting H_0 is defined as

$$H_0 : \quad p \geq \alpha \tag{4.20}$$

$$H_1 : \quad p < \alpha \tag{4.21}$$

and therefore there is evidence to reject H_0, if

$$X^2 > \chi^2_{1;0.95} \approx 3.841459 \tag{4.22}$$

Previous work has indicated that X^2 is not well approximated by the χ^2 distribution, if $(n_{01}+n_{10}) \lessapprox 25$, as this leads to a unreliable p-values [5]. In this case an exact binomial test has been suggested, which considers the imbalance in the discordants n_{01} and n_{10}. The interested reader is referred to [293] for a detailed explanation. However, in [87] it was also found that the exact test is overly conservative, leading to unnecessarily large p-values and poor power, and the authors propose the mid-p test instead.

As can be seen from the derivation above the McNemar test requires the availability of the contingency table (cf. Table 4.2), which is a disadvantage when only the accuracies are known. An easily computable alternative is the *z-test* as described by Dietterich [73]. Given a specific test set $\mathbf{D}_{test}$, let p_A and p_B denote the probabilities of correct classification of two systems A and B, i. e. the respective accuracies evaluated on this data set. Without loss of generality we assume that $p_B > p_A$. One can formulate the null hypothesis H_0 that the observed performance differences originate from random fluctuations of the probability of correct classification by either system, $p_{AB} = (p_A + p_B)/2$. Then, one rejects H_0 at a chosen significance level α. Assuming statistical independence of the prediction errors, one can formulate the null hypothesis that the number of correct classifications N_c on $\mathbf{D}_{test}$ follows a binomial distribution with success probability p_{AB} as

$$H_0 : N_c \sim B(N, p_{AB}) \tag{4.23}$$

where $N = |\mathbf{D}_{test}|$ is the number of instances in the test set. The probability of observing the improved accuracy of B is then computed as

$$P(N_c > p_B \cdot N) = 1 - P(N_c \leq p_B \cdot N) \tag{4.24}$$

If N is large enough ($N \gtrapprox 20$), then the binomial distribution is well approximated by the normal distribution

$$\mathcal{N}(N \cdot p_{AB}, N \cdot p_{AB}(1 - p_{AB})), \tag{4.25}$$

and the z-statistic $z^*_{c,B}$ becomes

$$z^*_{c,B} = \frac{p_B - p_{AB}}{\sqrt{p_{AB}(1 - p_{AB})}}\sqrt{N} \tag{4.26}$$

Following (4.20) and (4.21), H_0 can be rejected with significance level α, if

$$p = 1 - \Phi_{\mathcal{N}}(z^*_{c,B}) < \alpha, \tag{4.27}$$

where $\Phi_{\mathcal{N}}$ is the cumulative distribution function of the standard normal distribution. The assumption of independence of errors made in the derivation of the z-test is violated in some cases, e. g. when processing time series by recurrent neural networks, where there is an interdependence between predictions based on successive frames. While the McNemar test was found to have acceptable low Type I error, the z-test tends to underestimate p-values, i. e. it tends to have a high probability of Type I error [73]. Nonetheless, since it only requires the knowledge of the accuracies of both systems and the size of the test set, it is one of the most widely used statistical tests in the area of machine learning.

4.3.2 Significance Tests for Regression Problems

Analogously to the derivation of the z-test, let A and B be two regressors generating predictions on a common test set $\mathbf{D}$, and without loss of generality assume that $MAE_B < MAE_A$, with MAE being defined by the mean absolute error (4.12). One can then formulate the null hypothesis

$$H_0 : MAE_A - MAE_B = 0, \tag{4.28}$$

stating that any differences are only due to random fluctuations and compute the sample difference Δ_n of the absolute errors made by systems A and B on instance n:

$$\Delta_n = |\hat{y}_{A,n} - y_n| - |\hat{y}_{B,n} - y_n| \tag{4.29}$$

$$\mu_\Delta = \frac{1}{N}\sum_{n=1}^{N} \Delta_n \tag{4.30}$$

$$\sigma_\Delta = \frac{1}{N-1}\sum_{n=1}^{N}(\Delta_n - \mu_\Delta)^2 \tag{4.31}$$

where μ_Δ is the sample mean of Δ, σ_Δ the respective sample standard deviation, and $N = |\Delta|$ the sample size. The significance test is a so-called *t-test*, which tests if μ_Δ is significantly different from zero. It is based on *Student's t-distribution*, which only depends on the sample moments and size and is equivalent to the Normal distribution $\mathcal{N}$ for large sample sizes N. The *t-statistic* is given by

$$t = \frac{\mu_\Delta}{\sigma_\Delta}\sqrt{N} \tag{4.32}$$

and bears resemblance to the z-statistic (4.26). Similar to (4.27) one rejects the null hypothesis (4.28) with significance level α, if

$$p = 1 - F_t^N(t) < \alpha \tag{4.33}$$

where is the cumulative distribution function of Student's t-distribution with N degrees of freedom [293].

4.3.3 Performance Comparison Across Different Partitionings

Up to this point, derivations and statistical significance tests were based on a fixed (held-out) test set $\mathbf{D}_{test}$ and a corresponding training set $\mathbf{D}_{train}$. This approach has the shortcoming that it does not directly measure any variability due to the choice of $\mathbf{D}_{train}$ and $\mathbf{D}_{test}$. In order to tackle this problem, one can divide a given data set $\mathbf{D}$ into K partitionings, as done in K-fold cross-validation, and then train and evaluate two systems A and B on each partitioning k separately to obtain the evaluation measures $M_A^{(1)}, \ldots, M_A^{(K)}, M_B^{(1)}, \ldots, M_B^{(K)}$. M can be any evaluation measure for classification or regression discussed in this chapter. Further, it was found in [73] and confirmed in [30] that partitioning $\mathbf{D}$ into disjoint sets of approximately equal size leads to more stable significance tests than random resampling, since the latter leads to undesirable dependencies of the $M^{(k)}$ due to overlap in the trials. In each trial k one computes the performance differences as

$$M_D^{(k)} = M_A^{(k)} - M_B^{(k)} \tag{4.34}$$

and tests whether this difference is significantly different from zero, and performs a t-test with the t-statistic similar to (4.32):

$$t' = \frac{\mu_{M_D}}{\sigma_{M_D}}\sqrt{K} \tag{4.35}$$

$$\mu_{M_D} = \frac{1}{K}\sum_{k=1}^{K} M_D^{(k)} \tag{4.36}$$

$$\sigma_{M_D} = \frac{1}{K-1}\sum_{k=1}^{K}(M_D^{(k)} - \mu_{M_D})^2 \tag{4.37}$$

Analogously to (4.33), one rejects the null hypothesis, which states that there is no significant difference between systems A and B, if

$$p = 1 - F_t^K(t') < \alpha \tag{4.38}$$

with significance level α.

Chapter 5

Deep Neural Networks

In theory, there is no difference between theory and practice.
But, in practice, there is.

Jan L. A. van de Snepscheut

This chapter lays the theoretical and practical foundation for the experiments described in Part III of this thesis. It starts by giving a brief description of the biological morphology and information transmission principles which inspired many of the neural network models used nowadays. The subsequent section establishes a brief taxonomy of concepts found in current research of machine learning and deep neural networks (DNN) to allow a distinction and classification of the approaches and models adopted in this thesis from the manifold of proposed ideas in the literature. Section 5.3 introduces the most basic components, the *neurons*, of the mathematical models adopted in this thesis. As common in current research in computational paralinguistics only phenomenological models (cf. Section 5.2) are considered, i. e. the level of detail that physiological models offer is neglected. Thereafter, a number of sections describe different types of neural network models, which are used as building blocks for deep neural networks, e. g. by stacking or layering those blocks. This chapter then concludes with some important aspects of training these models, both supervised and unsupervised, in order to obtain good performance.

5.1 Biological Background

Scientists have long been intrigued by how the human brain makes sense of the world and over the past century biologists have accumulated an enormous amount of detailed knowledge about the structure and function of the brain [100]. Within the mammalian brain the *cerebral cortex* occupies the largest region and plays a

fundamental role in perception, attention, awareness, and consciousness, forming the basis for human communication and language. The human cortex contains a huge number of cell bodies, mainly of two types [207]:

Neuronal cells, or in short *neurons*, serve as the elementary processing units in the central nervous system and are connected to other neuronal (and non-neuronal) cells via *synapses*, which basically are electro-chemical information interfaces. Neurons thus form interconnected circuits with diverse, discrete functions – some to sense and extract features of the environment and to transmit this information to the brain, others for processing and storing this information. Recent research shows that the human brain contains approximately $86 \cdot 10^9$ neurons, only 19% of which are located in the cerebral cortex [14, 140], while the number of neocortical synapses amounts to approximately $1.0\text{-}1.6 \cdot 10^{14}$ [237], i. e. on average one cerebral neuron is connected to other neurons via approximately 1000 synapses. Individual neurons can even form synapses with up to 10.000 other neurons [207].

Glia cells, on the other hand, were historically considered to be mainly "supporter" cells, which supply nutrients and oxygen and give structural stabilization and protection to the neurons. Further they maintain homeostasis, destroy and remove pathogens and dead neurons from the brain. However, it is now recognized that glia cells also play an active role in brain function [207]. Recent research has shown that the number of glia cells in the human brain is approximately equal to the amount of neurons [14], contrary to prior belief that these cells outnumber cortical neurons by orders of magnitude.

Although there exist several different neuronal and glia cell types with distinct characteristics and properties, functional specialization of different brain regions primarily emerges from differences in circuit connectivity rather than from differences in the constituent cell types [207].

Neuronal Morphology and Information Transmission

Once fully differentiated, neurons take many different forms, but in general they all share a number of common key features: The rounded *soma* or cell body contains the nucleus and branches into tree-like projections termed *dendrites*, which are the components where signals are received from other neurons via the synapses. However, it is also possible that incoming synapses form directly on the soma. Particularly in the brain these dendrites can grow extremely long, which allows neurons to interconnect with up to thousands of other neurons. On the other side of the soma neuronal cells form a long, extended arm called the *axon*, which essentially functions as a transmission channel. It has been found that in humans, axons can grow to more

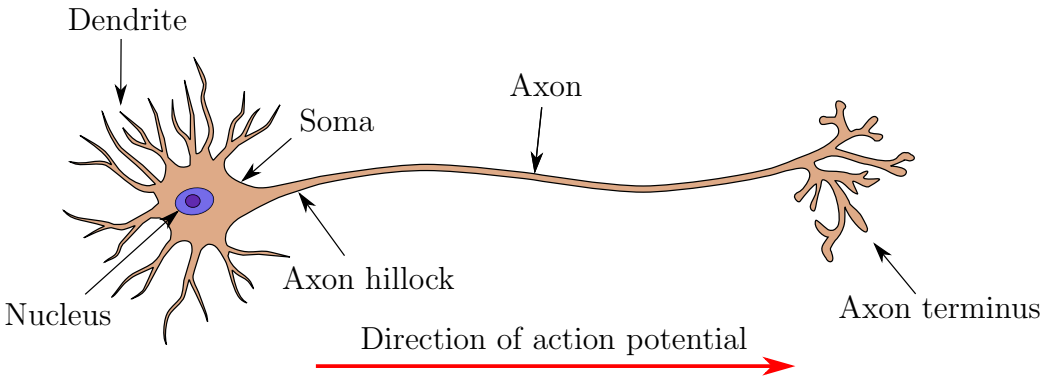


Figure 5.1: Typical morphology of a cortical neuron. Branched dendrites receive information from connected neurons. Small voltage changes in the dendrites, which are integrated over in the soma, can result in an action potential, generated in the axon hillock. The axon transmits the action potential to the synapses located at the axon termini, which connect with other neurons.

than a meter long [207]. Branch-like protrusions at the end of the axon (opposite the soma) are called *axon termini*. These structures are connected to the dendrites of other neurons via junctions called *synapses*. An axon terminus of a pre-synaptic cell contains synaptic vesicles, each of which is filled with a single type of neurotransmitter molecules. Once an electrical pulse or so-called *action potential* arrives at the synapse, it triggers the release of of a certain amount of neurotransmitter molecules. On the receiving side of the synapse, i. e. at the post-synaptic dendrite, these molecules cause a voltage change from negative to positive termed *depolarization*. It is in the soma of a neuron that the depolarizations from all its dendrites are integrated. If the resulting voltage in the soma exceeds a certain threshold an action potential is triggered, an electrical *spike* originating in the *axon hillock*. Hence, the neuronal integration of depolarizing and hyper-polarizing signals determines the likelihood of an action potential. Action potentials move along the axon to its termini at speeds of up to 100 meters per second. Moreover, neurons are able to fire repeatedly after a brief recovery period, usually in the order of milliseconds. In fact neuronal signals usually consist of a sequence of spikes termed *spike trains* [99].

It is important to note that action potentials – notwithstanding their analog nature on an electro-chemical level – effectively are binary, i. e. they are "all or none". This implies that the information processing happening inside a population of neurons is not based on the intensity of the action potentials, but rather on their relative timing and their frequency or *rate*. In fact, while peripheral neural circuits mainly convey information about what they sense in a rate code, circuits located close to or in the cortex increasingly code information via a time-dependent representation [226, 276, 325]. The synaptic connection between any two neurons

can either be *excitatory*, i. e. facilitating the transmission of the input signal by supporting the generation of a post-synaptic action potential, or *inhibitory*, i. e. by hindering transmission. Moreover, neurons receive slower neuro-modulatory inputs via regulating hormones which changes the threshold for excitation or inhibition [46].

Even though the typical morphology of a cortical neuron, as depicted in Figure 5.1, clearly indicates the unidirectional flow of information, a great number of recurrences exist in the brain by efferent connections, i. e. connections from neurons higher up in the cortical hierarchy to lower regions or even within the same brain region. In fact, this recurrence is necessary to enable time-dependence in sequence processing [76]. The complex networks formed by individual neurons and glia cells are not fixed, but instead their inter-connectivity changes through a process called *synaptic plasticity*, during which both the number of synaptic connections and their strength is modified, based on experience. This fundamental learning process is imitated by the training algorithms deployed everywhere nowadays in modern artificial neural networks.

5.2 Taxonomy

As alluded to in Section 5.1 neurons serve as the elementary processing units in the central nervous system and for this reason they form the biological substrate whose function one tries to replicate with more or less detailed mathematical models. There exists a plethora of models in the literature, but all share two commonalities.

First, there are at least two main objectives for modeling neurons or neural networks: On the one hand scientist strive to gain a better understanding of the underlying phenomena and how neurons function, and apart from laborious in-vivo cell experiments they use computer models to derive deeper knowledge. On the other hand researchers and engineers leverage powerful neural network models to perform inference and prediction on many in-lab as well as real-world tasks up to levels of performance previously deemed unreachable.

Second, all approaches are oriented towards "replicating", i. e. modeling, at least the basic functions a neuron performs. As discussed in Section 5.1 this includes (but is not limited to):

- *Weighting* the individual input signals: This step models the process at the transition *axon terminus* → *synapse* → *dendrites* and may include modeling the amount of neurotransmitters in the synapses w. r. t. the strength of the input signal, positive (excitatory) and negative (inhibitory) weighting, and constant potential offsets (often represented as *bias* in mathematical models, cf. Section 5.3.1).

- *Integration* or *summation* over the weighted signals: Collecting input from many connected neurons, this key feature of biological neurons enables the emergence of neural circuits and thus a higher-level, more powerful functionality. Be it based on rate coding or temporal coding, this phenomenon brings individual sources of information into a common context.

- *Transforming* the intermediate result in a sense models all neuronal processes that lead to the formation of action potentials. Again, this could reflect the frequency of action potentials (rate coding) or their temporal sequence (temporal coding), but could also include non-linear, molecular intra-cell effects. In mathematical models, as discussed in this thesis, this transformation operator often is chosen for reasons of mathematical commodity or algorithmic stability, rather than biological requirements.

Taking into account these two factors one can attempt to derive descriptors of model classes which may help to differentiate between different modeling approaches. Without claiming to be exhaustive, the following list indicates some criteria helpful to differentiate between different model types.

Rate coding vs. temporal coding
It has been mentioned above that both ways of representing information in spike trains exist in biological neurons and hence it only seems plausible to create respective models for both. While the idea of modeling rate coding – i. e. taking the frequency of spikes in a certain time interval as the basic "unit" of information – is adopted in the majority of current neural network models, modeling temporal coding via spike trains requires one to follow a completely different paradigm adopting *Spiking Neural Networks* (SNN) [267]. Although it is possible to compute SNN models on today's typically deployed serial Von-Neumann computer architectures, the latter are ill-suited for the inherently event-driven temporal, biological phenomena. SNNs are an emerging area of active research [28, 212] and beyond the scope of this thesis. Models based on rate coding instead perform static computation, usually executed at pre-determined intervals[1].

Computational complexity vs. level of detail
Even when restricting oneself to rate coding models the level of detail of neuronal models determines the computational complexity and hence the amount of compute power and memory one has to dedicate in order to perform simulations on these models. While a greater level of detail might reveal certain desirable characteristics and higher modeling power, this is not always the case. Thus, one must strike a balance between these two opposing characteristics. Most often researchers choose

[1] A typical value in the realm of voice signal analysis is e. g. 10 ms.

the minimum amount of detail necessary for a specific task at hand and try to minimize computational complexity, especially when large-scale experimentation is required. Apart from pragmatic reasons, such as limiting the time needed to train a model or to limit the amount of required computer resources, this is also helpful when training data is limited in order to avoid overfitting and numerical instabilities.

Biological models vs. phenomenological models
Research on *biological* neuron models has a long history and one of the earliest models described in the literature dates back to 1907, when the French neuroscientist Louis Lapicque described a first version of the Integrate-and-Fire model [2, 197]. Since then a plethora of studies has been published (e. g. see [106, 144, 157, 168, 231]) and in simplified terms biological models attempt to describe intra-neuronal and inter-neuronal physiological and bio-chemical processes to a greater level of detail than other models. And while research in this field is of great importance to the general understanding of how biological neurons process information, the use of such detailed models in medium- to large-scale experiments is limited due to their computational complexity, notwithstanding the ever-growing capabilities of today's computers. For this reason, researchers have turned towards *phenomenological* models as scientific models which attempt to describe the empirical relationship of phenomena between each other without modeling the exact mechanisms [90]. These models try to incorporate the principles and laws of the underlying biological phenomena, yet abstracting sufficiently enough to be computationally efficient. Most of the models described in this thesis fall into this latter category of models.

Categories of learning
In addition to the above-mentioned categorical classification, neural network models can be grouped into categories depending on the fundamental principle they are learned or trained with. These categories differ by the nature of the training data and by the manner and specific order in which training and test data are queried and provided to the learning process. Most learning schemes in current neural network research can be grouped into three fundamental types of learning [278]:

- In **supervised learning** a training set consist of N input-output pairs

  $$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N),$$

  where each *target* (or *label*) y_i is assumed to be generated by an unknown function $y = f(\mathbf{x})$, a function of the *observations* or *features* $\mathbf{x}$. The goal of the learning algorithm is to find a hypothesis $\hat{y}$ which approximates the true function y, i. e. a function that maps input to output minimizing the error between $\hat{y}$ and y [278]. Note that the error to be minimized can take on various different forms and that the final goal usually is not to merely perform well

(i. e. obtain low error) on the training set, but on unseen examples $(\mathbf{x}', y')$. If the learned hypothesis $\mathcal{H}$ performs well on unseen data points, it is said to *generalize*.

Supervised learning is the most common learning type in current neural network research, since it has proven to be extremely effective on a broad range of tasks. However, it requires the targets y to be available and generating these labels often proves to be an expensive, manual, and laborious task. In the field of computational paralinguistics this fact is particularly exacerbated by the fact that the true emotions as perceived by humans are highly subjective, and hence more than one annotator is needed to determine the targets [293].

Supervised learning can be broadly classified into **classification** and **regression** tasks. The learning problem is termed *classification*, when the output y is an element of a discrete set of categories, such as 'anger', 'fear', 'sadness', etc. In the particular case of only two categories (e. g. 'True' and 'False'), it is called binary classification. *Regression*, on the other hand, deals with the problem of estimating or predicting a continuous quantity, as occurs, for example, in predicting emotions in the arousal/valence space (cf. Figure 2.2).

- **Unsupervised learning**
  In many modern large-scale learning scenarios the amount of unlabeled data available by far exceeds that of labeled data. For model training it is usually beneficial to have more transcribed data, but as outlined above, the cost of labeling if often prohibitive, since it typically requires human effort to do so. Unsupervised algorithms attempt to resolve this problem by learning some hidden structure from unlabeled data. Obviously, since no target labels are available, calculating typical "success" measures, such as accuracy, is impossible in this case.

  There exists a great variety of unsupervised learning approaches, such as clustering algorithms (e. g. k-means), principal component analysis (PCA), independent component analysis (ICA), non-negative matrix factorization (NMF), or singular value decomposition (SVD), just to name a few [33, 78, 134]. Another central class of unsupervised learning algorithms is based on generative modeling, which is the statistical task of estimating a probability distribution that describes the process which generated the training data [146]. A good generative model is able to generate new data that resemble the training data in some sense. Especially in the field of neural network research this type of learning algorithm has gained wide-spread attention and is an active field of research. Common approaches belonging to this group are, for example, autoencoders, deep belief networks, generative adversarial networks (GAN), self-organizing maps (SOM), and hebbian learning.

- **Reinforcement learning** (RL) is an area of machine learning where a software agent needs to learn to take actions in an environment through maximizing some type of cumulative reward, which usually is not available right away, but often is presented to the agent with some delay [334]. It differs from supervised learning in that the observation/target pairs $(\mathbf{x}_i, y_i)$ are typically unavailable. Further, sub-optimal actions are often not corrected explicitly. Reinforcement learning commonly involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge) [176].

Besides these three main learning categories there are various generalizations or hybrids of the aforementioned learning types [228], for example:

- **Semi-supervised learning** [49, 386] Semi-supervised learning is a hybrid approach falling between supervised learning and unsupervised learning. As mentioned above, obtaining larger amounts of labeled data is usually unfeasible due to the cost associated with the labeling process, whereas acquisition of unlabeled data nowadays is often relatively cheap and simple. Semi-supervised learning leverages this situation by training on a large amount of unlabeled data in an unsupervised fashion, supported by supervised learning (or other types, such as reinforcement learning) on a small amount of labeled data. A number of researchers found that this approach produces considerable improvements in performance [69, 74, 382, 383, 384].

- **Active learning** [133] is a special form of semi-supervised learning in which the learning algorithm interactively queries an information source, e. g. the labeler or user, to obtain the required output (i. e. label or target value) at a new data point [311]. The most important question in this context is how to select the subset of data to be labeled [274], and there exist a wide variety of algorithms for this purpose, such as uncertainty sampling, query by committee, expected model change, etc. [239].

- **Transductive inference** was introduced by V. Vapnik [93] and is a form of reasoning from observed, specific, and labeled training data points to specific, unlabeled test data points. In this sense, it is similar to semi-supervised learning. However, the objective of transductive inference is to predict labels only for these particular test points [61, 228]. Transductive inference can yield predictions where *induction* fails, since the latter requires solving a more general problem before solving a more specific problem, i. e. reasoning from observed training cases to general rules, which are then applied to the test cases [49, 93].

- **On-line learning** refers to a machine learning approach in which the model is updated sequentially as new training data is available [29]. This is in contrast

to so-called *batch training*, where the model is trained on the full training set at once (as e. g. in second-order stochastic learning methods). Online learning is widely used in many areas of current machine learning research, for reasons which will be discussed in Section 5.7, but also because it allows training on very large data sets, which do not fit into computer memory all at once. In fact, a variant of online learning using *mini-batches* (i. e. a small number of observations or feature vectors) combined with backpropagation leads to Stochastic Gradient Descent (SGD), which currently is the most common training method for artificial neural networks.

It should be noted that the terms "model" and "learning" often are used loosely and interchangeably; for example, an unsupervised model basically refers to a model being trained in an unsupervised fashion, while the same model could possibly be trained in a supervised manner as well. This wide-spread use of terminology merely puts the model under consideration into a certain context.

5.3 Fundamental Neural Network Structure

Neural networks can be deployed as generative or discriminative models and can basically always be viewed as predictors or generators of values. The fundamental difficulty of leveraging neural networks is to find a topology (structure) and an appropriate training method to *learn* the model parameters, such that the predictions made by the model best match the underlying statistical distribution from which the values originate. The Kolmogorov theorem [190] in combination with the Universal Approximation theorem [65] essentially state that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate an arbitrary continuous function on compact subsets of $\mathbb{R}^N$, under mild assumptions on the activation function. However, it does not tell one how to determine the relevant parameters [82]. This section lays the foundation for the more complex model topologies described later in this chapter, by giving a mathematical definition of the term *neuron*, the atomic component of almost any current neural network model. It describes its parts and function and how it is used to form larger networks.

5.3.1 The Artificial Neuron

A single artificial neuron is commonly defined as a unit that performs the following computation [26]:

$$\phi(a) = \phi\left(\sum_{i=1}^{D} w_i z_i + b\right) = \phi\left(\mathbf{w}^\intercal \mathbf{z} + b\right), \tag{5.1}$$

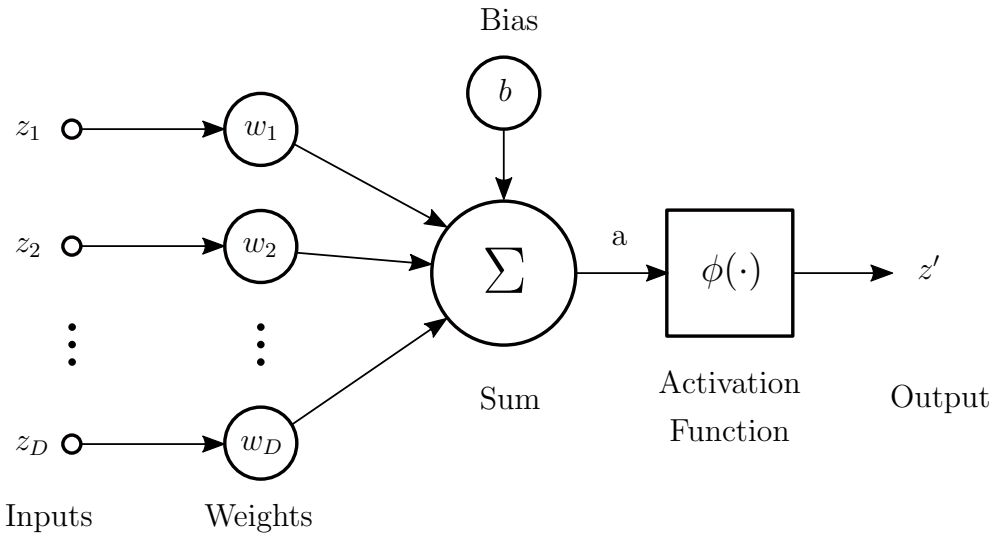


Figure 5.2: Schematic illustration of an artificial neuron.

i. e. the *activation* a of the neuron is obtained by forming a weighted linear (affine) combination of the D input values z_i with the weights w_i and adding the *bias* value b. The output of the neuron is then computed by feeding a to the *activation function* ϕ. There exists a multitude of activation functions, both linear or non-linear, and the most common ones will be discussed in Section 5.3.2. The number D of inputs z_i to a particular neuron is also referred to as *fan-in*. This term is commonly used when deriving appropriate weight initialization schemes (cf. Section 5.7.4). An example of an artificial neuron (5.1) is visualized in Figure 5.2.

When comparing this mathematical definition to biological neurons one can observe the analogy of the weighted linear transformation $\mathbf{w}^\intercal\mathbf{z}$ to the dendrite-synapse counterpart (cf. Section 5.1): Dendrites are the projections of neurons that serve to propagate the electrical stimulation received from other neural cells to its cell body (soma). These electrical signals are transmitted through an electro-chemical process from a large number of connected neurons to the dendrites via synapses, which are located at various points throughout the dendritic tree. Simply stated, each dendrite performs a multiplication by that dendrite's weight value, as modeled by w_i in (5.1), which is realized by adapting the amount of neurotransmitters in the synapses w. r. t. the strength of the input signal. Negative weights w_i can be used to model inhibition effects inside the synapses. Last but not least, the bias b can be justified from a biological point of view, representing the potential offset, or just as a mathematical commodity, in order to obtain beneficial mathematical and algorithmic properties. In physiological neurons the axon effectively samples the electrical potential resulting from the summation inside the soma and once reaching a certain potential, it will transmit a signal pulse down its length - it is said that it

fires. This temporal aspect of information processing and more detailed modeling of the processes inside a biological neuron will lead to the completely different paradigm termed *Spiking Neural Networks* [28, 212, 267]. Nonetheless, several non-linear processes can be observed in biological neurons, whose effects are attempted to be approximated by the non-linearity of the activation function ϕ [272].

5.3.2 The Activation Function

As seen in (5.1) the activation function ϕ takes the affine transformation of the neuron's input and applies a specific function on it, thus defining the output behavior of each node. While linear activation functions are used in certain circumstances, most of the time one desires to introduce non-linear properties to the neuron as a computational unit. This allows neural networks to learn a much wider range of complex, non-linear mappings between its in- and outputs and to become much more powerful function approximators. Many activation functions map the input to a limited output range, effectively compressing the activations, which is desirable in many cases, since it keeps the dynamic range of values under control. In order to allow backpropagation (cf. Section 5.7.1) to work, activation functions should satisfy two constraints: They should be *continuously differentiable*, i. e. their derivative should exist (ideally in closed form) everywhere, and they should be *monotonic* functions. In the following a number of common activation functions are defined and briefly described, to the extent needed for the work discussed in this thesis.

Linear (Identity)
The simplest activation function is the *linear*, or identity, function $\phi_{linear}(z)$ defined as:

$$\phi_{linear}(z) = z \tag{5.2}$$
$$\phi'_{linear}(z) = 1 \tag{5.3}$$

with $\phi'_{linear}(z)$ being its derivative. $\phi_{linear}(z)$ is depicted in Figure 5.4a. As (5.3) shows, the derivative of the linear activation function is constant, which means that the gradient has no relationship to its input z. As will be shown in Section 5.7.1 any parameter update will therefore be constant, which is often undesirable. Further, as alluded to above, the lack of non-linearity leads to a severe limitation in the capacity of the neuron to approximate more complex, interesting functions.

Sigmoid
One of the most widely known and adopted activation functions is the *sigmoid*, or logistic, function $\sigma(z)$

$$\phi_{sigmoid}(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{5.4}$$

$$\phi'_{sigmoid}(z) = \sigma'(z) = \sigma(z)(1 - \sigma(z)) \tag{5.5}$$

where $\phi'_{sigmoid}(z)$ denotes the derivative. $\phi_{sigmoid}(z)$ is depicted in Figure 5.3a and as one can see, its output values are limited to the range $[0, 1]$. Further, it saturates towards the tails of the function, i. e. big input values z are compressed. This saturation characteristic, however, can give rise to the so-called *vanishing gradient* problem during training (cf. Section 5.7.1): If the input values z to $\phi_{sigmoid}(z)$ are too big, e. g. due to inappropriate weight initialization, the output values are either close to 0 or 1 and the gradients become close to zero. In consequence, this leads to the training to be drastically slow or even stop. If one manages to control this situation, however, the gradient is very easy and fast to compute, since it uses the value $\sigma(z)$ of the activation itself, as can be seen from (5.5).

Hyperbolic Tangent

An alternative to the sigmoid activation function is the *hyperbolic tangent (tanh)*, given by

$$\begin{aligned} \phi_{tanh}(z) = \tanh(z) &= \frac{e^z - e^{-z}}{e^z + e^{-z}} \\ &= 2 \cdot \sigma(2z) - 1 \end{aligned} \tag{5.6}$$

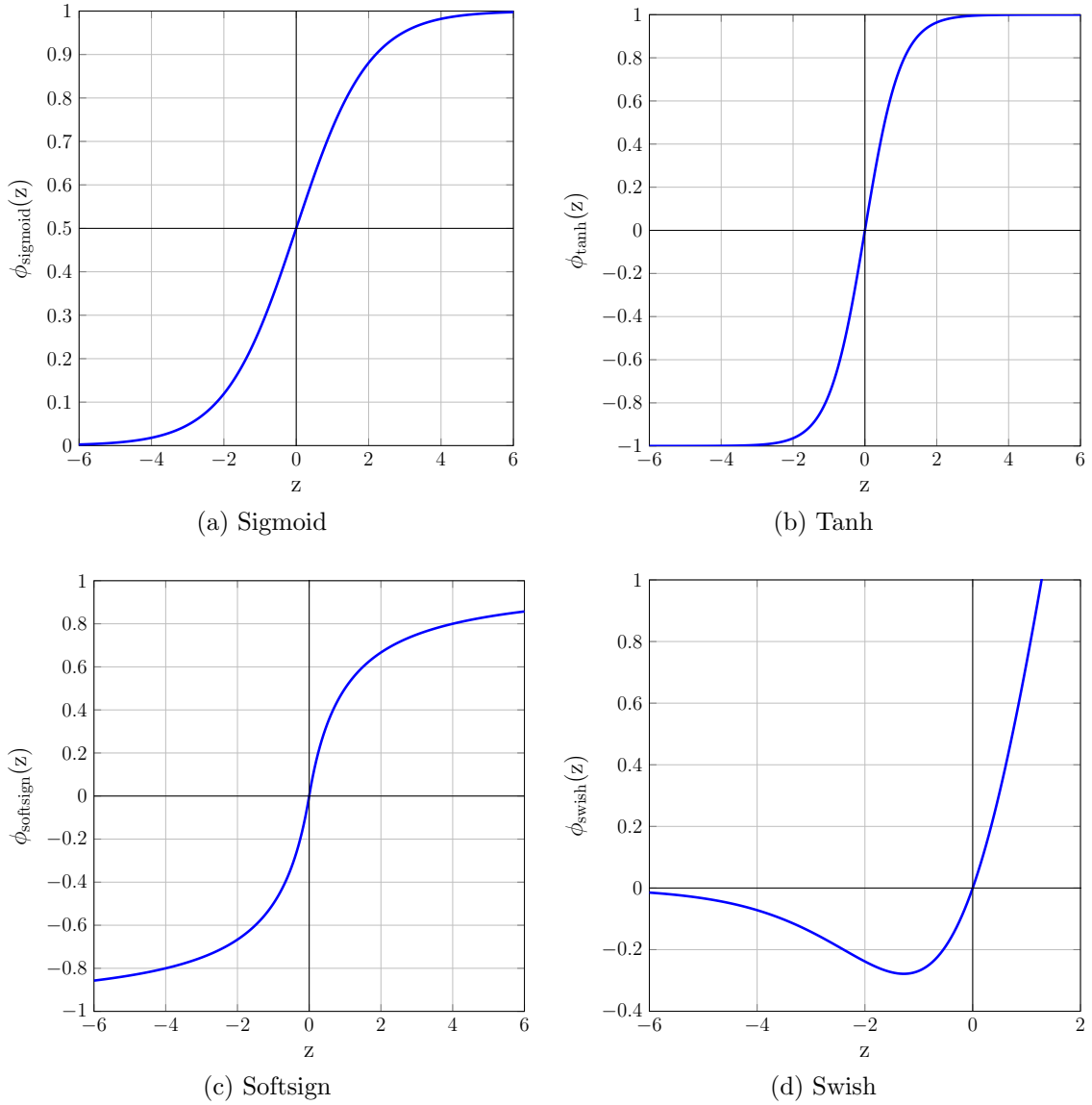
$$\phi'_{tanh}(z) = \tanh'(z) = 1 - \tanh^2(z) \tag{5.7}$$

again with $\phi'_{tanh}(z)$ representing the derivative of $\phi_{tanh}(z)$. It's graph is shown in Figure 5.3b. As can be observed from (5.6) there exists a simple relationship to the sigmoid function (5.4). In fact the hyperbolic tangent has the same s-shape, but $\phi_{tanh}(z) \in [-1, +1]$ and it is symmetric around $(0, 0)$. This symmetry and the fact that $\phi_{tanh}(z)$ can take on negative values have been found to be advantageous in some neural network topologies, e. g. for parts of Long Short-Term Memory networks (cf. Section 5.5.2), because it allows the gradients to flow backwards [104]. Nonetheless, the hyperbolic tangent potentially may also suffer from the vanishing gradient problem.

Softsign

The *softsign* activation function was initially proposed by Bergstra and colleagues [23] as a refined descriptive model of the visual area V1 of the human cortex and a replacement of the affine-sigmoidal function (5.4). In particular it offers a gentler non-linearity towards the tails than sigmoid or *tanh* functions, as can be seen from Figure 5.3c. It approaches its asymptotes much more slowly by design, since its tails are quadratic polynomials rather than exponentials. Since the detailed formulation is rather involved, an approximation was proposed as

$$\phi_{softsign}(z) = \frac{z}{1 + |z|} \tag{5.8}$$



(a) Sigmoid (b) Tanh

(c) Softsign (d) Swish

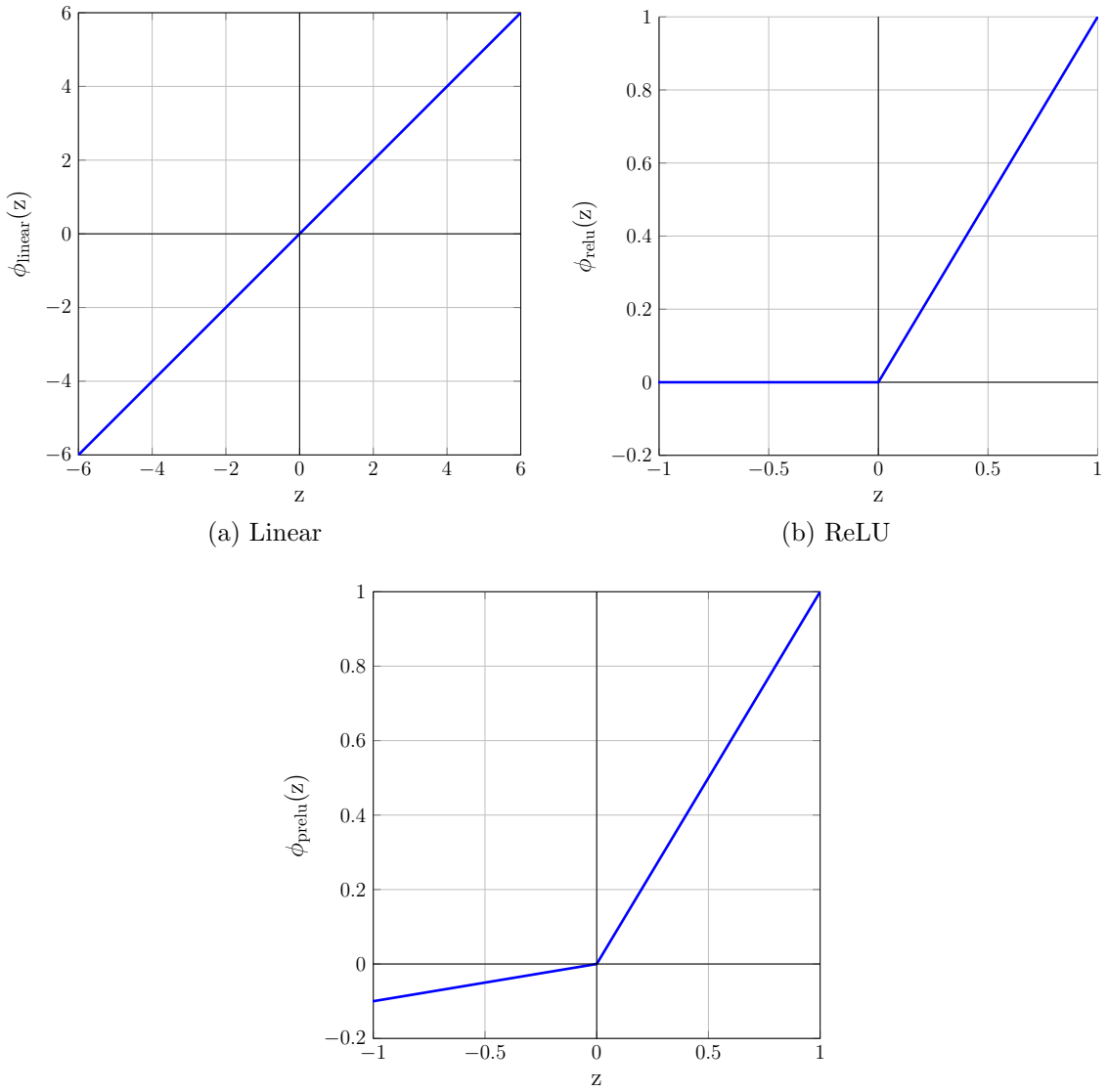
Figure 5.3: Common non-linear activation functions.

$$\phi'_{softsign}(z) = \frac{1}{(1 + |z|)^2} \tag{5.9}$$

Thus, $\phi_{softsign}(z)$ is similar to the hyperbolic tangent, but due to its smoother asymptotes potentially behaves differently regarding saturation.

Rectified Linear Unit (ReLU)

Even though the *Rectified Linear Unit (ReLU)* was first published by Hahnloser and colleagues [124, 125], it wasn't until the publication by Nair and Hinton [232] that it



(a) Linear

(b) ReLU

(c) PReLU ($\alpha = 0.1$)

Figure 5.4: Linear and piecewise-linear activation functions.

was intensively studied in many different machine learning areas, leading to a number of improvements, e. g. in image recognition [105] or speech recognition [211, 344]. It currently is the most widely used activation function in deep neural networks [201]. The ReLU is defined as

$$\phi_{relu}(z) = \begin{cases} 0 & \text{for } z < 0 \\ z & \text{for } z \geq 0 \end{cases} \tag{5.10}$$

$$\phi'_{relu}(z) = \begin{cases} 0 & \text{for } z < 0 \\ 1 & \text{for } z > 0 \end{cases} \tag{5.11}$$

and its graph is depicted in Figure 5.4b. It is essentially a half-wave rectified version of the linear function (5.2) and offers a number of desirable properties: It reduces the vanishing gradient problem w. r. t. the sigmoid and *tanh* activation functions, because it saturates only for $z \leq 0$. Furthermore, it is computed very efficiently, which allows for fast and effective training, even on very large data sets. In addition, the ReLU shows sparse activation, both for randomly initialized as well as for pre-trained networks: If the weights $\mathbf{w}$ in (5.1) of a specific neuron are updated in a way such that the affine transform $\mathbf{w}^\intercal \mathbf{z} + b \leq 0$, then the activation of that neuron will always be zero, i. e. $\phi_{relu}(z) = 0$, and hence the neuron will remain in-active and not contribute to any subsequent network computation. With an appropriate implementation this can lead to large computational savings using ReLU networks.

However, it is exactly this characteristic which also poses the biggest potential problem, leading to what is referred to as the *dying ReLU problem*: If the activation of a ReLU neuron is always zero, then no gradients flow backwards through the neuron during backpropagation and hence no parameter updates will occur on the weights and biases of this neuron. This effectively is another flavor of the already encountered vanishing gradient problem. While this sparsity can lead to computational savings it also effectively decreases the model capacity, i. e. the ability of the neuron or network to approximate certain functions. Another critique the ReLU sometimes suffers is its non-differentiability at zero, see (5.11); in practice, however, this is easily solved by arbitrarily assigning either 0 or 1 at $z = 0$. Finally, the ReLU is unbounded for $z > 0$, which can occasionally lead to problems during training, for example excessively large gradients. This situation can, however, be resolved by gradient clipping or similar counter-measures.

Parametric Rectified Linear Unit (PReLU)
In an attempt to mitigate the dying ReLU problem the Leaky ReLU [211] was proposed and later generalized by He and colleagues [136] to the *Parametric Rectified Linear Unit (PReLU)*, which is defined as

$$\phi_{prelu}(\alpha, z) = \begin{cases} \alpha z & \text{for } z < 0 \\ z & \text{for } z \geq 0 \end{cases} \tag{5.12}$$

$$\phi'_{prelu}(\alpha, z) = \begin{cases} \alpha & \text{for } z < 0 \\ 1 & \text{for } z \geq 0 \end{cases} \tag{5.13}$$

where $\alpha > 0$ accounts for a small positive slope for $z < 0$. The graph of the PReLU is shown in Figure 5.4c for $\alpha = 0.1$, a rather large value just chosen for graphical

clarity. The Leaky ReLU is obtained for $\alpha = 0.01$. The PReLU is an interesting extension to the ReLU, showing some improvements on image classification tasks at almost no computational cost [136]. Yet there is an ongoing debate about its usefulness due to inconsistent improvements on other tasks.

Sigmoid-Weighted Linear Unit (Swish)
As an alternative improvement of the ReLU, researchers from the Google Brain team proposed a new activation function called *Swish* [264], defined by

$$\phi_{swish}(z) = z \cdot \sigma(z) \tag{5.14}$$
$$\phi'_{swish}(z) = \phi_{swish} + \sigma(z)(1 - \phi_{swish}). \tag{5.15}$$

They demonstrated that by simply replacing all ReLU activation functions with Swish units they managed to improve the state-of-the-art results on a number of challenging image recognition tasks. Figure 5.3d shows the activation function around the origin. Contrary to other activation functions, it is a smooth, non-monotonic function and it is conjectured that precisely this non-monotonicity is responsible for the improved behavior. One can view (5.14) as a *self-gating* mechanism, where the input value z is gated by the sigmoid $\sigma(z)$.

Maxout
One particular, less often used, activation function is the *maxout* function [110], which is defined as

$$\phi_{maxout}(\mathbf{z}) = \max_i z_i \tag{5.16}$$
$$\frac{\partial \phi_{maxout}}{\partial z_j} = \begin{cases} 1 & \text{for } j = \operatorname{argmax}_i z_i \\ 0 & \text{for } j \neq \operatorname{argmax}_i z_i \end{cases} \tag{5.17}$$

Hence, $\phi_{maxout}(\mathbf{z})$ returns the maximum of its input values. The basic idea of maxout is to learn a piece-wise linear function, which is linear everywhere, except for $J - 1$ points, where J is the number of inputs of the maxout. According to the Stone-Weierstrass theorem such a function can approximate any continuous function and Goodfellow and colleagues [110] proof that any continuous function can be approximated arbitrarily well by a maxout network with two maxout units, since the maxout selects the maximum from k models (via its inputs). Maxout activation functions are often found in conjunction with convolutional neural networks, which are explained in Section 5.4.

Softmax

All previous activation functions result in exactly one output value, which in most cases serves as input to other neurons. However, one of the most widespread and useful activation functions for classification is the multinomial *softmax* function, given by

$$\phi^{(i)}_{softmax}(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \qquad \text{for } i = 1, ..., C \tag{5.18}$$

$$\frac{\partial \phi^{(i)}_{softmax}}{\partial z_j} = \phi^{(i)}_{softmax}(\mathbf{z})(\delta_{ij} - \phi^{(j)}_{softmax}(\mathbf{z})) \tag{5.19}$$

where $\partial \phi^{(i)}_{softmax} / \partial z_j$ denotes the partial derivative of the the i^{th} output value w. r. t. input z_j. The most important property of the softmax function is that it takes an un-normalized vector $\mathbf{z}$ and normalizes it so that each output element $\phi^{(i)}_{softmax}(\mathbf{z})$ is in the interval $[0, 1]$ and the sum of all output values sum up to one, i. e.

$$\phi^{(i)}_{softmax}(\mathbf{z}) \in [0, 1] \quad \forall\, i \tag{5.20}$$

$$\sum_{i=1}^{C} \phi^{(i)}_{softmax}(\mathbf{z}) = 1 \tag{5.21}$$

These properties allow the outputs to be interpreted as probabilities [26], and therefore the softmax is often used as the last layer in neural network-based classifiers to map un-normalized intermediate values to a probability distribution over predicted output classes.

For the case of two classes, i. e. $C = 2$, there exists the following interesting equivalence between the softmax and the the sigmoid activation function:

$$\begin{aligned} \phi^{(1)}_{softmax}(z_1, z_2) &= \frac{e^{z_i}}{e^{z_1} + e^{z_2}} \\ &= \frac{1}{1 + e^{-(z_1 + z_2)}} \\ &= \phi^{(1)}_{sigmoid}(z_1 - z_2) \end{aligned} \tag{5.22}$$

$$\phi^{(2)}_{softmax}(z_1, z_2) = \phi^{(2)}_{sigmoid}(z_2 - z_1) \tag{5.23}$$

This means that for binary classification both softmax or sigmoid activation functions can be used equivalently. One advantage of the sigmoid over the softmax, however, is that using the former with a threshold allows for tuning the decision of the winning class by selecting a threshold different from 0.5.

5.3.3 Feed-Forward Neural Network

By composing multiple neuronal units, each of which computes (5.1), much more powerful function approximators can be built than what could be achieved by a single neuron alone. Building a neural network on the one hand is accomplished by duplicating a neuron and connecting the copies to the same inputs (or a subset of them), but with separate weights and biases - this collection of computational units is commonly referred to as a *layer*. On the other hand the neuronal outputs can be used as inputs to other neurons[2], forming a series of functional transformations. The organization of neurons into layers and stacking these layers into networks constitutes a basic design principle for many neural network topologies in use today and in the following a number of them will be explained in detail.

The simplest possible network consists of an input layer, to which the input values $\mathbf{x}_i$ (e. g. feature vectors) are fed, and an output layer, which generates some prediction $\hat{\mathbf{y}}_\mathbf{i}$. More powerful networks add one or more so-called *hidden* layers in-between, which increases the capacity and the predictive power of the network. Figure 5.5 shows an illustration of a feed-forward network with two hidden layers. In such a network there are no interconnections between neurons of the same layer and the information flow is uni-directional from input to output. Again, the reader is referred to later sections for more elaborate extensions of this type of network.

Extending the formulation of a single neuron (5.1) to the general case, let

$$a_j^{(l)} = \sum_{i=0}^{D} w_{ji}^{(l)} z_i^{(l)} \tag{5.24}$$

be the activation of neuron j in layer l for D input values $z_i^{(l)}$. Here we have absorbed the bias $b_j^{(l)} = w_{j0}^{(l)}$ into the weight vector by defining an additional input variable z_0 whose value is clamped at $z_0 = 1$. Each of the activations $a_j^{(l)}$ is then transformed using the activation function $\phi(\cdot)$ to obtain $z_j^{(l+1)} = \phi^{(l)}(a_j^{(l)})$. Note that the activation functions do not need to be identical for all neurons in the network. As pointed out before the outputs $z_j^{(l+1)}$ often serve as the input for the next layer $l+1$ in the network. The weights (and biases) $w_{ji}^{(l)}$ can be subsumed in the weight matrix $\mathbf{W}^{(l)}$ and the same can be done for the inputs, activations, and outputs of any layer. This results in the shortened notation:

$$\mathbf{z}^{(l+1)} = \phi^{(l)}(\mathbf{W}^{(l)}\mathbf{z}^{(l)}) \tag{5.25}$$

[2]The number of neurons a particular neuron is connected to is referred to as *fan-out*. This will be of importance for weight initialization schemes discussed in Section 5.7.4.

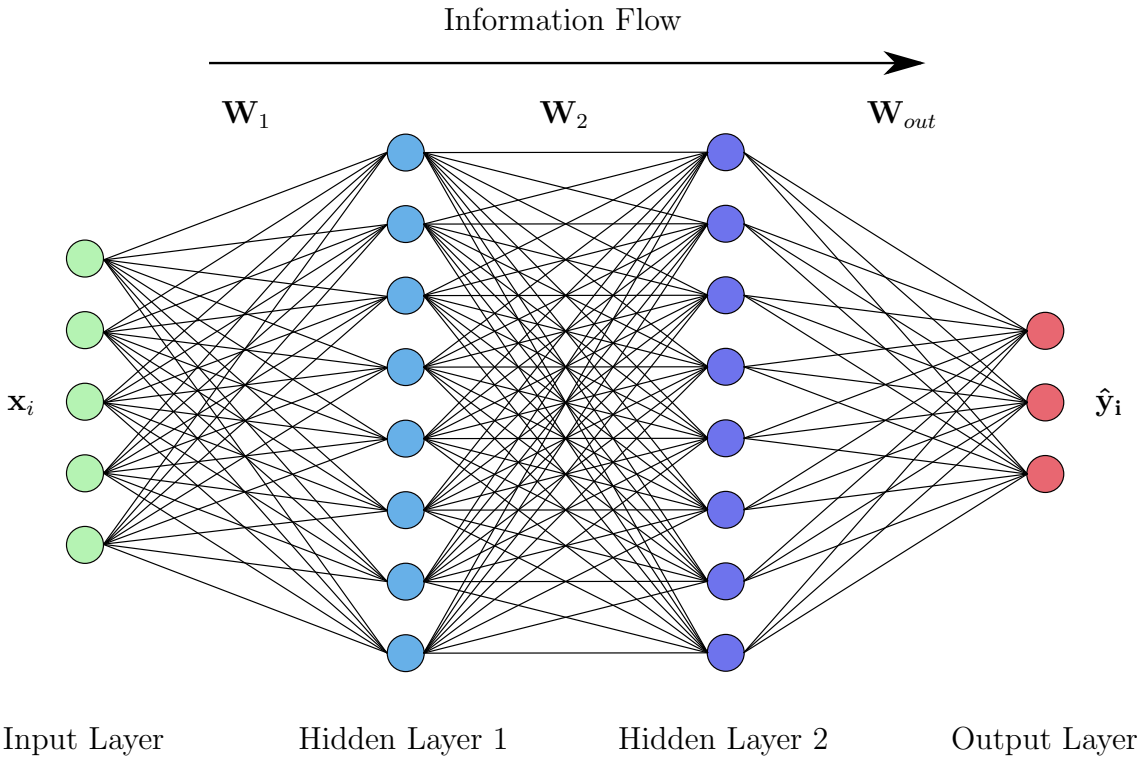


Figure 5.5: Illustration of a feed-forward neural network with two hidden layers.

where $\phi^{(l)}(\cdot)$ is applied element-wise. In the following the superscript l, indicating the layer, will be dropped where confusion is unlikely. Combining the various layer computations, an L-layer *feed-forward neural network* computes the function $\hat{\mathbf{y}}(\mathbf{x})$

$$\hat{\mathbf{y}} = \phi^{(L)}(\mathbf{W}^{(L)}\phi^{(L-1)}(\mathbf{W}^{(L-1)} \dots \phi^{(1)}(\mathbf{W}^{(1)}\mathbf{x}))) \tag{5.26}$$

The layers with indices $l = 1, \dots, L-1$ are the hidden layers and one generally refers to a network as a *deep neural network (DNN)* if it has at least two hidden layers. For historic reasons, networks with one hidden layer are also called *multi-layer perceptrons* (MLP) [273].

5.4 Convolutional Neural Networks

In this section, a special class of DNN called *convolutional neural network* (CNN) will be described, which have played a very important role in the history of deep learning. They were one of the first effective deep neural network models, long before the advent of modern deep learning, and were also successfully used in early commercial applications [200]. CNNs are exceptional representatives of biologically

inspired artificial intelligence, with its history dating back to the Nobel prize winning work by the neurophysiologists Hubel and Wiesel describing how the mammalian vision system works in the primary visual cortex (V1) [161, 162, 163]. CNNs are designed to roughly capture the following aspects of their findings [109]:

- V1 is organized as a two-dimensional spatial map mirroring the structure of the image in the retina. CNNs accordingly arrange their features in terms of 2-D maps.

- V1 contains a great amount of so-called *simple cells*, which can be approximately modeled by a linear function of the image in small and localized neighborhoods, termed *receptive fields*. Neurons tuned to a particular receptive field exhibit strong activation, while not responding to other stimuli; hence its name. The kernels of CNNs, as described below, emulate these properties.

- Besides these simple cells, V1 also contains *complex cells*, which respond to more complex patterns in the input image and in particular exhibit some invariance to its position, scale, and rotation. This characteristic inspired the use of pooling, which will be described later in this section.

The repetition of subsequent alternations of detection via kernels and pooling is conjectured to exist as one moves deeper through multiple anatomical areas of the cortex, eventually leading to individual cells which respond very specifically to particular concepts and which are invariant to strong transformations of the input. These cells, for example, have been shown to exists in the medial temporal lobe of the human brain [261].

Fukushima [91, 92] proposed a hierarchical, multilayered neural network, called the *neocognitron*, which is capable to learn robust visual pattern recognition and which incorporates most of the design elements of CNNs. However, it used an unsupervised, layer-wise clustering learning algorithms. About the same time Lang and Hinton [195] introduced supervised training via backpropagation to train *time-delay neural networks* (TDNN), which essentially can be viewed as one-dimensional CNNs applied to time series. Based on the success of this work LeCun and colleagues [200] developed and proposed the basic CNN structure and trained it on image objects via 2-D convolutions. Historically, the evolution of CNNs was strongly motivated and influenced by image recognition problems, but recently has also gained strong popularity in other fields, such as ASR [3, 4, 101, 188, 279] or computational paralinguistics [302, 305, 336, 346, 385].

Before describing the components of a CNN model in detail, it is helpful to observe some fundamental characteristics of convolutional networks and differences to conventional feed-forward networks. Due to the full connectivity of the latter

and the large number of free parameters this entails, they often suffer from the *curse of dimensionality*, which either requires very large amounts of data or quickly leads to overfitting on smaller data sets. As will be become apparent below, due to *weight sharing* this is much less of a problem in CNNs. Further, FF-DNN treat input features which are far apart (both in time and frequency) in the same way as features which are close to each other. In contrast, CNNs take into account the spatial structure of data, enforcing a local connectivity pattern between neurons of adjacent layers, i. e. each neuron is connected to only a small, coherent region of the previous layer [121]. This characteristic is important when selecting the type of input features, as for a CNN to work well the features should be locally correlated (e. g. as a spectrogram or log-mel features). Decorrelated features, such as MFCC, instead are less suited and often lead to sub-optimal results [165].

5.4.1 The Convolution Operation

As the name *convolutional* neural network implies, the central mechanism of a CNN is constituted by the operation of convolution, which in general is defined as

$$\int i(u)\, k(v-u)\, du, \tag{5.27}$$

with $i(\cdot), k(\cdot) \in \mathbb{C}$. In the discrete domain, if $k(\cdot)$ has finite support in $\{-M, ... + M\}$, the (discrete) convolution can be re-written as

$$(i * k)(n) = \sum_{m=-M}^{M} i(m)\, k(n-m). \tag{5.28}$$

In practice, however, the convolution is commonly replaced by cross-correlation, given by

$$o(n) = (i * k)(n) = \sum_{m=-M}^{M} \overline{i(m)}\, k(n+m). \tag{5.29}$$

For the scope of this thesis, and of most of current research described in the literature, $i(\cdot), k(\cdot) \in \mathbb{R}$ and hence (5.28) and (5.29) only differ in the fact that the latter inverts the order of its coefficients. However, since $k(\cdot)$ is to be learned from data and therefore the ordering is irrelevant, cross-correlation can safely be used instead of convolution without affecting the results. In the common terminology of convolutional networks, $i(m)$ is referred as the *input* of the CNN and $k(n)$ as the *kernel*. The resulting output $o(n) = (i * k)(n)$ is usually called a *feature map*. In general, the inputs, kernels, and outputs are represented by multidimensional arrays (*tensors*) and the kernel (weights) are the free parameters to be learned from data. Specifically, in audio applications one often performs convolution (correlation) over mini-batches

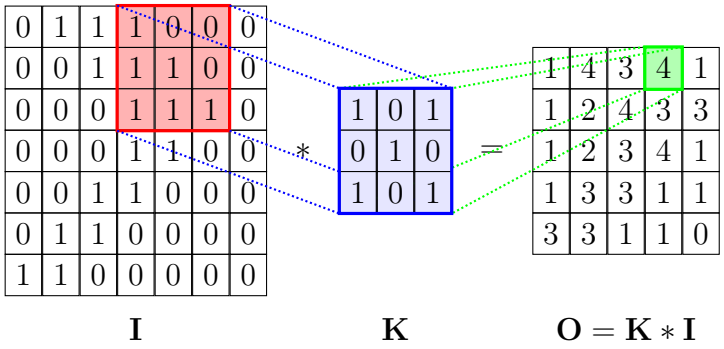


Figure 5.6: Illustration of the convolution operation performed by kernel **K** over the input **I**. For illustration purposes the input and kernel values are chosen to be binary.

of two-dimensional inputs; for example, a spectrogram or a sequence of mel-filterbank vectors form the input "image" **I** and the shape of the resulting input tensor is (mini-batch size, number of timesteps, number of frequency bands). The resulting 2-D cross-correlation operation over each item in a mini-batch is then given by

$$\mathbf{O}(i,j) = (\mathbf{K} * \mathbf{I})(i,j) = \sum_m \sum_n \mathbf{I}(i-m, j-n) \cdot \mathbf{K}(m,n), \tag{5.30}$$

where use of the commutative property is made and the fact is exploited that the input is assumed to be real. This formulation in fact reflects the most common implementation of convolution[3] in many deep learning frameworks, where the kernel **K** is sliding across the input **I** to compute the output feature map **O**. An illustration of this mechanism is given in Figure 5.6.

In summary, a convolutional layer computes its output as follows: A kernel (also called filter) **K** computes the dot product between the kernel weights and the input values to obtain the activation, which is stored in a feature map, at the current location. The area the kernel covers is referred to as the receptive field, in analogy to the biological archetype explained above. Then the kernel proceeds to the next location, which is determined by the *stride* S, the step size the kernel moves each time. A stride of one means that the kernel slides from one feature or time step to the next, while $S > 1$ results in sliding over the input in larger intervals, which leads to less overlap between the feature map outputs. Note that there can be separate strides for the two directions of the input. In order to capture different characteristics of the input each CNN layer usually computes several different feature maps using different kernels. These feature maps are then combined into the output tensor of the CNN for

[3]This thesis follows the convention of many deep learning libraries of implementing cross-correlation while calling it convolution.

further processing. In order to render the output non-linear, an activation function is applied to each element of this feature map tensor. Furthermore, *padding* might be added to the inputs **I**: the size of the feature maps is always smaller than the size of the inputs which leads to a continuous shrinking if one stacks several convolutional layers on top of each other. By padding the input with 0's (or some other appropriate values) one can compensate for this loss. Further, one can leverage padding to ensure that kernel sizes and strides match the input. Eventually, by introducing zeros between the kernel elements, called *dilation*, the size of the convolutional layer's receptive field can be increased and cover more relevant information [374]. This can be important for tasks which require more context to make a prediction. The main hyperparameters to chose, when defining a CNN model topology, are therefore the number of feature maps, also called the *depth* of the convolutional layer, the size of the kernel $\mathbf{K}$, the stride S, the padding applied to the input $\mathbf{I}$, and the dilation factor D. Based on these hyper-parameters the size of the feature maps for a particular convolutional layer can be computed as

$$O = \left\lfloor \frac{I - K + 2 \cdot P}{S} \right\rfloor + 1, \tag{5.31}$$

where O is the size of the (output) feature map, I the corresponding size of input $\mathbf{I}$, K the size of the kernel $\mathbf{K}$, P the the number of padding elements, and S the stride, for one dimension. Typically, these values are chosen to be identical for the two (or all) dimensions of the input, but this need not be so.

As Goodfellow and colleagues [109] point out, the convolution component of CNNs leverages three important aspects that presumably help to improve a machine learning system and possibly are the reason for the success of convolutional neural networks:

- *Sparse connectivity* is achieved by designing the kernel K to be smaller than the input I. This reduces the number of free, learnable parameters and hence leads to higher statistical efficiency of the network, e. g. smaller risk of overfitting. Further, the general network size becomes smaller, which is of interest for practical applications. Besides, this fact also implies that computing the outputs requires fewer operations: For m inputs and n outputs, feed-forward networks require computations on the order of $O(m \times n)$ per example, while their CNN counterparts require only $O(k \times n)$, where k is the number of kernel parameters. Since k is usually much smaller than m this leads to substantial computational savings.

- *Parameter sharing* refers to re-using the same (kernel) parameters for many computations in a model. This has the favorable implication that only one set of parameters needs to be learned for every location in the input feature map,

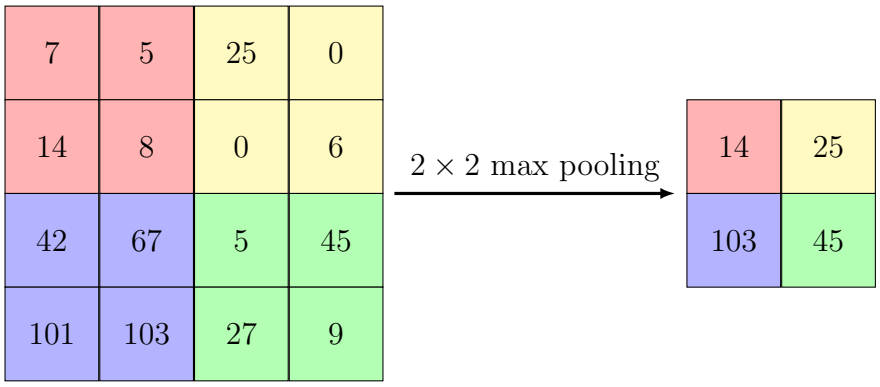


Figure 5.7: Illustration of 2×2 max pooling.

instead of many separate sets as done in conventional feed-forward networks. This dramatically can reduce the memory requirements of a CNN model w. r. t. dense neural networks in the order of magnitudes.

- *Equivariance* emerges from the parameter sharing of CNNs and describes the translation invariance caused by the convolution operation. It denotes the property that the output changes in the same way as the input, which is highly useful when one wants to detect certain structures, e. g. edges in a spectrogram due to harmonic structure, which appear many times in the input at different time and frequency locations.

5.4.2 Pooling

Pooling in a very important concept of convolutional neural networks and is a type of non-linear down-sampling of its input. Pooling typically is implemented by partitioning the input feature map into non-overlapping areas and computing a single output for each such sub-region. A pooling layer hence progressively reduces the extent of the feature map representation, which reduces the number of parameters and therefore the computational and memory requirements of the network. This further enhances the above-mentioned translation invariance and generally helps in controlling overfitting. Therefore, in many current CNN models a pooling layer is inserted after a convolutional layer. The most common pooling function is *max pooling* [31], which outputs the maximum of its inputs, and which is illustrated in Figure 5.7. Max pooling is essentially identical to the maxout activation function, defined by (5.16). However, many other pooling functions have been defined in the literature, for example average pooling, L_p-norm pooling, stochastic pooling, spatial pyramid pooling, etc. [119].

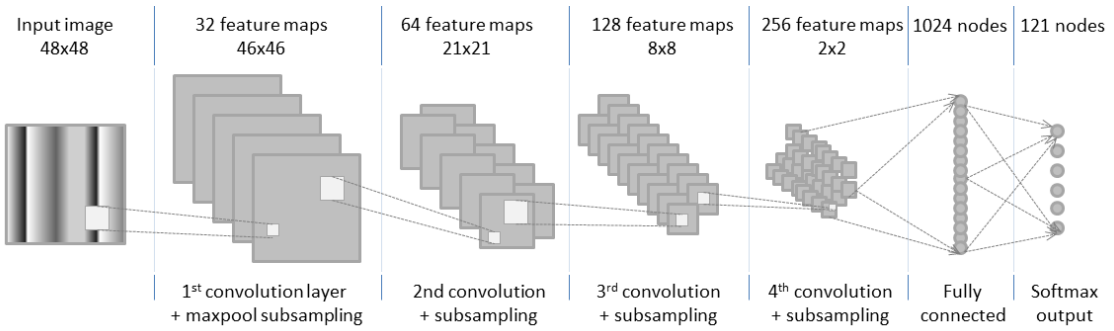


Figure 5.8: Example of full CNN model.

5.4.3 Model Structure

As alluded to above, a convolution layer is commonly followed by a pooling layer, to the degree that such a layer pair is often seen as a single CNN layer. Terminology in this regard varies in the literature. However, at the output of the last layer one or more *fully-connected* (FC) layers are usually appended, which are standard feed-forward layers. All (pooled) feature maps are flattened into a column vector, which is fed into the (first) FC layer. The FC layer component operates on the high-level features extracted by the initial layers of the CNN and tries to learn non-linear combinations of these features. Therefore, the FC layers represent the classification or regression stage performing high-level reasoning of the inputs. Further, they have full access to all feature maps which allows them to perform the non-local processing, which was intentionally hidden in the lower convolutional/pooling layers, but on higher-level features. Alternating convolutional and pooling layers, followed by dense fully-connected (or RNN) output layers, many different CNN architectures can be created. Figure 5.8 shows an example of such a network. In fact, a large number of models based on or incorporating CNNs have been proposed in the research literature in recent years, many of which were competition- and challenge-winning designs. Among the most prominent ones are AlexNet [194], ZF Net [378], GoogLeNet or Inception network [338], and VGGNet [313], just to name a few. A comprehensive review of recent advances in the research of CNNs can be found in [119].

5.5 Recurrent Neural Networks

The network structure described in the previous sections do not contain any recurrent cycles and therefore can be represented by a directed acyclic graph (DAG). If one instead allows cyclical connections as well, one obtains a class of models called *recurrent neural networks (RNN)*. RNNs essentially form a directed graph along a temporal sequence and one of their main characteristics is that they show temporally

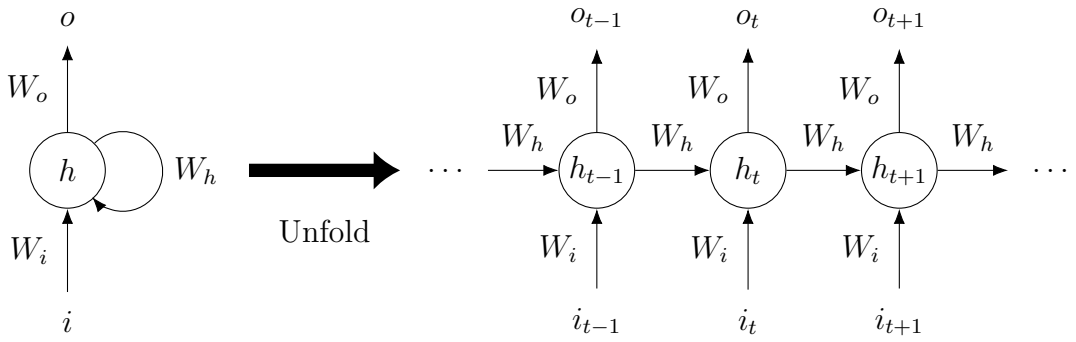


Figure 5.9: Illustration of the unfolding principle in a recurrent neural network.

dynamic behavior [244]. This makes them especially suitable for processing sequential data. While regular feed-forward networks only map input to output vectors, RNNs can leverage their internal state, which serves as a memory of the history of previous inputs and states, to predict the output. As pointed out by Graves [112] the equivalent result to the universal approximation theory for feed-forward networks is that an RNN with a sufficient number of hidden units can approximate any measurable sequence-to-sequence mapping to arbitrary accuracy [126]. A large variety of RNNs have been proposed in the research literature, including Jordan networks [173], Elman networks [81], Hopfield networks [213], Time-Delay Neural Networks [196] and Echo State Networks [169]. One commonality of all these networks is that they share the weights across multiple time steps.

Computing the forward pass in an RNN is analogous to that of a feed-forward network with a single hidden layer. The only exception is that the hidden layer state information from the previous timestep is added to the (external) input. Let $x_i(t)$ be the value of the i-th of I input units at time step $t = 1, \ldots, T$. Further, let J be the number of hidden units in the recurrent network layer. Then, the output of hidden unit k at each time step t is given by

$$h_k(t) = \phi\left(\sum_{i=1}^{I} w_{ki} \cdot x_i(t) + \sum_{j=1}^{J} w_{kj} \cdot h_j(t-1)\right). \tag{5.32}$$

This formulation differs from Equations (5.24) and (5.25) in the feed-forward network case only in additionally considering the input from the hidden layer outputs of the previous time step (apart from the slightly changed notation). The full sequence of hidden states is computed by starting at $t = 1$ and recursively applying (5.32), incrementing t at each time step. At $t = 1$ the hidden states $h_k(0)$ need to be initialized properly before performing any computations. Often this is done by setting $h_k(0) = 0$ for all hidden states, but it has also been found that the stability

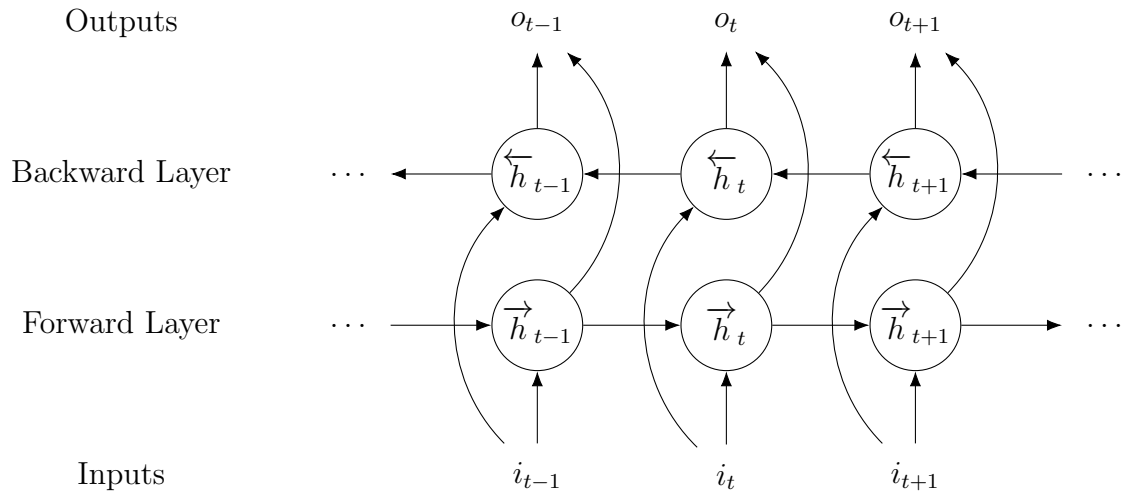


Figure 5.10: Illustration of a bi-directional recurrent neural network.

and performance can sometimes be improved by using nonzero initial values [388].

It is instructive to visualize RNNs by *unfolding* the update graph along the temporal input sequence. Figure 5.9 shows the idea behind it for a small neighborhood around the current time step t. The unfolded graph does not contain any cycles and this helps to generalize to networks with more complex update dependencies.

5.5.1 Bidirectional Recurrent Neural Networks

As shown in the previous section RNNs can theoretically access and make use of all past information presented to the network. However, in certain circumstances it is helpful to also have access to future input. This idea leads to *Bidirectional Recurrent Neural Networks* (BRNN), introduced by Schuster and Paliwal [308]. The basic idea behind them is to process the full input sequence in two separate recurrent layers: one layer in the conventional way (forward direction) for $t = 1, \ldots, T$, and one operating on the time-inverted sequence (backward direction), i. e. for $t = T, \ldots, 1$. After processing both directions both outputs are connected to the same output layer, as illustrated in Figure 5.10. This approach provides the complete past and future context of the input sequence to the output layer.

While this approach has shown to result in significant improvements over unidirectional RNNs in several domains, it suffers the problem of non-causality, which renders bidirectional algorithms unfeasible for many real-time algorithms and applications which require minimal delay. However, there also exist many problems, where the violation of causality is not an issue, for example if the input sequences

are of spatial nature or in situations where off-line processing of data is feasible. BRNNs have proven to be highly competitive in many research areas, such as text processing [266], clinical situations [252], speech recognition [114], and computational paralinguistics [37].

As an intermediate solution to the problem of non-causality and the induced delay, it is also possible to either process sub-segments of a longer sequence, e. g. segments between pauses in a speech utterance, or to leverage a limited temporal look-ahead of a certain duration, which is acceptable for a specific real-time application. This look-ahead (and its involved delay) represents the future context used on the forward and backward passes of the BRNN computations.

5.5.2 Long Short-Term Memory Neural Networks

As discussed in the previous sections, RNNs have the ability to leverage temporal context information in mapping input sequences to output sequences and theoretically they can keep track of arbitrarily long dependencies in the input. However, in practice the range of context that standard RNNs can make use of is fairly limited. Hochreiter [153, 155] found that the reason for this problem lies in the *vanishing gradient problem*: In backpropagation training (cf. Section 5.7.1) the update of each parameter in a neural network is proportional to the partial derivative of the loss function w. r. t. the parameter. Following the unfolding principle shown in Figure 5.9, each additional time step in the computation of the RNN is equivalent to adding another layer in a corresponding feed-forward network. However, this means that the impact of the input at a particular time step diminishes the further away the loss function is computed. Accordingly, the respective gradient of the loss w. r. t. to this input becomes vanishingly small, hence preventing the parameter from changing its value. Similarly the *exploding gradient problem* [20, 242] describes an excessive increase of the norm of the gradient due to the explosion of the long term components, which can grow exponentially more than short term ones, leading to unstable or divergent training [243].

Hochreiter and Schmidhuber [154] proposed the *Long Short-Term Memory* (LSTM) network in an attempt to partially solve the vanishing gradient problem by adding memory cells, self-loops, and input and output control gates which produce paths through which the gradients can flow over long durations. Later Gers and colleagues [98] introduced the forget gate, enabling the LSTM to reset its own state and to erase previously stored information. Finally, Gers and Schmidhuber [97] added so-called *peephole connections*, which are connections from the cell to the gates, which facilitates learning precise timings. Today, LSTM networks are the fundamental components of many of the most successful deep learning systems, both in the research community and industry. For a detailed description of several LSTM

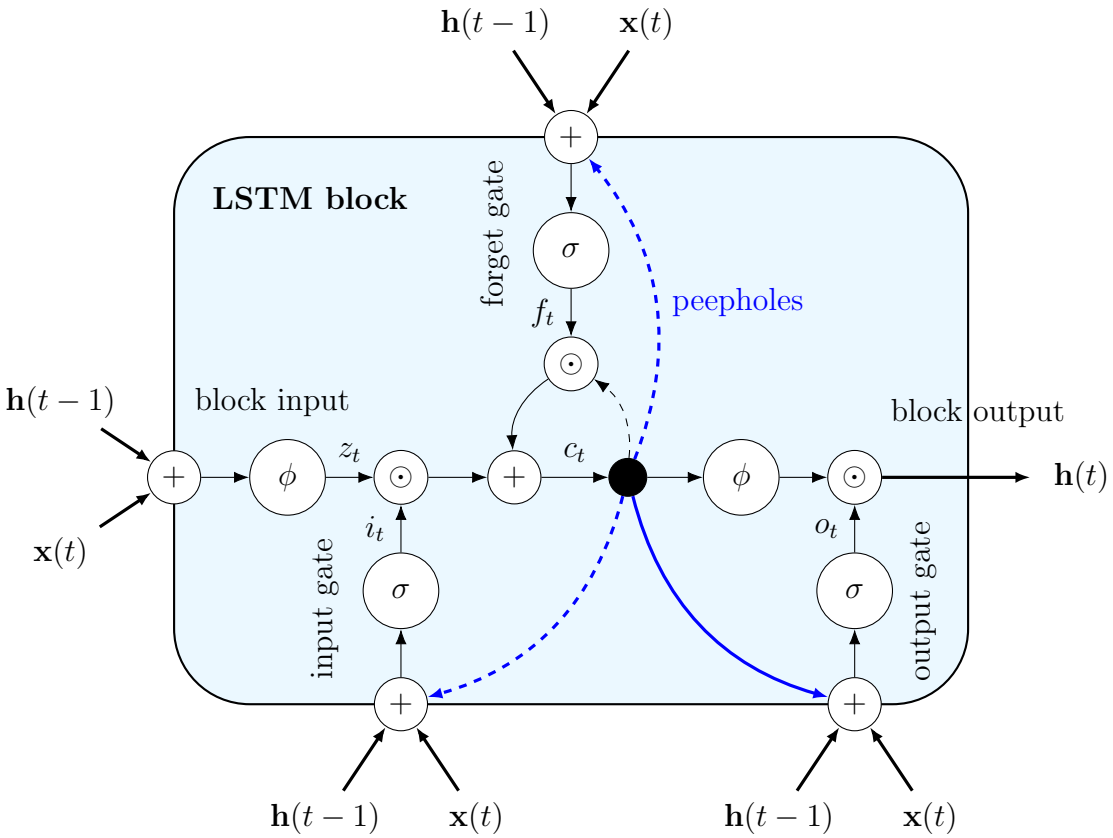


Figure 5.11: Long Short-Term Memory block with one memory cell and peephole connections. Dashed lines are connections with time-delay, which are only effective at the next time step $t+1$. Blue lines are the peephole connections. σ are gating activation functions, while ϕ are input and output activation functions. $\odot$ denotes element-wise multiplication.

variants, their historical evolution, and a summarization of their general performance the interested reader is directed to [115].

As already mentioned before, there exist several variants of LSTM networks, and since it is most relevant for the work in this thesis, an LSTM architecture with peephole connections will be described in the following. It consists of a number of recurrently connected *memory blocks*. Each such block contains one or more *cells* and three *gates*, the input, output, and forget gate, which control the read, write, and reset operations of the cells. Figure 5.11 presents a schematic illustration of an LSTM block with a single memory cell. An LSTM layer composed of these blocks is formally defined as follows [115]: Let $\mathbf{x}(t)$ be the input vector at time t, D be the dimensionality of the input vectors, and N be the number of blocks in an LSTM

layer. Then the parameters of this layer are defined as:

- Input weights: $\mathbf{W}_z, \mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o \in \mathbb{R}^{N \times D}$
- Recurrent weights: $\mathbf{R}_z, \mathbf{R}_i, \mathbf{R}_f, \mathbf{R}_o \in \mathbb{R}^{N \times N}$
- Biases: $\mathbf{b}_z, \mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o \in \mathbb{R}^N$
- Peephole weights: $\mathbf{p}_i, \mathbf{p}_f, \mathbf{p}_o \in \mathbb{R}^N$

Given these parameters the formula of an LSTM layer forward pass are given by

$$\mathbf{z}(t) = \phi(\mathbf{W}_z \mathbf{x}(t) + \mathbf{R}_z \mathbf{h}(t-1) + \mathbf{b}_z) \quad \text{block input} \quad (5.33)$$
$$\mathbf{i}(t) = \sigma(\mathbf{W}_i \mathbf{x}(t) + \mathbf{R}_i \mathbf{h}(t-1) + \mathbf{p}_i \odot \mathbf{c}(t-1) + \mathbf{b}_i) \quad \text{input gate} \quad (5.34)$$
$$\mathbf{f}(t) = \sigma(\mathbf{W}_f \mathbf{x}(t) + \mathbf{R}_f \mathbf{h}(t-1) + \mathbf{p}_f \odot \mathbf{c}(t-1) + \mathbf{b}_f) \quad \text{forget gate} \quad (5.35)$$
$$\mathbf{c}(t) = \mathbf{i}(t) \odot \mathbf{z}(t) + \mathbf{f}(t) \odot \mathbf{c}(t-1) \quad \text{cell state} \quad (5.36)$$
$$\mathbf{o}(t) = \sigma(\mathbf{W}_o \mathbf{x}(t) + \mathbf{R}_o h(t-1) + \mathbf{p}_o \odot \mathbf{c}(t) + \mathbf{b}_o) \quad \text{output gate} \quad (5.37)$$
$$\mathbf{h}(t) = \mathbf{o}(t) \odot \phi(\mathbf{c}(t)) \quad \text{block output} \quad (5.38)$$

where ϕ and σ are activation functions and $\odot$ denotes element-wise multiplication. The outputs $\mathbf{h}(t)$ are the inputs to any subsequent, higher layer, and simultaneously represent the state of the current hidden layer, which is fed into all LSTM blocks at the next time step $t+1$. From this formulation the number of parameters in a LSTM layer, Π_{LSTM}, can be deduced as

$$\Pi_{LSTM} = N \cdot (4 \cdot (N + D + 1) + 3), \quad (5.39)$$

where N is the number of (LSTM) units, D is the input dimensionality, the additional 1 accounts for the biases, and the 3 reflect the peephole connections.

The cell lies in the heart of an LSTM block and keeps track of the dependencies between the elements in the input. It is able to store and access information over long periods of time, thereby mitigating the vanishing gradient problem [112]. The input gate instead controls the information flow into the cell, while the output gate determines to which extent the cell state contributes to the output of the LSTM block. Finally, the forget gate controls the keep/forget characteristics of the cell. For example, if the input gate is closed, $i(t) = 0$, no input arrives at the cell and hence $c(t)$ remains uninfluenced by the input. The forget gate can reduce the current cell state value, by multiplying it with a value less then 1 or keeping it unchanged by setting $f(t) = 1$. Eventually, if $o(t) = 0$, no output is generated from the LSTM block.

All gates operate as a kind of *information throttle* and hence should output a value between 0 and 1, which multiplicatively controls the information flow through the LSTM block. For this reason the activation function typically adopted for

this purposed is the sigmoid function (5.4), as shown in Equations (5.34), (5.35), and (5.37). On the other hand, the activation function ϕ, which operates on the block input, should be of a type whose second derivative can sustain for a long range before going to zero, in order to overcome the vanishing gradient problem. The tanh activation function (5.6) is therefore commonly used.

5.5.3 Gated Recurrent Units

Based on the wide-spread use and the success of LSTM architectures, researchers have sought to identify the elements of LSTM which are really necessary and those which might be simplified or even eliminated, for example in order to reduce the number of parameters to fight overfitting or to reduce the computation time both during inference and training. One such alternative is the *Gated Recurrent Unit (GRU)* [53, 55, 56], which is defined as

$$\mathbf{u}(t) = \sigma(\mathbf{W}_z\mathbf{x}(t) + \mathbf{R}_z\mathbf{h}(t-1) + \mathbf{b}_z) \qquad \text{update gate} \qquad (5.40)$$

$$\mathbf{r}(t) = \sigma(\mathbf{W}_r\mathbf{x}(t) + \mathbf{R}_r\mathbf{h}(t-1) + \mathbf{b}_r) \qquad \text{reset gate} \qquad (5.41)$$

$$\tilde{\mathbf{h}}(t) = \phi(\mathbf{W}_h x(t) + \mathbf{R}_h(\mathbf{r}(t) \odot \mathbf{h}(t-1)) + \mathbf{b}_h) \qquad \text{candidate activation} \qquad (5.42)$$

$$\mathbf{h}(t) = (1 - \mathbf{u}(t)) \odot \mathbf{h}(t-1) + \mathbf{u}(t) \odot \tilde{\mathbf{h}}(t) \qquad \text{activation} \qquad (5.43)$$

Similar to the LSTM unit, a GRU possesses gates which modulate the information flow inside the unit, but it does not have separate memory cells. Equation (5.43) states that the activation at time step t, $\mathbf{h}(t)$, is a linear interpolation between activation at the previous time step and the candidate activation $\tilde{\mathbf{h}}(t)$. The update gate $\mathbf{u}(t)$ decides how much the unit updates its activation (cf. (5.40)). While the computation of a linear sum between the existing state and the newly computed state is similar to the LSTM unit, there is no control mechanism in the GRU for the amount of exposition of the internal state. Instead a GRU exposes its internal state each time [55]. The reset gate defined by (5.41) enables the GRU to forget its previously computed state, if $r(t) \approx 0$. As noted in [55] is is unclear a-priori, if LSTM or GRU networks perform better on a particular task. One possible advantage of the GRU over the LSTM, however, is the smaller number of parameters in the GRU architecture, Π_{GRU}, which is given by

$$\Pi_{GRU} = 3 \cdot N \cdot (N + D + 1), \qquad (5.44)$$

where as previously N is the number of (GRU) units, D is the input dimensionality, and the additional 1 accounts for the biases. For completeness it is mentioned here that there exists a great number of variants to the described GRU formulation; the interested reader is referred to [71, 139] for more details.

5.6 Residual Networks

Many of the successful deep learning algorithms and models used nowadays have originated from the research field of visual object recognition and the depth of neural networks was often found to be of crucial importance [313, 338]. However, stacking more layers does not automatically lead to better network performance, a phenomenon also observed in computational paralinguistics, e. g. by the author and his colleague [35]. Instead adding more layers to a suitably deep model can lead to higher training error, as shown in [137], and hence this degradation is not due to overfitting. It was found that gradients vanished from bottom to top layers, which leads to worse performance. In order to solve this problem the authors proposed a solution by adding (identity) *skip connections* or *shortcuts* to the network, effectively bypassing some layers. The resulting *deep residual networks (ResNet)* consist of repeatedly stacked building blocks called *residual units*. Each such unit can be expressed as

$$\mathbf{a}^{(l)} = \mathcal{F}(\mathbf{z}^{(l)}; \mathcal{W}^{(l)}) + S(\mathbf{z}^{(l)}) \tag{5.45}$$

$$\mathbf{z}^{(l+1)} = \phi^{(l+1)}(\mathbf{a}^{(l)}) \tag{5.46}$$

where $\mathbf{z}^{(l)}$ represents the input to the l-th residual unit, $\mathcal{W}^{(l)} = \{\mathbf{W}^{(l,k)} | 1 \leq k \leq K\}$ the set of the units weights and biases, and K the number of layers composing the residual unit. The residual function is denoted by $\mathcal{F}$ and the function $\phi^{(l+1)}$ is the output activation function after element-wise addition of the residual and the input. In the original formulation in [137] the shortcut function S is the identity mapping, i. e. $S(\mathbf{z}^{(l)}) = \mathbf{z}^{(l)}$, except for cases where the dimension of $\mathbf{z}^{(l)}$ is changed by $\mathcal{F}$. In these cases S is needed to match the dimensions and set to be a linear transformation, i. e. $S(\mathbf{z}^{(l)}) = \mathbf{W}_s \mathbf{z}^{(l)}$. An example of a residual unit with $K = 2$ is shown in Figure 5.12. Commonly the number of layers K per residual unit is small, i. e. $K \in [2, 3]$, in order to avoid the vanishing gradient problem mentioned above, but there is no strict limit to this.

A similar approach, inspired by Long Short-Term Memory recurrent neural networks, was introduced by Srivastava and his colleagues [322, 323]. The proposed *Highway Networks* borrow the idea of an adaptive gating mechanism from LSTM networks to allow for computation paths along which information can flow across many layers without attenuation, the so-called *information highways*. This enables training of very deep networks even with conventional gradient-based methods. Let the computation of a normal feed-forward network be formulated as

$$\mathbf{z}^{(l+1)} = H(\mathbf{z}^{(l)}, \mathbf{W}_H^{(l)}), \tag{5.47}$$

which is a generalization of (5.25). A Highway Network additionally defines two non-linear transforms, the *transform gate* $T(\mathbf{z}^{(l)}, \mathbf{W}_T^{(l)})$ and the *carry gate* $C(\mathbf{z}^{(l)}, \mathbf{W}_C^{(l)})$,

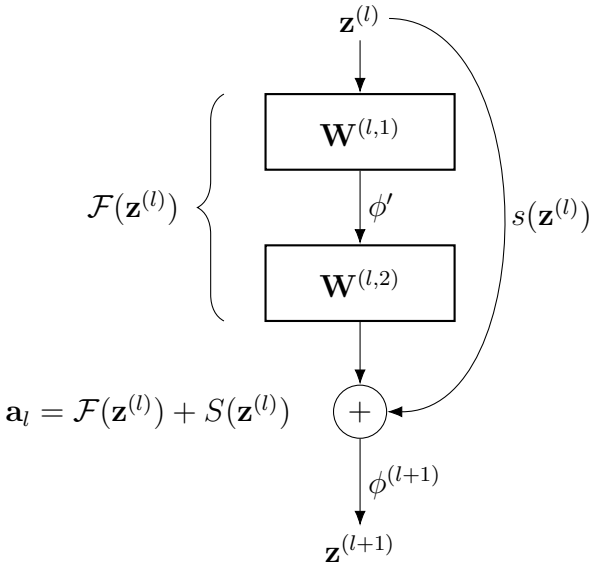


Figure 5.12: Residual unit for $K = 2$ layers with associated weight layers $\mathbf{W}^{(l,1)}$ and $\mathbf{W}^{(l,2)}$, where the biases are subsumed in the matrices (cf. Section 5.3.3). Note that the internal activation function ϕ' do not need to be identical to the output activation function $\phi^{(l+1)}$.

such that [323]

$$\mathbf{z}^{(l+1)} = H(\mathbf{z}^{(l)}, \mathbf{W}_H^{(l)}) \cdot T(\mathbf{z}^{(l)}, \mathbf{W}_T^{(l)}) + \mathbf{z}_l \cdot C(\mathbf{z}^{(l)}, \mathbf{W}_C^{(l)}). \tag{5.48}$$

Commonly, the carry gate is set as $C = 1 - T$, which yields the common definition

$$\mathbf{z}^{(l+1)} = H(\mathbf{z}^{(l)}, \mathbf{W}_H^{(l)}) \cdot T(\mathbf{z}^{(l)}, \mathbf{W}_T^{(l)}) + \mathbf{z}_l \cdot (1 - T(\mathbf{z}^{(l)}, \mathbf{W}_T^{(l)})). \tag{5.49}$$

For the Jacobian of the layer transform,

$$\frac{d\mathbf{z}^{(l+1)}}{d\mathbf{z}^{(l)}} = \begin{cases} \mathbf{I}, & \text{if } T(\mathbf{z}^{(l)}, \mathbf{W}_T^{(l)}) = \mathbf{0} \\ H'(\mathbf{z}^{(l)}, \mathbf{W}_H^{(l)}), & \text{if } T(\mathbf{z}^{(l)}, \mathbf{W}_T^{(l)}) = \mathbf{1} \end{cases} \tag{5.50}$$

which states that depending on the output of the transform gate, a highway layer can smoothly vary its behavior between that of a normal layer and that of a layer which simply passes through its inputs [323]. Note that the transform gates T and the carry gates C should output values in the range $[0, 1]$ to appropriately fulfill their gating functionality. Hence, commonly the sigmoid activation function (5.4) is chosen for both. A very detailed, recent study by He and colleagues [138] expounds the impact of the identity mapping or short-cut path in ResNets, investigating many different connection schemes.

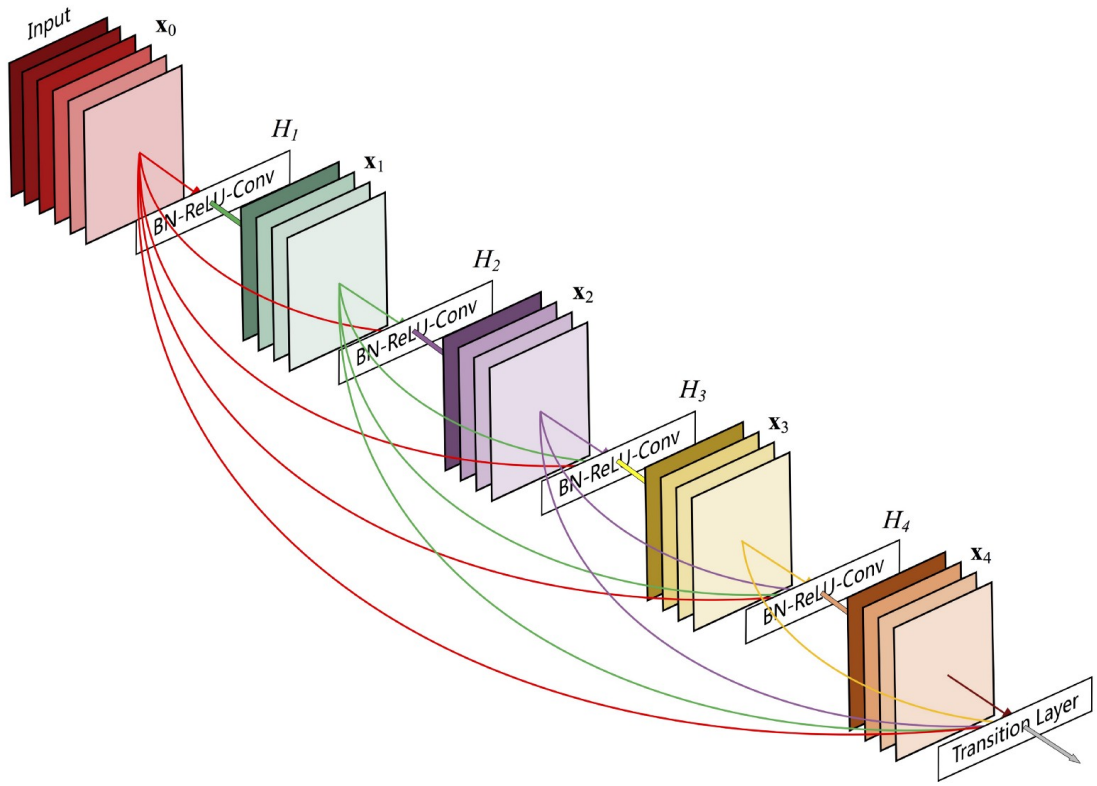


Figure 5.13: Illustration of the DenseNet architecture [160].

Some very interesting studies have recently been performed regarding the implications of ResNet architectures and their connection to other network types: First, Greff and colleagues [116] found that randomly dropping layers in a ResNet has little impact on the output representation and the performance of very deep networks, defying a popular view of deep learning that higher layers learn a increasingly abstract representation of lower layers. Instead they argue that successive layers iteratively refine their estimates of the same features instead of computing an entirely new representation. This finding was confirmed by Huang et al. [159]. Second, Liao and Poggio [204] show that a shallow RNN is exactly equivalent to a very deep ResNet with weight sharing among the layers and that such a RNN leads to a performance similar to the corresponding ResNet, although is has orders of magnitude fewer parameters. Further, they conjecture that a class of moderately deep RNNs is a biologically-plausible model of the ventral stream in visual cortex. Third, besides feed-forward networks the idea of creating information highways has also been extended to LSTM networks by introducing *Recurrent Highway Networks* [387]. These networks allow to train multi-layer state transitions in recurrent neural networks.

ResNets and Highway Networks share a key characteristic: they create shortcut connections between lower and higher layers. The insights gained from the study on these approaches led to an extended architecture called *DenseNet*. To ensure maximum information flow between the layers in the network, all layers (with matching feature map sizes) are connected directly to each other, i. e. each layer obtains additional inputs from all preceding layers and passes on its own feature maps to all subsequent layers [160]. An illustration of this network is given in Figure 5.13. Contrary to ResNets and Highway Networks, the feature streams are combined by concatenation, not summation, before they are passed on to the next layer. Usually, DenseNet layers are rather narrow containing a relatively small number of filters per layer. Hence, given the dense connection scheme each layer adds only a small amount to the "collective knowledge" [160] of the network. This is feasible since the final classification layer(s) are connected to all feature maps in the network and can leverage this distributed information. This also leads to an improved flow of information through the network, especially regarding gradients, which is helpful in training. Since each layer has direct access to the gradients from the loss function and to the original input signal, this implicitly constitutes deep supervision [204]. The authors of [160] also observed a regularizing effect caused by the dense connections, which reduces overfitting on tasks with smaller training set sizes. Additionally, given the particular connection scheme and the relatively fewer feature maps per layer, DenseNets require fewer parameters than conventional CNNs.

5.7 Supervised Network Training

Besides choosing an appropriate model topology for a given task at hand, training encompasses many equally important aspects of generating a network that generalizes well to unseen test data. Selecting the right training algorithm and its associated hyper-parameters often makes the difference between a model which excels and one that fails. Generally, the model parameters $\{\mathbf{W}, \mathbf{b}\}$ in a DNN are unknown a-priori and need to be estimated from a set of training examples $\mathbf{D}_{train}^{(sup)} = \{\mathbf{x}_i, \mathbf{y}_i\}_1^N$, where $\mathbf{x}_i$ is the i-th input, $\mathbf{y}_i$ the corresponding vector of targets (or labels), and N the total number of training examples. This situation, where the targets $\mathbf{y}_i$ are available is called *supervised training* (cf. Section 5.2), since the targets guide the training process to learn a desired mapping function from inputs to outputs. On the other hand, for *unsupervised training*, where no targets are available, the training set reduces to $\mathbf{D}_{train}^{(unsup)} = \{\mathbf{x}_i\}_1^N$. The process of learning the model parameters is usually called the parameter estimation process or *training process* and is determined by a learning algorithm and a loss function, as will be described in the following.

5.7.1 Parameter Estimation via Error Backpropagation

Supervised training is normally carried out by minimizing some error or cost function, commonly also called *loss function* $\mathcal{L}(\theta)$, where θ represents the free model parameters to be learned from the training data. As a short-hand notation, $\mathcal{L}(\theta)$ is often referred to as just the *loss*. This loss reflects the ability of the network to accurately model the function which maps the inputs $\mathbf{x}$ to the corresponding outputs $\mathbf{y}$; i. e. the better the model predicts the true targets $\mathbf{y}$, the smaller the loss. For the training algorithms described in this thesis, as applies for most current research, the loss function is defined to be a scalar value and ideally it should match the evaluation metric used for a certain problem. However, this is not always the case: for example, the AUC often used in paralinguistic settings is non-differentiable and thus unsuited for the gradient-based learning methods described in the following. In general, however, the loss function should be easy to evaluate and highly correlated to the final evaluation metric of the task so that any improvements on the cost function directly lead to improvements in the final evaluation score.

Formalizing this intuition, the model parameters need to be modified in a way so that they minimize the expected loss

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x},y}(L(\theta; \mathbf{x}, \mathbf{y})) = \frac{1}{N} \sum_{i=1}^{N} L(\theta; \mathbf{x}_i, \mathbf{y}_i), \tag{5.51}$$

where $L(\theta; \mathbf{x}_i, \mathbf{y}_i)$ is the per-example loss for the i-th input-output pair $(\mathbf{x}_i, \mathbf{y}_i)$, i. e. the global loss is assumed to be the average of the per-example losses. For regression problems a very popular loss functions is the *mean squared error* (MSE)

$$L_{MSE}(\theta; \mathbf{x}_i, \mathbf{y}_i) = \frac{1}{2} \|\mathbf{y}_i - \hat{\mathbf{y}}_\mathbf{i}\|^2 = \frac{1}{2} (\mathbf{y}_i - \hat{\mathbf{y}}_\mathbf{i})^T (\mathbf{y}_i - \hat{\mathbf{y}}_\mathbf{i}), \tag{5.52}$$

where $\hat{\mathbf{y}}$ denotes the network output, i. e. the prediction of the true target $\mathbf{y}$. For classification problems the loss function is often chosen to be the *cross-entropy* (CE)

$$L_{CE}(\theta; \mathbf{x}_i, \mathbf{y}_i) = -\sum_{c=1}^{C} y_i^{(c)} \cdot \log \hat{y}_i^{(c)}, \tag{5.53}$$

where C denotes the number of discrete target classes. It has been shown that minimizing the CE loss is equivalent to minimizing the Kullback-Leibler divergence (KLD) between the empirical probability distribution and the probability distribution estimated from the DNN [373]. The final goal is to find a set of model parameters θ^* which minimizes the global loss

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta), \tag{5.54}$$

where θ^* is also referred to as the *global optimum*. A common way to minimize the loss is compute its gradient w. r. t. to the model parameters,

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta L(\theta; \mathbf{x}_i, \mathbf{y}_i), \tag{5.55}$$

and to adjust the model parameters θ accordingly. However, except for very simple cases, minimizing $\mathcal{L}(\theta)$ is usually a non-convex problem, since it shows highly nonlinear dependencies on the model parameters. Often, there exist multiple *local minima* in the loss function where the gradient vanishes, i. e. $\nabla_\theta \mathcal{L}(\theta) = 0$. Therefore, one usually resorts to iterative numerical procedures, such as *gradient descent*, which updates the current set of model parameters θ_τ by following the estimated gradient downhill to obtain a new set of parameters

$$\theta_{\tau+1} = \theta_\tau - \lambda \cdot \nabla_\theta \mathcal{L}(\theta_\tau). \tag{5.56}$$

Here λ, called the *learning rate*, is the step size determining how far to change the model parameters in the negative direction of the gradient. Starting by computing the gradient of the error function at the output of a neural network and using the chain rule to iteratively compute the gradients for each layer down through the network leads to the well-known *backpropagation* algorithm [199, 272].

Gradient descent has been criticized for being unreliable and slow and for this reason *stochastic gradient descent* (SGD) was proposed as a stochastic approximation of gradient descent. Instead of computing the gradient over the full training data set, SGD computes gradients over parts of the data, so-called *mini-batches*:

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta L(\theta; \mathbf{x}_i, \mathbf{y}_i), \tag{5.57}$$

where B is the size of the mini-batch. Hence, the model parameter updates are performed after each mini-batch, leading to many more parameter updates per epoch[4] than with standard gradient descent. Although the trajectory of parameter updates in SGD is "noisier" than in regular gradient descent, SGD has shown to be very robust and superior in many applications [29, 186, 199, 277, 331].

For recurrent neural networks a particular variant of backpropagation is commonly adopted, called *backpropagation through time* (BPTT) [362]. As for all recurrent neural networks the training data is an ordered sequence of K input-output pairs, $(\mathbf{x}_i, \mathbf{y}_i)_1^K$. Here, K can be the length of an audio clip or just some shorter segment; however, it must contain consecutive features of the same context, since recurrent

[4] An epoch is defined as processing the full training data once.

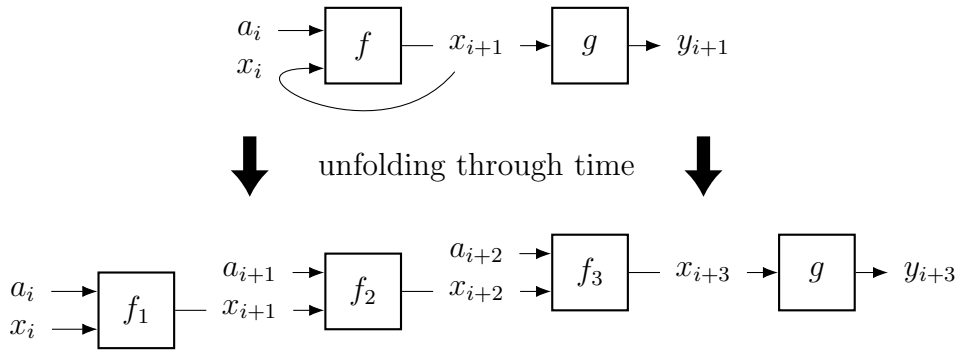


Figure 5.14: Illustration of unfolding through time in the BPTT algorithm for three time steps.

networks try to learn the underlying, time-dependent structure. Further, it is necessary to specify some initial value $\mathbf{h}_0$ to the hidden state of the recurrent network units. Often a vector of zeros is used for this purpose, but it is also possible to keep the final state from a previous BPTT step as the initial value for the next one, or to randomly initialize it to some other appropriate value. BPTT essentially is equivalent to unfolding the recurrent network in time and can thus be viewed as a feed-forward network with K layers, i. e. one layer per time step, but with all layers sharing the same model parameters and feeding the input $\mathbf{x}_i$ to the i-th layer. This procedure is depicted in Figure 5.14. The BPTT algorithm computes the gradient of the loss function w. r. t. all the network parameters, and hence, the model parameter updates is the average over all gradients in the BPTT data chunk.

5.7.2 Learning Rates and Momentum

One of the main hyperparameters for the successful training of neural networks is the (initial) learning rate λ of gradient descent, as shown in (5.56). It determines the magnitude of the weight updates in the search to minimize the loss functions and needs to be chosen appropriately: if it is set too small, only very small updates will be made to the network weights, and hence training will progress very slowly. Further, training can also get stuck in a local minimum and fail to explore the loss landscape. If, on the other hand, the learning rate is set too large the weight updates will be too large as well, which can lead to an explosion of the loss and to divergence of training. The optimal learning rate depends on the shape of the loss landscape, which in turn depends on both the chosen model topology and the dataset used for training.

A commonly employed approach, known as *learning rate annealing*, is to start with a relatively high λ and then reduce it over time following some schedule. The main intuition behind this approach is that with a high initial learning rate the

training process searches more globally in the loss landscape for good, low-loss regions, while the annealing process allows to narrow down the search to the exact minimum in a more fine-grained fashion. One of the most popular annealing schedules is the linear or *step decay*

$$\lambda_{\tau+1} = c_\lambda \cdot \lambda_\tau, \qquad 0 < c_\lambda < 1, \tag{5.58}$$

where c_λ is the decay rate. Other commonly used annealing schedules are the *exponential decay*, defined as

$$\lambda_{\tau+1} = \lambda_\tau \cdot e^{-c_\lambda p} \qquad c_\lambda > 0 \tag{5.59}$$

and the 1/t decay, given by

$$\lambda_{\tau+1} = \frac{\lambda_\tau}{1 + c_\lambda p} \qquad c_\lambda > 0 \tag{5.60}$$

where c_λ is a decay parameter and p some counter. It is pointed out that the event triggering a learning rate decay can be of differing nature: It can, for example, be based on time, so that annealing is triggered after each epoch or after having processed a certain number of mini-batches. Alternatively, it could be based on a performance evaluation on the development data, triggering a learning rate decay whenever the validation error stops to improve or falls below a certain threshold. In addition to lowering the learning rate more elaborate techniques for finding a suitable λ have been suggested: Smith [315], for example, proposes a cyclical learning rate schedule which varies between two bound values.

A classical technique for improving the convergence rate of DNN training is the *momentum* method. It is strongly motivated by the following analogy: if the loss is visualized as the location in a hilly terrain, then the set of model parameters θ at a certain optimization step bears analogy to a particle rolling in this landscape. This particle can be assigned a location, velocity, and momentum at any given time. The accumulation of a velocity matrix in directions of persistent reduction in the objective is equivalent to the accumulation of velocity in directions of low curvature that persist across multiple iterations, leading to accelerated progress in such directions compared to gradient descent [250, 331]. Formalizing this intuition leads to the following pair of equations for the (standard) momentum:

$$\mathbf{v}_\tau = m_\tau \, \mathbf{v}_{\tau-1} - \lambda \nabla \mathcal{L}_\theta(\theta_{\tau-1}) \tag{5.61}$$

$$\theta_\tau = \theta_{\tau-1} + \mathbf{v}_\tau \tag{5.62}$$

where $m_\tau \in [0, 1]$ is the momentum constant, $\lambda > 0$ the learning rate, $\nabla \mathcal{L}_\theta(\theta_\tau)$ an unbiased estimate of the loss gradient at θ_τ defined by (5.55), and $\mathbf{v}_\tau$ the velocity matrix. The momentum constant m_τ is used to control the decay of $\mathbf{v}_\tau$, where larger

values lead to higher velocities, since the gradient information persists across more update cycles. A typical choice of the momentum constant is between 0.5 and 0.9.

An improved variant of standard momentum, which recently has gained popularity, is *Nesterov's accelerated gradient* (NAG). It is an iterative algorithm that was originally derived for non-stochastic gradients [331] and is shown to enjoy stronger theoretical converge guarantees for convex functions [21]. Nesterov momentum is defined as [333]:

$$\mathbf{v}_\tau = m_{\tau-1}\,\mathbf{v}_{\tau-1} - \lambda \nabla \mathcal{L}_\theta(\theta_{\tau-1} + m_{\tau-1}\mathbf{v}_{t-1}) \tag{5.63}$$

$$\theta_\tau = \theta_{\tau-1} + \mathbf{v}_\tau \tag{5.64}$$

where the momentum constant should be chosen to be $m_\tau \approx 1 - 3/(5+\tau)$ [235]. Ignoring this momentum schedule, the key difference between momentum and Nesterov's accelerated gradient is that momentum computes the gradient before applying the velocity, while Nesterov's accelerated gradient computes the gradient after doing so [331]. In practice, this seems to lead to a stabler behavior.

5.7.3 Adaptive Learning Rates

Given the importance of the learning rates and the difficulty of finding appropriate initial values a number of algorithms with adaptive learning rates have been proposed. Most of these algorithms adapt individual learning rates for each specific trainable model parameter during the course of training.

AdaGrad

One representative of this group is the *AdaGrad* algorithm. The update rule for the i-th model parameter at update step τ, $\theta_{\tau+1,i}$, is given as

$$\theta_{\tau+1,i} = \theta_{\tau,i} - \frac{\lambda}{\sqrt{\overline{g}_{\tau,i}}} \cdot g_{\tau,i} \tag{5.65}$$

$$g_{\tau,i} = \nabla_{\theta_i} \mathcal{L}_\tau(\theta_i) \tag{5.66}$$

$$\overline{g}_{\tau,i} = \sum_{t=1}^{\tau-1} (g_{t,i})^2 \tag{5.67}$$

where (5.66) defines a shorthand notation for the gradient of the loss at update step τ for the i-th model parameter. Equation (5.67) states that $\overline{g}_{\tau,i}$ is the sum of the squares of the gradients w. r. t. θ_i up to step τ. Obviously, AdaGrad individually adapts the learning rates for each model parameter individually by scaling it inversely proportional to the square root of the sum of all of its previous squared values [77]. This means that it performs smaller updates to parameters associated

with frequently occurring features and larger updates to parameters associated with infrequent ones. The main advantage of the AdaGrad algorithm is that there is no need to manually anneal the learning rate. However, due to the accumulation of the squared gradients during training the effective learning rate shrinks excessively fast, eventually becoming infinitesimally small.

In order to reduce this negative side-effect, the *AdaDelta* algorithm was proposed by Zeiler [377]. Instead of accumulating all previous gradients, it accumulates the squares of previous gradients only over a sliding window of fixed size, recursively defined as a decaying average.

RMSprop

Another adaptive learning rate optimization algorithm which tries to eliminate AdaGrad's diminishing learning rate is the *RMSprop* algorithm, proposed by Tieleman and Hinton [341]. Similar to AdaDelta, it modifies the AdaGrad gradient accumulation into an exponentially weighted moving average to discard the gradient history from the far past as

$$\overline{g}_{\tau,i} = \lambda \cdot \overline{g}_{\tau-1,i} + (1-\lambda) \cdot (g_{\tau,i})^2 \tag{5.68}$$

$$\theta_{\tau+1,i} = \theta_{\tau,i} - \frac{\lambda}{\sqrt{\overline{g}_{\tau,i}}} \cdot g_{\tau,i} \tag{5.69}$$

The use of the moving average introduces a new hyperparameter, which controls the window size of the moving average. Empirically, RMSprop has been shown to be an effective and practical optimization algorithm for deep neural networks, including recurrent neural network models [331].

Adam

Adaptive Moment Estimation (Adam) [184] is yet another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients $g^2_{\tau,i}$ like AdaDelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients $g_{\tau,i}$. Similar to momentum, which can be visualized as a ball running downhill, Adam behaves like a heavy ball with friction, which prefers flat minima in the error surface [142]. Adam starts by computing estimates of the mean (first moment), $m_{\tau,i}$, and the (uncentered) variance (second moment), $v_{\tau,i}$ of the gradients per model parameter, $g_{\tau,i}$:

$$m_{\tau,i} = \beta_1\, m_{\tau-1,i} + (1-\beta_1)\, g_{\tau,i} \tag{5.70}$$

$$v_{\tau,i} = \beta_2\, v_{\tau-1,i} + (1-\beta_2)\, g^2_{\tau,i} \tag{5.71}$$

The authors in [184] point out that the moment estimates are biased towards zero, especially during the initial timesteps, since the moving averages are initialized as

0's. This especially happens when the decay rates are small (i. e. the βs are close to 1) and they propose to correct this initialization bias by

$$\hat{m}_{\tau,i} = \frac{m_{\tau,i}}{1-\beta_1^\tau} \tag{5.72}$$

$$\hat{v}_{\tau,i} = \frac{v_{\tau,i}}{1-\beta_2^\tau} \tag{5.73}$$

followed by the parameter update as in (5.68) to yield the Adam update rule:

$$\theta_{\tau+1,i} = \theta_{\tau,i} - \frac{\lambda}{\sqrt{\hat{v}_{\tau,i}}} \cdot \hat{m}_{\tau,i} \tag{5.74}$$

The suggested default values are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. As noted in [331], Adam is generally regarded as being fairly robust to the choice of hyperparameters and compares favorably to other adaptive learning methods.

5.7.4 Weight Initialization

In most current neural network model topologies and learning algorithms the network weights $\mathbf{W}$ and the associated biases $\mathbf{b}$ are the most relevant learnable parameters. It has been well known for a long time [104, 236, 368] that proper initialization is essential for good convergence of training and for the final performance of the trained model. To start with, setting all initial weights to zero will lead to a useless model, because all neurons in the network will compute the same output and hence all gradients computed during backpropagation will lead to the exact same parameter updates and therefore to identical weights across the network.

In order to break this symmetry problem the most obvious solution is to initialize the weights with small random numbers sampled from a uniform distribution or, alternatively, a normal distribution (commonly with zero mean and unit variance). Randomness guarantees that all neurons in a network compute unique values and thus obtain distinct updates. The initial values should be small enough to start in the linear regime of the adopted activation function of the neurons (cf. Section 5.3.2), which ensures that the problem of vanishing gradients caused by saturating non-linearities of the activation function is minimized. At the same time the initial values should not be too small, since very small weights lead to very small gradients, which diminishes the gradient flow during backpropagation and slows down training.

In principle, when initializing a deep neural network it is advantageous to strive for a constant variance of the inputs to each layer in order to avoid exploding or vanishing gradients [109]. In almost all situations the weights of a model are initialized to values drawn randomly from either a uniform or a Gaussian distribution.

Even though no exhaustive studies exist, the preference of one over the other does not seem to have a particular effect. What seems to matter most is the scale of the initial distribution and it is important to remember that the variance of the output distribution of randomly initialized neurons grows with the number of the neurons' inputs. In order to keep the scale of the gradients approximately the same in all layers Glorot and Bengio [104] proposed the *normalized initialization*, often referred to as the *Glorot* or *Xavier* initialization. If a uniform distribution is desired the initial weight values should be drawn from

$$\mathbf{W} \sim U\left[\sqrt{\frac{6}{N_{in}+N_{out}}}, \sqrt{\frac{6}{N_{in}+N_{out}}}\right], \tag{5.75}$$

where N_{in} is the number of input units (*fan-in*) and N_{out} the number of output units (*fan-out*). If a Gaussian distribution is chosen then the weights should be initialized according to

$$\mathbf{W} \sim \mathcal{N}\left(0, \frac{2}{N_{in}+N_{out}}\right). \tag{5.76}$$

A detailed analytical description of initialization-dependent phenomena can be found in [285]. An slightly modified initialization scheme specifically tailored to (P)ReLU networks instead was proposed in [136], given as:

$$\mathbf{W} \sim \mathcal{N}\left(0, \frac{2}{N_{in}}\right). \tag{5.77}$$

Less care has to be taken to initialize the layer biases and in most situations they can be set to zero, since the symmetry breaking characteristic is achieved by the random weight initialization. However, for ReLU-type networks the biases sometimes are set to a small positive constant, e. g. 0.01, so that all ReLU neurons start in the positive, linear regime and thus the initial gradients are non-vanishing. Furthermore, for LSTM networks Jozefowicz and his colleagues [174] advised to initialize the forget gate to a large value, such as 1 or 2, leading to an initial forget value close to 1 and thus enabling gradient flow at the beginning of training.

5.7.5 Regularization

In machine learning one generally trains a model on a finite set of training data and at least in the case of supervised training the neural networks, which are the working substrate of this thesis, essentially learn a mapping from the input value to the output values. It has been shown by Hornik and his colleagues [158] that neural networks are universal function approximators, given they possess the necessary capacity, which among other factors is determined by their topology and size. If the network is chosen too small then the model cannot adequately capture the underlying structure of the (training) data and *underfitting* occurs. On the other hand, if the

network is chosen to large, it will be able to model the training data extremely well and might "memorize" the individual training data points. This effect is referred to as *overfitting*. The problem with the latter is that an overfitted model will work very well on the training data, but usually very poorly on unseen test data, incurring a large *generalization error*. In general, one strives for a network which generalizes well to new data, but often one tends towards using larger networks to avoid underfitting.

One way to combat overfitting, even with larger networks, is to employ *regularization*. This generally involves imposing some sort of smoothness constraint on the learned model [103]. This section discusses some of the common regularization methods used in this thesis. In general, regularization can be formalized by adding a cost to the loss function from (5.51).

$$\tilde{\mathcal{L}}(\theta; \mathbf{X}, \mathbf{Y}) = \mathcal{L}(\theta; \mathbf{X}, \mathbf{Y}) + \alpha\, \Omega(\theta) \tag{5.78}$$

where $\alpha \in [0, \infty)$ is the regularization coefficient, which controls the relative contribution of the regularization cost $\Omega(\theta)$ to the overall loss function $\tilde{\mathcal{L}}(\theta; \mathbf{X}, \mathbf{Y})$. Setting $\alpha = 0$ results in no regularization and reduces to the previously used loss $\mathcal{L}(\theta; \mathbf{X}, \mathbf{Y})$. Are more detailed theoretical background on regularization can be found in [135].

Parameter-Norm Regularization

The most common parameter norm regularizer is the L_2 *regularization*, also known as *weight decay* or Tikhonov regularization, which is given by:

$$\Omega(\theta) = \frac{1}{2}\|\mathbf{W}\|_2^2 = \frac{1}{2}\sum_{i,j}|w_{i,j}|^2. \tag{5.79}$$

It can be shown that this leads to a multiplicative shrinking of the weight matrix $\mathbf{W}$ by a constant factor at each parameter update step and that L_2 regularization causes the learning algorithm to "perceive" the input $\mathbf{X}$ as having higher variance, which makes it shrink the weights on features whose covariance with the output target $\mathbf{Y}$ is low compared to this added variance [109]. Alternatively, one can employ L_1 *regularization*, defined as

$$\Omega(\theta) = \|\mathbf{W}\|_1 = \sum_{i,j}|w_{i,j}| \tag{5.80}$$

i. e. as the sum of absolute values of the individual model parameters. Goodfellow and colleagues [109] note that the effect of L_1 regularization is quite different from that of L_2 regularization, as its contribution to the gradient no longer scales linearly with each individual parameter, but instead is a constant factor with a sign equal to $\text{sgn}(w_{i,j})$. Further, L_1 regularization results in sparser solutions than L_2 regularization, which by some researchers has been used as a feature selection mechanism [340].

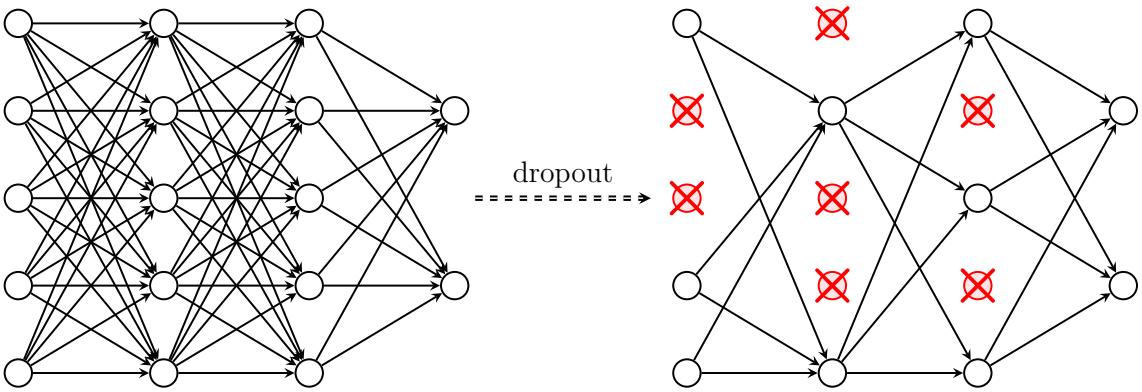


Figure 5.15: An example of reducing a neural network with two hidden layers by applying dropout. Crossed units have been dropped.

It should be noted that parameter-norm regularization is typically applied only tp the network weights, excluding the biases. As pointed out in [109], each weight $w_{i,j}$ in a network specifies the interaction of two variables and adapting $w_{i,j}$ hence requires to observe those variables in a variety of conditions. On the other hand, each bias controls only a single variable and therefore requires less data to be fit. Furthermore, at times a different regularization coefficient α is employed for each network layer, in order to be more selective when regularizing the network weights. As a concluding remark it should be noted that in addition to improving the generalization error, regularization of the network weights was found to also speed up the convergence of training [206].

Dropout

Dropout is a stochastic regularization technique minimizing an expected loss function under a noise distribution. During the training phase, a random fraction $1 - p$ of the nodes of a layer or network is removed for each training example, together with its corresponding incoming and outgoing edges [321]. This leaves a reduced network as illustrated in Figure 5.15. Since the random selection of nodes happens in a different manner for each training example, dropout can be regarded as a form of model averaging. During the testing phase all nodes and activations are used, but the weights must be reduced by a factor of p in order to account for the reduced activation inputs during training.

Dropout is a very efficient way of performing model averaging and has achieved remarkable improvements in research areas such as image classification [194] and speech recognition [66]. Further, it was shown by Srivastava and his colleagues [321] that dropout is more effective than other standard, computationally inexpensive regularizers, such as weight decay or sparse activity regularization. Nonetheless, it

can be combined with other forms of regularization to yield a further improvement. A big advantage of dropout is that it can be used with many different types of networks and training algorithms. Besides feed-forward neural networks, it has shown to also perform well on probabilistic models, such as RBMs [321] and recurrent neural networks [18, 247, 310, 376].

Additive Noise

Apart from the more theoretical regularization methods described above there exist a number of very simple, heuristic approaches to improve generalization. One way is to add (random) noise to the input features at the start of each epoch. This enforces the constraint that the model should learn the underlying structure contained in the input necessary to predict the output, which should be invariant to small variations of the input features. Besides applying additive noise to the input features some researchers have also suggested to add noise to the neural unit activation functions [120] or the gradients of the loss function [233]. A comparison study [9] on the application of additive noise to the network inputs, outputs, weight connections, and weight changes found that input noise and weight noise encourage the neural-network output to be a smooth function of the input or its weights, respectively. In the weak-noise limit, noise added to the output of the neural networks only changes the objective function by a constant. Hence, it cannot improve generalization. Input noise introduces penalty terms in the objective function that are related to, but distinct from those found in the regularization approaches. However, weight noise is found to be effective in improving the generalization performance only for the classification problem. Other forms of noise have practically no effect on generalization.

Training Set Randomization

Shuffling of randomizing the input features, be it individual feature frames, mini-batches, or data chunks for recurrent neural networks (where the order of frames in the training chunk must be kept contiguous), prevents the same examples from always appearing in the same order. This is especially beneficial for the most widely used SGD-based optimization algorithms (cf. Section 5.7.1), since in this case the update steps depending on the respective mini-batch gradients are randomized, too. This reduces the susceptibility to local minima in the loss function.

Early Stopping

One of the most commonly used forms of regularization in deep learning is *early stopping* [371], due to its simplicity and effectiveness: given a training set, $\mathbf{D}_{train}$, and a validation set, $\mathbf{D}_{valid}$, training is executed as normal and after a pre-defined period (e. g. after each epoch or after a certain number of iterations) the validation set error is computed. If this error improves, a copy of the model parameters is stored. The algorithm terminates once the validation set error has not improved over the best one

for some number of steps and the best model parameters are returned as the final model parameters. This way the error on the validation set represents a proxy for the generalization error, i. e. one assumes that the validation set error is a good indicator for the error obtained on an unseen test set [256]. Alternatively, one can resort to cross-validation as described in Section 4.2, where multiple partitions of the data into $\mathbf{D}_{train}$ and $\mathbf{D}_{valid}$ are created. However, even this simple procedure in practice is complicated by the fact that the validation error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when overfitting has truly begun [256]. The interested reader is referred to [109] for an interesting and detailed treatise of how early stopping actually acts as a regularizer.

5.7.6 Parameter Normalization Techniques

Batch Normalization

One of the most popular recent advances in optimizing DNNs is *batch normalization*, which was introduced by Ioffe and Szegedy in 2015 [167] in order to mitigate the difficulty of training very deep models. In these models the input to each layer is determined by the parameters of all preceding layers. During training the parameter updates are determined by the respective gradients, which are computed under the assumption that the parameters of the preceding layers remain unchanged. However, in practice all updates are performed simultaneously leading to unexpected effects. Due to the changes of the parameters in a DNN the respective distributions of its nodes change as well and require them to continuously adapt to the new distributions. This leads to an effect coined *internal covariate shift* [167] and batch normalization was invented to mitigate this effect.

The basic idea is to normalize the inputs of each layer to have zero mean activation and a standard deviation of one, similar to z-score normalization applied to the input features (cf. Section 3.2). In fact the output of one layer is the input to the next layer in the network, just as the input features are the input to the first hidden layer in a DNN. As the name suggests, batch normalization performs the normalization on each training mini-batch. Further, it is important to note that the normalization occurs before applying the activation function. The batch normalization algorithm performs the operations defined in Algorithm 1.

Batch normalization offers a number of advantages for training deep neural networks: First, it allows for much higher learning rates and thus faster convergence during training. This stems from the fact that the affine, linear transformation $\mathbf{W}^\intercal\mathbf{z} + \mathbf{b}$ (cf. Equation (5.1)) might drive its outputs into the non-linear, saturated regime of the activation function (ReLU being an exception), which leads to vanishing gradients and slow training, especially for large values in $\mathbf{W}$. By normalizing the

Algorithm 1 Batch normalization transform

Input: Values of z over a mini-batch: $\mathcal{B} = \{z_1, z_2, \ldots, z_B\}$;
Parameters to be learned: γ, β
Output: $\{z_i' = \mathbf{BN}_{\gamma,\beta}(z_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{B}\sum_{i=1}^{B} z_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{B}\sum_{i=1}^{B}(z_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\hat{z}_i \leftarrow \frac{z_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalization}$$

$$z_i' \leftarrow \gamma\hat{z}_i + \beta \equiv BN_{\gamma,\beta}(z_i) \qquad \text{// scale \& shift}$$

values of $\mathbf{W}$ this effect is reduced. Second, with batch normalization one has to pay less attention to correct initialization of the network parameters, which was shown to be highly important in Section 5.7.4. And finally it provides some regularization, since it adds some form of noise to the network, which reduces or even eliminates the need for dropout in many cases. It was shown in [166] that the effectiveness of batch normalization diminishes when the minibatch size is small or when the minibatches do not consist of independent samples, due to differences between the training and inference stages of batch normalization. In these cases the authors propose a modified variant called *batch renormalization*.

Layer Normalization
The effect of batch normalization is dependent on the mini-batch size and it is not straight-forward to apply to RNNs. Ba and colleagues [15] proposed a modification called *layer normalization*, suggesting to compute the mean and variance used in normalization on a single training case instead of on a mini-batch. This means that, unlike batch normalization, layer normalization performs exactly the same computation at training and test time. Further the dependence on the mini-batch size is eliminated and in the context of recurrent neural networks the normalization statistics can be computed separately at each time step. In [15] it is shown that layer normalization stabilizes the hidden state dynamics in RNNs very effectively.

Weight Normalization
Another modification to batch normalization, which can also be applied to recurrent neural networks, is called *weight normalization* and was proposed by Salimans and Kingma [284]. This method constitutes a reparameterization of the model's weight

vectors $\mathbf{w}$ by decoupling the length of $\mathbf{w}$ from its direction. Contrary to batch normalization it is a deterministic variant which does not add any noise to the gradients. This characteristic lends the method to be useful in noise-sensitive applications such as generative models, for which batch normalization is less appropriate. In addition, the authors show that the computational overhead added by weight normalization is lower than that in batch normalization. Similar to all parameter normalization methods discussed above weight normalization speeds up the convergence of optimization algorithms, such as stochastic gradient descent.

5.7.7 Software Frameworks

For running machine learning experiments researchers nowadays heavily rely on a number of different software frameworks. Since the advent of deep learning in the last decade the research community has welcome and strongly relied on the appearance and development of these frameworks. Required characteristics of any such framework to be beneficial are reliability, speed of execution, easy extensibility, public availability, and a broad developer and user base, in order to maintain the code and eradicate any errors (*bugs*) as soon as possible. Current frameworks offer a wide variety of functionality and differ in the trade-off between the above criteria. Some of the most widely known frameworks are Theano [24], Caffe [172], PyTorch [179], the Microsoft Cognitive Toolkit (formerly known as CNTK) [309], MXNet [51], Chainer [343], TensorFlow [1], and Keras [54], just to name a few.

One of the cornerstones of the more recent frameworks is the presence of *automatic differentiation*, i. e. the automatic numerical evaluation of derivatives of functions of a model [17, 263]. This is achieved by applying the chain rule repeatedly on elementary arithmetic operations, such as addition, subtraction, multiplication, and division, and on elementary functions, e. g. exp, log, tanh, etc., for which the derivatives are known either in closed form or for which efficient algorithmic implementations exist.

Due to the rapid development of algorithms and, hence, frameworks in the field of deep learning, a couple of different tools were used in the experiments described in this thesis. The first is CURRENT, a C++–based toolkit developed at the Technical University Munich by Weninger and colleagues [360]. While it offers fast execution, it is limited in the model topologies and optimization algorithms it supports and was superseded by Theano, which was one of the first publicly available frameworks offering a wide variety of model types and automatic differentiation. However, since the appearance of Tensorflow (and other toolkits) it has suffered a strong decrease of its user base and in consequence its development and maintenance was ceased in 2017. Therefore, all recent experiments presented in this thesis were conducted with Tensorflow.

5.8 Unsupervised Network Training

It was mentioned in Section 5.2 that the amount of unlabeled data often by far exceeds the amount of labeled data. This is especially true in the field of paralinguistics, where the available data sets are fairly small caused by the significant effort and cost required for data annotation. While supervised training is essential to obtain good results, unsupervised training can support and often improve the performance of the former by learning some underlying structure inherent in the data. A typical use case of unsupervised training is pre-training a network on unlabeled data to initialize the network parameters to a set of values which represent a better starting point for subsequent supervised training than random initialization. In this chapter two common categories of unsupervised neural network training will be expounded, the first based on *Restricted Boltzmann Machines (RBM)* and the other on *Autoencoder (AE)* variants.

5.8.1 Restricted Boltzmann Machines

Restricted Boltzmann Machines are a variant of Boltzmann Machines [150, 282] and are a class of undirected graphical models forming a bipartite graph: one part consists of a layer of observed or *visible* variables, the other part of latent or *hidden* variables. Each visible unit is connected to each hidden unit and vice versa, but there are no connections between nodes within a group. An schematized example of a RBM graph is shown in Figure 5.16. RBMs were initially proposed by Smolensky [316] under the name of *Harmonium*, but only became popular when Hinton and colleagues used them to pre-train deep learning algorithms [145, 149]. As described in [89, 145] an RBM assigns an energy to every joint configuration, $(\mathbf{v}, \mathbf{h})$, of visible and hidden state vectors, represented by $\mathbf{v}$ and $\mathbf{h}$, respectively. For binary visible units, an RBM with V visible and H hidden units is described by the following energy function:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{i=1}^{V} v_i b_i^{(v)} - \sum_{j=1}^{H} h_j b_j^{(h)} \tag{5.81}$$

where v_i and h_j are the binary states of visible unit i and hidden unit j, $b_i^{(v)}$ and $b_j^{(h)}$ are the respective biases, and w_{ij} is the weight between them. Under this energy function, the conditional probabilities for each visible and hidden unit given the others are

$$p(h_j = 1|\mathbf{v}) = \sigma\left(\sum_i v_i w_{ij} + b_j^{(h)}\right) \tag{5.82}$$

$$p(v_i = 1|\mathbf{h}) = \sigma\left(\sum_j h_j w_{ij} + b_j^{(v)}\right) \tag{5.83}$$

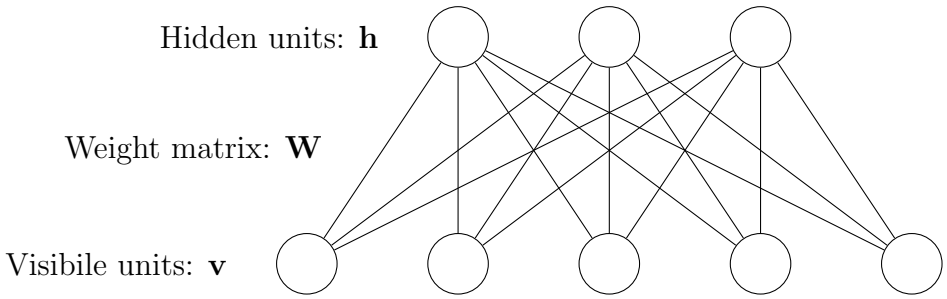


Figure 5.16: Schematized RBM graph.

where

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.84}$$

is the logistic or sigmoid function. The network assigns a probability to every possible joint configuration $(\mathbf{v}, \mathbf{h})$ via the energy function as

$$p(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z} = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{u},\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}} \tag{5.85}$$

where Z is called the partition function. The marginal distribution of the visible units is obtained by summing over all possible hidden states as

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}) \tag{5.86}$$

and the gradient of the log probability of a training vector with respect to a weight is simply

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{5.87}$$

where the angle brackets are used to denote expectations under the distribution specified by the subscript. This formulation leads to a very simple update rule adopting stochastic steepest ascent in the log probability of the training data:

$$\Delta w_{ij} = \epsilon \cdot (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}), \tag{5.88}$$

where ϵ is the learning rate controlling the size of the update steps. It is easy to get an unbiased sample of $\langle v_i h_j \rangle_{data}$ using the sample data $\mathbf{v}$, because there are neither connections between the visible nodes nor between the hidden nodes. However, it is very difficult to obtain an unbiased sample of $\langle v_i h_j \rangle_{model}$, since its computation involves the normalization constant Z, which cannot generally be computed efficiently (being a sum of an exponential number of terms), as explained in [145]. To avoid the difficulty in computing the log-likelihood gradient, Hinton [148] proposed the

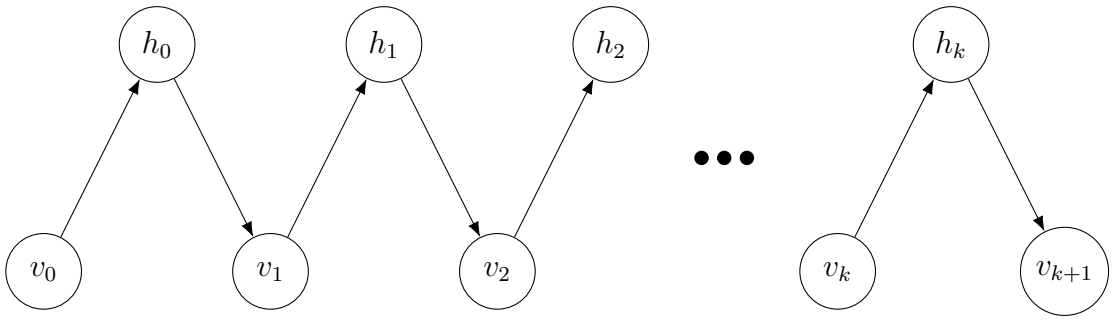


Figure 5.17: Illustration of k-step alternating Gibbs sampling.

Contrastive Divergence (CD) algorithm which approximately follows the gradient of the difference of two divergences [47]:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} \approx \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon} \tag{5.89}$$

The algorithm starts by initializing the states of the visible units $\mathbf{v}$ with a training vector. All binary states of the hidden units are subsequently computed in parallel using Equation (5.82). After sampling the hidden states a "reconstruction" is generated by setting each $v_i = 1$ with a probability specified by Equation (5.83). This procedure called *alternating Gibbs sampling* is then repeated for k steps, before collecting the statistics to compute $\langle v_i h_j \rangle_{recon}$. Alternating Gibbs sampling is illustrated in Figure 5.17. Although RBMs typically learn better models if more steps of alternating Gibbs sampling are used, i. e. $k > 1$, in most situations one-step ($k = 1$) Gibbs sampling is sufficient [332].

To deal with real-valued instead of binary input data, one can resort to an RBM with Gaussian visible units and binary hidden units, called *Gaussian-Bernoulli Restricted Boltzmann Machine (GBRBM)*. For the GBRBM the energy function needs to be modified, for example as proposed by Cho [52]:

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^{V} \frac{(v_i - b_i^{(v)})^2}{2\sigma_i^2} - \sum_{i=1}^{V} \sum_{j=1}^{H} \frac{v_i}{\sigma_i^2} h_j w_{ij} - \sum_{j=1}^{H} h_j b_j^{(h)} \tag{5.90}$$

Under this modified energy function, the conditional probabilities for each visible and hidden unit given the others are

$$p(v_i = v|\mathbf{h}) = \mathcal{N}\left(\mathbf{v} \;\middle|\; (\sum_j h_j w_{ij} + b_i^{(v)}), \sigma_i^2\right) \tag{5.91}$$

$$p(h_j = 1|\mathbf{v}) = \phi\left(\sum_i \frac{v_i}{\sigma_i^2} w_{ij} + b_j^{(h)}\right) \tag{5.92}$$

where $\mathcal{N}(\cdot \mid \mu, \sigma^2)$ denotes the Gaussian probability density function with mean μ and variance σ. Note that the variance parameter σ_i^2 is a learnable one in this formulation, but it is not necessarily equivalent to the variance of the input data.

5.8.2 Autoencoders

An autoencoder (AE) is a type of neural network which attempts to learn a compressed representation of its input. The basic idea behind it is rather simple: in a general formulation the AE consists of an encoder $\xi : \mathcal{X} \to \mathcal{R}$, which maps an input feature vector $\mathbf{x} \in \mathcal{X}$ to a latent representation $\mathbf{r} \in \mathcal{R}$, also called the *code*. The decoder $\psi : \mathcal{R} \to \mathcal{X}'$ takes this representation $\mathbf{r}$ and maps it to a *reconstruction* $\mathbf{x}'$ of the same shape of $\mathbf{x}$. This operation can be formally written as

$$\mathbf{x}' = (\psi \circ \xi)(\mathbf{x}) \tag{5.93}$$

$$\mathbf{r} = \xi(\mathbf{x}) \tag{5.94}$$

Ideally, the reconstruction equals the input, $\mathbf{x}' = \mathbf{x}$, and hence training the AE network consists in minimizing the MSE loss

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2 \tag{5.95}$$

which leads to the encoder/decoder solution:

$$\xi^*, \psi^* = \underset{\xi,\psi}{\operatorname{argmin}} \, ||\mathbf{X}_{train} - (\psi \circ \xi)(\mathbf{X}_{train})||^2, \tag{5.96}$$

i. e. the optimum encoder (ξ^*) and decoder (ψ^*) parameter sets are determined on a training set $\mathbf{X}_{train}$ in an unsupervised manner by setting the output equal to the input. In general, the parameters represented by ξ and ψ differ from each other, but could also be chosen to be shared, especially the network weights.

In order to successfully train an autoencoder network (and to obtain a compressed representation of the input) the latent representation $\mathbf{r}$ must be smaller than the input layer $\mathbf{x}$, i. e. the encoder should exhibit a *bottleneck* architecture. Otherwise the AE network will essentially learn the identity function, which is the trivial solution to the optimization problem (5.96). However, using one of the variants described below this constraint can be avoided [19]. An illustration of an autoencoder structure is presented in Figure 5.18.

As pointed out earlier [36], the main motivation for adopting autoencoder networks is to pre-train - possibly deep - neural networks in an unsupervised manner. To this end, the autoencoder, as depicted in Figure 5.18, is trained to convergence and the encoder, including the code layer, is used as (part of) the network, while the

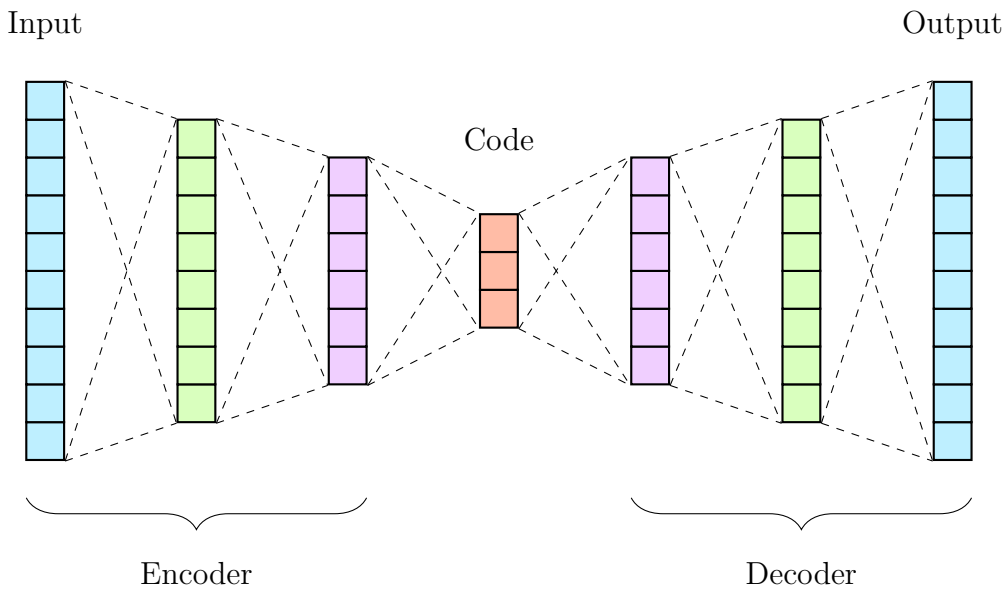


Figure 5.18: Illustration of the autoencoder architecture with the bottleneck code layer.

decoder part is discarded. This pre-training moves the network parameters close to an optimum and thus gives a good initialization to a subsequent fine-tuning step, e. g. by running Stochastic Gradient Descent (SGD). Moreover, it is possible to stack the resulting, pre-trained autoencoders to form a deep *stacked autoencoder* to get a good initialization for a deep network, which can subsequently be fine-tuned. As shown in the previous section an alternative approach to autoencoder pre-training is to utilize Restricted Boltzmann Machines (RBM). It is unclear whether RBMs or AEs lead to better performance and in practice both seem to give comparable results on many tasks. However, AEs are usually simpler and faster to train than RBMs and have become very popular in the literature in recent years.

An number of variants to the basic AE approach have been proposed in the literature, in an attempt to learn a richer and more robust representation and to improve the ability to capture important information contained in the input:

Denoising autoencoders corrupt the input, for example by adding a certain amount of random noise, but try to recover the original, undistorted input signal. Thus, the loss function can be written as

$$\mathcal{L}(\tilde{\mathbf{x}}, \mathbf{x}') = ||\mathbf{x} - (\psi \circ \xi)(\tilde{\mathbf{x}})||^2, \tag{5.97}$$

where $\tilde{\mathbf{x}}$ is the corrupted input. This simple concept was used with great success for different neural architectures [219, 349, 350] and was shown to lead to robust

representations.

Sparse autoencoders instead sparsify the elements of the code layer $\mathbf{r}$, i. e. they take measures to reduce the average activity of the code layer's elements. This can be formalized by adding a sparsity penalty (L_0-norm), $||\mathbf{r}||_0$, to the loss function $\mathcal{L}$ as follows:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2 + \lambda_s ||\mathbf{r}||_0, \tag{5.98}$$

where $||\mathbf{r}||_0$ counts the non-zero elements of $\mathbf{r}$ and $\lambda_s \geq 0$ controls the sparsity. There are multiple ways to accomplish this, but a simple approach is k-sparse autoencoders, proposed by Makhzani and Frey [214]. This algorithms finds the k highest activations in the code layer activations and sets all other elements to zero. Note that the error signal is backpropagated only through the k active nodes during training. Sparsification eventually leads to a further compression of the latent representation $\mathbf{r}$, but not in a fixed way, because over time different elements of $\mathbf{r}$ can be zeroed out.

Variational autoencoders (VAE) [185], instead of mapping the input $\mathbf{x}$ to a fixed vector $\mathbf{r}$, learn a latent variable model and map $\mathbf{x}$ onto a probability distribution. The bottleneck latent representation is replaced by two vectors, a vector representing the mean of the distribution and one which represents the standard deviation. One then samples from this distribution to generate inputs for the decoder, which tries to reconstruct the original input data $\mathbf{x}$. In order for VAE to be trainable, the loss function needs to extended by the Kullback-Leibler divergence D_{KL} between the true and the approximate posterior as [41]

$$\log p_\psi(\mathbf{x}|\mathbf{r}) = D_{KL}(q(\mathbf{r}|\mathbf{x}) \,||\, p(\mathbf{r})) + \mathcal{L}(\xi, \psi; \mathbf{x}, \mathbf{r}). \tag{5.99}$$

Since backpropagation cannot be applied to a sampling operator the so-called "reparameterization trick" is applied. This trick consists in parameterizing each random variable $r_i \sim q_\xi(r_i|\mathbf{x}) = \mathcal{N}(\mu_i|\sigma_i)$ as a differentiable transformation of a noise variable $\epsilon \sim \mathcal{N}(0, 1)$ as [41]

$$r_i = \mu_i + \sigma_i \, \epsilon. \tag{5.100}$$

Contractive autoencoders (CAE) [268] instead use the Frobenius norm of the Jacobian matrix, in order to learn a representation which is less sensitive to small variation in the input. In this case the loss function is given by

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2 + \lambda_c ||J_r(\mathbf{x})||_F^2 \tag{5.101}$$

$$||J_r(\mathbf{x})||_F^2 = \sum_{ij} \left(\frac{\partial r_j(x)}{\partial x_i} \right) \tag{5.102}$$

In a CAE the encoder mapping exhibits the property of locality, i. e. small changes in $\mathbf{x}$ lead to small changes of $\mathbf{r}$.

Part III

Applications

Chapter 6

Likability Classification

This is the Law of Likability: The real you is the best you.

Michelle T. Lederman

With respect to other paralinguistic phenomena, such as age, gender, emotion, or social signals, the prediction and classification of *likability* is a rarely addressed topic and hence a rather limited number of publications can be found in the research literature. This might be due to the fact that likability is a highly subjective and complex phenomenon, which depends on many facets of inter-human communication. The Likability Sub-Challenge of the Interspeech 2012 Speaker Trait Challenge [291] was one of the first attempts to establish a basis for comparable research on this topic by supplying an accessible research database and some reference baseline results for future research. Based on this dataset this chapter examines the classification of the likability of human voices by employing deep neural networks and presents results exceeding the baseline results by a fair margin. This research work results from the participation in the Interspeech 2012 Speaker Trait Challenge and was previously presented in [35].

6.1 The Likability of Voices

The concept of the likability of a speaker's voice is a fascinating topic, but its definition is an indistinct and intricate topic. As deduced from Table 2.1, likability is a highly subjective measure and hence it is difficult to objectively determine what makes a voice agreeable. There is a substantial body of research from the psychological realm that attributes this paralinguistic phenomenon to a number of different elements [299]: Some studies link likability to sexual attraction and reciprocal liking [12, 60, 156]. In a professional environment likability has been shown to be

related to competence [234]. Furthermore, Zuckerman and colleagues [390] found that attractive voices can be associated with the presence of confidence, lack of tension, and favorable ratings of personality. Besides the interest in the analysis of likability these and related findings are also interesting for text-to-speech (TTS) synthesis of pleasant voices. For example, Syrdal et al. [337] found gender-specific differences and a positive correlation of likability to features related to spectral tilt and the power of unvoiced speech segments. In the context of commercial advertisement it was found that low pitch and faster than normal syllable speed was preferred by listeners [50]. Several studies on the relevant features for the attractiveness and pleasantness of voices have observed that prosodic features, in particular fundamental frequency (F_0), are highly correlated [192, 203]. For example, male voices are judged as being more attractive if they have low mean F_0 and closely spaced, low-frequency harmonics [60, 156]. Moreover, voice quality features, such as the normalized amplitude ratio and the related breathiness, have been reported to impact the attractiveness of voices [45]. Furthermore, Strangert and Gustafson [328] confirmed the importance of fundamental frequency, but also concluded that temporal features were not related to the overall rating of a speaker.

Related work

Some of the earliest attempts to study the classification of likability was made by Burkhardt and colleagues [43]. They collected a database which laid the foundation for the Likability Sub-Challenge of the Interspeech 2012 Speaker Trait Challenge [291], described in the next section. Further, based on a large, supra-segmental feature set, which is smaller than, but similar to the one used in this study, they used a decision/regression tree to tackle the problem. Another study [358] found that all sentences from the same speaker were rated similarly but the agreement between different listeners for the same speaker was low. They also observed that the most significant results were mostly dependent on the gender of the speaker. Pinto-Coelho et al. [58, 249] instead combined a Support Vector Machine (SVM) with a Gaussian Mixture Model (GMM)/Naive Bayes classifier using a late fusion scheme using a limited set of input features. They reported that using only 6 features they obtained best performance on their Portuguese data set.

As mentioned above the Interspeech 2012 Speaker Trait Challenge provided the first reasonably sized, accessible database for likability classification, which allows to compare results across different proposed approaches. Based on this dataset Gonzalez and Anguera [108] extracted and evaluated a set of perceptually inspired features and obtained an absolute improvement of 3.2% with respect to the baseline results using a linear SVM classifier and only 7 of the official baseline features. Another approach was followed by Montacié and Caraty [229]: they experimented with 4 large supra-segmental feature sets with sizes between 8,348 and 10,342 features and a SVM classifier. They obtained a UAR of 64.1% with one of these feature sets and a UAR of 65.8% with a fusion of their 4 submission systems.

Table 6.1: Distribution of age and gender groups in the Speaker Likability Database [43]. Numbers denote the absolute number of speakers, while the values in parentheses the age range of the respective groups.

Young (15-24)		Adult (25-54)		Senior (55-80)		Σ	
Male	Female	Male	Female	Male	Female	Male	Female
112	121	129	135	156	147	397	403
233		264		303		800	

6.2 Speech Database

The experimental results reported in this chapter are based on the *Speaker Likability Database* (SLD), which was introduced by Burkhardt and colleagues [43] and later used in the Likability Sub-Challenge of the Interspeech 2012 Speaker Trait Challenge [291]. It is a subset of the German aGender database [42], which was originally collected to study the recognition of age and gender of individuals. The audio signals were recorded at a sampling rate of 8 kHz over fixed and mobile telephone lines and each utterance is an instance of one of 18 different utterance types sampled from a list described in [42]. Among other, this list contains command words, embedded commands, commands relative to time and date, names, numbers, and yes/no commands. From the aGender database 800 adult speakers were selected in an age and gender balanced manner. In the attempt to minimize the effort required for annotation by multiple raters the longest sentence per speaker was selected, based on the number of words, and the annotation w. r. t. likability was used for all utterances of the respective speaker. Table 6.1 shows the distribution of the speakers w. r. t. age and gender for the three age groups "young" (age 15-24), "adult" (age 25-54), and "senior" (age 55-80).

The likability ratings of the data were determined in the following way [291]: Each audio file was assigned to one of six blocks according to its age-gender group in order to control any effect of gender and age on the likability ratings. The six groups are identical to the ones shown in Table 6.1. Then the recordings were presented to 32 raters (17 male, 15 female, age 20-42), where each rater rated only three out of the six blocks in order to minimize any effects of boredom or fatigue which might arise during the rating process. Further, both the order of the blocks as well as the order of audio files within each block was randomized. The raters were instructed to rate each stimulus according to its likability on a seven point Likert scale [205], trying to ignore the content of speech or the transmission quality of the recording. In order to form a general agreement from the individual likability ratings (16 per stimulus), the evaluator weighted estimator (EWE) (2.1) was adopted. The EWE weight for rater k,

Table 6.2: Partitioning of the Speaker Likability Database into training, development and test sub-splits (L: likable / NL: non-likable).

	Train	Devel	Test	Σ
L	189	92	119	400
NL	205	86	109	400
Σ	394	178	228	800

r_k (cf. Equation (2.2)), obviously was only computed on the recordings belonging to this rater's block. It was found [291] that the raters reliability was $0.057 \leq r_k \leq 0.697$.

For the classification task of the Interspeech 2012 Speaker Trait Challenge the EWE ratings were discretized into the classes *likable (L)* and *non-likable (NL)*, based on the median EWE rating of all recordings in the database. Finally, the data was partitioned into training, development, and test sub-splits as shown in Table 6.2.

6.3 Experiments and Results

6.3.1 Experimental Setup

The feature set used in this experiment is the supra-segmental feature set described in the baseline paper of the Interspeech 2012 Speaker Trait Challenge [291]. It contains 6125 features derived from applying functionals to the underlying LLDs, very similar to the ones described in Chapter 3.1.2. This feature set is a predecessor to the supra-segmental feature set shown in Table 3.2. It differs from the latter in two aspects: first the number of LLDs is 64 instead of 65, because the spectral centroid is not included. Second, fewer functionals are applied to the LLDs. The computed features are normalized via z-score normalization (3.24), where the moments are computed on the training sub-set. Interestingly, the normalized feature set turns out be approximately Gaussian distributed, which is beneficial when pre-training the employed neural networks with the Gaussian-Bernoulli RBM method (cf. Section 5.8.1), which by design is matched to inputs with a Gaussian distribution.

Given that the utterance-level, supra-segmental features are relatively few in number and hence the risk of overfitting is imminent, each of the examined neural networks is pre-trained via GBRBM for 50 epochs using contrastive divergence (5.89) with one Gibbs step per mini-batch. For the pre-training, the learning rate λ is set to 10^{-3} for the weights and biases and 10^{-6} for the variance parameter σ of the GBRBM. Informal experiments show that momentum is not helpful, instead L_2-regularization (5.79) is applied. Additionally, it is helpful to enforce a moderate

amount of weight sparsification to the first layer of the RBM. This is achieved by setting all weights below a certain sparsity threshold to zero after each epoch. In these experiments the optimal sparsity threshold is found to be 0.08 by preliminary experiments. Given the Gaussian distribution of the normalized input features GBRBM training is used to pre-train the first layer of the neural network, while all subsequent layers are pre-trained via the regular RBM approach.

As common in RBM training each layer is trained until convergence in an unsupervised fashion on shuffled input features using a batch size of 20. Then, the layer parameters are frozen and the next layer is stacked on top of the lower layers. Subsequently, the RBM training is repeated, adapting only the top layer parameters. For the binary classification task of the Likability Sub-Challenge the output layer is chosen to be a logistic regression layer with one output node.

Finally, the full network is fine-tuned using mini-batch stochastic gradient descent using the cross-entropy loss function. Interestingly, preliminary experiments show that a very small batch size of 2 proves to give optimum results. Fine-tuning is run until the loss does not improve on the development set for 10 epochs. Further, the optimal results are obtained with a learning rate of $\lambda = 0.05$, an L_2-regularization coefficient of 10^{-4}, and no weight sparsification during the fine-tuning stage.

6.3.2 Results

The evaluation measure in the Interspeech 2012 Speaker Trait Challenge was unweighted accuracy (UA), which is identical to the unweighted average recall (UAR), defined in Equation (4.4). Since the classes are reasonably balanced, i. e. the number of utterances with labels 'L' and 'NL' are approximately equally distributed for all data sub-splits (cf. Table 6.2), the UAR should be approximately similar to the accuracy. It is noted that the results, which were made available in a pre-publication by Brueckner and Schuller [35], are reported for the experiments that were submitted to the Challenge site and which were returned by the Challenge organizers. Participants were allowed to submit only five uploads of their predictions on unlabeled test data. Therefore, a number of preliminary experiments were conducted to find a reasonably optimal starting point for the final evaluations. Table 6.3 shows the results of the Challenge baseline, a one-layer feed-forward DNN without unsupervised pre-training and pre-trained networks varying the number of layers. Each layer is composed of 2048 nodes, which proves to be the optimum number of units per layer on the development set.

A number of interesting conclusions can be drawn from the table: first, the 1-layer network without pre-training performs worse on the development set, but exceeds

Table 6.3: Results for the classification task ('L' vs. 'NL') of the Likability Sub-Challenge. All FF-DNN layers contained 2048 nodes. 'no pre-train.' denotes a network which was not pre-trained via RBMs. Test results are reported for the experiments that were submitted to the Challenge site and which were returned by the Challenge organizers. Participants were allowed to submit only five uploads of their predictions on unlabeled test data.

	Test UAR [%]	Devel UAR [%]
Baseline (random forests) [291]	59.0	57.6
FF-DNN (1 layer, no pre-train.)	60.9	56.4
FF-DNN (1 layer)	**64.0**	**57.2**
FF-DNN (2 layers)	62.9	56.2
FF-DNN (3 layers)	62.2	56.0
FF-DNN (4 layers)	–	54.1

the baseline test results. Second, unsupervised pre-training outperforms both the baseline and network without pre-training on both the test and the development set. Finally, adding additional layers on top of the first layer does not improved performance. Instead performance decreases with each additional layer. This might indicate the occurrence of overfitting given the small dataset using utterance-level, supra-segmental features of large size. The best model topology, a 1-layer FF-DNN pre-trained with GBRBM, significantly (at a significance level of $\alpha = 0.05$) improves the baseline results of 59.0 % UAR to 64.0 % UAR, which constitutes a relative 8.5 % improvement.

Instead of stacking multiple layers with a large number of nodes, the number of subsequent layers can be gradually reduced to obtain a *pyramidal* neural network structure, similar to a bottleneck structure. Such models were successfully applied in previous challenges, e. g. in the Interspeech 2010 Paralinguistic Challenge [363]. However, in the current experiments no expansion of layers after the bottleneck layer is carried out. As before all networks are pre-trained with RBM and fine-tuned afterwards. The results obtained by different pyramidal topologies are depicted in Table 6.4. It can be observed that this approach does not lead to further improvements over the 1-layer network with pre-training. While the first two networks actually beat the baseline performance they all are inferior to the best network from Table 6.3.

Given the huge amount of parameters in most of the described networks and the limited amount of training data in relation it should be conjectured that the main reason for the limited performance of the deeper networks is overfitting. However,

Table 6.4: Results for the classification task of the Likability Sub-Challenge for pyramidal network topologies. The number is parentheses denote the number of nodes per layer. Test results are reported for the experiments that were submitted to the Challenge site and which were returned by the Challenge organizers. Participants were allowed to submit only five uploads of their predictions on unlabeled test data.

	Test UAR [%]	Devel UAR [%]
Baseline (random forests) [291]	59.0	57.6
FF-DNN (6125-2048-1024-256)	60.2	60.3
FF-DNN (6125-2048-1024-256-32)	–	59.1
FF-DNN (6125-1024-256-32)	–	53.6
FF-DNN (6125-1024-256-32-8)	–	56.2

informal, previous experiments exploring the topology space showed that smaller networks did not perform better or even worse. Further is was verified that the the UAR evaluated on the training data was not much better than on the development data, which confirms that overfitting is not the root cause for the described observations.

6.4 Conclusions

In this chapter the application of deep feed-forward neural networks to the task of classifying the likability of voices was expounded, a difficult paralinguistic task given its highly subjective nature. The experiments were trained and evaluated on the Speaker Likability Database, which was the official dataset in the Likability Sub-Challenge of the Interspeech 2012 Speaker Trait Challenge, using the large supra-segmental, utterance-level, ComParE challenge feature set. In contrast to the other paralinguistic tasks described in the following chapters, increasing the neural network depth did not lead to improved performance with respect to a (1-layer) MLP. Instead, unsupervised pre-training of the network via a regularized Gaussian-Bernoulli Restricted Boltzmann Machine (GBRBM) gave significantly higher performance over the corresponding network without pre-training. This shows that unsupervised pre-training can be particularly important in situations where the dimension of features is large and the amount of training data relatively limited. With this approach the best performing network topology, a one-layer network composed of 2048 nodes, improved the challenge baseline result of 59.0 % UAR to 64.0 % UAR which constitutes a relative improvement of 8.5 %.

Chapter 7

Social Signal Detection

Nonverbal communication forms a social language
that is in many ways richer and more fundamental than our words.

Leonard Mlodinow

Contrary to the utterance level prediction problem in the previous chapter, the research objective in this study, which was presented by the author and his colleague in [36, 37, 39], is the problem of detecting social signals on a frame level resolution. This is first addressed by investigating the use of feed-forward DNNs and the effect of pre-training via stacked autoencoders. It is further demonstrated how higher-order posterior networks improve the performance by effectively smoothing the posterior trajectories. Then, the non-linear characteristics of recurrent neural networks, in particular stacked BLSTMs, are exploited to improve upon the previous approach. Moreover, it is proposed to adopt a hierarchical DNN-BLSTM network topology to obtain state-of-the-art results on the Interspeech 2013 Computational Paralinguistics Challenge. Finally, it is shown how a real-time and resource-efficient approach using LSTMs and GRUs can be achieved, and a details study demonstrates its effectiveness in both mono-lingual and cross-lingual scenarios.

7.1 Social Signal Detection in Speech

As alluded to in Chapter 2 nonverbal communication is an essential component of human interaction, complementing verbal communication, i. e. the semantics and content of speech. This nonverbal communication is not limited to the acoustic modality and includes facial expressions, gesture, posture, etc. It plays a fundamental role in human communication, because nonverbal cues are one of the main carriers of *social signals*, which have been defined to be *"acts or structures that influence the*

behavior or internal state of other individuals" [222], *"actions whose function is to bring about some reaction or to engage in some process"* [40], or *"communicative or informative signals which [...] provide information about social facts"* [251].

In the context of paralanguage, i. e. the nonverbal communication based on vocal cues, *laughter* and *filler* are two of the most important social signals that can be observed in social interactions: they carry information about a speaker's emotional state [352], personality, or other speaker-related traits [293]. Laughter is commonly associated with spontaneous or contrived affective expressions, such as happiness, amusement, but also embarrassment or discomfort [16, 275, 339, 348]. Fillers on the other hand are vocalizations, such as "ah", "eh", or "uhm", and are often used in human conversations to hold the floor in situations of uncertainty or hesitations [57].

Humans effortlessly elaborate nonverbal cues [354]. In order to build socially intelligent technologies, such as virtual assistants, it is hence important to automatically detect, analyze, and even generate such social signals. This could help to provide a more natural and successful dialog. Since both laughter and fillers can occur basically at any point in the audio stream, using an expert model to detect the begin and end of these events can be beneficial in many use cases [37].

Related work

As pointed out by the author and his colleague [37], one of the earliest attempts to detect social signals was presented by Kennedy and Hauptmann [177] who trained Hidden Markov Models (HMMs) to recognize non-word sounds in television broadcasts, dedicating a small number of HMM parameters to these sound events. Schuller et al. [290] later investigated different strategies for the discrimination between four types of non-verbal vocalizations: laughter, breathing, hesitation, and consent. They adopted HMMs, Support Vector Machines, and Hidden Conditional Random Fields using a broad selection of diverse acoustic low-level descriptors and statistical functionals. They found that HMMs outperform other classifiers. Wagner et al. [356] instead applied a SVM classifier to phonetic patterns extracted from raw speech transcriptions, using a sliding-window scheme computing histograms of phoneme occurrences including some temporal context. Janicki [171] also resorted to a SVM classifier, but on a mixed set of differential and absolute log-likelihood scores of a GMM model with a high number of Gaussians and a relatively long context window. Finally, Gupta et al. [122] used linear low-pass filtering and masking techniques followed by a stacked generalization framework in order to smooth the fluctuant posterior time trajectories of their approach.

First attempts of applying neural networks for detecting non-verbal vocalizations from speech, and especially laughter, appeared already a decade ago [187], but only used a single-layer feed-forward (FF) neural network. Schuller and colleagues [290]

applied several different approaches based on dynamic modeling and Hidden Markov Models (HMM), Conditional Random Fields (CRF), and Support Vector Machines (SVM), or Non-Negative Matrix Factorization (NMF) on this problem. The Social Signals Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge (ComParE) [292] then finally laid a basis to objectively compare research research efforts on laughter and filler detection [8, 193]. More recently the research community has seen work on this problem employing genetic algorithms [111] and context-aware probabilistic decisions [123].

7.2 Databases

The experiments in this chapter are executed on two different databases. The first one is the *SSPNet Vocalisation Corpus (SVC)*, which was used in the Social Signals Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge (ComParE) [292] and is described in the next section. The second corpus is the recent *Automatic Sentiment in the Wild (SEWA)* database[1], which will be described in Section 7.2.2.

7.2.1 SSPNet Vocalisation Corpus

The *SSPNet Vocalisation Corpus* [283] consists of 2763 audio clips, each 11 seconds long, which are annotated in terms of laughter and fillers. This corpus was extracted from the SSPNet-Mobile Corpus [253] involving 120 subjects (63 females and 57 males), which were engaged in phone calls, where the subjects were asked to address the Winter Survival Exercise [283, 355], a scenario often used in behavioral experiments. The conversations were recorded on the phones (model Nokia N900) of each participant of the call and sum up to a total duration of 8 hours and 25 minutes. Since the clips were extracted from the microphone recordings, each audio instance only contains the voice of one individual speaker. It was ensured that each clip contains at least one instance of laughter or filler of length $1.5s \leq t \leq 9.5s$. Further it was guaranteed that clips from the same speaker never overlap, while clips from two speakers of the same conversation might overlap; this can happen e. g. in the case of simultaneous laughter. Altogether, the SVC corpus contains approximately 3.0k filler events and 1.2k laughter events. Further, both types of vocalization can be considered fully spontaneous [292].

The task of the Social Signals Sub-Challenge is to perform a frame-wise classification of the three vocalization classes *laughter*, *filler*, and *garbage*, which comprises all other vocalizations, such as speech, but also including silence. In all experiments the

[1] `http://sewaproject.eu`

Table 7.1: Partitioning of the SSPNet Vocalisation Corpus into train, development, and test sets. Values denote the absolute number of utterances, vocalization segments, and frames. [292]

	Train	Devel	Test	Σ
Utterances				
Σ	1,583	500	680	2,763
Segments				
Laughter	649	225	284	1,158
Filler	1,710	556	722	2,988
Frames				
Laughter	59,294	25,750	23,994	109,038
Filler	85,034	29,432	35,459	149,925
Garbage	1,591,442	492,607	684,937	2,768,986
Σ	1,735,770	547,789	744,390	3,027,949

same data division is applied as in the Challenge: All data are divided into speaker disjoint subsets for training (70 speakers, calls 1-35), development (20 speakers, calls 36-45), and testing (30 speakers, calls 46-60), and are manually segmented into 'garbage' (∼2.8 million frames), 'laughter' (∼109,000 frames), and 'filler' segments (∼150,000 frames). The resulting partitioning is shown in Table 7.1.

7.2.2 SEWA Corpus

The *Automatic Sentiment in the Wild (SEWA)* database [191] is a recent audio-visual database collected from 398 subjects (201 male, 197 female) from 6 different cultural backgrounds: British, German, Hungarian, Greek, Serbian, and Chinese. It further exhibits a broad distribution in gender and age. All recordings were made 'in the wild', i. e. not under laboratory settings but on arbitrary desktop PCs or notebooks with standard webcams and microphones, and hence exhibit spontaneous and natural behavior. The SEWA data collection was conducted using a website specifically built for this task, which allowed the participants to be recorded in truly unconstrained in-the-wild environments. All subjects participated in pairs, staying in different rooms, either at their home or in an office. Each subject had to watch 4 different commercials, while being recorded. The spots had been chosen with the intent to evoke various emotions, such as compassion, joy, or boredom. After watching the last spot of 90 s duration the subjects were asked to discuss about this last clip in a video chat. There were no restrictions on the aspects to discuss; the maximum length of the conversation was 3 minutes, but participants were allowed to finish at any

Table 7.2: Distribution statistics for the SEWA database for British English and German [39].

	British	German
Number of subjects	66	64
Total duration (min)	90	89
Number of frames	546,233	533,470
- Laughter	10,843 (2.0 %)	16,700 (3.1 %)
- Filler	3,2701 (6.0 %)	32,017 (6.0 %)

time earlier. It was required that both subjects know their partner (either relatives, friends, or colleagues), to ensure that an unreserved discussion could develop. The pairs were balanced w. r. t. gender (female-female, female-male, male-male). Different age ranges (18+) are represented in the database; however, about half of the subjects are between 18 and 30 years old. The complete SEWA database was manually transcribed, including the nonverbal vocalizations 'laughter' and 'filler'. Given the fact that most of these events occur during the video chat sessions and not during the sessions of subjects watching advertisements, the experiments are restricted to the *video chats* only. Since this thesis focuses on acoustic-based approaches of deep learning only the audio part of the recordings was taken into account. Further, since at the time of running these experiments not all languages were supplied with robust labels of the relevant vocalization classes only *British* and *German* recordings were considered. [39] Table 7.2 shows the distribution statistics computed on the SEWA database for British English and German.

7.3 Acoustic Feature Sets

Since the objective of the task in this chapter is the frame-wise detection and localization of social signals, the supra-segmental, utterance-level features often used in paralinguistic tasks are disused. Instead only a relatively small set of frame-wise descriptors is used in this study. For the experiments using the SSPNet Vocalisation Corpus, the LLD features from the Social Signals Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge (ComParE) [292] are used. Using the TUM openSMILE open-source feature extractor [84], frame-wise low-level descriptors (LLDs) and functionals are extracted every 10 ms adopting a frame size of 25 ms. In particular, MFCCs 1–12 and logarithmic energy are computed along with their first and second order delta regression coefficients. These are augmented by voicing probability, HNR, F_0 and zero-crossing rate, as well as their first order delta coefficients. Then, for each frame-wise LLD the arithmetic mean and standard deviation across the frame itself and eight of its neighboring frames (four before and

four after) are calculated. This results in $47 \times 3 = 141$ descriptors per frame. For the experiments on the SEWA database, however, a slightly different feature set is adopted. Again, the openSMILE toolkit is used to extract frame-based LLD vectors every 10 ms, but this time the LLDs from Chapter 3.1.1, as shown in Table 3.1, are used. All feature sets are subsequently z-score normalized, i. e. they are transformed to have zero mean and unit variance, where the respective first-order moments are computed on the corresponding training data set.

7.4 Experiments and Results

7.4.1 Regular Posterior Baseline System

7.4.1.1 Experimental Setup

As a baseline reference, feed-forward DNNs are trained on the frame-level class targets on the full training set of the SSPNet Vocalisation Corpus. In the following, the predicted network outputs without any additional smoothing are referred to as *regular posteriors*, in order to distinguish them from the smoothed, higher-order extension, as will be explained below. The input to all networks are the frame-wise features described in the previous section. Further, context windows of n subsequent feature frames, with $1 \leq n \leq 15$, are built as follows: A sliding window from $t-(n-1)/2$ to $t+(n-1)/2$ is applied to merge n successive D-dimensional feature vectors $\mathbf{x}(t) \in \mathbb{R}^D$ to an $(n \cdot D)$-dimensional extended feature vector $\mathbf{x}'(t)$, i. e.

$$\mathbf{x}'(t) = \left[\mathbf{x}\left(t - \frac{n-1}{2}\right), \ldots, \mathbf{x}(t), \ldots, \mathbf{x}\left(t + \frac{n-1}{2}\right)\right] \quad \text{for} \quad \frac{n-1}{2} < t \leq T - \frac{n-1}{2}\,. \tag{7.1}$$

Given the default frame duration of 20 ms and a frame shift of 10 ms this amount to a maximum temporal context of approximately 160 ms per context window, which is known to be in the range of average phone durations of human speech [389].

The networks are trained via Stochastic Gradient Descent (SGD) using Nesterov momentum and L_2-regularization on the layer weights. The training process is stopped once the cross-entropy (CE) loss of the development set has not improved for at least 10 consecutive epochs. Then the model which has achieved the lowest CE error is used during inference. Informal tests have shown that this is a reliable indicator for the final UAAUC performance on this imbalanced data set. Further, all meta-parameters used in training the neural networks, such as the number and size of the hidden layers, learning rate, momentum, batch size, etc., are chosen to be the ones that give the highest unweighted average area-under-the-curve (UAAUC)

Table 7.3: Regular posteriors: Comparison of a single-hidden layer MLP and a two-hidden layer SAE for different hidden layer sizes on the development set.

UAAUC [%]	Number of hidden units per layer				
	64	128	256	512	1024
MLP	92.5	92.8	**93.0**	92.8	92.7
2-layer SAE	93.1	93.4	**93.7**	93.4	93.3

Table 7.4: Regular posteriors: Effect of the number of hidden layers in a deep SAE on the UAAUC (development set). Each layer consists of 256 hidden units.

UAAUC [%]	Number of hidden layers				
	1	2	3	4	5
Deep SAE	93.0	**93.7**	93.4	93.2	93.0

(cf. Equation (4.9)) value on the development set. As explained in Chapter 4.1.1 the UAAUC is a suitable classification measure in multi-category scenarios and was also the official target measure in the Interspeech 2013 Computational Paralinguistics Challenge (ComParE) [292]. This type of training approach is followed for all subsequent experiments in this chapter, unless otherwise noted.

7.4.1.2 Results

Two different network setups are investigated: a single-layer FF-DNN without pre-training and a deep DNN, pre-trained with the stacked auto-encoder (SAE) approach. Contrary to the results in the likability task (cf. Chapter 6) pre-training the single-layer FF-DNN does not improve performance. Further, initial tuning shows that a context of 11 frames (5 left + center + 5 right) consistently gives best results. Table 7.3 shows the comparison of the UAAUC for a single-hidden layer MLP and a two-hidden layer SAE for different layer sizes. For both networks the use of 256 hidden units per layer give best results and one can observe that the 2-layer DNN with SAE pre-training outperforms the one-layer network.

Fixing the number of hidden units to 256 the number of layers is varied, where each layer is again pre-trained with the stacked auto-encoder method. The results, reported in Table 7.4, show that 2 layers indeed are optimal for the given number of hidden units. Stacking additional layers on top leads to a continuous degradation of performance.

7.4.2 Enhanced Posteriors and Posterior Smoothing

7.4.2.1 Experimental Setup

In frame-wise detection problems, in particular the current problem of detecting laughter and fillers, quite often spurious drops or spikes in the trajectories of the posterior probabilities (network outputs) can be observed. One way to reduce these artifacts is to a apply a sliding window smoothing filter, e. g. an ARMA filter [330]. Another way to improve upon the performance of posterior-based systems is to build a second network on top of the first one, thus building a *hierarchical* neural network. This approach has been shown to to improve results in the realm of ASR systems [178]. In the following, let the model which contains all layers from the input to the first softmax be called the *feature model*. This model processes the input features and generates posteriors at its (softmax) output. These posteriors will be referred to as *regular* or *first-order* posteriors. Any subsequent network, which takes the regular posteriors as input will be called *posterior model*, and the smoothed, modified posteriors shall be termed *enhanced* or *higher-order posteriors*.

Higher-order posteriors are formed the same way as stacked input features: The regular posteriors, i. e. the outputs of the feature model, are stacked into a posterior context window as in (7.1) and are fed into the posterior model, which now has the ability to learn long-term inter- and intra-dependencies between class evidences (posteriors) in the training data and to transform the regular posteriors into enhanced posteriors. Figure 7.1 shows a schematic example of this process. The long term dependencies captured by the posterior model leads to an enhancement of the quality of the overall class posteriors. As conjectured by [178], the rationale is that at the output of every neural network, the information stream gets simpler (converging to a sequence of binary posterior vectors), and can thus be further processed (using a simpler classifier) by looking at a larger temporal window.

A graphical representation of the posterior trajectories over time is referred to as a *posteriorgram* [107]. Figure 7.2 depicts a typical example of a posteriorgram for both regular and enhanced posteriors of one of the utterances in the SSPNet Vocalization database. Evidently, the enhanced posterior trajectories are much smoother than their regular counterparts, which usually leads to fewer false detections. A disadvantage of this smoothing is that the transitions at the class boundaries are also washed out to some degree. It should hence be expected that there will be more errors in these transition areas.

For training the *second-order* enhanced posterior networks n consecutive frames of the three-dimensional regular posteriors ("laughter", "filler", and "garbage") are stacked with $3 \leq n \leq 201$, which amounts to a maximum temporal context of

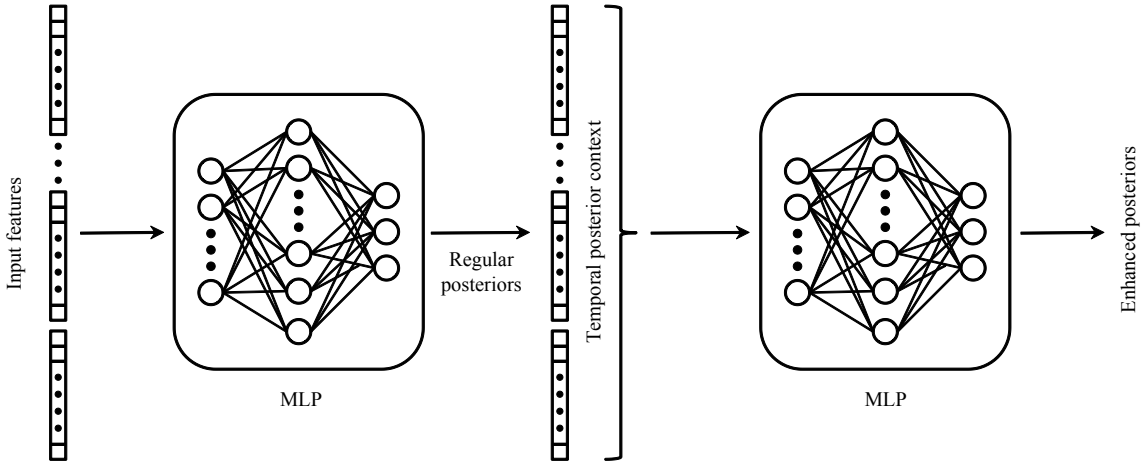


Figure 7.1: Hierarchical network to generate enhanced posteriors: The feature model transforms stacked (acoustic) features into regular posteriors. A temporal context of those posterior vectors is created by frame stacking. The posterior model processes the temporal context of regular posteriors and learns long term dependencies to estimate enhanced posteriors.

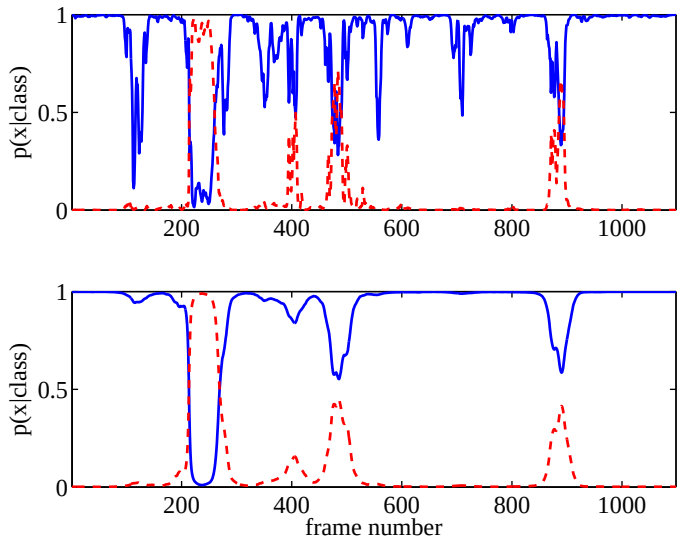


Figure 7.2: Example of posteriorgrams showing the posterior trajectories over time for one utterance. The plot on the top shows the posteriorgram of the regular posteriors for the two classes *garbage* (solid blue line) and *laughter* (dotted red line). The plot on the bottom shows the posteriorgram of the enhanced posteriors for the same classes and utterance.

approximately 2 seconds. As previously the set of meta-parameters is chosen to be the one which yields the highest UAAUC value on the development set. Notice that

Table 7.5: Enhanced posteriors: Comparison of the effect of the temporal context of stacked regular posteriors for a MLP with 256 hidden units on the development set.

Nr. of context frames	51	75	101	151	175	201
UAAUC [%]	96.6	96.9	97.1	**97.3**	97.2	97.1

Table 7.6: Enhanced posteriors: Comparison of the effect of the number of hidden units for a MLP using an input context of 151 frames on the development set.

Nr. hidden units	64	128	256	512	1024	2048
UAAUC [%]	96.8	97.1	**97.3**	97.2	97.2	97.1

the enhanced posterior network is trained of fixed inputs, i. e. first the input network is trained to convergence and then its parameters are frozen. The enhanced posterior network is then added and trained, without fine-tuning the input network.

7.4.2.2 Results

Taking the 2-layer SAE network with 256 hidden units per layer as the feature model, as found in the previous section, a single-layer neural network (MLP) of size 256 is stacked on top to learn the relationships between neighboring posterior values and to generate enhanced posteriors. Table 7.5 shows the impact of varying the context size of the stacked regular posteriors. Best results are obtained for a context of 151 frames, which accumulates evidence across approximately 1.5 seconds. At first glance, this seems to be a fairly large context. However, presumably the feature model, which produces the regular posteriors, uses a much smaller window of approximately 160 ms to predict the presence of social signals with a finer time resolution, while the smoothing posterior model collects this evidence to smooth out outliers and improve the robustness of the final prediction. Nonetheless, it is reassuring to observe, that the performance is not overly sensitive to the context size. Thus, one could reduce the context size, for example to reduce delay in real-time systems. Tuning the number of hidden units in the posterior model shows that 256 is actually a good size, as shown in Table 7.6. Again, the decrease in performance is rather small as one deviates from the optimum number of hidden units.

7.4.3 Higher-Order Posteriors and Comparison

7.4.3.1 Experimental Setup

Extending the idea of enhanced posteriors from the previous chapter, the resulting second-order posteriors can again be fed into another neural network to create so-called *higher-order* posteriors. The performance of third-order posteriors is also

Table 7.7: Third-order posteriors: Results obtained for a 2nd-order MLP on the development set. The first row shows the number of frames of regular posteriors (output from the feature model) used to build the input of the posterior model (MLP). The second row shows the number of frames of enhanced posteriors (output from the posterior model) used to build the input to the third MLP.

Nr. of context frames (regular)	51		151	
Nr. of context frames (enhanced)	51	151	51	151
UAAUC [%]	96.8	96.9	**97.1**	96.8

examined in this study. Training is executed as described above for the other networks. In particular, the final, higher-order posterior network is trained on fixed inputs, as described in Section 7.4.2.1.

7.4.3.2 Results

Table 7.7 depicts the results for higher-order posteriors for two different context window sizes, 51 and 151, respectively. Comparing these results to those shown in Table 7.5 one can observe that for shorter, sub-optimal context lengths (51 is this case) higher-order posteriors actually give rise to a slight improvement. However, for the optimum context length of 151 frames the performance decreases minimally. Is is presumed that this is due to the effect of overly smoothing the posterior trajectories, especially at the transition boundaries between classes. One can summarize that for the task at hand going beyond second-order posteriors does not lead to performance improvements.

A synopsis of the best results obtained with the described approaches on the SSPNet Vocalization database can be found in Table 7.8, together with detailed results for the social signal classes 'laughter' and 'filler'. The table also contains the results from the Challenge baseline. Leveraging enhanced posteriors a UAAUC of 97.3% is achieved on the development set and 92.4% on the test set, which significantly outperforms the Challenge baseline results by 9.7% and 9.1% UAAUC absolute, respectively, at a significance level of $\alpha = 0.001$. Further, it also significantly ($\alpha = 0.001$) excels the performance of the winner of the Challenge [122] by 2.4% and 0.9% UAAUC absolute. Note that for the results of the baseline on the test set the respective models were retrained on the union of the training and development sub-sets. Retraining the posterior networks on the combined sub-sets, the results slightly worsen, so the results on the test set are based on networks that are trained on the training set only.

Table 7.8: Summary of best results on the SSPNet Vocalization database. Depicted are results on the development ('dev') and the test set using models trained on the full training set. Note that the test results for the baseline are obtained training on the combination of training and development set.

		Dev set	Test set
Baseline	AUC [Laughter]	86.2	82.9
	AUC [Filler]	89.0	83.6
	UAAUC [%]	87.6	**83.3**
Regular posteriors	AUC [Laughter]	92.8	90.5
	AUC [Filler]	94.5	88.0
	UAAUC [%]	93.7	**89.2**
Enhanced posteriors	AUC [Laughter]	98.1	94.9
	AUC [Filler]	96.5	89.9
	UAAUC [%]	97.3	**92.4**

7.4.4 BLSTM Networks

7.4.4.1 Experimental Setup

Since BLSTM networks have shown excellent performance on frame-wise tasks in recent years on several tasks [114, 364], it is of interest to explore if they can also compete in the social signals domain. Therefore, firstly, experiments are carried out in the regular posterior setup by replacing the FF-DNN by a single BLSTM layer (i. e. one layer per input direction), where each BLSTM memory block contains one single memory cell. After determining the optimum number of BLSTM memory blocks of this feature model, a BLSTM network is stacked on the outputs of the first network, operating on the regular posteriors to compute enhanced posteriors, but in a recurrent way. This essentially forms a bottleneck architecture, where the bottleneck layer consists of the (regular) posteriors after the softmax operation. This novel approach of creating a deep BLSTM, which uses a self-learned context to model the time trajectories of the posteriors, is shown to be highly competitive w. r. t. previous approaches.

7.4.4.2 Results

Table 7.9 shows the results of varying the number of memory blocks in the BSLTM network. Best results are obtained when using 50 blocks. Comparing the results to the ones obtained by FF-DNNs (shown in Table 7.8) one can conclude that the single-network BLSTM outperforms the regular posterior networks significantly on

Table 7.9: Regular posteriors (BLSTM): Comparison of the number of memory blocks.

BLSTM network topology	UAAUC [%] Devel	Test
141-30-3	96.3	91.5
141-40-3	96.6	92.1
141-50-3	**97.0**	**93.0**
141-60-3	96.3	91.8
141-80-3	96.3	91.5

both the development and test sets. In fact, it even excels the test set results of the enhance posterior network from 92.4% to 93.0%, an 8% relative improvement, even though it performs slightly worse on the development set.

As shown in Table 7.10, adding another BLSTM on top of the regular posteriors increases the performance on the test set by an additional 0.4% to 93.4%, while not showing any gain on the development set. The relatively small gain compared to the gain achieved in the FF-DNN case clearly shows that the first BLSTM network already learns much of the temporal structure necessary to predict smooth posteriors for the task, presumably extending the effective context beyond the context seen by the FF-DNN sliding windows. Yet, the gain achieved by the posterior network is significant on the test set, which indicates that there is still valuable information contained in the temporal structure of the class posteriors generated by the first network that can be exploited by the higher-layer network. As a control experiment a 'regular' deep BLSTM without the intermediate output layer (of size 3) is constructed and evaluated. The number of memory blocks in each layer is optimized on the development set and the result of the best-performing topology is shown on the bottom line of Table 7.10. Evidently, the deep bottleneck network offers superior performance over the deep, regular BLSTM. The reasons for this behavior is unclear, but it is possible that the bottleneck layer functions as a strong regularizer.

A remaining question is the fact that the UAAUC on the development set does not improve in the same way as the UAAUC on the test set. One possible reason could be that the CE loss on the development set, which is used as an early stopping criterion, ignores the data imbalance of the data set and hence is not directly correlated to the *absolute* value of the UAAUC. However, *within* each experimental group the dev set UAAUC is a good indicator for the performance on the test set.

Table 7.10: Enhanced posteriors: Comparison of the number of memory blocks in the second layer BLSTM.

Stacked BLSTM network topology	UAAUC [%] Devel	Test
141-50-3-10-3	96.8	93.2
141-50-3-20-3	**96.9**	**93.4**
141-50-3-30-3	96.7	93.1
141-50-3-40-3	96.6	92.9
141-50-3-50-3	96.7	93.0
141-50-20-3	96.6	91.7

7.4.5 Hierarchical DNN-BLSTM Networks

7.4.5.1 Experimental Setup

In a final experiment, the feature model BLSTM network of the hierarchical bottleneck architecture is replaced by the network that produced the best results from Section 7.4.2, while the posterior model network remains a BLSTM. This effectively forms a hierarchical, deep DNN-BLSTM network. The DNN in this study consists of two hidden layers of size 256, pre-trained as a stacked autoencoder and subsequently fine-tuned using stochastic gradient descent.

7.4.5.2 Results

The results of the combined DNN-BLSTM hierarchical network are given in Table 7.11. A deep DNN-BLSTM network, which consists of two hidden layers with 256 units each in the feature model and a posterior model BLSTM network with 20 memory blocks, achieves a UAAUC of 94.0% on the test set and 97.2% on the development set. To the author's knowledge this represents the best results on the SSPNet Vocalization dataset published so far. Interestingly, the DNN performs better than a BLSTM when used as the first network module. It seems to capture structure of the feature set that is not conveyed by its temporal characteristics, but some other form of inherent information.

Table 7.12 summarizes the key results obtained in this study and compares it to the baseline results of the Social Signals Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge (ComParE) [292]. Further, the performance obtained with (one-directional) LSTMs is shown for comparison. Evidently, BLSTMs manage to exploit future time context in each utterance, which boosts its performance and shows the modeling power inherent in BLSTMs w. r. t. to LSTM models.

Table 7.11: Deep DNN-BLSTM: Comparison of the number of memory blocks in the second layer BLSTM.

Effective network topology	UAAUC [%] Devel	Test
141-256-256-3-16-3	96.7	92.2
141-256-256-3-20-3	**97.2**	**94.0**
141-256-256-3-24-3	96.8	93.0
141-256-256-3-30-3	96.9	93.3
141-256-256-3-50-3	96.7	92.5

Table 7.12: Summary of results obtained in this study on the the SSPNet Vocalization database.

Model architecture	Network topology	UAAUC [%] Devel	Test
Baseline [292]	–	87.6	83.3
Regular posterior	141-256-256-3	93.7	89.2
Enhanced posterior	141-256-256-3	97.3	92.4
LSTM	141-50-3	95.3	90.9
BLSTM	141-50-3	97.0	93.0
LSTM-LSTM	141-50-3-20-3	95.1	90.7
BLSTM-BLSTM	141-50-3-20-3	96.9	93.4
DNN-LSTM	141-256-256-3-20-3	95.2	90.3
DNN-BLSTM	141-256-256-3-20-3	**97.2**	**94.0**

7.4.6 Mono- and Cross-Lingual Social Signals 'in the Wild'

7.4.6.1 Experimental Setup

The research objectives in this study, which was presented by the author and his colleague in [39], are three-fold: To start with, it gives a first evaluation of the detection of social signals on the recent SEWA database. In addition, it presents the first mono-lingual and cross-lingual results on the detection of laughter and fillers in conversational and spontaneous speech collected 'in the wild'. The final goal is to give an extensive comparison of BLSTM, LSTM, and resource-efficient GRU networks on this task. This is particularly important for applications which cannot afford long time delays or have limited compute constraints.

The experiments are carried out for British English and German, and for each language the set of utterances is divided into a training (17/18 speaker pairs), validation (7 pairs), and test (8 pairs) subset. Care is taken to ensure gender-pair balancing, i. e. that the proportions of female-female, male-male, and male-female pairs are approximately constant across the subsets. All examined models are trained on the respective training set using the Adam optimizer and CE loss, until the the early stopping criterion is met; i. e. training is stopped once there is no improvement of the UAAUC on the validation set for more than 5 epochs. After early stopping the model that achieves the highest UAAUC value on the validation set is used for inference. This approach was found to be robust in previous studies [36, 37]. The parameters for the Adam optimizer are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is varied between $10e^{-4} \leq \lambda \leq 10e^{-2}$, and since ADAM is an adaptive learning-rate algorithm, not explicit learning rate annealing is applied. Further, the (recurrent) models are trained based on data chunks of size 32, i. e. the forward pass is computed on the model for 32 consecutive frames before the error is backpropagated via BPTT and the model parameters are updated. At the start of each chunk the hidden states of the recurrent models are initialized to 0. This means that no history context is carried over from one chunk to the next, which allows the randomization of chunks and which is known to speed up convergence and to improve generalization. For the frames within each chunk no context expansion is applied, i. e. the input at each time frame is one single feature frame, as described in Section 7.3.

In the *mono-lingual* study each language is examined separately, in order to gain some understanding of the performance of the different model architectures and to find a suitable topology that works well on the SEWA database. Further, the results serve as a reference for the cross-lingual experiments. The use of the training, validation, and test splits follow this subdivision. In the *cross-lingual* study instead, for each language, the combination of the training and validation sets of that language are used for training (in order to increase the amount of training data) and the test set of that language as the validation set (e. g. for early stopping). Then, the evaluation is carried out on the other language's full data set.

7.4.6.2 Results

Starting with the *mono-lingual* case it is necessary to find a suitable model topology that works well on the SEWA database. To this end, a grid search is executed over the number of cells N_{cell} in each layer with $N_{cell} \in [4, 512]$ and over the number of layers $N_{layers} \in [1, 2, 3]$. Table 7.13 shows the top-3 model topologies for all three different model types (BLSTM, LSTM, and GRU) for both languages. Stunningly, for both languages and all three model types a two-layer, inverse pyramidal topology with 32 cells in the first layer and 16 cells in the second layer works best. The results compare very favorably against the previously reported numbers on the

Table 7.13: UAAUC [%] for mono-lingual training and testing without posterior smoothing for three different model types and three different topologies per type.

Model	Topology	British		German	
		Valid	Test	Valid	Test
BLSTM	130-32-3	79.7	82.7	82.7	83.8
	130-**32-16**-3	84.7	**87.0**	83.0	**86.3**
	130-32-32-32-3	83.4	85.0	83.0	86.1
LSTM	130-32-3	79.9	80.2	81.1	82.9
	130-**32-16**-3	80.4	**81.6**	81.6	**83.1**
	130-32-32-32-3	80.7	81.6	78.6	77.6
GRU	130-32-3	77.4	78.9	81.6	84.3
	130-**32-16**-3	80.0	**84.0**	83.3	**85.9**
	130-32-32-32-3	80.7	81.6	82.8	85.6

SSPNet Vocalisation Corpus from the previous sections, given the more difficult recording conditions of the SEWA database. Further, the performance for both languages is very similar. Even though one can expect some cultural differences in the expression of social signals, the chosen approaches exhibit robustness and language-independence. While it is to be expected that BLSTM models outperform LSTM networks, due to their access to future context information, it is surprising that GRUs compare very favorably to the BLSTM models. One reason that GRUs perform significantly better than LSTM models might be due to their lower complexity, i. e. the fewer number of parameters, which could lead to improved generalization.

In the previous sections it was shown that smoothing the posteriors at the output of the feature model is highly beneficial to the overall performance. This idea is also followed in this study by taking the models with the best model topology (130-32-26-3) as the feature model and stacking another (posterior) model on top of it to generate enhanced posteriors. Contrary to the previous approach, however, in this case the inputs to the posterior model are the outputs of the feature model *before* applying the softmax nonlinearity, i. e. the so-called *logits*. In this experiment, the posterior model is trained while keeping the parameters of the feature model fixed. Further, the feature model and posterior model are always of matching type, i. e. for a BLSTM feature model a BLSTM posterior model is used, etc. A preliminary grid search performed over the number of cells $N_{cell} \in [1; 64]$ showed that the optimal number of cell in the posterior network is around $N_{cell}^{posterior} = 8$, which is used in the remaining experiments. Table 7.14 shows the effect of combining the best feature model topology from Table 7.13 with the posterior model, resulting in a full network

Table 7.14: UAAUC [%] for mono-lingual training and testing with posterior smoothing for the optimal topology 130-32-16-3-8-3.

Model	British		German	
	Posterior smoothing		Posterior smoothing	
	no	yes	no	yes
BLSTM	87.0	**87.5**	86.3	**86.7**
LSTM	81.6	82.7	83.1	83.9
GRU	84.0	84.3	85.9	86.1

topology of 130-32-16-3-8-3. The overall gain due to posterior smoothing on the test set lies between 0.2 % and 1.1 % UAAUC. It should be noted that this gain depends on the amount of 'laughter' and 'filler' events found in the data, since it also includes the dominant class 'garbage'.

Finally, a *cross-lingual* experiment is carried out, to check for language independence of social signal detection. For both language combinations (British-German and German-British) all model types are studied, both with and without posterior smoothing, and again performing a grid search over the number of layers and the number of memory blocks inside each layer, as previously. Similar to the mono-lingual case the best performing network topology for all model types turns out to be 130-32-16-3 for the feature model, and 16 for BLSTM or 8 for LSTM/GRU, respectively, for the posterior model. Table 7.15 depicts the results for the best respective setups. As in the mono-lingual case, the BLSTM models outperforms the LSTM and GRU models, but the difference is relatively small. Also, GRUs again beat the LSTM models, and in the German-British setup it is only approximately 2.0% worse than the BLSTM results. This finding is of high importance, since it shows that for on-line or low-resource applications resorting to GRU models constitutes a viable approach and the expected decrease in performance is very limited. Finally, posterior smoothing is again found to consistently and significantly improve the unsmoothed results in all experiments. Interestingly, the gains are smallest for the GRU models and largest for the LSTM models.

7.5 Conclusions

In this chapter several effective methods for detecting social signals in speech signals have been introduced. Results on the SSPNet Vocalisation Corpus have demonstrated that a single-layer MLP network already outperforms the baseline results from the Social Signals Sub-Challenge of the Interspeech 2013 Computational Paralinguistics

Table 7.15: UAAUC [%] for cross-lingual setups British (train & validation) – German (test) and German-British for various deep neural architectures, all with topology 130-32-16-3 (no posterior smoothing) and 130-32-16-3-16(blstm)/8(lstm,gru)-3 (with posterior smoothing). For a detailed description refer to the text.

Train/Validation – Test	British-German						German-British					
Model	BLSTM		LSTM		GRU		BLSTM		LSTM		GRU	
Smoothing	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes
Approx. # parameters	48k	51k	24k	25k	18k	18k	48k	51k	24k	25k	18k	18k
Valid	85.6	87.6	82.4	85.0	86.9	86.8	88.0	88.7	86.6	77.9	86.6	86.8
Test	**83.7**	**84.4**	**78.4**	**79.6**	**81.1**	**81.3**	**85.0**	**85.6**	**80.6**	**82.4**	**83.7**	**83.8**

Challenge, the performance of which can significantly be improved by unsupervised pretraining using a stacked autoencoder approach. Furthermore, a novel posterior smoothing technique was introduced, which intelligently and effectively smoothes the highly variable posterior trajectories resulting from the investigated neural networks by stacking additional neural networks which operate on the posteriors output by the underlying network. It has been shown that this process can be repeated to obtain an even higher degree of context smoothing. Leveraging the temporal learning capabilities of recurrent neural networks, this approach was extended to BLSTM networks and hierarchical DNN-BLSTM networks, which raises the performance to a new level. The proposed approaches were demonstrated to achieve state-of-the-art results on the SSPNet Vocalization Corpus. Finally, the findings were verified and extended on the recent SEWA database, which is a collection of recordings under realistic conditions, exhibiting spontaneous and natural behavior of the participants. The presented study constitutes the first evaluation of the detection of social signals on this database. In addition, it presents the first mono-lingual and cross-lingual results on the detection of laughter and fillers in conversational and spontaneous speech collected 'in the wild'. Eventually, it shares a first comparison of efficient, resource-efficient models to more conventional recurrent neural models on this task and demonstrates that the proposed models and approaches are highly effective.

Chapter 8

Conflict Detection

The aim of argument and of discussion should not be victory, but progress.

Joseph Joubert

Motivated by the fact that human speech contains valuable information about the valence and level of conflict, this chapter expounds the application of deep neural networks to the problem of automatic detection of conflict in speech. The experiments are conducted on the Conflict Sub-Challenge data set of the Interspeech 2013 Computational Paralinguistics Challenge (ComParE) [292] and the results show that the proposed approach surpasses the baseline results by a consistent margin, as was shown in a study by the author and his colleague in [38]. In fact, at the time this work was published the results were the best reported so far in the literature on both the classification and the regression task. This shows the feasibility and appropriateness of the described methods for this problem.

After describing the database, three different feature sets are presented, which are used in the experiments for the actual prediction of conflict as well as for predicting the overlap ratio, an auxiliary feature, which helps to improve the overall performance. Then, a BLSTM model is proposed to reliably estimate the overlap ratio from real speech signals and the different feature sets and methods are evaluated and compared to the Challenge baseline results. Parts of this work were presented earlier by the author and his colleague in [38].

8.1 Detection of Conflict in Speech

Although there does not seem to be a universally accepted definition of conflict, it undoubtedly is an intrinsic component of human relations, inevitably arising in

inter-human communication, manifested in incompatibility, disagreement, or difference within or between social entities (individuals, groups, organizations, etc.) [262]. Traditionally, conflict has been defined as opposing interests involving scarce resources and goal divergence and frustration [254] or incompatible activities, where one person's actions interfere, obstruct, or in some way get in the way of another person's action [59, 70]. In any case, one key element of conflict is that it is a dynamic process that does not appear suddenly, but takes time to evolve, and passes through several stages [320]. Further, it usually is the process resulting from the tension in interpersonal interactions because of real or perceived differences [68, 357].

Reliably detecting conflict is of high interest for many application scenarios, ranging from safety- and security-critical situations, such as air traffic control or public and private surveillance, over customer centers and the increasingly wide-spread artificial intelligence agents, to general data mining use cases. In all these situations both real-time, on-line deployments and off-line situations are abundant, and all focus on detecting and extracting conflictual segments as a basis for further processing and action. Since speech is a predominant carrier of information about level of conflict, this chapter focuses on the detection of conflict from audio-based signals.

Related work

Early works on the automatic detection of conflicting situations in multi-party conversations date back almost two decades [366], where the authors examined, if involvement could be reliably judged by human listeners. They concluded that human raters demonstrated significant agreement in discriminating involved from non-involved speech, despite the subjective nature of the problem. It was also found that the temporal trajectories of fundamental frequency (F_0) and speech energy represent reliable acoustic cues of involvement. A survey of audio-visual cues of agreement and disagreement was presented by Bousmalis and colleagues [32]. In this study visual cues, such as head gestures, facial and hand actions, were found to be more relevant than auditory information, for example throat clearing and sighing. Nonetheless, pauses in speech, delays, interruptions, and utterance length also play an important role. Pesarin et al. [245] proposed a semi-automatic approach, using a generative statistical technique based on Markov chains, for detecting conflict in conversational dialogues. This way they were able to identify structural patterns linking the organization of speaker turns to the presence of conflict. Feature selection was the basis for the work by Kim et al. [182]: The authors successfully adopted different kinds of regression models for automatic and manual diarization, applying Support Vector Regression, Gaussian Process models, and Bayesian Linear Regression. In an extension to this work they added the detection of conflict escalation in political debates [181, 183].

As in many other paralinguistic areas, all this work on conflict detection suffered from reduced comparability of results due to the different data sets underlying the research. In order to remedy this situation the Conflict Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge [291] introduced a benchmark dataset to allow for the objective comparison of approaches on the detection of conflict and the prediction of conflict level. Based on this data set, one of the challenge participants, Grèzes et al. [117], found that the ratio between overlapping and non-overlapping speech, the so-called *overlap ratio*, represents the single best feature for predicting the conflict level.

Given the importance of overlapping speech, a number of studies have presented approaches on how to robustly estimate overlapping speech segments in multi-party conversations. Support Vector Machines and Support Vector Regression using microphone arrays were used by Yamamoto et al. [369] to estimate the number of sound sources. Discriminant Capability Analysis was adopted by Boakye et al. [27] as a feature analysis method achieving almost oracle performance on the underlying database. In [379] Zelenák and his colleagues complemented short-term spectral features with long-term, prosody-based features, followed by a feature selection stage based on the minimal-redundancy-maximal-relevance (mRMR) criterion. Geiger et al. [96] instead used a conventional HMM system based on a feature set extracted via convolutive non-negative sparse coding. By improving the overlap detection rate they managed to reduce the overall diarization error rate. In a later study the authors added Long Short-Term Memory (LSTM) recurrent neural networks to their baseline model to improve upon their earlier results [95]. A similar idea was also used by Yella and Bourlard [372] for speaker diarization in meeting room conversations.

8.2 Speech Database

The *SSPNet Conflict Corpus* (SC^2) [182] is a subset of the Canal 9 Corpus [351], which is a publicly available collection of 45 broadcasted Swiss political debates in French language, and was also used in the Conflict Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge [292]. The data set has a total duration of 11.9 hours and the debates were segmented into 30-second long, uniform, non-overlapping clips, involving at least two speakers or more and assuming that the levels of conflict are stationary within the time period. It includes 110 subjects in total, 18 females (1 moderator and 17 participants) and 92 males (1 moderator and 91 participants).

The SC^2 corpus includes a rich set of socially relevant annotations, such as turn-taking (who speaks when and how much), agreement and disagreement between participants, and the role played by the people involved. Each debate includes one

Table 8.1: Partitioning of the SSPNet Conflict Corpus into train, validation, and test sets for binary classification ('Low' $\equiv [-10, 0[$, 'High' $\equiv [0, +10]$).

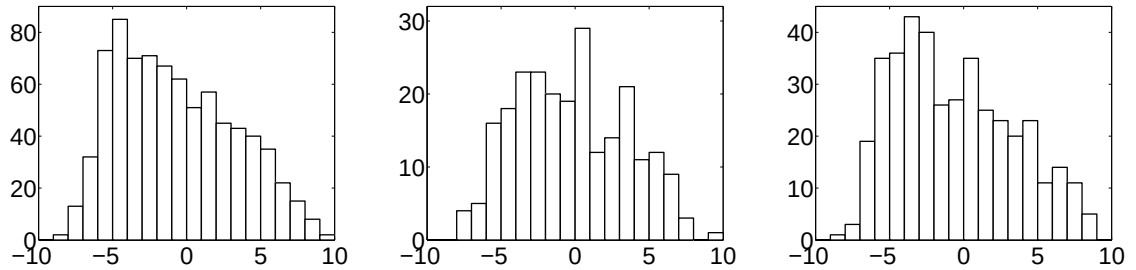
#	Train	Valid	Test	Σ
Low	471	127	226	824
High	322	113	171	606
Σ	793	240	397	1.430

moderator and two coalitions opposing each other on the issues of the day. Each clip was annotated using a questionnaire including physical (objective observation) and inferential (subjective interpretation) questions about the occurrence of conflict [353]. Annotations were conducted by 551 assessors via the Amazon Mechanical Turk online system, each clip being rated in terms of their conflict level by 10 assessors in two ways: First, a continuous conflict score in the range [-10, +10] was assigned to each clip, which allows to perform a straightforward regression task (*score*). Second, based in these score labels each clip was classified to be either of *high* conflict or *low* conflict, depending if the score value assigned to it is ≥ 0 or < 0, respectively, thus giving rise to a classification task (*class*) [38, 380].

As several subjects occur in debates with different moderators, a truly speaker-independent partitioning of the data is not possible. Since all participants (apart from the moderators) do not occur more than a couple of times (most of them only once), the following strategy was followed to reduce speaker dependence to a minimum: All broadcasts with the female moderator (speaker no. 50) were assigned to the training set. The validation set consists of all broadcasts moderated by the (male) speaker no. 153 and the test set comprises the rest of the broadcasts, containing all remaining male moderators. This further ensures that the validation and test sets are similar, in case the gender of the moderator should have an influence. This data split leads to the training set comprising approximately 55% of the data, the validation set 17%, and the test set 28%, with a strict speaker-disjoint partition not being feasible. The resulting distribution of the data is shown in Table 8.1 along with the respective binary class labels. Histograms of the continuous score ratings over the partitions are depicted in Figure 8.1.

8.3 Acoustic Feature Sets

To assess the relevance of different acoustic features three different feature sets are examined in the following experiments: **Feature set I** is the baseline feature set used in the ComParE Conflict Sub-Challenge [292] and consists in the supra-segmental feature set (cf. Section 3.1.2) listed in Table 3.2. In the experiments expounded in



(a) train (b) valid (c) test

Figure 8.1: Histograms for the level of conflict ($\in [-10, +10]$) for the Challenge partitions of the SSPNet Conflict Corpus.

this chapter one segment per utterance is used. Hence, after extracting the LLDs (cf. Table 3.1) as described in Section 3.1.1 a series of functionals are applied to extract higher level statistics over the full utterance, resulting in 6,373 features.

For time-recursive neural networks such as LSTMs supra-segmental features, especially if computed on a per-utterance basis, are an unsuitable choice, since they hinder the networks to exploit the information distributed across time. But even for regular feed-forward networks it might be beneficial to learn intermediate representations on smaller context windows of the input signal. For this reason a frame-based **feature set II** is constructed as follows: the speech waveform is split into segments of 20 ms length, shifting the segments by half their size, i. e. with a frame shift of 10 ms. For each such segment the logarithmic energy and the Mel-frequency cepstral coefficients (MFCC) 1-12 together with their first and second order delta (Δ) regression coefficients are computed. This bears resemblance to the features commonly used in automatic speech recognition. In addition, these features are augmented by the probability of voicing, the harmonic-to-noise ratio (HNR), the fundamental frequency F_0, and the zero-crossing rate (ZCR), as well as their respective first order Δ. Based on these LLDs the arithmetic mean and the standard deviation across a context window of size $n = 9$ (central frame + four to the left + four to the right) are computed and appended to the LLD features. Overall, this results in $47 \times 3 = 141$ descriptors per frame. The reason for selecting this particular set of features is that it limits the number of features to a relatively small number and that it was used with great success in the ComParE Vocalization Sub-Challenge in previous work [36, 37].

A completely different approach is used in constructing **feature set III**, which is a modification of the feature set proposed by Kim et al. [182]. It is composed of two parts, a conversational and a prosodic one. The conversational part consists of statistics about speaker turn information extracted from the speech signal: it includes turn duration statistics, namely the mean, median, minimum, maximum,

and variance of speaker turn durations, as well as the number of speaker turns. Additionally, it is complemented by the mean, median, minimum, maximum, and variance of the total speaking time for individual speakers in each recording, as well as the number of speakers present. Eventually, the overlap ratio and the turn keeping/turn stealing ratio are added; the latter defines the ratio between the number of times a speaker change happens and the number of times a speaker change does not happen after an overlap.

The conversational part of feature set III is complemented by prosodic features based on per-utterance statistics: mean, median, standard deviation, minimum, maximum, and quantiles (0.01, 0.25, 0.75 and 0.99) of pitch and intensity statistics obtained from the entire audio recording. The pitch and intensity features are identical to the ones in feature set I. These general prosodic features are augmented by the mean, median, and standard deviation of pitch and intensity computed from individual speaker turns. Overall, feature set III contains 38 features. It is important to notice that the statistics described above are estimated on single-talker segments, as well as overlapping speech segments.

8.4 Overlap Prediction Generator Network

It was found by Grèzes et al. [117] that the *overlap ratio*, i. e. the proportion between overlapping and non-overlapping speech represents the best isolated feature for predicting the level of conflict in audio signals. Geiger et al. [95] used LSTM networks to robustly estimate this feature and the author and his colleague [38] extended this idea to BLSTM models to generate frame-wise overlap predictions. Let

$$\mathbf{X} = \mathbf{x}(1), \ldots, \mathbf{x}(T) \tag{8.1}$$

be the sequence of T feature vectors $\mathbf{x}(t)$ extracted from an audio signal. For overlap prediction this sequence is fed into a neural network with a single sigmoidal output node to obtain $\hat{y}(t)$, which represents an estimate of the the presence of speaker overlap in the audio signal for each time step t. Since a BLSTM is used in these experiments, $\hat{y}(t)$ depends on both past and future input, up to and including time step t in both directions (cf. Section 5.5.1):

$$\hat{y}(t) = g_f\big(\mathbf{x}(1), \ldots, \mathbf{x}(t)\big) + g_b\big(\mathbf{x}(T), \ldots, \mathbf{x}(t)\big)\,, \tag{8.2}$$

where g_f and g_b denote the functions computed by the forward and backward pass of the BLSTM, respectively. The labels for training the network are given by

$$y(t) = \begin{cases} 1, & \text{if } \mathbf{x}(t) \in \text{overlap} \\ 0, & \text{else} \end{cases} \tag{8.3}$$

i. e. the task is treated as a binary classification problem. Therefore, the output $\hat{y}(t)$ of the trained network can be used for predicting overlap at each time step t:

$$o(t;\theta) = \begin{cases} 1, & \text{if } \hat{y}(t) \geq \theta \\ 0, & \text{if } \hat{y}(t) < \theta \end{cases} \tag{8.4}$$

where $o(t;\theta)$ is the classification result based on the decision threshold θ, which can be varied to select a specific operating point, in order to achieve a desired trade-off between precision and recall and hence to optimize overall performance. In the experiments described in this chapter the activation function at the output node was chosen to be the sigmoid (5.4) and the threshold was set to its canonical value $\theta = 0.5$. Eventually, the overlap ratio for a speech segment or full utterance is given as

$$r_o(\theta) = \frac{\sum_{t=1}^{T} o(t;\theta)}{T}, \tag{8.5}$$

i. e. the ratio between the number of frames characterized by two or more speakers simultaneously talking over the number of total frames in the audio segment.

8.5 Experiments and Results

For the conflict detection task the ComParE Conflict Sub-Challenge [292] baseline results will serve as a reference, as they show very good performance on the binary classification and real-valued regression tasks. Results are reported for two scenarios: First, the classification task, in which each audio recording is classified to be either conflictual (*high* in Table 8.1) or non-conflictual (*low*). For this problem the evaluation measure was chosen to be the unweighted average recall (UAR) as defined by (4.4), since it accounts well for the class imbalance in the SC^2 corpus. Second, the regression task, in which the scores predicted by the networks are evaluated against the rater's reference score values in the range $[-10, +10]$. For this problem the Pearson correlation coefficient ρ (4.14), denoted by CC in [292], is used as the evaluation criterion. The Challenge's baseline results were computed adopting a linear kernel Support Vector Machine (SVM) trained using Sequential Minimal Optimization (SMO) [26]. The SVM complexity parameter $C \in \{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ which achieved the best UAR on the development set was chosen for the reference results and logistic models were fit to the SVM hyperplane distance based on the training set to obtain (pseudo) class posteriors. The best baseline results are depicted in Table 8.2.

Table 8.2: Challenge baseline results. C: Complexity parameter in SVM training (tuned on development set). devel: Result on development set by training on training set. test: Result on test set by training on the training and development sets.

Task	[%]	C	Devel	Test
Classification	UAR	0.1	79.1	80.8
Regression	ρ (CC)	0.001	81.6	82.6

8.5.1 Supra-Segmental Features

8.5.1.1 Experimental Setup

As a baseline experiment, the supra-segmental feature set I as described in 8.3 is used as a static, utterance-level input to a neural network. All feature vectors are normalized via z-score normalization as defined by (3.24), where the statistical moments are computed on the training set. A favorable side-effect of this normalization is that the resulting normalized features approximately follow a Gaussian distribution, which is beneficial for the unsupervised pre-training using Gaussian-Bernoulli Restricted Boltzmann Machines (cf. Chapter 5.8.1) examined in this section.

In a first step the normalized features are used to train a one-layer neural network, i. e. MLP, in order to investigate the effect of the activation function in the hidden layer units on the results. For this purpose, the commonly used sigmoidal activation function (5.4) is evaluated against the piece-wise linear ReLU function (5.10). Furthermore, both variants are trained with either randomly initialized or pre-trained weights. As explained in Section 5.8 pre-training can help to combat overfitting, which is often encountered for larger networks in the paralinguistics research field, due to the relatively small data sets available. In this experiment GBRBMs are used to pre-train the respective networks. After this initial experiment, several DNNs are trained varying the number of hidden layers, in order to investigate the impact of depth on the performance.

For all experiments, prior to training, all weights are initialized with the uniform Glorot initialization scheme (5.75), also those that undergo pre-training. All networks are trained on the training set using standard SGD with momentum as described in Section 5.7.2, using the cross entropy (CE) loss function (5.53) in the classification task and the mean squared error (4.13) for the regression task. Early stopping, determined on the development set, is adopted, i. e. as soon as the loss function starts to rise on the development set the training is stopped. Finally, it is found that dropout regularization helps when using the ReLU activation function in these experiments, while this is not the case for the sigmoid units. All hyperparameters

are tuned on the development set of the SC^2 corpus and the reported results reflect the optimal settings.

8.5.1.2 Results

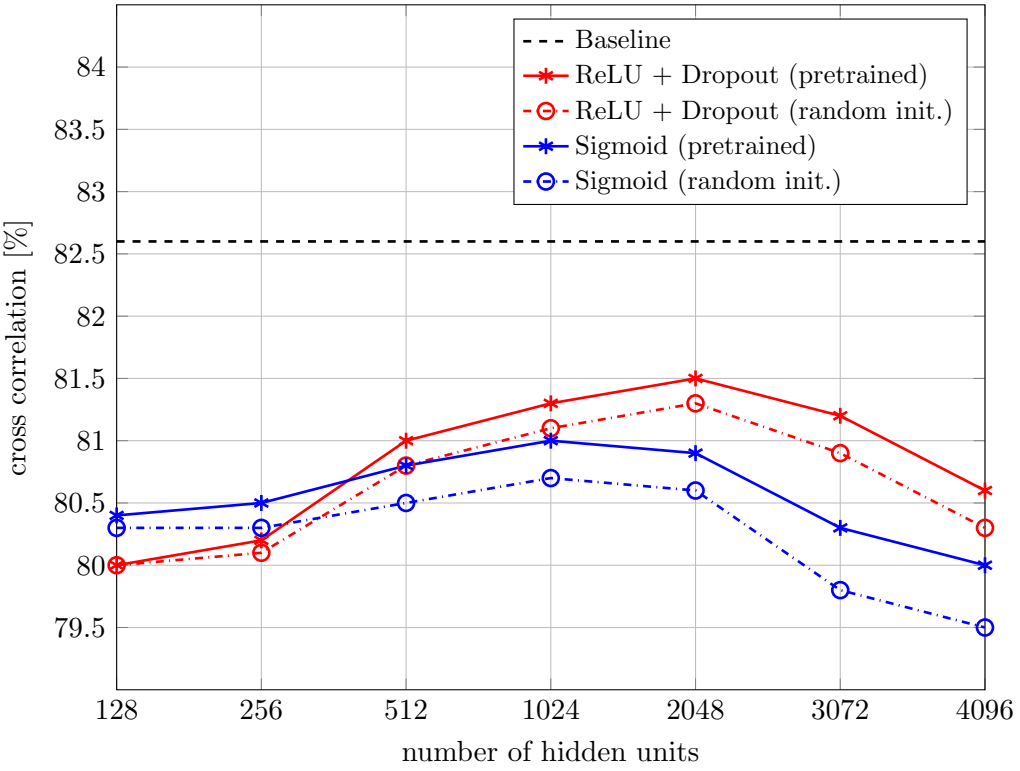


Figure 8.2: Regression task: test set results based on the baseline feature set I for a one-hidden layer MLP for varying hidden layer sizes. Shown are the graphs for networks trained with ReLUs + dropout vs. sigmoid hidden units and with vs. without pre-training the networks.

Figure 8.2 shows the best test set results obtained on the regression task for different sizes of the hidden layer in the MLP. For hidden layer sizes $L \leq 256$ the sigmoidal networks prove to be slightly better than the ReLU counterparts. On the other hand, for larger layer sizes $L \geq 512$ the ReLU networks' performance excels. In all configurations pre-training improves the randomly initialized networks by a small, but consistent gain. Interestingly, in the experiments the sigmoid networks seem to profit more from pre-training than the ReLU networks, which is probably due to the regularization effect of dropout, only used with the ReLU activation functions. Furthermore, pre-training has a bigger impact on larger hidden layers, presumably because larger networks with their higher number of parameters benefit more from

Table 8.3: Results on the classification and regression tasks for pre-trained ReLU networks trained on feature set I for different model topologies. Shown are the best results obtained on the development set (devel) and on the test set (test). Values represent UAR for the classification task and Pearson's ρ for the regression task, denoted as percentages.

Task	Classification		Regression	
Data sub-set	Devel	Test	Devel	Test
ComParE baseline [292]	79.1	80.8	81.6	82.6
MLP (6373-2048-1)	78.3	79.8	80.9	81.5
DNN (6373-1024-1024-1)	**79.7**	**80.9**	**81.5**	**82.1**
DNN (6373-1024-1024-1024-1)	78.9	80.2	81.1	81.8

its regularization effect than smaller networks.

Nonetheless, even though the performance of the best MLP reaches $\rho_{mlp} = 81.5\%$, indicating the usefulness of simple MLP networks, it still remains below the baseline system's performance of $\rho_{baseline} = 82.6\%$. For this reason the depth of the ReLU networks is increased by adding additional hidden layers and examining the impact of the number of hidden layers on the performance. Pre-training using GBRBM is applied as before and a wide range of DNN model topologies are trained. The best results are reported in Table 8.3 for both the classification and the regression task. Again the values denote UAR for the former and Pearson's ρ for the latter, all denoted as percentages for easy comparison. As can be seen, for the supra-segmental, utterance-level features set the best results are obtained with a 2-layer DNN with 1024 hidden units per layer. For the classification task the neural network approach shows slightly better performance than the ComParE Conflict Sub-Challenge baseline results, while for the regression task the results are slightly worse.

8.5.2 Overlap Ratio

8.5.2.1 Experimental Setup

As alluded to above, the frequency and duration of overlap in speech provides valuable information for predicting the level of conflict in discourse. The SC^2 corpus provides manually labeled meta-data containing speaker-turn information as well as annotations of overlapping speech segments. From this information the relative percentage of overlap with respect to the length of the audio recording is computed for each utterance. Since it is based on the reference annotation this value is termed *oracle overlap ratio*. Note that this feature is a scalar, supra-segmental, utterance-level feature. In order to assess the predictive power of the oracle overlap ratio, it is used

as a sole feature to feed-forward neural networks of varying depth, i. e. 1–3 hidden layers, and width, changing the number of hidden ReLU units from 2 to 128. As in the previous experiments the neural networks are trained on the training set using SGD, aborting training once the performance on the development set start to saturate.

In a second experiment the oracle overlap ratio is added to feature set I, in order to assess whether it contains complementary information w. r. t. the baseline features. For this large feature set the number of hidden units per layer is varied from 64 to 4096 and again, the number of layers change from 1–3, as before.

In realistic scenarios it is of course important to extract the overlap ratio automatically from speech. To this end, a BLSTM classifier is trained on the the oracle overlap ratio to output overlap ratio estimates, as described in Section 8.4. As a recurrent network architecture, the BLSTM requires frame-level features and therefore the 141-dimensional feature set II is used to train the network. Since both the input as well as the (single) output are of small dimension, a single-layer BLSTM network is sufficient to accurately predict overlap vs. non-overlap. Informal experiments suggest that a single-layer network consisting of 50 BLSTM memory cells is a good choice. The initial weights are initialized with values drawn from a uniform distribution in the range $[-0.1, +0.1]$. In order to increase the robustness of the network predictions, the input features are distorted with additive noise sampled from a zero-mean Gaussian distribution with $\sigma = 0.1$, which acts as a form of regularization and enhances generalization. Each network is trained via the Backpropagation Through Time (BPTT) algorithm (cf. Section 5.7.1) with a learning rate of 10^{-5} and a standard momentum of 0.9, until the loss function does not show any improvement on the development set for 10 epochs. For the classification task cross-entropy is used as a loss function, while for the regression task the MSE loss is used. After stopping the training the model which shows best performance on the development set is used for inference. From the frame-level predictions of overlap being present or absent the utterance-level overlap ratio is computed as defined in (8.5) for the full audio clip. The predicted overlap ratio is then used either as a single input feature or as a complementary feature to feature set I, analogous to the remarks for the oracle overlap ratio.

8.5.2.2 Results

Table 8.4 shows the best results obtained for the oracle overlap ratio as a single feature (top) and as a complementary feature to feature set I (bottom), for different numbers of hidden layers. Interestingly, even when used as a single input feature the oracle overlap ratio yields very good performance on both classification and regression tasks. A network with two hidden layers with 32 nodes each proved to be optimal, with smaller and larger networks being slightly less effective. As a

Table 8.4: Results for the **oracle** overlap ratio as a single feature (top) and as a complementary feature to feature set I (bottom) for different numbers of hidden layers on the classification and the regression task. Shown are the best results obtained on the development set (devel) and on the test set (test). The values reported denote UAR for the classification task and CC for the regression task, both as percentages.

Feature(s)	Task	Classification		Regression	
	Data sub-set	Devel	Test	Devel	Test
Overlap ratio	MLP (1-32-1)	75.5	76.7	78.6	79.1
	DNN (1-32-32-1)	**76.4**	**77.2**	**79.2**	**79.6**
	DNN (1-32-32-32-1)	75.4	76.7	78.9	79.2
Overlap ratio + Feature set I	MLP (6374-2048-1)	80.6	81.8	81.3	81.8
	DNN (6374-1024-1024-1)	**81.4**	**82.5**	**81.9**	**82.5**
	DNN (6374-1024-1024-1024-1)	80.5	81.9	81.3	81.7

complementary feature to feature set I results strongly improve to surpass the results from the previous section.

The results for the experiments based on the predicted overlap ratio are depicted in Table 8.5. Again, a feed-forward network with two hidden layers with 32 nodes each gives best results for the predicted overlap ratio as a single feature. Interestingly, the performance of the predicted overlap ratio proves to be significantly better than the oracle overlap ratio ($\alpha < 0.05$, according to a z-test for classification and a t-statistic for regression, respectively), increasing the UAR on the test set of the classification task from 77.2% to 82.9%, and improving the results of the test set on the regression task from 79.6% to 82.7%. As in the case for the oracle overlap ratio, adding the predicted overlap ratio to the feature set I improves the results further, yielding optimal performance for a feed-forward DNN with two hidden layers with 1024 nodes each, as before. Once more, the predicted overlap ratio seems to be advantageous over the oracle overlap ratio. The reason for this behavior remains unclear and merits further investigation.

8.5.3 Conversational-Prosodic Features

8.5.3.1 Experimental Setup

Inspired by the improvements obtained by leveraging the overlap ratio, this feature is also added to feature set III, which is a mixture of conversational and prosodic features and which was successfully used by Kim and his colleagues [182]. In order to facilitate the experiments the speaker-turn features are computed from the manual

Table 8.5: Results for the **predicted** overlap ratio as a single feature (top) and as a complementary feature to feature set I (bottom) for different numbers of hidden layers on the classification and the regression task. Shown are the best results obtained on the development set (devel) and on the test set (test). The values reported denote UAR for the classification task and CC for the regression task, both as percentages.

Feature(s)	task	Classification		Regression	
	Data sub-set	Devel	Test	Devel	Test
Overlap ratio	MLP (1-32-1)	80.5	82.3	81.3	82.1
	DNN (1-32-32-1)	**80.8**	**82.9**	**81.9**	**82.7**
	DNN (1-32-32-32-1)	80.5	82.5	81.2	81.9
Overlap ratio + Feature set I	MLP (6374-2048-1)	82.0	83.2	82.0	82.6
	DNN (6374-1024-1024-1)	**82.3**	**83.7**	**82.5**	**83.2**
	DNN (6374-1024-1024-1024-1)	82.0	83.1	82.1	82.6

annotations provided by the data set. However, as it gives better results, the predicted overlap ratio is used instead of the oracle overlap ratio, as described in Section 8.5.2. As in the preceding experiments, the neural network topologies are chosen to be feed-forward DNNs with 1–3 hidden layers, each composed of a number of ReLU hidden units, where the optimal number are experimentally identified on the development set. As before the networks are trained via SGD with momentum and dropout, using early stopping. All parameters and loss functions are as described above.

8.5.3.2 Results

The results for the optimal network configurations are shown in Table 8.6. First, the results show that the conversational-prosodic feature set III further improves upon the previously optimal results, when using a DNN with two hidden layers with 512 ReLU nodes each, both for the classification and the regression task. As in the above experiments, two hidden layers give better performance than one or three hidden layers. However, the number of hidden nodes per layer needs to be smaller than before, presumably since the number of features in feature set III is much smaller than the number of features in feature set I.

On the classification task the best network achieves a test set UAR = 84.3%, which exceeds the baseline result by 3.5% and the best result in the Conflict Sub-Challenge reported by Räsäsen et al. [265] by 0.4% absolute. On the regression task the improvements are slightly smaller, yet still raising the benchmark of the Challenge correlation coefficient from 82.6% to 83.8% on the test set.

Table 8.6: Results for feature set III varying the number of hidden layers on the classification and regression task. Shown are the best results obtained on the development set (devel) and on the test set (test). The percentages reported denote UAR for the classification task and CC for the regression task. For comparison the baseline results and the highest published competition results of the Conflict Sub-Challenge are shown as well.

Task	Classification		Regression	
Data sub-set	Devel	Test	Devel	Test
MLP (38-512-1)	82.5	83.8	82.2	83.2
DNN (38-512-512-1)	**83.1**	**84.3**	**83.0**	**83.8**
DNN (38-512-512-512-1)	82.6	84.0	82.7	83.4
Challenge baseline [291]	79.1	80.8	81.6	82.6
Räsänen et al. [265]	—	83.9	—	—
Grèzes et al. [117]	—	83.1	—	—

8.6 Conclusions

In this chapter an approach was presented how to successfully apply feed-forward DNNs to the problem of automatically detecting conflict in spontaneous, multi-party conversations. The experiments were performed on the SSPNet Conflict Corpus, which was also used in the Conflict Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge. Different utterance-level feature sets were evaluated on two separate tasks, namely classification and regression. As a first result, it was proven that replacing the traditionally used sigmoid hidden units with ReLU units and pre-training the networks using a variant of the RBM algorithm – combined with dropout as an advanced regularization method – improves performance and yields a baseline performance on-par with the challenge baseline performance. Next, it was shown that the oracle overlap ratio, i. e. the ratio of overlapping speech to non-overlapping speech obtained from manual segmentation, by itself as well as a complementary feature to the baseline feature set improves upon the baseline results.

In order to make the use of overlap prediction feasible for real-life applications, a new method was proposed for predicting the ratio of overlapping speech using a BLSTM network, which operates on a frame-level basis, and which was shown to be highly effective. In the light of the research objectives of this thesis, it is interesting to observe that the performance gain brought about by the predicted overlap ratio excels that of the oracle overlap ratio.

Finally, it was demonstrated that combining this predicted overlap ratio feature to a previously suggested conversational-prosodic feature set and using a moderately sized feed-forward DNN with two hidden layers and 512 ReLU units each, outperforms the Conflict Sub-Challenge baseline and the best challenge contributions on both the classification task and the regression task. For the classification task the model achieves a UAR = 84.3% on the test set, which improves the baseline result by 3.5%, and the best result reported for the Conflict Sub-Challenge by Räsäsen et al. [265] by 0.4%. For the regression task the relative improvements are smaller, still raising the benchmark of the Challenge correlation coefficient of 82.6% to 83.8%, measured on the test set. It should further be noted that while the baseline results for the classification and regression tasks were obtained by SVM with different complexity parameters C, the topologies of the adopted DNNs in the experiments were the same, although the respective networks were trained separately for each task (and with different objective functions).

Future work should focus on further automating the feature extraction process by automatic speaker turn detection and diarization, since the used features partially rely on manual speaker-turn annotations provided by the data base. Further, directly learning an optimal feature set in an end-to-end fashion could turn out be a very promising research directive.

Chapter 9

End-to-End Emotion Recognition

When dealing with people, remember
you are not dealing with creatures of logic,
but with creatures of emotion.

Dale Carnegie

With the ground-breaking improvements achieved with deep neural networks in the last decade in research areas such as object, speech, or speaker recognition, the field of computational paralinguistics has made the paradigm shift to deep learning as well. In particular, emotion and affect recognition is an area of high interest, because besides academic curiosity it is also highly relevant for commercial applications. In this vein, a large number of new neural network architectures have been proposed, such as autoencoder networks, convolutional neural networks (CNN), or memory enhanced neural network models such as Long Short-Term Memory (LSTM) models [288], and have shown their favorable property to model inherent structure contained in the speech signal [147]. Some recent studies have started to look into end-to-end optimization utilizing as little human a-priori knowledge as possible [113]. Yet, the majority of these works make use of commonly hand-engineered input features, such as Mel-Frequency Cepstral Coefficients (MFCC), Perceptual Linear Prediction (PLP) coefficients, and supra-segmental features (used in the series of ComParE [292] and AVEC challenges [271]), all of which build upon knowledge gained in decades of auditory research and have shown to be robust for many speech domains. A new trend in the general machine learning community has emerged attempting to learn intermediate representations of the underlying signal directly from the speech waveform. The motivation behind this *end-to-end* learning is that without any specific manual constraints a model can learn interesting and relevant characteristics for the task at hand in a more specific, tailored manner, which could lead to improved final performance. This study, which was published earlier by the

author and his colleagues [346], proposes a new model and approach to effectively and successfully train an emotion recognition system in such an end-to-end fashion.

Related work

One of the first investigations that studied learning higher-level representations directly from the raw speech waveform was carried out by Jaitly and Hinton [170]. They used Restricted Boltzmann Machines in the context of a phoneme recognition system and learned filters which exhibited the bandpass behavior of the inner ear in the human auditory system. On the TIMIT phoneme recognition task they achieved state-of-the-art results at that time. Bhargava and Rose [25] investigated acoustic models for ASR using deep neural networks directly taking windowed speech waveforms as input. They found that the DNNs automatically acquired internal representations similar to Mel-filter banks with any predefined information and obtained results which were only marginally worse than MFCC features evaluated on the same architectures. Sainath et al. proposed a stacked architecture of CNN layers, LSTM layers, and fully connected layers, which they called CLDNN, which they trained on 2,000 hours of speech data and matched the performance of a large-vocabulary speech recognition system based on log-Mel filterbank features [280, 281]. They observed that the time convolution layer helps reducing temporal variations, the frequency convolution layer for preserving locality and reducing frequency variations, whereas the LSTM layers assist in temporal modeling of the acoustic signal. Palaz et al. [240, 241] attempted to estimate phoneme class conditional probabilities employing CNNs directly trained on the speech waveforms and found that the features which were learned between the first two convolution layers of the CNN tend to model the spectral envelope of sub-segmental speech signals, leading to an improved noise-robustness with respect to MFCC features. In the context of music information retrieval [72] and polyphonic music transcription [312] deep end-to-end learning was successfully applied. In the area of computational paralinguistics, some studies have been conducted utilizing CNNs in the context of feature learning, for example by Milde and Biemann [225] or Mao et al. [216]. Yet, all investigations rest on low-dimensional Mel filterbank features and lack the full end-to-end training aspect, which is investigated and presented in this study.

9.1 Speech Database

The *REmote COLlaborative and Affective interactions Corpus (RECOLA)* [269] is a multimodal (audio, video, ECG, and EDA) corpus of spontaneous collaborative and affective interactions in French. The recordings were made from 46 participants (27 females, 19 males, mean age: 22 years, standard deviation: 3 years) during a video conference while completing a task requiring collaboration, where the mood of participants was manipulated and balanced in dyads. The audio signals used in this

Table 9.1: 26 eGeMAPS low-level descriptors (LLD); [1]computed on voiced and unvoiced frames, respectively; [2]computed on voiced, unvoiced and all frames, respectively. [271]

1 Energy-related LLD	**Group**
Sum of Auditory Spectrum (Loudness)	Prosodic
9 Spectral/Cepstral LLD	**Group**
α Ratio (50-1000 Hz, 1-5 kHz)[1]	Spectral
Spectral Slope (0-500 Hz, 0.5-1.5 kHz)[1]	Spectral
Hammarberg Index[1]	Spectral
MFCC 1–4[2]	Cepstral
Spectral Flux[2]	Spectral
16 Voicing-related LLD	**Group**
F_0 (Linear, Log)	Prosodic
Formants 1, 2, 3 (Frequency, Bandwidth, Amplitude)	Voice Quality
Harmonic Ratios H1-H2, H1-A3	Voice Quality
Logarithmic HNR, Jitter (Local), Shimmer (Local)	Voice Quality

study were recorded with unidirectional headset microphones (AKG C520L) at a sampling rate of 44.1 kHz (16 bit) and resampled to 16 kHz. For each audio recording only the first 5 minutes were annotated in continuous-time and continuous-valued scale for arousal and valence. In order to guarantee consistency in the annotations, all sequences were annotated by all 6 French-speaking annotators. Furthermore, zero-mean normalization was applied to the annotations, to remove an eventual bias in the annotation values, followed by synchronization, in order to tackle the problem of having different time reactions between the annotators. The ground truth of a sequence was then estimated by mean filtering the annotations provided by all annotators. The analysis of the annotations shows a good inter-annotator agreement rate for the affective dimensions [269]. In this study all annotated audio recordings are used and are split into three partitions, training (16 subjects), validation (15 subjects), and test (15 subjects), by stratifying the age and gender of the speakers.

9.2 Acoustic Feature Sets

Two different feature sets are used in this study to compare the proposed signal-based approach against. First, the *extended Geneva Minimalistic Acoustic Parameter Set (eGeMAPS)* [86] is used. The acoustic low-level descriptors (LLD) cover spectral, cepstral, prosodic, and voice quality information, as shown in Table 9.1, and are

extracted with the openSMILE toolkit [85]. All LLDs are smoothed over time with a symmetric moving average filter of length 3, and the arithmetic mean and coefficient of variation (standard deviation normalized by the arithmetic mean) are applied as functionals. In addition, the following functionals are applied to loudness and pitch: percentiles 20, 50 and 80, the range of percentiles 20 – 80 and the mean and standard deviation of the slope of rising/falling signal parts. Furthermore, the average RMS energy and 6 temporal features are included, which are: the rate of loudness peaks per second, mean length and standard deviation of continuous voiced and unvoiced segments and the rate of voiced segments per second, approximating the pseudo syllable rate [271]. Note that some functionals are applied only in voiced or unvoiced regions, depending on the type of the LLD (cf. Table 9.1). The interested reader is referred to [83] for the exact details on how the features are computed. Altogether, the eGeMAPS feature set contains 88 features per time frame. Each frame is computed on overlapping fixed length audio segments of 3 s length with a frame shift of 40 ms.

The second feature set investigated this study is based on the ComParE LLDs listed in Table 3.1. On these 65 LLDs the functionals *max*, *min*, *mean*, *range*, and *standard-deviation* [270] are applied with the same window length and frame shift as for the eGeMAPS feature set.

9.3 Experiments and Results

9.3.1 Model Design

As expounded in Chapter 3 feature engineering has dominated basically all fields of pattern recognition for decades, i. e. features used in classification, regression, or detection tasks were carefully designed incorporating human knowledge, drawn from a long research history in many research areas. In speech recognition, for example, the still most common features are the Mel-frequency cepstral coefficients (MFCC), which loosely replicate some central aspects of the human auditory system. In most computational paralinguistic tasks, features are first extracted and then passed to a machine learning algorithm as well. In contrast, this study aims at learning the feature extractor jointly with the subsequent predictor (regressor) in a true end-to-end fashion.

To make training feasible, the raw input speech waveforms are first conditioned to have zero mean and unit variance (z-score normalization, cf. Equation (3.24)). This accounts for any bias and loudness differences in the recordings. Thereafter, the waveforms are segmented into sequences each 6 s long. Since the sampling rate $f_s = 16$ kHz, each segment contains 96,000 samples grouped into one vector.

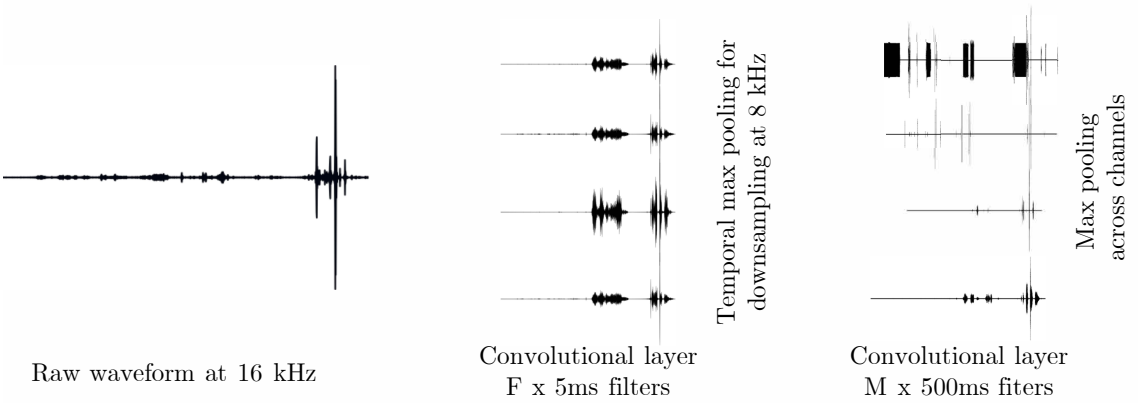


Figure 9.1: Illustration of the proposed one-dimensional convolutional neural network (1-D CNN) operating on the raw speech waveform.

Conventional approaches to feature extraction often include some form filtering via finite impulse response (FIR) or infinite impulse response (IIR) filters to achieve a time-frequency decomposition, which has shown to mitigate the negative effects of background noise [152]. Inspired by research of the auditory system of animals and humans, more complicated, engineered kernels, such as gammatone filters [287] or gammachirp filters [259] have been proposed. In this study, a one-dimensional (1-D) temporal CNN, as described in Section 5.4, is used to extract an intermediate representation directly from the segments of (normalized) raw speech. Based on preliminary experiments $K = 40$ FIR filters are used on 5 ms windows, in order to extract localized, spectral information from the high sampling rate acoustic signal. As will be shown in the results, the learned filters resemble auditory filter banks and hence are suitable to perform a time-frequency decomposition, reminiscent of the DFT conventionally used in feature extraction. The output of this first temporal convolution layer is passed through a half-wave rectifier, $z' = max(0, z)$, mimicking the cochlear transduction step in the human ear, and subsequently downsampled to 8 kHz by applying a max pooling operation of size 2. In order to extract more long-term characteristics of the underlying speech signal (e. g. roughness), another 1-D temporal convolution layer using $M = 40$ FIR filters operating on windows of length 500 ms is applied. This layer again is followed by another max pooling layer; however, this time the pooling is applied across channels, with a pool size of 20. This operation reduces the dimensionality of the representation while preserving the necessary statistics of the underlying speech signal. This processing chain is exemplified in Figure 9.1. The sequences of this intermediate representation, which are approximately 6 s long (depending on padding etc.), are segmented into smaller sub-sequences to match the time resolution (40 ms) of the gold standard annotation. Each sub-sequence is then fed into a recurrent BLSTM network composed of two bidirectional LSTM layers with 128 memory blocks each.

9.3.2 Loss Function

Ringeval et al. [270, 271] proposed to utilize the *concordance correlation coefficient* (CCC), ρ_c (cf. Equation (4.15)), to evaluate the agreement level between the predictions of the network and the gold-standard derived from the annotations. Yet, for training the networks the mean-squared error was used as a loss function. Presumably, this leads to sub-optimal results. Hence, in this study, the neural networks are directly optimized by integrating the CCC into the loss function $\mathcal{L}_c$ as follows:

$$\mathcal{L}_c = 1 - \rho_c = 1 - \frac{2\sigma_{xy}^2}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2} \tag{9.1}$$

$$= 1 - 2\sigma_{xy}^2 \psi^{-1} \tag{9.2}$$

where $\psi = \sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2$ is used as a shorthand notation, $\mu_x = \mathbb{E}(x)$ and $\mu_y = \mathbb{E}(y)$ are the expectations over the inputs x and gold standard variables y, σ_x^2 and σ_y^2 the respective variances, and σ_{xy}^2 the covariance of x and y. In order to maximize the CCC, one preferably minimizes $\mathcal{L}_c$ by backpropagating the gradient of the last layer weights with respect to $\mathcal{L}_c$ as

$$\frac{\partial \mathcal{L}_c}{\partial x} \propto 2\frac{\sigma_{xy}^2(x - \mu_y)}{\psi} + \frac{\mu_y - y}{\psi} \tag{9.3}$$

9.3.3 Experimental Setup

To compare the results obtained with the proposed approach a standard baseline machine learning algorithm is evaluated. A Support Vector Regression (SVR) model with a linear kernel is trained on the RECOLA dataset using the LIBSVM library [48]. The linear kernel provides superior results w. r. t. the polynomial and radial basis function (RBF) kernels and hence is chosen to provide a strong baseline. The complexity parameter is optimized via a logarithmic grid search in the range $C \in [10e^{-6}, 1]$.

For a second baseline result, a BLSTM network composed of three hidden layers with 64 units per layer is used, adhering to the architecture proposed by [270, 271]. Noise sampled from a Gaussian distribution according to $\mathcal{N}(0, 0.1)$ is added to the input features as a form of regularization. The network is trained via SGD with a batch size of 5 sequences, until the performance on the validation set does not improved for 5 consecutive epochs (early stopping). Furthermore, the learning rate λ is optimized on the validation set for emotional dimension (arousal, valence) and objective function (MSE, CCC) separately, where ρ_c from (4.15) is used to measure the performance during evaluations. In accordance with approach defined in the AV$^+$EC 2015 challenge [271], ρ_c is computed on the gold standard and the predictions concatenated over all recordings.

Both parts of the model (CNN and BLSTM) are trained jointly using the loss function from (9.1), on mini-batches of 50 samples, and using the ADAM optimizer, with an initial learning rate of $2 \cdot 10^{-3}$. In addition, dropout is used with $p = 0.5$ for all non-recurrent layers. This strong regularization is important to prevent overfitting, given the large number of parameters (≈ 1.5 million). Again, early stopping is adopted, i. e. training is stopped once the performance on the validation set does not increase for 5 consecutive epochs.

As alluded to in Chapter 2, continuous-valued problems, especially those involving subjective ratings, such as emotion recognition, pose some particular challenges, due to the variation in the annotation process or caused by the delay between the physical evidence leading to the perception of an emotion and the actual perception itself. In order to partially compensate for these influences, a number of postprocessing steps are examined on the validation set predictions, for both the baseline and the proposed methods:

1. *Median filtering* is applied using a window size between 0.4s and 20s, effectively smoothing the predictions and hence improving the quality of noisy predictions [271]. This removes potential outliers and also takes into account the continuity of the annotation process.

2. *Centering*, by removing the bias between the gold standard annotation and the prediction [175], in order to account for any systematic offset. The reference mean is computed on the full training set.

3. *Scaling* follows the same argumentation as centering [175]. It either uses a min-max normalization, where the reference extrema are computed on the training set, or it uses the ratio between the standard deviation of the gold standard and that of the predictions as a scaling factor.

4. *Time-shifting* the labels (or predictions) usually turns out to be highly beneficial in continuously evaluated emotion tasks, since it compensates for the reaction lag of the evaluators [217]. In this study the predictions are shifted forwards in time with values ranging from 40 ms to 10 s.

If the respective post-processing step leads to an improvement of ρ_c on the validation set, the step is kept for all further evaluation (both validation and test), else it is discarded.

9.3.4 Results

The results are shown in Table 9.2 in terms of ρ_c. First of all, the prediction of arousal is much better than that for valence, for any loss function and any approach. This observation confirms previous findings that it is easier to predict arousal than

Table 9.2: Results (in terms of ρ_c) for different classifiers and loss functions on the RECOLA database for prediction of arousal and valence. The values denote performance on the test set, the values in parenthesis the performance on the validation set. a) models were optimized w. r. t. MSE. b) models were optimized w. r. t. CCC. [346]

Predictor	Features	Arousal	Valence
a) MSE loss			
SVR	eGeMAPS	.318 (.489)	.169 (.210)
SVR	ComParE	.366 (.491)	.180 (.178)
BLSTM	eGeMAPS	.300 (.404)	.192 (.187)
BLSTM	ComParE	.132 (.221)	.117 (.152)
End-to-end	Raw signal	.684 (.728)	.249 (.312)
b) CCC loss			
BLSTM	eGeMAPS	.316 (.445)	.195 (.190)
BLSTM	ComParE	.382 (.478)	.187 (.246)
End-to-end	Raw signal	**.686 (.741)**	**.261 (.325)**

valence from speech signals [270, 347]. Second, using the MSE loss, the SVR baseline is consistently better than the BLSTM system for arousal. Interestingly, the BLSTM performs much worse using the ComParE feature set than any other feature-system combination when trained on the MSE loss, for both arousal and valence. The reason for this is unclear. However, this artifact disappears when training on the CCC loss function. Overall, utilizing the CCC loss leads to significant improvements over the MSE loss for arousal and valence. Most importantly, however, the proposed end-to-end model outperforms the baseline systems by a large margin, irrespective of the chosen loss function. Best results are obtained when using the CCC loss for training the end-to-end model on the raw audio signals.

There is general agreement in the paralinguistic research community that certain acoustic and prosodic cues, such as the fundamental frequency F_0, pitch range, mean speech intensity, or loudness are most relevant for predicting the affective state of human speakers [13, 286]. It is thus to be expected that some of these characteristics are captured by the proposed end-to-end model, which is directly trained on the audio waveforms. Yet, relatively little research has been conducted in this direction of visualization. In order to gain a better understanding of what the proposed model learns, the statistics of the gate activations, i. e. the values of the hidden-to-output connections, of several different cells in the BLSTM network are examined and compared to a selection of acoustic-prosodic features known to affect

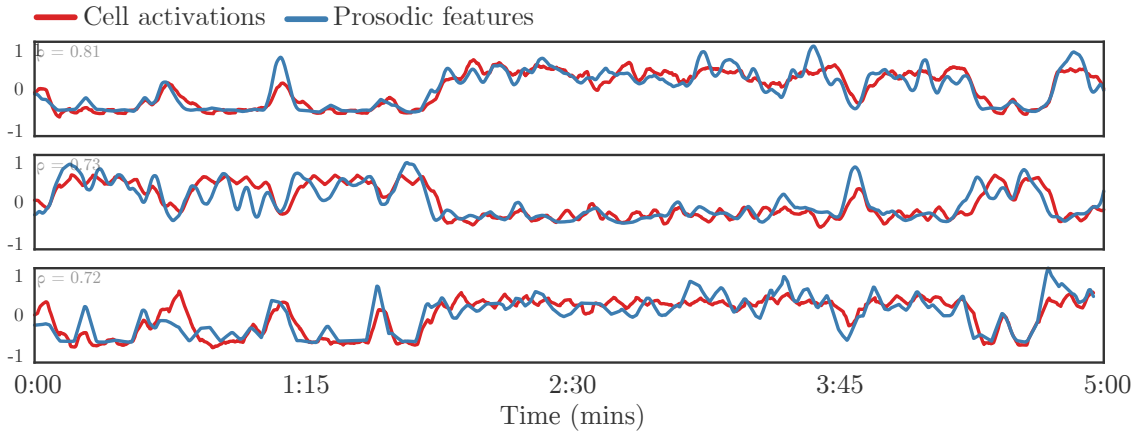


Figure 9.2: Visualization of three different gate activations w. r. t. different prosodic features known to affect arousal. From top to bottom: range of RMS energy ($\rho = 0.81$), loudness ($\rho = 0.73$), and mean of fundamental frequency F_0 ($\rho = 0.72$).

arousal. Figure 9.2 presents a visualization of three selected gate activations with respect to RMS energy (Pearson correlation $\rho = 0.81$), loudness ($\rho = 0.73$), and the mean of the fundamental frequency F_0 ($\rho = 0.72$). It shows that certain cells of the model indeed learn to respond to particular characteristics in the speech signal in a way to closely resemble certain acoustic-prosodic features.

9.4 Conclusions

In this chapter a fully end-to-end approach to the prediction of spontaneous emotion in human speech was proposed. To this end, a convolutional, recurrent neural network model (CNN-BLSTM) was introduced, which automatically learns intermediate representations directly from the unprocessed, raw speech signal. This intermediate representation was then utilized to predict the level of arousal and valence in a time-continuous manner on the audio part of the REmote COLlaborative and Affective interactions Corpus (RECOLA), a multimodal corpus of spontaneous collaborative and affective interactions. It was shown that the proposed method achieves significantly higher performance compared to two baseline systems, a Support Vector Regressor and a deep BLSTM neural network, evaluated on two common paralinguistic frame-level feature sets, the eGeMAPS and the ComParE low-level descriptors. This shows that learning the intermediate representations directly from the speech signal can be a highly effective method to adapt these representations to the task at hand. Furthermore, this study introduced the direct optimization of the proposed system on the concordance correlation coefficient (CCC) instead of the commonly adopted MSE. The former is often used to evaluate the agreement between

the reference gold standard and the model predictions, but not for optimization of the network parameters. This study demonstrates that is highly beneficial to do so, leading to further gains in performance. As a final contribution, internal activations of the recurrent output BLSTM network were compared to acoustic-prosodic features, which are known to affect arousal and which are commonly used in computational paralinguistic tasks. It was found that certain cell activations are highly correlated with those prosodic features.

Part IV

Concluding Remarks

Chapter 10

Concluding Remarks

Science never solves a problem without creating ten more.

George Bernard Shaw

10.1 Summary

One of the main pillars of computational paralinguistics is the study of the automated analysis of non-verbal communication. To robustly detect, predict, and classify the events which are the carriers of the manifold human paralinguistic characteristics is of paramount importance in order to improve human-machine interaction. The research presented in this thesis is an attempt to advance the knowledge about how to leverage the new research area of deep learning in the realm of computational paralinguistics. Given the breadth of the latter, with its many different paralinguistic tasks, and the rapidly evolving and intensifying research of the former, this work can only exemplarily address a selection of previously unexplored avenues, laying the base for further research. This section summarizes the main achievements of this thesis in this respect.

After a an introduction to the general motivation and the contributions of this work in Chapter 1, an overview over some general aspects of computational paralinguistics is given in Chapter 2, followed by a description of commonly employed features in this field in Chapter 3. Chapter 4 then expounds the mathematical background necessary for the evaluation of the proposed approaches and is succeeded by an extensive account of the basic principles, architectures, and training aspects of deep neural networks. Based on these principles Part III describes the findings and results obtained by applying deep neural networks to a selection of four different tasks of computational paralinguistics:

First, in Chapter 6 feed-forward neural networks, the cornerstone of deep neural networks, are applied to the classification of likability of human voices, a difficult paralinguistic task given its highly subjective nature. This contribution is a result from the participation in the Likability Sub-Challenge of the Interspeech 2012 Speaker Trait Challenge [291]. It is based on the challenge's supra-segmental, utterance-level, feature set and is demonstrated to be superior to the challenge baseline results, achieving state-of-the-art results. Yet, it is shown that on this task a wide, but shallow one-layer neural network excels all deeper network architectures, if pre-trained via a regularized Gaussian-Bernoulli Restricted Boltzmann Machine approach in an unsupervised manner.

Second, for the task of detecting social signals ("laughter" and "filler") from human speech a number of effective neural network architectures and methods are proposed in Chapter 7. Evaluated on the dataset of the Social Signals Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge [292], unsupervised pretraining via a stacked autoencoder network is shown to yield significant gains in performance. Furthermore, a novel, deep posterior smoothing method is proposed and shown to be highly effective. This approach is extended to recurrent and hierarchical networks leading to state-of-the-art results. These findings are verified to the recent SEWA database, presenting the first mono-lingual and cross-lingual results and a first comparison of resource-efficient recurrent models on this database.

Third, in Chapter 8 feed-forward neural networks are successfully applied to the problem of automatically detecting conflict in spontaneous, multi-party conversations. Based on the SSPNet Conflict Corpus, used in the Conflict Sub-Challenge of the Interspeech 2013 Computational Paralinguistics Challenge [292], several different supra-segmental feature sets are investigated on classification and regression tasks and some architectural and training improvements are proposed. Furthermore, an enhanced method for predicting the ratio of overlapping speech using a BLSTM network is introduced and it is shown that this feature, combined with a conversational-prosodic feature set, leads to state-of-the-art results significantly outperforming the challenge baseline.

Finally, a novel, fully end-to-end approach to the prediction of spontaneous emotion in human speech is presented in Chapter 9. In this approach a convolutional, recurrent neural network model (CNN-BLSTM) is directly trained on the raw speech signal and learns an intermediate feature representation which leads to significant improvements in the prediction of the level of arousal and valence from speech in a time-continuous fashion. Evaluated on the recent RECOLA database, the proposed method surpasses two strong baseline systems, each one trained on two common paralinguistic frame-level feature sets. An additional gain in performance is achieved by optimizing the proposed system on the concordance correlation coefficient instead

of the commonly adopted mean-squared error. As a final contribution, it is shown that certain internal, recurrent cell activations from the learned network are highly correlated with particular prosodic features known to affect arousal and commonly used in computational paralinguistic tasks.

In summary, all experiments presented in this thesis demonstrate that deep learning approaches have the potential to be highly effective in a range of paralinguistic problems. A number of improvements and novel methods have be introduced, and in order to facilitate comparison with other approaches in the literature, they are evaluated on commonly accessible databases, most of which provided to the research community under the umbrella of the Interspeech paralinguistic challenges in recent years.

10.2 Outlook

As the introductory quote in this chapter states, any progress made in a scientific area opens new avenues and creates new ideas for further research. And as alluded to earlier the many different facets of computational paralinguistics and the rapidly evolving field of deep learning offers multifarious opportunities to continue to explore open questions in both fields of research. Based on the work and the results presented in this thesis, a short list of potentially beneficial topics is briefly addressed in order to guide further research.

Novel deep learning architectures It has been alluded to several times in this thesis that the field of deep learning in general has increased in dynamics during recent years, which has led to a large body of literature proposing and describing new deep learning architectures. Sometimes these proposals rely on novel combinations of established network topologies, sometimes completely new network types are proposed. Many of the novel ideas originate in the research areas of object, speech, or speaker recognition, and hence many of these approaches need yet to be explored in computational paralinguistics. An example of a highly interesting topic is given by the recent (self-)attention mechanism [370]. This idea could be especially helpful for approaches which make predictions of paralinguistic events on a segment level, but where annotations are available only on a coarser time-resolution, e. g. on an utterance level. For this to work, all segment predictions must somehow be merged into one final prediction, which is to be compared against the gold standard. This is often performed by simple average-pooling, but self-attention implicitly learns the weights reflecting the importance of the respective segment contributions w. r. t. the final annotation label. Of course, there is no guarantee that a working approach in one particular field also means success in computational paralinguistics, given the constraints and peculiarities of this domain.

Tackling the data scarcity problem Due to the high annotation cost paralinguistic databases are usually fairly small compared to the large datasets found in areas such as speech recognition, where training data can amount up to thousands of hours. This scarcity of training data typically leads to overfitting with larger neural network models and can only partially be resolved by regularization methods, such as L_2-regularization or dropout. Another way to improve upon this situation is to better leverage the limited amount of training data by pre-training neural network models in an unsupervised fashion. As pointed out earlier, nowadays often there is an abundance of unlabeled data which can be exploited for this purpose. As has been shown in many experiments in this study, many approaches even benefit from unsupervised pre-training on the same training data used for supervised training, obviously leveraging additional information inherent in the data, which cannot be fully utilized by supervision. Either way, this combination of unsupervised and supervised training gives rise to the concept of *semi-supervised training* (cf. Chapter 5.2).

In addition, similar concepts like *Dynamic Active Learning* [380] or *Cooperative Learning* [130] were recently proposed for semi-supervised labeling, exploiting confidence-based information from a prediction network or from the evaluation of the interrater agreement to decide which audio recordings need to be annotated by human raters and if so, by how many. These approaches can dramatically reduce the effort and cost of annotation and thus allow for larger annotated datasets. Often in these techniques, a large part of the data can be automatically annotated by an existing predictor. This process can be continued iteratively in order to improve system performance and to continuously increase the amount of training data.

Towards holistic paralinguistics Instead of devising and building single-task systems for each paralinguistic problem separately, a different line of thought, called *holistic paralinguistics* [380] attempts to exploit the relations between certain tasks and build a single multi-task framework, potentially even for multiple modalities at once. This can, for example, be achieved by sharing parts of the model parameters across several different tasks. One advantage of this approach is that the aggregate of the composing datasets can be used to train all shared parameters, hence reducing overfitting and related negative effects. Beyond that, hidden synergies might be exploited by cross-domain transfer learning.

End-to-end learning The results of Chapter 9 confirm the potential of end-to-end training, already demonstrated in several different research areas [72, 113, 312, 319, 347], and prove that this approach is not merely able to match, but also to surpass previous state-of-the-art results. Intuitively, true end-to-end learning possesses the ability to closely adapt to a model's input, completely remediating the hand-crafted feature extraction step. Of course, by doing so one abandons the robustness of

proven features, which often are the result of decades of feature engineering and trial and error. Some of the current shortcomings of signal-based end-to-end approaches could be due to normalization issues leading to undesirable artifacts in the internal representations. Yet, large-scale, unsupervised training of the early steps of such models, which are to replace the conventional feature extraction, might lead to additional gains and remove the need for hand-crafted or even task-specific feature extraction.

Spiking neural networks Finally, a more forward-looking bet into the future concerns the fundamental structure of current neural network architectures: As mentioned in Chapter 5.2 most current phenomenological models follow the rate coding paradigm by modeling the firing rate of biological neurons. While this has shown to be effective and well matched to the typical Von-Neumann architectures and derivates (CPU, GPU, TPU), modeling time codes with *spiking neural networks* (SNN) might exhibit a number of advantages: First, neurons based on temporal codes - except for peripheral neurons shown to approximately operate on a rate code - are highly memory-efficient, requiring magnitudes less energy than current hardware [246]. Second, it has been shown that the precise time-relations across spike populations allow to encode significantly more information in an energy-efficient manner than rate coding [212, 267]. There has been some increase in interest in the application of SNNs recently [164, 365, 367, 375], yet much research needs to be conducted to make them competitive to current deep learning architectures. Furthermore, a paradigm shift to event-driven, neuromorphic hardware might be required to achieve this [329]: Research has shown that the communication mechanism must be taken into account in hardware design [307] in order to allow for efficient SNN implementations including regular and irregular spike events [209].

Acronyms

AE Autoencoder

ACC Accuracy

ACF Auto Correlation Function

AI Artificial Intelligence

AUC Area Under The Curve

ANN Artificial Neural Network

ARMA Auto-Regressive Moving Average

ASR Automatic Speech Recognition

AVEC Audio/Visual Emotion Challenge and Workshop

BLSTM Bi-directional Long Short-Term Memory

BN Batch Normalization

BPTT Backpropagation Through Time

CAE Contractive Autoencoder

CC Cross-Correlation

CCC Concordance Correlation Coefficient

CD Contrastive Divergence

CE Cross-Entropy

CNN Convolutional Neural Network

CP Computational Paralinguistics

CRF Conditional Random Field

CV Cross-Validation

DAG Directed Acyclic Graph

DBN Deep Belief Network

DCT Discrete Cosine Transform

DET Detection Error Tradeoff

DFT Discrete Fourier Transform

DNN Deep Neural Network

ECG Electrocardiogram

EDA Electrodermal Activity

EER Equal Error Rate

EWE Evaluator Weighted Estimator

FAR False Acceptance Rate, False Alarm Rate

FF Feed-Forward

FFT Fast Fourier Transform

FIR Finite Impulse Response

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

GAN Generative Adversarial Networks

GBRBM Gaussian-Bernoulli Restricted Boltzmann Machine

GMM Gaussian Mixture Model

HNR Harmonics-To-Noise Ratio

HMM Hidden Markov Model

ICA Independent Component Analysis

IIR Infinite Impulse Response

KLD Kullback-Leibler Divergence

LLD Low-Level Descriptor

LOSGO Leave One Speaker-Group Out

LOSO Leave One Speaker Out

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MFCC Mel-Frequency Cepstral Coefficients

MLP Multi-Layer Perceptron

mRMR Minimal-Redundancy-Maximal-Relevance

MSE Mean Squared Error

NAG Nesterov Accelerated Gradient

NLU Natural Language Understanding

NMF Non-Negative Matrix Factorization

NN Neural Network

PCA Principal Component Analysis

RBF Radial Basis Function

RBM Restricted Boltzmann Manchine

RF Random Forest

RL Reinforcement Learning

RMS Root-Mean Square

RMSE Root-Mean Square Error

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

SGD Stochastic Gradient Descent

SHS Sub-Harmonic Summation

SNN Spiking Neural Network

SOM Self-Organizing Network

SVD Singular Value Decomposition

SVM Support Vector Machine

SVR Support Vector Regression

TN True Negative

TNR True Negative Rate

TP True Positive

TPR True Positive Rate

TTS Text-To-Speech

TUM Technische Universität München

UAAUC Unweighted Average Area Under The Curve

UA Unweighted Accuracy

UAR Unweighted Average Recall

VA Virtual Agents

VAE Variational Autoencoder

ZCR Zero-Crossing Rate

List of Symbols

Paralinguistics

K	Number of raters
N	Number of instances
r_k	Reliability of rater k
σ_n	Standard deviation of annotations for instance n
$y_{n,EWE}$	EWE-smoothed annotation of instance n
$y_{n,k}$	Annotation of instance n by rater k
$\overline{y}_n$	Mean of all annotations of instance n over all raters

Acoustic Features

a	Minimum value of min-max normalization
a_0	Hamming window constant
A	Peak-to-peak amplitude
b	Maximum value of min-max normalization
$c(n)$	Cepstrum at time step n
$c_s(n)$	Cepstrum of the source signal at time step n
$c_f(n)$	Cepstrum of the filter at time step n
d	Filter bank index
D	Dimensionality of feature vectors
E_{RMS}	Root-mean square energy of audio signal
f	Frequency in Hz
F_0	Fundamental frequency

$\mathcal{F}\{x\}$	Fourier transform of x
$\mathcal{F}^{-1}\{X\}$	Inverse Fourier transform of X
$\mathscr{F}$	Functional
g_k	k-th feature extraction step/component
h	Predictor model
J_p	Delta jitter
J_{pp}	Local jitter
k	Frequency index
L	Liftering coefficient
m	Mel frequency
M	Number of Mel-frequency filters
n	Discrete time step index
N	Number of time samples = window size
p_v	Probability of voicing
S_{pp}	Shimmer
T_0	Fundamental period
$\overline{T}_0$	Average of fundamental period (over a short time-frame)
$w(n)$	Window function at time step n
$x(n)$	Audio signal at time step n
$\boldsymbol{x}_i$	i-th (input) feature vector
$\boldsymbol{x}'_i$	i-th normalized feature vector
$x^{(w)}(n)$	Amplitude of windowed audio signal at time step n
$X(k)$	Complex spectrum at frequency index k
$\boldsymbol{X}_{aud}$	Auditory spectrum
$\boldsymbol{X}_{mel}$	Mel-frequency spectrum
$\boldsymbol{X}_{mfcc}$	Mel-frequency cepstral coefficients
$\boldsymbol{X}'_{mfcc}$	Liftered Mel-frequency cepstral coefficients
$\hat{y}_i$ or $\hat{\boldsymbol{y}}_i$	Label or target prediction(s) associated with $\boldsymbol{x}_i$

Measures of Success

Acc	Accuracy
α	Significance level
B	Binomial distribution
C	Number of classes
C_p	Positive class
C_n	Negative class
$\boldsymbol{D}$	Data set
$\boldsymbol{D}_{train}$	Training data split
$\boldsymbol{D}_{cv}$	Cross-validation or validation data split
$\boldsymbol{D}_{test}$	Test data split
$\delta_{i,j}$	Kronecker delta between i and j
Δ_n	Sample-based error difference between two systems
$f(\cdot)$	Arbitrary function
F_1	Score function
F_t	Cumulative distribution function for Student's t-distribution
H_0	Null hypothesis
H_1	Alternative hypothesis
μ	Mean of data sample or population
M	Evaluation measure
N	Total number of data instances
N_c	Number of classes belonging to class c
N_{TP}	Number of true positive classifications
N_{TN}	Number of true negative classifications
N_{FP}	Number of false positive classifications
N_{FN}	Number of false negative classifications
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and std. devation σ
p	Probability distribution
P	Probability

$\Phi_{\mathcal{N}}$ ..... Cumulative distribution function of $\mathcal{N}(\mu, \sigma)$
ψ ..... Mean error parameter
$\mathbb{R}$ ..... The set of real numbers
ρ ..... Pearson's correlation coefficient
ρ_s ..... Spearman's rank correlation coefficient
ρ_c ..... Concordance correlation coefficient
σ ..... Standard deviation of data sample or population
σ^2 ..... Variance of data sample or population
θ ..... Operating point threshold value
t ..... t-statistic
$\boldsymbol{x}_i$ ..... The i-th example from a data set
$\boldsymbol{X}$ ..... Data matrix (e. g. acoustic features)
χ^2 ..... Chi-square probability distribution
X ..... McNemar test statistic
X^2 ..... Chi-square random variable
$\hat{y}_i$ or $\hat{\boldsymbol{y}}_i$ ..... Label or target prediction(s) associated with $\boldsymbol{x}_i$
y_i or $\boldsymbol{y}_i$ ..... Label or target value(s) associated with $\boldsymbol{x}_i$
$z^*_{c,B}$ ..... z-statistic

Deep Neural Networks

α ..... Regularization coefficient
a ..... Activation of a neuron
B ..... Size of mini-batch
b_i ..... Bias of neuron i
$\boldsymbol{b}$ ..... Bias vector
β ..... Batch normalization shift parameter
β_1, β_2 ..... Parameters for Adam optimizer
C ..... Number of discrete classes (in classification problem)
c_λ ..... Learning rate decay parameter

D	Number of inputs to a neuron
$\boldsymbol{D}_{train}$	Training data split
$\boldsymbol{D}_{valid}$	Validation data split
D_{KL}	Kullback-Leibler Divergence
E	Energy function
γ	Batch normalization scale parameter
H	Number of hidden units
$\mathcal{H}$	Hypothesis
$\boldsymbol{h}$	Hidden state vector in RNN models, hidden unit in RBM model
$\tilde{\mathbf{h}}$	Hidden state value vector
J_r	Jacobian matrix of function generating r
K	Length of BPTT data chunk
λ	Learning rate
$\mathcal{L}$	Global loss
L	Per-example loss
m_τ	Momentum constant at step τ
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and std. devation σ
ϕ	Activation function
ϕ'	Derivative of activation function ϕ
N	Dimensionality of vector
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and std. devation σ
∇	Gradient
Ω	Regularization cost
p	Step/epoch counter; dropout keep probability
ϕ	Activation function
Π	Number of model parameters
ψ	Decoder in the autoencoder model
r	Latent representation in autoencoder model
σ	Sigmoid activation function

σ^2 ..... Variance
τ ..... Time step
$\boldsymbol{\theta}$ ..... Model parameters
$\boldsymbol{\theta}^*$ ..... Optimal model parameters
U ..... Uniform distribution
V ..... Number of visible units
$\boldsymbol{v}$ ..... Visible unit in RBM model
$\boldsymbol{v}_t$ ..... Momentum velocity matrix at step t
w_{ij} ..... Weight from input j to neuron i
$\boldsymbol{W}$ ..... Weight matrix
$\boldsymbol{x}_i$ ..... The i-th example from a data set
$\boldsymbol{X}$ ..... Data matrix (e. g. acoustic features)
$\tilde{\mathbf{x}}$ ..... Noise-corrupted input to an autoencoder model
ξ ..... Encoder in the autoencoder model
$\hat{y}_i$ or $\hat{\boldsymbol{y}}_i$ ..... Label or target prediction(s) associated with $\boldsymbol{x}_i$
y_i or $\boldsymbol{y}_i$ ..... Label or target value(s) associated with $\boldsymbol{x}_i$
z ..... Input to a neuron

Likability Classification

α ..... Significance level
λ ..... Learning rate
r_k ..... Reliability of rater k

Social Signal Detection

β_1, β_2 ..... Adam optimizer hyper-parameters
D ..... Dimensionality of input vectors
λ ..... Learning rate
n ..... Number of subsequent frames stacked into an extended, stacked feature vector

N_{cell} ..... Number of cells
N_{layers} ..... Number of layers
t ..... Time step
T ..... Maximum time step
$\mathbf{x}$ ..... Input feature vector
$\mathbf{x}'$ ..... Extended, stacked feature vector

Conflict Detection

C ..... SVM complexity parameter
g_b ..... Backward function of BLSTM forward pass
g_f ..... Output function of BLSTM forward pass
L ..... Layer size := number of neurons per layer
o ..... Overlap
p ..... p-value for significance tests
ρ ..... Pearson's correlation coefficient
r_o ..... Overlap ratio
θ ..... Decision threshold for overlap prediction classification
$\boldsymbol{x}(t)$ ..... Feature vector at time step t
$y(t)$ ..... Binary target label at time step t
$\hat{y}(t)$ ..... Target prediction at time step t

End-to-End Emotion Recognition

$\mathbb{E}$ ..... Expectation operatior
F_0 ..... Fundamental frequency
K ..... Number of FIR filters (5 ms)
$\mathcal{L}_c$ ..... CCC-based loss function
M ..... Number of FIR filters (500 ms)
μ ..... Mean

ψ ..... Denominator in CCC loss function

ρ_c ..... Concordance correlation coefficient

σ ..... Standard deviation

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A System for Large-scale Machine Learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[2] L. F. Abbott, "Lapiceque's Introduction of the Integrate-And-Fire Model Neuron (1907)," *Brain Research Bulletin*, vol. 50, no. 5-6, pp. 303–304, 1999.

[3] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, and G. Penn, "Applying Convolutional Neural Networks Concepts to Hybrid NN-HMM Model for Speech Recognition," in *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan: IEEE, 2012, pp. 4277–4280.

[4] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 10, pp. 1533–1545, 2014.

[5] A. Agresti, *Categorical Data Analysis*. John Wiley & Sons, 2003.

[6] J. Allen, "Short Term Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 3, pp. 235–238, 1977.

[7] M. Aly, "Survey on Multiclass Classification Methods," *Neural Networks*, vol. 19, pp. 1–9, 2005.

[8] G. An, D.-G. Brizan, and A. Rosenberg, "Detecting Laughter and Filled Pauses Using Syllable-based Features," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013, pp. 178–181.

[9] G. An, "The Effects of Adding Noise During Backpropagation Training on a Generalization Performance," *Neural Computation*, vol. 8, no. 3, pp. 643–674, 1996.

[10] G. Andria, M. Savino, and A. Trotta, "Windows and Interpolation Algorithms to Improve Electrical Measurement Accuracy," *IEEE Transactions on Instrumentation and Measurement*, vol. 38, no. 4, pp. 856–863, 1989.

[11] S. Arlot and A. Celisse, "A Survey of Cross-Validation Procedures for Model Selection," *Statistics Surveys*, vol. 4, pp. 40–79, 2010.

[12] E. Aronson, T. D. Wilson, R. M. Akert, and S. R. Sommers, *Social Psychology*, 9th ed. Harlow, UK: Pearson Education Limited, 2018.

[13] M. M. H. E. Ayadi, M. S. Kamel, and F. Karray, "Survey on Speech Emotion Recognition: Features, Classification Schemes, and Databases," *Pattern Recognition*, vol. 44, pp. 572–587, 2011.

[14] F. A. C. Azevedo, L. R. B. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. L. Ferretti, R. E. P. Leite, W. J. Filho, R. Lent, and S. Herculano-Houzel, "Equal Numbers of Neuronal and Nonneuronal Cells Make the Human Brain an Isometrically Scaled-Up Primate Brain," *The Journal of Comparative Neurology*, vol. 513, no. 5, pp. 532–541, 2009.

[15] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[16] J.-A. Bachorowski1, M. Smoski1, and M. Owren, "The Acoustic Features of Human Laughter," *The Journal of the Acoustical Society of America*, vol. 110, no. 3, pp. 1581–1597, 2001.

[17] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic Differentiation in Machine Learning: A Survey," *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018.

[18] J. Bayer and C. Osendorfer, "Learning Stochastic Recurrent Networks," *arXiv preprint arXiv:1411.7610*, 2014.

[19] Y. Bengio, *Learning Deep Architectures for AI*. Now Publishers, Inc., 2009.

[20] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[21] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in Optimizing Recurrent Networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada: IEEE, 2013, pp. 8624–8628.

[22] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[23] J. Bergstra, G. Desjardins, P. Lamblin, and Y. Bengio, "Quadratic Polynomials Learn Better Image Features," University of Montreal, Tech. Rep. 1337, 2009.

[24] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, and Y. Bengio, "Theano: a CPU and GPU Math Expression Compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Austin, TX, USA, 2010, Oral Presentation.

[25] M. Bhargava and R. Rose, "Architectures for Deep Neural Network Based Acoustic Models Defined Over Windowed Speech Waveforms," in *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Dresden, Germany: ISCA, 2015.

[26] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. New York, NY, USA: Springer, 2006.

[27] K. Boakye, O. Vinyals, and G. Friedland, "Improved Overlapped Speech Handling for Speaker Diarization," in *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Florence, Italy: ISCA, 2011, pp. 941–944.

[28] S. M. Bohte, "Spiking Neural Networks," PhD Thesis, Universiteit Leiden, Leiden, The Netherlands, 2003.

[29] L. Bottou, "Online Algorithms and Stochastic Approximations," in *Online Learning and Neural Networks*, D. Saad, Ed. Cambridge, UK: Cambridge University Press, 1998.

[30] R. R. Bouckaert and E. Frank, "Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 3–12.

[31] Y. Boureau, J. Ponce, and Y. LeCun, "A Theoretical Analysis of Feature Pooling in Visual Recognition," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 111–118.

[32] K. Bousmalis, M. Mehu, and M. Pantic, "Spotting Agreement and Disagreement: A Survey of Nonverbal Audiovisual Cues and Tools," in *Proceedings of the 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII)*, vol. 2. Los Alamitos, CA, USA: IEEE, 2009.

[33] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to Statistical Learning Theory," in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Computer Science, vol. 3176. Springer, 2003, pp. 169–207.

[34] A. P. Bradley, "The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[35] R. Brueckner and B. Schuller, "Likability Classification - A Not So Deep Neural Network Approach," in *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Portland, OR, USA: ISCA, 2012.

[36] ——, "Hierarchical Neural Networks and Enhanced Class Posteriors for Social Signal Classification," in *Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Olomouc, Czech Republic: IEEE, 2013, pp. 361–364.

[37] ——, "Social Signal Classification Using Deep BLSTM Recurrent Neural Networks," in *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy: IEEE, 2014, pp. 4856–4860.

[38] ——, *Be at Odds? Deep and Hierarchical Neural Networks for Classification and Regression of Conflict in Speech*. Cham: Springer International Publishing, 2015, ch. Conflict and Multimodal Communication: Social Research and Machine Intelligence, pp. 403–429.

[39] R. Brueckner, M. Schmitt, M. Pantic, and B. W. Schuller, "Spotting Social Signals in Conversational Speech over IP: A Deep Learning Perspective," in *Proceedings of the 18th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Stockholm, Sweden: ISCA, 2017, pp. 2371–2375.

[40] P. M. Brunet and R. Cowie, "Towards a Conceptual Framework of Research on Social Signal Processing," *Journal on Multimodal User Interfaces*, vol. 6, no. 3-4, pp. 101–115, 2012.

[41] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding Disentangling in β-VAE," *arXiv preprint arXiv:1804.03599*, 2018.

[42] F. Burkhardt, M. Eckert, W. Johannsen, and J. Stegmann, "A Database of Age and Gender Annotated Telephone Speech," in *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*. Valletta, Malta: ELRA, 2010.

[43] F. Burkhardt, B. Schuller, B. Weiss, and F. Weninger, "'Would You Buy A Car From Me?' - On the Likability of Telephone Voices," in *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Florence, Italy: ISCA, 2011, pp. 1557–1560.

[44] D. Byrd, "Relations of Sex and Dialect to Reduction," *Speech Communication*, vol. 15, no. 1-2, pp. 39–54, 1994.

[45] N. Campbell and P. Mokhtari, "Voice Quality: The 4th Prosodic Dimension," in *Proceedings of the 15th International Congress of Phonetic Sciences (ICPhS)*. Barcelona, Spain: IPA, 2003, pp. 2417–2420.

[46] I. Carcea and R. C. Froemke, "Cortical Plasticity, Excitatory-Inhibitory Balance, and Sensory Perception," *Progress in Brain Research*, vol. 207, pp. 65–90, 2013.

[47] M. Á. Carreira-Perpigñán and G. E. Hinton, "On Contrastive Divergence Learning," in *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Bridgetown, Barbados: JMRL, 2005, pp. 33–40.

[48] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011.

[49] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[50] A. Chattopadhyay, D. W. Dahl, R. J. B. Ritchie, and K. N. Shahin, "Hearing Voices: The Impact of Announcer Speech Characteristics on Consumer Response to Broadcast Advertising," *Journal of Consumer Psychology*, vol. 13, no. 3, pp. 198–204, 2003.

[51] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems," *arXiv:1512.01274*, 2015.

[52] K. Cho, A. Ilin, and T. Raiko, "Improved Learning of Gaussian-Bernoulli Restricted Boltzmann Machines," in *Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN)*, ser. Lecture Notes in Computer Science, T. Honkela, W. Duch, M. Girolami, and S. Kaski, Eds., vol. 6791. Espoo, Finland: Springer, 2011, pp. 10–17.

[53] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.

[54] F. Chollet, "Keras: The Python Deep Learning Library," https://keras.io, 2015.

[55] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 27. Montreal, Canada: MIT Press, 2014.

[56] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated Feedback Recurrent Neural Networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 2067–2075.

[57] H. Clark and J. F. Tree, "Using uh and um in Spontaneous Speaking," *Cognition*, vol. 84, no. 1, pp. 73–111, 2002.

[58] L. Coelho, D. Braga, M. Sales-Dias, and C. Garcia-Mateo, "An Automatic Voice Pleasantness Classification System Based on Prosodic and Acoustic Patterns of Voice Preference," in *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Florence, Italy: ISCA, 2011.

[59] P. T. Coleman, M. Deutsch, and E. C. Marcus, Eds., *The Handbook of Conflict Resolution: Theory and Practice*, 3rd ed. San Francisco, CA, USA: John Wiley & Sons, 2014.

[60] S. A. Collins, "Men's Voices and Women's Choices," *Animal Behaviour*, vol. 60, no. 6, pp. 773–780, 2000.

[61] C. Cortes and M. Mohri, "On Transductive Regression," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 19. Whistler, Canada: MIT Press, 2006, pp. 305–312.

[62] D. Crystal, "A Perspective for Paralanguage," *Le Maître Phonétique*, vol. 120, pp. 25–29, 1963.

[63] ——, "Paralinguistics," *The Body as a Medium of Expression*, pp. 162–174, 1975.

[64] ——, *A Dictionary of Linguistics and Phonetics*. John Wiley & Sons, 2011, vol. 30.

[65] G. Cybenko, "Approximation by Superposition of Sigmoidal Functions," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[66] G. Dahl, T. Sainath, and G. Hinton, "Improving Deep Neural Networks for LVCSR Using Rectified Linear Units and Dropout," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada: IEEE, 2013, pp. 8609–8613.

[67] S. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[68] C. K. W. De Dreu and L. R. Weingart, "Task Versus Relationship Conflict, Team Performance, and Team Member Satisfaction: A Meta-Analysis," *Journal of Applied Psychology*, vol. 88, no. 4, p. 741, 2003.

[69] J. Deng, X. Xu, Z. Zhang, S. Frühholz, and B. Schuller, "Semisupervised Autoencoders for Speech Emotion Recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 1, pp. 31–43, 2018.

[70] M. Deutsch, *The Resolution of Conflict: Constructive and Destructive Processes*, ser. Carl Hovland Memorial Lectures Series. London, UK: Yale University Press, 1977.

[71] R. Dey and F. M. Salemt, "Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks," in *Proceedings of the 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. Medford, MA, United States: IEEE, 2017, pp. 1597–1600.

[72] S. Dieleman and B. Schrauwen, "End-to-End Learning for Music Audio," in *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy: IEEE, 2014, pp. 6964–6968. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6854950

[73] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[74] A. Diment, T. Heittola, and T. Virtanen, "Semi-Supervised Learning for Musical Instrument Recognition," in *Proceedings of the 21st European Signal Processing Conference (EUSIPCO)*. Marrakech, Morocco: IEEE, 2013, pp. 1–5.

[75] Y. Dodge, *The Concise Encyclopedia of Statistics*. Springer Science & Business Media, 2008.

[76] R. J. Douglas and K. A. C. Martin, "Recurrent Neuronal Circuits in the Neocortex," *Current Biology*, vol. 17, no. 13, pp. 496–500, 2007.

[77] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[78] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, USA: John Wiley & Sons, 2001.

[79] A. L. Edwards, "Note on the 'Correction for Continuity' in Testing the Significance of the Difference Between Correlated Proportions," *Psychometrika*, vol. 13, no. 3, pp. 185–187, 1948.

[80] J. P. Egan, *Signal Detection Theory and ROC Analysis*. Academic Press, 1975.

[81] J. L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[82] K. Eswaran and V. Singh, "Some Theorems for Feed Forward Neural Networks," *arXiv preprint arXiv:1509.05177*, 2015.

[83] F. Eyben, *Real-time Speech and Music Classification by Large Audio Feature Space Extraction*. Springer, 2015.

[84] F. Eyben, M. Wöllmer, and B. Schuller, "openSMILE – The Munich Versatile and Fast Open-Source Audio Feature Extractor," in *Proceedings of the 18th ACM International Conference on Multimedia*. Florence, Italy: ACM, 2010, pp. 1459–1462.

[85] F. Eyben, F. Weninger, F. Gross, and B. Schuller, "Recent Developments in openSMILE, the Munich Open-Source Multimedia Feature Extractor," in *Proceedings of the 21st ACM International Conference on Multimedia*. Barcelona, Spain: ACM, 2013, pp. 835–838.

[86] F. Eyben, K. R. Scherer, B. W. Schuller, J. Sundberg, E. André, C. Busso, L. Y. Devillers, J. Epps, P. Laukka, S. S. Narayanan, and K. P. Truong, "The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing," *IEEE Transactions on Affective Computing*, vol. 7, no. 2, pp. 190–202, 2016.

[87] M. W. Fagerland, S. Lydersen, and P. Laake, "The McNemar Test for Binary Matched-Pairs Data: Mid-p and Asymptotic are Better Than Exact Conditional," *BMC Medical Research Methodology*, vol. 13, no. 1, p. 91, 2013.

[88] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[89] A. Fischer and C. Igel, "Training Restricted Boltzmann Machines: An Introduction," *Pattern Recognition*, vol. 47, no. 1, pp. 25–39, 2014.

[90] R. Frigg and S. Hartmann, "Models in Science," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2018.

[91] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[92] ——, "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.

[93] A. Gammerman, V. Vovk, and V. Vapnik, "Learning by Transduction," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 148–155.

[94] E. A. Garcia and H. He, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge & Data Engineering*, vol. 21, pp. 1263–1284, 2008.

[95] J. Geiger, F. Eyben, B. Schuller, and G. Rigoll, "Detecting Overlapping Speech with Long Short-Term Memory Recurrent Neural Networks," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013, pp. 1668–1672.

[96] J. T. Geiger, R. Vipperla, S. Bozonnet, N. Evans, B. Schuller, and G. Rigoll, "Convolutive Non-Negative Sparse Coding and New Features for Speech Overlap Handling in Speaker Diarization," in *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Portland, OR, USA: ISCA, 2012.

[97] F. A. Gers and J. Schmidhuber, "Recurrent Nets that Time and Count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, vol. 3. Como, Italy: IEEE, 2000, pp. 189–194.

[98] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[99] W. Gerstner and W. M. Kistler, *Spiking Neuron Models - Single Neurons, Populations, Plasticity*. Cambridge, UK: Cambridge University Press, 2002.

[100] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks And Models of Cognition*. Cambridge, UK: Cambridge University Press, 2002.

[101] P. Ghahremani, V. Manohar, D. Povey, and S. Khudanpur, "Acoustic Modelling from the Signal Domain Using CNNs," in *Proceedings of the 17th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. San Francisco, CA, USA: ISCA, 2016, pp. 3434–3438.

[102] L. Gillick and S. Cox, "Statistical Significance Tests for Speech Recognition Algorithms," in *Proceedings of the 14th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Glasgow, Scotland: IEEE, 1989, pp. 532–535.

[103] F. Girosi, M. Jones, and T. Poggio, "Regularization Theory and Neural Networks Architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 1995.

[104] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9. Chia Laguna Resort, Sardinia, Italy: JMRL, 2010, pp. 249–256.

[105] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 15. Fort Lauderdale, FL, USA: JMLR, 2011, pp. 315–323.

[106] K. Godfrey, *Compartmental Models and Their Application*. London, UK: Academic Press Inc., 1983.

[107] M. J. R. Gomez and D. P. W. Ellis, "Error Visualization for Tandem Acoustic Modeling on the Aurora Task," in *Proceedings of the 27th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Orlando, FL, USA: IEEE, 2002, pp. 4176–4179.

[108] S. Gonzalez and X. Anguera, "Perceptually Inspired Features for Speaker Likability Classification," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada: IEEE, 2013, pp. 8490–8494.

[109] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, vol. 1.

[110] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout Networks," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 28, no. 3, Atlanta, GA, USA, 2013, pp. 1319–1327.

[111] G. Gosztolya, "Detecting Laughter and Filler Events by Time Series Smoothing with Genetic Algorithms," in *Proceedings of the 18th International Conference on Speech and Computer (SPECOM)*, ser. Lecture Notes in Artificial Intelligence. Budapest, Hungary: Springer, 2016, pp. 232–239.

[112] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Springer, 2012, vol. 385.

[113] A. Graves and N. Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Beijing, China, 2014, pp. 1764–1772.

[114] A. Graves and J. Schmidhuber, "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures," *Neural Networks*, vol. 18, pp. 602–610, 2005.

[115] K. Greff, R. Srivastava, J. Koutník, B. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, no. 99, pp. 1–11, 2015.

[116] K. Greff, R. K. Srivastava, and J. Schmidhuber, "Highway and Residual Networks Learn Unrolled Iterative Estimation," *arXiv preprint arXiv:1612.07771*, 2016.

[117] F. Grèzes, J. Richards, and A. Rosenberg, "Let Me Finish: Automatic Conflict Detection Using Speaker Overlap," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013, pp. 200–204.

[118] M. Grimm and K. Kroschel, "Evaluation of Natural Emotions Using Self Assessment Manikins," in *Proceedings of the 2005 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. San Juan, Puerto Rico: IEEE, 2005, pp. 381–385.

[119] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent Advances in Convolutional Neural Networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[120] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio, "Noisy Activation Functions," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, New York, NY, USA, 2016, pp. 3059–3068.

[121] B. B. Gupta and Q. Z. Sheng, *Machine Learning for Computer and Cyber Security: Principle, Algorithms, and Practices*. CRC Press, 2019.

[122] R. Gupta, K. Audhkhasi, S. Lee, and S. S. Narayanan, "Paralinguistic Event Detection Using Probabilistic Time-Series Smoothing and Masking," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013, pp. 173–177.

[123] R. Gupta, K. Audhkhasi, S. Lee, and S. Narayanan, "Detecting Paralinguistic Events in Audio Stream Using Context in Features and Probabilistic Decisions," *Computer Speech and Language*, vol. 36, no. C, pp. 72–92, 2016.

[124] R. H. R. Hahnloser and H. S. Seung, "Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 14. Vancouver, Canada: MIT Press, 2001, pp. 217–223.

[125] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit," *Nature*, vol. 405, no. 6789, p. 947, 2000.

[126] B. Hammer, "On the Approximation Capability of Recurrent Neural Networks," *Neurocomputing*, vol. 31, no. 1-4, pp. 107–123, 2000.

[127] D. J. Hand and R. J. Till, "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001.

[128] J. A. Hanley and B. J. McNeil, "The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[129] E. Hänsler and G. Schmidt, *Speech and Audio Processing in Adverse Environments*. Springer Science & Business Media, 2008.

[130] S. Hantke, C. Stemp, and B. W. Schuller, "Annotator Trustability-based Cooperative Learning Solutions for Intelligent Audio Analysis," in *Proceedings*

of the 19th Annual Conference of the International Speech Communication Association (INTERSPEECH). Hyderabad, India: ISCA, 2018, pp. 3504–3508.

[131] J. Harrigan, R. Rosenthal, K. R. Scherer, and K. Scherer, *New Handbook of Methods in Nonverbal Behavior Research*. Oxford University Press, 2008.

[132] F. J. Harris, "On The Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[133] M. Hasenjäger and H. Ritter, "Active Learning in Neural Networks," in *New Learning Paradigms in Soft Computing*. Springer, 2002, pp. 137–169.

[134] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed., ser. Springer Series in Statistics. New York, USA: Springer, 2016.

[135] S. S. Haykin, *Neural Networks and Learning Machines*. New York, NY, USA: Prentice Hall, 2009.

[136] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. Washington, DC, USA: IEEE, 2015, pp. 1026–1034.

[137] ——, "Deep Residual Learning for Image Recognition," in *Proceedings of the 29th Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, 2016, pp. 770–778.

[138] ——, "Identity Mappings in Deep Residual Networks," in *Proceedings of the 14th European Conference on Computer Vision (ECCV)*. Amsterdam, The Netherlands: Springer, 2016, pp. 630–645.

[139] J. C. Heck and F. M. Salem, "Simplified Minimal Gated Unit Variations for Recurrent Neural Networks," in *Proceedings of the 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. Medford, MA, United States: IEEE, 2017, pp. 1593–1596.

[140] S. Herculano-Houzel, "The Remarkable, Yet Not Extraordinary, Human Brain As a Scaled-Up Primate Brain and its Associated Cost," in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 109, no. 1. PNAS, 2012, pp. 10 661–10 668.

[141] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "RASTA-PLP Speech Analysis Technique," in *Proceedings of the 17th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. San Francisco, CA, USA: IEEE, 1992, pp. 121–124.

[142] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in *Advances in Neural Information Processing Systems (NIPS)*. Long Beach, CA, USA: MIT Press, 2017, pp. 6626–6637.

[143] A. A. Hill, *Introduction to Linguistic Structures: from Sound to Sentence in English*. Harcourt, Brace, 1958.

[144] J. L. Hindmarsh and R. M. Rose, "A Model of Neuronal Bursting Using Three Coupled First Order Differential Equations," *Proceedings of the Royal Society B: Biological Sciences*, vol. 221, no. 1222, pp. 87–102, 1984.

[145] G. E. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," Machine Learning Group, University of Toronto, Technical Report 2010-003, 2010.

[146] G. Hinton and T. J. Sejnowski, Eds., *Unsupervised Learning: Foundations of Neural Computation*, ser. Computational Neuroscience. Cambridge, MA, USA: MIT Press, 1998.

[147] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[148] G. E. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[149] G. E. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[150] G. E. Hinton and T. J. Sejnowski, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Rumelhart, David E. and McClelland, James L. and the PDP Research Group, 1986, vol. 1, no. 282-317, ch. Learning and Relearning in Boltzmann Machines, p. 36.

[151] H.-G. Hirsch and D. Pearce, "The Aurora Experimental Framework for the Performance Evaluation of Speech Recognition Systems Under Noisy Conditions," in *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.

[152] H.-G. Hirsch, P. Meyer, and H.-W. Rühl, "Improved Speech Recognition Using High-Pass Filtering of Subband Envelopes," in *Proceedings of the 2nd European Conference on Speech Communication and Technology (EUROSPEECH)*. Genova, Italy: ISCA, 1991, pp. 413–416.

[153] S. Hochreiter, "Untersuchungen zu Dynamischen Neuronalen Netzen," Diploma Thesis, Technische Universität München, Germany, 1991.

[154] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[155] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*, Kremer and Kolen, Eds. IEEE Press, 2001.

[156] C. R. Hodges-Simeon, S. J. C. Gaulin, and D. A. Puts, "Different Vocal Parameters Predict Perceptions of Dominance and Attractiveness," *Human Nature*, vol. 21, no. 4, pp. 406–427, 2010.

[157] A. L. Hodgkin and A. F. Huxley, "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[158] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[159] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision*. Springer, 2016, pp. 646–661.

[160] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the 30th Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA: IEEE, 2017, pp. 4700–4708.

[161] D. H. Hubel and T. N. Wiesel, "Receptive Fields of Single Neurones in the Cat's Striate Cortex," *The Journal of Physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[162] ——, "Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[163] ——, "Receptive Fields and Functional Architecture of monkey Striate Cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[164] D. Huh and T. J. Sejnowski, "Gradient Descent for Spiking Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada: MIT Press, 2018, pp. 1440–1450.

[165] M. Huzaifah, "Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks," *arXiv preprint arXiv:1706.07156*, 2017.

[166] S. Ioffe, "Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models," in *Advances in Neural Information Processing Systems (NIPS)*. Long Beach, CA, USA: MIT Press, 2017, pp. 1945–1953.

[167] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv preprint arXiv:1502.03167*, 2015.

[168] E. M. Izhikevich, "Simple Model of Spiking Neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[169] H. Jaeger, "Adaptive Nonlinear System Identification with Echo State Networks," in *Advances in Neural Information Processing Systems (NIPS)*, no. 16. Vancouver, Canada: MIT Press, 2003, pp. 609–616.

[170] N. Jaitly and G. Hinton, "Learning a Better Representation of Speech Soundwaves using Restricted Boltzmann Machines," in *Proceedings of the 36th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Prague, Czech Republic: IEEE, 2011, pp. 5884–5887.

[171] A. Janicki, "Non-linguistic Vocalisation Recognition Based on Hybrid GMM-SVM Approach," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013, pp. 153–157.

[172] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*. Orlando, FL, USA: ACM, 2014, pp. 675–678.

[173] M. Jordan, "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine," in *Proceedings of the 8th Annual Conference of the Cognitive Science Society*. Taylor & Francis Group, 1986.

[174] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An Empirical Exploration of Recurrent Network Architectures," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 2342–2350.

[175] M. Kächele, P. Thiam, G. Palm, F. Schwenker, and M. Schels, "Ensemble Methods for Continuous Affect Recognition: Multi-modality, Temporality, and Challenges," in *Proceedings of the of the 23rd ACM International Conference on Multimedia (AVEC@ACM)*. Brisbane, Australia: ACM, 2015, pp. 9–16.

[176] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[177] P. Kennedy and A. Hauptmann, "Laughter Extracted from Television Closed Captions as Speech Recognizer Training Data," in *Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH)*. Budapest, Hungary: ISCA, 1999, pp. 663–666.

[178] H. Ketabdar and H. Bourlard, "Enhanced Phone Posteriors for Improving Speech Recognition Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 1094–1106, 2010.

[179] N. Ketkar, "Introduction to PyTorch," in *Deep Learning with Python*. Springer, 2017, pp. 195–208.

[180] A. Kießling, *Extraktion und Klassifikation Prosodischer Merkmale in der Automatischen Sprachverarbeitung*. Shaker, 1996.

[181] S. Kim, F. Valente, and A. Vinciarelli, "Automatic Detection of Conflicts in Spoken Conversations: Ratings and Analysis of Broadcast Political Debates," in *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan: IEEE, 2012, pp. 5089–5092.

[182] S. Kim, M. Filippone, F. Valente, and A. Vinciarelli, "Predicting the Conflict Level in Television Political Debates: An Approach Based on Crowdsourcing, Nonverbal Communication and Gaussian Processes," in *Proceedings of the 20th ACM International Conference on Multimedia*. Nara, Japan: ACM, 2012, pp. 793–796.

[183] S. Kim, S. H. Yella, and F. Valente, "Automatic Detection of Conflict Escalation in Spoken Conversations," in *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Portland, OR, USA: ISCA, 2012.

[184] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[185] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[186] K. C. Kiwiel, "Convergence and Efficiency of Subgradient Methods for Quasiconvex Minimization," *Mathematical programming*, vol. 90, no. 1, pp. 1–25, 2001.

[187] M. Knox and N. Mirghafori, "Automatic Laughter Detection Using Neural Networks," in *Proceedings of the 8th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Antwerp, Belgium: ISCA, 2007, pp. 2973–2976.

[188] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A Study on Data Augmentation of Reverberant Speech for Robust Speech Recognition," in *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA, USA: IEEE, 2017, pp. 5220–5224.

[189] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *International Joint Conferences on Artificial Intelligence Organization*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[190] A. N. Kolmogorov, "On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition," in *Doklady Akademii Nauk*, vol. 114, no. 5. Russian Academy of Sciences, 1957, pp. 953–956.

[191] J. Kossaifi, R. Walecki, Y. Panagakis, J. Shen, M. Schmitt, F. Ringeval, J. Han, V. Pandit, B. Schuller, K. Star *et al.*, "SEWA DB: A Rich Database for Audio-Visual Emotion and Sentiment Research in the Wild," *arXiv preprint arXiv:1901.02839*, 2019.

[192] J. Kreiman, B. R. Gerratt, and K. Precoda, "Listener Experience and Perception of Voice Quality," *Journal of Speech, Language, and Hearing Research*, vol. 33, no. 1, pp. 103–115, 1990.

[193] T. F. Krikke and K. P. Truong, "Detection of Nonverbal Vocalizations Using Gaussian Mixture Models: Looking for Fillers and Laughter in Conversational Speech," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, Framce: ISCA, 2013, pp. 163–167.

[194] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Cassification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*. Lake Tahoe, NV, USA: MIT Press, 2012, pp. 1097–1105.

[195] K. J. Lang and G. Hinton, "The Development of the Time-Delay Neural Network Architecture for Speech Recognition," Carnegie Mellon University, Tech. Rep. CMU-CS-88-152, 1988.

[196] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A Time-Delay Neural Network Architecture for Isolated Word Recognition," *Neural Networks*, vol. 3, no. 1, pp. 23–43, 1990.

[197] L. Lapicque, "Recherches Quantitatives sur l'Excitation Electrique des Nerfs Traitée comme une Polarization," *Journal de Physiologie et Pathologie General*, vol. 9, pp. 620–635, 1907.

[198] I. Lawrence and K. Lin, "A Concordance Correlation Coefficient to Evaluate Reproducibility," *Biometrics*, pp. 255–268, 1989.

[199] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the trade*, G. Orr and K.-R. Müller, Eds. Springer, 1998, ch. 2, pp. 9–50.

[200] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[201] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[202] J. Lee Rodgers and W. A. Nicewander, "Thirteen Ways to Look at the Correlation Coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[203] A. Li and H. Wang, "Friendly Speech Analysis and Perception in Standard Chinese," in *Proceedings of the 8th International Conference on Spoken Language Processing (INTERSPEECH)*. Jeju Island, Korea: ISCA, 2004.

[204] Q. Liao and T. Poggio, "Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex," *arXiv preprint arXiv:1604.03640*, 2016.

[205] R. Likert, "A Technique for the Measurement of Attitudes," *Archives of Psychology*, 1932.

[206] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the Computational Efficiency of Training Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada: MIT Press, 2014, pp. 855–863.

[207] H. Lodish, A. Berk, C. A. Kaiser, M. Krieger, A. Bretscher, H. Ploegha, A. Amon, and K. C. Martin, *Molecular Cell Biology*. New York, USA: W. H. Freeman and Company, 2016.

[208] I. Luengo, I. Saratxaga, E. Navas, I. Hernáez, J. Sanchez, and I. Sainz, "Evaluation of Pitch Detection Algorithms Under Real Conditions," in *Proceedings of the 32nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Honolulu, HI, USA: IEEE, 2007, pp. 1057–1060.

[209] Y. Luo, L. Wan, J. Liu, J. Harkin, L. McDaid, Y. Cao, and X. Ding, "Low Cost Interconnected Architecture for the Hardware Spiking Neural Networks," *Frontiers in Neuroscience*, vol. 12, 2018.

[210] D. A. Lyon, "The Discrete Fourier Transform, Part 4: Spectral Leakage," *Journal of Object Technology*, vol. 8, no. 7, 2009.

[211] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 30, no. 1, Atlanta, GA, USA, 2013, p. 3.

[212] W. Maass, "Computation with Spiking Neurons," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 2003, pp. 1080–1083.

[213] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[214] A. Makhzani and B. Frey, "K-sparse Autoencoders," *arXiv preprint arXiv:1312.5663*, 2013.

[215] B. S. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface*. John Wiley & Sons, 2002, vol. 1.

[216] Q. Mao, M. Dong, Z. Huang, and Y. Zhan, "Learning Salient Features for Speech Emotion Recognition Using Convolutional Neural Networks," *IEEE Transactions on Multimedia*, vol. 16, pp. 2203–2213, 2014.

[217] S. Mariooryad and C. Busso, "Correcting Time-Continuous Emotional Labels by Modeling the Reaction Lag of Evaluators," *IEEE Transactions on Affective Computing*, vol. 6, pp. 97–108, 2015.

[218] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET Curve in Assessment of Detection Task Performance," National Institute of Standards and Technology (NIST), Gaithersburg, MD, Tech. Rep., 1997.

[219] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," in *Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN)*. Springer, 2011, pp. 52–59.

[220] Q. McNemar, "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[221] N. Mehra and S. Gupta, "Survey on Multiclass Classification Methods," *International Journal of Computer Science and Information Technologies*, vol. 4, no. 4, pp. 572–576, 2013.

[222] M. Mehu and K. R. Scherer, "A Psycho-Ethological Approach to Social Signal Processing," *Cognitive Processing*, vol. 13, no. 2, pp. 397–414, 2012.

[223] M. L. Meistrell and K. A. Spackman, "Evaluation of Neural Network Performance by ROC Analysis: Examples from the Biotechnology Domain," *Computer Methods and Programs in Biomedicine*, vol. 32, pp. 73–80, 1989.

[224] A. Mencattini, E. Martinelli, F. Ringeval, B. Schuller, and C. Di Natale, "Continuous Estimation of Emotions in Speech by Dynamic Cooperative Speaker Models," *IEEE Transactions on Affective Computing*, vol. 8, no. 3, pp. 314–327, 2017.

[225] B. Milde and C. Biemann, "Using Representation Learning and Out-Of-Domain Data for a Paralinguistic Speech Task," in *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Dresden, Germany: ISCA, 2015.

[226] R. Miller, Ed., *Time and the Brain*, ser. Conceptual Advances in Brain Research. Amsterdam, The Netherlands: Harwood Academic Publishers, 2000, vol. 3.

[227] A.-R. Mohamed, G. Hinton, and G. Penn, "Understanding How Deep Belief Networks Perform Ccoustic Modelling," in *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan: IEEE, 2012, pp. 4273–4276.

[228] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2012.

[229] C. Montacié and M.-J. Caraty, "Pitch and Intonation Contribution to Speakers' Traits Classification," in *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Portland, OR, USA: ISCA, 2012, pp. 526–529.

[230] A. Moore, "Carnegie Mellon Dean Of Computer Science On The Future Of AI," 2017. [Online]. Available: https://www.forbes.com/sites/peterhigh/2017/10/30/carnegie-mellon-dean-of-computer-science-on-the-future-of-ai

[231] C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber," *Biophysical Journal*, vol. 35, no. 1, pp. 193–213, 1981.

[232] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 807–814.

[233] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding Gradient Noise Improves Learning for Very Deep Networks," *arXiv preprint arXiv:1511.06807*, 2015.

[234] M. S. Nesler, D. M. Storr, and J. T. Tedeschi, "The Interpersonal Judgment Scale: A Measure of Liking or Respect?" *The Journal of Social Psychology*, vol. 133, no. 2, pp. 237–242, 1993.

[235] Y. Nesterov, "A Method for Solving the Convex Programming Problem with Convergence Rate $O(1/k^2)$," in *Soviet Mathematics Doklady*, vol. 27, 1983, pp. 372–367.

[236] D. Nguyen and B. Widrow, "Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights," in *International Joint Conference on Neural Networks (IJCNN)*. San Diego, CA, USA: IEEE, 1990, pp. 21–26.

[237] T. Nguyen, "Total Number of Synapses in the Adult Human Neocortex," *Undergraduate Journal of Mathematical Modeling: One + Two*, vol. 3, no. 1, pp. 1–13, 2010.

[238] E. Nöth, A. Batliner, A. Kießling, R. Kompe, and H. Niemann, "Verbmobil: The Use of Prosody in the Linguistic Components of a Speech Understanding System," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp. 519–532, 2000.

[239] F. Olsson, "A Literature Survey of Active Machine Learning in the Context of Natural Language Processing," Swedish Institute of Computer Science, Tech. Rep., 2009.

[240] D. Palaz, R. Collobert, and M. Magimai-Doss, "Estimating Phoneme Class Conditional Probabilities from Raw Speech Signal using Convolutional Neural Networks," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013.

[241] D. Palaz, M. Magimai-Doss, and R. Collobert, "Analysis of CNN-based Speech Recognition System Using Raw Speech as Input," in *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Dresden, Germany: ISCA, 2015.

[242] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the Exploding Gradient Problem," *CoRR, abs/1211.5063*, vol. 2, 2012.

[243] ——, "On the Difficulty of Training Recurrent Neural Networks," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 30, no. 1, Atlanta, GA, USA, 2013, pp. 1310–1318.

[244] B. A. Pearlmutter, "Dynamic Recurrent Neural Networks," 1990.

[245] A. Pesarin, M. Cristani, V. Murino, and A. Vinciarelli, "Conversation Analysis at Work: Detection of Conflict in Competitive Discussions through Automatic Turn-Organization Analysis," *Cognitive Processing*, vol. 13, no. 2, pp. 533–540, 2012.

[246] M. Pfeiffer and T. Pfeil, "Deep Learning with Spiking Neurons: Opportunities and Challenges," *Frontiers in Neuroscience*, vol. 12, 2018.

[247] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout Improves Recurrent Neural Networks for Handwriting Recognition," in *Proceedings of International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Crete, Greece: IEEE, 2014, pp. 285–290.

[248] R. W. Picard, *Affective Computing*. Cambridge, MA, USA: MIT Press, 2000.

[249] L. Pinto-Coelho, D. Braga, M. Sales-Dias, and C. Garcia-Mateo, "On the Development of an Automatic Voice Pleasantness Classification and Intensity Estimation System," *Computer Speech & Language*, vol. 27, no. 1, pp. 75–88, 2013.

[250] D. C. Plaut, "Experiments On Learning by Back Propagation," 1986.

[251] I. Poggi and F. D'Errico, "Social Signals: a Framework in Terms of Goals and Beliefs," *Cognitive Processing*, vol. 13, no. 2, pp. 427–445, 2012.

[252] F. B. Pokorny, B. W. Schuller, P. B. Marschik, R. Brueckner, P. Nyström, N. Cummins, S. Bölte, C. Einspieler, and T. Falck-Ytter, "Earlier Identification of Children with Autism Spectrum Disorder: An Automatic Vocalisation-Based Approach," in *Proceedings of the 18th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Stockholm, Sweden: ISCA, 2017, pp. 309–313.

[253] A. Polychroniou, H. Salamin, and A. Vinciarelli, "The SSPNet-Mobile Corpus: Social Signal Processing Over Mobile Phones," in *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC)*. Reykjavik, Iceland: ELRA, 2014, pp. 1492–1498.

[254] L. R. Pondy, "Organizational Conflict: Concepts and Models," *Administrative Science Quarterly*, vol. 12, no. 2, pp. 296–320, 1967.

[255] D. M. Powers, "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[256] L. Prechelt, "Early Stopping - But When?" in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 55–69.

[257] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*. Macmillan New York, 1992, vol. 2.

[258] F. J. Provost and T. Fawcett, "Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions," in *Proceedings of the Third International Conference on Knowlegde Discovery and Data Mining (KDD)*, vol. 97. Newport Beach, CA, USA: AAAI, 1997, pp. 43–48.

[259] J. Qi, Y. Jiang, and R. Liu, "Auditory Features Based on Gammatone Filters for Robust Speech Recognition," in *Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. Bejing, China: IEEE, 2013, pp. 305–308.

[260] Y. Qi and R. E. Hillman, "Temporal and Spectral Estimations of Harmonics-to-Noise Ratio in Human Voice Signals," *The Journal of the Acoustical Society of America*, vol. 102, no. 1, pp. 537–543, 1997.

[261] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried, "Invariant Visual Representation by Single Neurons in the Human Brain," *Nature*, vol. 435, no. 7045, p. 1102, 2005.

[262] M. A. Rahim, *Managing Conflict in Organizations*, 4th ed. New York, NY, USA: Routledge, 2017.

[263] L. B. Rall, *Automatic Differentiation: Techniques and Applications*, ser. Lecture Notes in Computer Science. Springer, 1981, vol. 120.

[264] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *arXiv preprint arXiv:1710.05941*, 2018.

[265] O. Räsänen and J. Pohjalainen, "Random Subset Feature Selection in Automatic Recognition of Developmental Disorders, Affective States, and Level of Conflict from Speech," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013, pp. 210–214.

[266] A. Ray, S. Rajeswar, and S. Chaudhury, "Text Recognition using Deep BLSTM Networks," in *Proceedings of the 8th International Conference on Advances in Pattern Recognition (ICAPR)*. Kolkata, India: IEEE, 2015, pp. 1–6.

[267] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code*. Cambridge, MA, USA: MIT Press, 1999.

[268] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive Auto-Encoders: Explicit Invariance During Feature Extraction," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, WA, USA, 2011, pp. 833–840.

[269] F. Ringeval, A. Sonderegger, J. Sauer, and D. Lalanne, "Introducing the RECOLA Multimodal Corpus of Remote Collaborative and Affective Interactions," in *Proceedings of the 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. Shanghai, China: IEEE, 2013, pp. 1–8.

[270] F. Ringeval, F. Eyben, E. Kroupi, A. Yuce, J.-P. Thiran, T. Ebrahimi, D. Lalanne, and B. Schuller, "Prediction of Asynchronous Dimensional Emotion Ratings from Audiovisual and Physiological Data," *Pattern Recognition Letters*, vol. 66, pp. 22–30, 2015.

[271] F. Ringeval, B. Schuller, M. Valstar, S. Jaiswal, E. Marchi, D. Lalanne, R. Cowie, and M. Pantic, "AV$^+$ EC 2015: The First Affect Recognition Challenge Bridging Across Audio, Video, and Physiological Data," in *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*. Brisbane, Australia: ACM, 2015, pp. 3–8.

[272] R. Rojas, *Neural Networks - A Systematic Introduction*. Berlin, Germany: Springer, 2016.

[273] F. Rosenblatt, "Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms," Cornell Aeronautical Lab Inc., Buffalo, NY, USA, Tech. Rep., 1961.

[274] N. Rubens, D. Kaplan, and M. Sugiyama, *Active Learning in Recommender Systems*. Boston, MA, USA: Springer, 2011, pp. 735–767.

[275] W. Ruch and P. Ekman, "The Expressive Pattern of Laughter," *Emotion, Qualia, and Consciousness*, pp. 426–443, 2001.

[276] R. V. Rullen and S. J. Thorpe, "Rate Coding Versus Temporal Order Coding: What the Retinal Ganglion Cells Tell the Visual Cortex," *Neural Computation*, vol. 13, no. 6, pp. 1255–1283, 2001.

[277] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[278] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., ser. Series in Artificial Intelligence. Essex, UK: Pearson Education Limited, 2014.

[279] T. N. Sainath and B. Li, "Modeling Time-Frequency Patterns with LSTM vs. Convolutional Architectures for LVCSR Tasks," in *Proceedings of the 17th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. San Francisco, CA, USA: ISCA, 2016, pp. 813–817.

[280] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks," in *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, Australia: IEEE, 2015, pp. 4580–4584.

[281] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the Speech Front-End with Raw Waveform CLDNNs," in *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Dresden, Germany: ISCA, 2015.

[282] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann Machines," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Clearwater Beach, FL, USA: JMRL, 2009, pp. 448–455.

[283] H. Salamin, A. Polychroniou, and A. Vinciarelli, "Automatic Detection of Laughter and Fillers in Spontaneous Mobile Phone Conversations," in *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Manchester, United Kingdom: IEEE, 2013, pp. 4282–4287.

[284] T. Salimans and D. P. Kingma, "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*. Barcelona, Spain: MIT Press, 2016, pp. 901–909.

[285] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks," *arXiv preprint arXiv:1312.6120*, 2013.

[286] K. R. Scherer, "Vocal Communication of Emotion: A Review of Research Paradigms," *Speech Communication*, vol. 40, pp. 227–256, 2003.

[287] R. Schlüter, I. Bezrukov, H. Wagner, and H. Ney, "Gammatone Features and Feature Combination for Large Vocabulary Speech Recognition," in *Proceedings*

of the 32nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Honolulu, HI, USA: IEEE, 2007, pp. 649–652.

[288] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[289] B. Schuller, "Intelligent Audio Analysis – Speech, Music, and Sound Recognition in Real-Life Conditions," Habilitation Thesis, Technische Universität München, Munich, Germany, 2012.

[290] B. Schuller, F. Eyben, and G. Rigoll, "Static and Dynamic Modelling for the Recognition of Non-Verbal Vocalisations in Conversational Speech," in *Perception in Multimodal Dialogue Systems: 4th IEEE Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems*, ser. Lecture Notes on Computer Science (LNCS). Heidelberg: Springer, 2008, pp. 99–110.

[291] B. Schuller, S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, A. Burkhardt, R. van Son, F. Weninger, F. Eyben, T. Bocklet, G. Mohammadi, and B. Weiss, "The INTERSPEECH 2012 Speaker Trait Challenge," in *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Portland, OR, USA: ISCA, 2012.

[292] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, F. Chetouani, M. Weninger, F. Eyben, E. Marchi, M. Mortillaro, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The INTERSPEECH 2013 Computational Paralinguistics Challenge: Social Signals, Conflict, Emotion, Autism," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013, pp. 148–152.

[293] B. Schuller and A. Batliner, *Computational Paralinguistics: Emotion, Affect and Personality in Speech and Language Processing*. Chichester, UK: John Wiley & Sons, 2014.

[294] B. Schuller, S. Steidl, and A. Batliner, "The INTERSPEECH 2009 Emotion Challenge," in *Proceedings of the 10th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Brighton, UK: ISCA, 2009.

[295] B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. Müller, and S. S. Narayanan, "The INTERSPEECH 2010 Paralinguistic Challenge," in *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Makuhari, Japan: ISCA, 2010.

[296] B. Schuller, S. Steidl, A. Batliner, F. Schiel, and J. Krajewski, "The INTERSPEECH 2011 Speaker State Challenge," in *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Florence, Italy: ISCA, 2011.

[297] B. Schuller, S. Steidl, A. Batliner, J. Epps, F. Eyben, F. Ringeval, E. Marchi, and Y. Zhang, "The INTERSPEECH 2014 Computational Paralinguistics Challenge: Cognitive & Physical Load," in *Proceedings of the 15th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Singapore, 2014.

[298] B. Schuller, S. Steidl, A. Batliner, S. Hantke, F. Hönig, J. R. Orozco-Arroyave, E. Nöth, Y. Zhang, and F. Weninger, "The INTERSPEECH 2015 Computational Paralinguistics Challenge: Nativeness, Parkinsons's & Eating Condition," in *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Dresden, Germany: ISCA, 2015.

[299] B. Schuller, S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, F. Burkhardt, R. van Son, F. Weninger, F. Eyben, T. Bocklet, G. Mohammadi, and B. Weiss, "A Survey on Perceived Speaker Traits: Personality, Likability, Pathology, and the First Challenge," *Computer Speech & Language*, vol. 29, no. 1, pp. 100–131, 2015.

[300] B. Schuller, S. Steidl, A. Batliner, E. Bergelson, J. Krajewski, C. Janott, A. Amatuni, M. Casillas, A. Seidl, M. Soderstrom *et al.*, "The INTERSPEECH 2016 Computational Paralinguistics Challenge: Addressee, Cold & Snoring," in *Proceedings of the 18th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Stockholm, Sweden: ISCA, 2017, pp. 3442–3446.

[301] B. Schuller, F. Weninger, Y. Zhang, F. Ringeval, A. Batliner, S. Steidl, F. Eyben, E. Marchi, A. Vinciarelli, K. Scherer, M. Chetouani, and M. Mortillaro, "Affective and Behavioural Computing: Lessons Learnt from the First Computational Paralinguistics Challenge," *Computer Speech & Language*, vol. 53, pp. 156–180, 2019.

[302] B. W. Schuller, "Big Data, Deep Learning–At the Edge of X-Ray Speaker Analysis," in *Proceedings of the 19th International Conference on Speech and Computer (SPECOM)*, ser. Speech and Computer. Hatfield, Herfordshire, UK: Springer, 2017, pp. 20–34.

[303] B. W. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. MüLler, and S. Narayanan, "Paralinguistics in Speech and Language—State-Of-The-Art

and the Challenge," *Computer Speech & Language*, vol. 27, no. 1, pp. 4–39, 2013.

[304] B. W. Schuller, S. Steidl, A. Batliner, J. Hirschberg, J. K. Burgoon, A. Baird, A. C. Elkins, Y. Zhang, E. Coutinho, and K. Evanini, "The INTERSPEECH 2016 Computational Paralinguistics Challenge: Deception, Sincerity & Native Language," in *Proceedings of the 17th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, vol. 2016. San Francisco, CA, USA: ISCA, 2016, pp. 2001–2005.

[305] B. W. Schuller, S. Steidl, A. Batliner, P. B. Marschik, H. Baumeister, F. Dong, S. Hantke, F. Pokorny, E.-M. Rathner, K. D. Bartl-Pokorny, C. Einspieler, D. Zhang, A. Baird, S. Amiriparian, K. Qian, Z. Ren, M. Schmitt, P. Tzirakis, and S. Zafeiriou, "The INTERSPEECH 2018 Computational Paralinguistics Challenge: Atypical & Self-Assessed Affect, Crying & Heart Beats," in *Proceedings of the 19th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Hyderabad, India: ISCA, 2018, pp. 122–126.

[306] B. W. Schuller, A. Batliner, C. Bergler, F. B. Pokorny, J. Krajewski, M. Cychosz, R. Vollmann, S.-D. Roelen, S. Schnieder, E. Bergelson, A. Cristia, A. Seidl, A. Warlaumont, L. Yankowitz, E. Nöth, S. Amiriparian, S. Hantke, and M. Schmitt, "The INTERSPEECH 2019 Computational Paralinguistics Challenge: Styrian Dialects, Continuous Sleepiness, Baby Sounds & Orca Activity," in *Proceedings of the 20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Graz, Austria: ISCA, 2019.

[307] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A Survey of Neuromorphic Computing and Neural Networks in Hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[308] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[309] F. Seide, "Keynote: The Computer Science Behind the Microsoft Cognitive Toolkit: An Open Source Large-Scale Deep Learning Toolkit for Windows and Linux," in *IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. Austin, TX, USA: IEEE, 2017.

[310] S. Semeniuta, A. Severyn, and E. Barth, "Recurrent Dropout Without Memory Loss," *arXiv preprint arXiv:1603.05118*, 2016.

[311] B. Settles, "Active Learning Literature Survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[312] S. Sigtia, E. Benetos, and S. Dixon, "An End-to-end Neural Network for Polyphonic Piano Music Transcription," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 5, pp. 927–939, 2016. [Online]. Available: https://doi.org/10.1109/TASLP.2016.2533858

[313] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[314] J. O. Smith, *Mathematics of the Discrete Fourier Transform (DFT): With Audio Applications*. Julius Smith, 2007.

[315] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*. Santa Rosa, CA, USA: IEEE, 2017, pp. 464–472.

[316] P. Smolenski, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Rumelhart, David E. and McClelland, James L. and the PDP Research Group, 1986, vol. 1, no. 194–281, ch. Information Processing in Dynamical Systems: Foundations of Harmony Theory, p. 87.

[317] M. E. Sobel, *The Analysis of Contingency Tables*. Boston, MA, USA: Springer, 1995, pp. 251–310.

[318] J. Sola and J. Sevilla, "Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems," *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, 1997.

[319] W. Song and J. Cai, "End-to-End Deep Neural Network for Automatic Speech Recognition," Stanford University, Tech. Rep. CS224D, 2015.

[320] K. Spaho, "Organizational Communication and Conflict Management," *Management: Journal of Contemporary Management Issues*, vol. 18, no. 1, pp. 103–118, 2013.

[321] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks From Overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[322] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training Very Deep Networks," in *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada: MIT Press, 2015, pp. 2377–2385.

[323] ——, "Highway Networks," *arXiv preprint arXiv:1505.00387*, 2015.

[324] T. J. Steichen and N. J. Cox, "A Note on the Concordance Correlation Coefficient," *The Stata Journal*, vol. 2, no. 2, pp. 183–189, 2002.

[325] C. F. Stevens and A. Zador, "Neural Coding: The Enigma of the Brain," *Current Biology*, vol. 5, no. 12, pp. 1370–1371, 1995.

[326] S. S. Stevens, J. Volkmann, and E. B. Newman, "A Scale for the Measurement of the Psychological Magnitude Pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[327] A. Stevenson, Ed., *Oxford Dictionary of English*, 3rd ed. Oxford, UK: Oxford University Press, 2010.

[328] E. Strangert and J. Gustafson, "What Makes a Good Speaker? Subject Ratings, Acoustic Measurements and Perceptual Evaluations," in *Proceedings of the 9th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Brisbane, Australia: ISCA, 2008.

[329] E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, "An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data," *Frontiers in neuroscience*, vol. 11, p. 350, 2017.

[330] Y. Sun, D. Willett, R. Brueckner, R. Gruhn, and D. Bühler, "Experiments on Chinese Speech Recognition with Tonal Models and Pitch Estimation using the Mandarin Speecon Data," in *Proceedings of the 9th International Conference on Spoken Language Processing (INTERSPEECH)*. Pittsburgh, PA, USA: ISCA, 2006.

[331] I. Sutskever, "Training Recurrent Neural Networks," Ph.D. dissertation, University of Toronto, Toronto, Canada, 2013.

[332] I. Sutskever and T. Tieleman, "On the Convergence Properties of Contrastive Divergence," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9. Chia Laguna Resort, Sardinia, Italy: JMRL, 2010, pp. 789–795.

[333] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the Importance of Initialization and Momentum in Deep Learning," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 30, no. 1, Atlanta, GA, USA, 2013, pp. 1139–1147.

[334] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2011.

[335] J. A. Swets, *Signal Detection Theory and ROC Analysis in Psychology and Diagnostics: Collected Papers*. Psychology Press, 2014.

[336] Z. S. Syed, J. Schroeter, K. Sidorov, and D. Marshall, "Computational Paralinguistics: Automatic Assessment of Emotions, Mood, and Behavioural State from Acoustics of Speech," in *Proceedings of the 19th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Hyderabad, India: ISCA, 2018, pp. 511–515.

[337] A. K. Syrdal, A. Conkie, and Y. Stylianou, "Exploration of Acoustic Correlates in Speaker Selection for Concatenative Synthesis," in *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP)*. Sydney, Australia: ISCA, 1998.

[338] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the 28th Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, 2015, pp. 1–9.

[339] H. Tanaka and N. Campbell, "Acoustic Features of Four Types of Laughter in Natural Conversational Speech," in *Proceedings of the 17th International Congress of Phonetic Sciences (ICPhS)*. Hong Kong, China: IPA, 2011, pp. 1958–1961.

[340] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[341] T. Tieleman and G. Hinton, "Lecture 6.5-Rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[342] K. M. Ting, *Precision and Recall*. Boston, MA, USA: Springer, 2010, pp. 781–781.

[343] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a Next-Generation Open Source Framework for Deep Learning," in *Proceedings of the Workshop on Machine Learning Systems (LearningSys) in the 29h Annual Conference on Neural Information Processing Systems (NIPS)*, vol. 5. Montreal, Canada: MIT Press, 2015, pp. 1–6.

[344] L. Tóth, "Phone Recognition with Deep Sparse Rectifier Neural Networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada: IEEE, 2013, pp. 6985–6989.

[345] G. L. Trager, "Paralanguage: A First Approximation," *Studies in Linguistics*, vol. 13, pp. 1–11, 1958.

[346] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu Features? End-to-End Speech Emotion Recognition using a Deep Convolutional Recurrent Network," in *Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China: IEEE, 2016, pp. 5200–5204.

[347] P. Tzirakis, G. Trigeorgis, M. A. Nicolaou, B. W. Schuller, and S. P. Zafeiriou, "End-to-End Multimodal Emotion Recognition Using Deep Neural Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, pp. 1301–1309, 2017.

[348] J. Vettin and D. Todt, "Laughter in Conversation: Features of Occurrence and Acoustic Structure," *Journal of Nonverbal Behavior*, vol. 28, no. 2, pp. 93–115, 2004.

[349] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, New York, NY, USA, 2008, pp. 1096–1103.

[350] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[351] A. Vinciarelli, A. Dielmann, S. Favre, and H. Salamin, "Canal9: A database of Political Debates for Analysis of Social Interactions," in *Proceedings of the 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII)*, 2009, pp. 1–4.

[352] A. Vinciarelli, M. Pantic, and H. Bourlard, "Social Signal Processing: Survey of an Emerging Domain," *Image and Vision Computing*, vol. 27, no. 12, pp. 1743–1759, 2009.

[353] A. Vinciarelli, S. Kim, F. Valente, and H. Salamin, "Collecting Data for Socially Intelligent Surveillance and Monitoring Approaches: The Case of Conflict in Competitive Conversations," in *Proceedings of the 5th International Symposium on Communications, Control and Signal Processing (ISCCSP)*. IEEE, 2012, pp. 1–4.

[354] A. Vinciarelli, M. Pantic, D. Heylen, C. Pelachaud, I. Poggi, M. Schröder, and et al., "Bridging the Gap Between Social Animal and Unsocial Machine: A Survey of Social Signal Processing," *IEEE Transactions on Affective Computing*, vol. 3, no. 1, pp. 69–87, 2012.

[355] R. J. Volkema and R. J. R. H. Gorman, "The Influence of Cognitive-Based Group Composition on Decision-Making Process and Outcome," *Journal of Management Studies*, vol. 35, no. 1, pp. 105–121, 1998.

[356] J. Wagner, F. Lingenfelser, and E. André, "Using Phonetic Patterns for Detecting Social Cues in Natural Conversations," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Lyon, France: ISCA, 2013, pp. 168–172.

[357] J. A. Wall Jr. and R. R. Callister, "Conflict and its Management," *Journal of Management*, vol. 21, no. 3, pp. 515–558, 1995.

[358] B. Weiss and F. Burkhardt, "Voice attributes affecting likability perception," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[359] F. Weninger, F. Eyben, B. W. Schuller, M. Mortillaro, and K. R. Scherer, "On the Acoustics of Emotion in Audio: What Speech, Music, and Sound Have in Common," *Frontiers in Psychology*, vol. 4, 2013.

[360] F. Weninger, J. Bergmann, and B. Schuller, "Introducing CURRENNT: The Munich Open-Source CUDA RecurREnt Neural Network Toolkit," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 547–551, 2015.

[361] F. J. Weninger, *Intelligent Single-Channel Methods for Multi-Source Audio Analysis*, 2015.

[362] P. J. Werbos, "Backpropagation Through Time: What It Does and How to Do It," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[363] M. Wöllmer, F. Weninger, F. Eyben, and B. Schuller, "Acoustic-Linguistic Recognition of Interest in Speech with Bottleneck-BLSTM Nets," in *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Florence, Italy: ISCA, 2011, pp. 77–80.

[364] M. Wöllmer, Y. Sun, F. Eyben, and B. Schuller, "Long Short-Term Memory Networks for Noise Robust Speech Recognition," in *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Makuhari, Chiba, Japan: ISCA, 2010, pp. 2966–2969.

[365] S. Woźniak, A. Pantazi, and E. Eleftheriou, "Deep Networks Incorporating Spiking Neural Dynamics," *arXiv preprint arXiv:1812.07040*, 2018.

[366] B. Wrede and E. Shriberg, "Spotting 'Hot Spots' in Meetings: Human Judgments and Prosodic Cues," in *Proceedings of the 8th European Conference on*

Speech Communication and Technology (EUROSPEECH). Geneva, Switzerland: ISCA, 2003, pp. 2805–2808.

[367] X. Xie, H. Qu, Z. Yi, and J. Kurths, "Efficient Training of Supervised Spiking Neural Network via Accurate Synaptic-Efficiency Adjustment Method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1411–1424, 2017.

[368] J. Y. F. Yam and T. W. S. Chow, "A Weight Initialization Method for Improving Training Speed in Feedforward Neural Network," *Neurocomputing*, vol. 30, no. 1-4, pp. 219–232, 2000.

[369] K. Yamamoto, F. Asano, T. Yamada, and N. Kitawaki, "Detection of Overlapping Speech in Meetings Using Support Vector Machines and Support Vector Regression," *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, vol. 89-A, no. 8, pp. 2158–2165, 2006.

[370] B. Yang, L. Wang, D. F. Wong, L. S. Chao, and Z. Tu, "Convolutional Self-Attention Network," *arXiv preprint arXiv:1805.08318*, vol. abs/1805.08318, 2018.

[371] Y. Yao, L. Rosasco, and A. Caponnetto, "On Early Stopping in Gradient Descent Learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[372] S. H. Yella and H. Bourlard, "Overlapping Speech Detection Using Long-Term Conversational Features for Speaker Diarization in Meeting Room Conversations," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1688–1700, 2014.

[373] D. Yu and L. Deng, *Automatic Speech Recognition - A Deep Learning Approach*. Springer, 2016.

[374] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[375] D. Zambrano, R. Nusselder, H. S. Scholte, and S. Bohte, "Efficient Computation in Adaptive Artificial Spiking Neural Networks," *arXiv preprint arXiv:1710.04838*, 2017.

[376] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[377] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *arXiv preprint arXiv:1212.5701*, 2012.

[378] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Proceedings of the 13th European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science. Zurich, Switzerland: Springer, 2014, pp. 818–833.

[379] M. Zelenák and J. Hernando, "The Detection of Overlapping Speech with Prosodic Features for Speaker Diarization," in *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Florence, Italy: ISCA, 2011, pp. 1041–1044.

[380] Y. Zhang, "Machine Learning Techniques for Holistic Computational Paralinguistics," Ph.D. dissertation, Imperial College London, 2018.

[381] Z. Zhang, "Semi-Autonomous Data Enrichment and Optimisation for Intelligent Speech Analysis," Ph.D. dissertation, Technische Universität München, 2015.

[382] Z. Zhang and B. Schuller, "Semi-Supervised Learning Helps in Sound Event Classification," in *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan: IEEE, 2012, pp. 333–336.

[383] Z. Zhang, F. Ringeval, B. Dong, E. Coutinho, E. Marchi, and B. Schuller, "Enhanced Semi-Supervised Learning for Multimodal Emotion Recognition," in *Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China: IEEE, 2016, pp. 5185–5189.

[384] Z. Zhang, J. Han, J. Deng, X. Xu, F. Ringeval, and B. Schuller, "Leveraging Unlabeled Data for Emotion Recognition With Enhanced Collaborative Semi-Supervised Learning," *IEEE Access*, vol. 6, pp. 22 196–22 209, 2018.

[385] J. Zhao, X. Mao, and L. Chen, "Speech Emotion Recognition using Deep 1D & 2D CNN LSTM Networks," *Biomedical Signal Processing and Control*, vol. 47, pp. 312–323, 2019.

[386] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

[387] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber, "Recurrent Highway Networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 4189–4198.

[388] H. G. Zimmermann, R. Grothmann, A. M. Schaefer, and C. Tietz, "Identification and Forecasting of Large Dynamical Systems by Dynamical Consistent Neural Networks," in *New Directions in Statistical Signal Processing: From*

Systems to Brains, S. Haykin, J. C. Principe, T. J. Sejnowski, and J. McWhirter, Eds. Cambridge, MA, USA: MIT Press, 2006, pp. 203–242.

[389] B. Ziółko and M. Ziółko, "Time Durations of Phonemes in Polish Language for Speech and Speaker Recognition," in *Proceedings of the 4th Language and Technology Conference (LTC)*. Poznań, Poland: Springer, 2009, pp. 105–114.

[390] M. Zuckerman, H. Hodgins, and K. Miyake, "The Vocal Attractiveness Stereotype: Replication and Elaboration," *Journal of Nonverbal Behavior*, vol. 14, no. 2, pp. 97–112, 1990.

[391] E. Zwicker, *Psychoakustik*. Springer, 1982.

[392] E. Zwicker and H. Fastl, "Loudness," in *Psychoacoustics*. Springer, 1999, pp. 203–238.

[393] ——, *Psychoacoustics: Facts and Models*. Springer Science & Business Media, 2013, vol. 22.