Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

DOCTORAL THESIS

On Representation Learning in Speech Processing and Automatic Speech Recognition

Benjamin Milde

An der Universität Hamburg eingereichte Dissertation
zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr.rer.nat.).

Fakultät für Mathematik, Informatik und Naturwissenschaften, Fachbereich Informatik.

2022

Examiners:

1. Prof. Dr. Chris Biemann
2. Prof. Dr.-Ing. Timo Gerkmann
3. Prof. Dr.-Ing. Florian Metze

Date of the doctoral examination: 27.10.2022

Declaration of Authorship

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

I herewith declare on oath that I have written this dissertation thesis myself and have not used any sources and aids other than those indicated.

Signed:

Date:

"We can only see a short distance ahead, but we can see plenty that needs to be done."

- Alan Turing

Abstract

On Representation Learning in Speech Processing and Automatic Speech Recognition

Speech processing is a difficult task for computers, owing to many factors of variance present in any speech signal. In Automatic Speech Recognition (ASR), a person's voice, the environment and how and where speech sounds are recorded can drastically alter the appearance of a speech signal without changing the content of what is being said. Meanwhile, humans deal seemingly effortlessly with these factors of variance in understanding spoken language.

A central question in automatic speech processing is how and what representations to use, to facilitate further processing and to apply machine learning methods to automate speech processing tasks. A focus in this thesis is on learning models and representations from speech data itself. Artificial neural networks have recently reemerged as an important ingredient of acoustic and language modelling and have produced promising results and error reductions over previous methods. They are now a widespread tool in learning good and robust representations for speech signals in ASR and are also typically used in language modelling. After an introduction to speech processing in Chapter 1, this thesis provides an overview of common (deep) neural network techniques and models in Chapter 2. An introduction to speech processing and ASR is given in Chapter 3.

In Chapter 4, a study on transfer learning is conducted on an isolated paralinguistic speech task, namely eating condition recognition. With the system presented in this chapter, we also participated in a paralinguistic speech challenge. The model was pre-trained on a language identification task and transfer learning was successfully used for the target task with little training data.

In Chapter 5 of this thesis, we propose Unspeech context embeddings. Unspeech models are trained on unannotated speech data using contrastive learning, with Siamese convolutional neural networks. The model is built on the idea that speech sounds that are close in time share the same contexts. The model can be trained on vast amounts of unannotated data, as evidenced by training it on up to 10,000 hours of speech data. We evaluate the model and its embeddings in automatic speech recognition tasks and several other downstream tasks, such as short command recognition, emotion recognition and speaker recognition.

In Chapter 6, we propose Sparsespeech, a neural model for discrete acoustic unit discovery in unannotated speech data. In the proposed model, we aim to represent speech as discretized units together with a context embedding. Unspeech embeddings can be used as context embedding or an alternative implicit context vector. The model is evaluated with the ABX error measure on English read speech data, using the largescale ASR benchmark Libri-light for ASR learning scenarios that are either unsupervised or with limited supervision.

Multitask learning for grapheme-to-phoneme (G2P) conversion is proposed in Chapter 7, where a neural G2P model based on the sequence-to-sequence architecture is trained on multiple languages (English and German). An error evaluation reveals that irregularly pronounced words, such as English loan words, are often wrongly predicted by the model. In Chapter 8, a recipe to train TDNN-HMM ASR models for German ASR on 1,700 hours of freely available audio data from different sources is presented. The models are evaluated on the Tuda-De test set, with the best model yielding strong WER results that beat previously published end-to-end systems. It is also evaluated on the Verbmobil test set, showing that good results can also be obtained for conversational speech.

Real world use is demonstrated by applying the model to automatic subtitle generation, where additional tasks such as subtitle segmentation and punctuation reconstruction need to be considered. Insights gained from Chapter 7 are used to extend a freely available lexicon to add English loan words as well as current German vocabulary to the lexicon. The words are added semi-manually, by predicting multiple possible pronunciations and by using an active learning approach with Text-To-Speech (TTS) feedback. In another demo application, the feasibility of online decoding in a speech application that transcribes meetings on-the-fly is demonstrated with the application being able to summarize meetings as well.

In Chapter 9, another speech application is presented that ambiently researches relevant information, in the form of proposing relevant documents to users who passively listen to spoken language. In contrast to other personal assistants, this system is not specifically triggered, as it unobtrusively listens to speech streams in the background and implicitly queries an index of documents.

In Chapter 10, the thesis concludes that representation learning is key to a wide range of speech processing tasks and that alternative machine learning paradigms, other than purely supervised ones promise to harvest untapped potential.

Zusammenfassung

Die Verarbeitung der gesprochenen Sprache stellt Computer vor besondere Herausforderungen, da einzelne Sprachsignale eine hohe Varianz aufzeigen können. Bei der automatischen Spracherkennung (engl. Automatic Speech Recognition, ASR) können die Stimme einer Person, die Umgebung und die Art und Weise, wie und wo Sprache aufgezeichnet wird, das Erscheinungsbild eines Sprachsignals drastisch beeinflussen, ohne den Inhalt des Gesagten zu verändern. Gleichzeitig kommt der Mensch mit diesen Faktoren beim Verstehen gesprochener Sprache scheinbar mühelos zurecht.

Eine zentrale Frage bei der automatischen Sprachverarbeitung ist daher, wie und welche Repräsentationen verwendet werden sollen, um die weitere Verarbeitung zu erleichtern und um Methoden des maschinellen Lernens zur Automatisierung der Sprachverarbeitung einzusetzen. Ein Schwerpunkt dieser Arbeit ist das Lernen von Modellen und Repräsentationen aus den Sprachdaten selbst. Künstliche neuronale Netze sind inzwischen ein wichtiger Bestandteil der akustischen und sprachlichen Modellierung und haben vielversprechende Ergebnisse und Fehlerreduzierungen gegenüber früheren Methoden erbracht. Sie sind heute ein weit verbreitetes Werkzeug zum Erlernen guter und robuster Repräsentationen für Sprachsignale und werden auch typischerweise in der Sprachmodellierung eingesetzt. Nach einer Einführung in die Sprachverarbeitung in Kapitel 1 gibt diese Arbeit in Kapitel 2 einen Überblick über gängige (tiefe) neuronale Netzwerktechniken und -modelle. Eine Einführung in die Sprachverarbeitung und ASR wird in Kapitel 3 gegeben.

In Kapitel 4 wird eine Studie zum Transferlernen an einer paralinguistischen Sprachanwendung, der Erkennung von Schmatzgeräuschen beim Reden und gleichzeitigen Essen, durchgeführt. Mit dem vorgeschlagenen System wurde auch an einem paralinguistischen Wettbewerb teilgenommen. Das Modell wurde mit einer Spracherkennungsaufgabe vortrainiert und das Transferlernen wurde erfolgreich für die Zielaufgabe mit wenigen Trainingsdaten eingesetzt.

In Kapitel 5 dieser Arbeit schlagen wir Unspeech-Kontexteinbettungen vor. Unspeech-Modelle werden auf nichtannotierten Sprachdaten mit kontrastivem Lernen und siamesischen neuronalen Netzen trainiert. Das Modell basiert auf der Idee, dass zeitlich nahe beieinander liegende Sprachlaute denselben Kontext teilen. Das Modell kann auf riesigen Mengen nichtannotierter Daten trainiert werden, was durch das Training mit bis zu 10.000 Stunden Sprachdaten bewiesen wurde. Wir evaluieren das Modell und seine Einbettungen mit einem ASR-System und verschiedenen anderen nachgelagerten Aufgaben, wie z.B. der Erkennung kurzer Befehle, sowie der Erkennung von Emotionen in der gesprochenen Sprache und der Sprechererkennung.

In Kapitel 6 wird Sparsespeech vorgeschlagen, ein neuronales Modell zur Erkennung diskreter akustischer Einheiten in nichtannotierten Sprachdaten. Das vorgeschlagene Modell zielt darauf ab, Sprache als diskrete Einheiten zusammen mit einer

Kontexteinbettung darzustellen. Nichtsprachliche Einbettungen können als Kontexteinbettung oder als alternativer impliziter Kontextvektor verwendet werden. Das Modell wird mit dem ABX-Fehlermaß für englische gelesene Sprachdaten evaluiert, wobei der große ASR-Benchmark Libri-light für ASR-Lernszenarien verwendet wird, die entweder unbeaufsichtigt oder mit begrenzter Überwachung durchgeführt werden.

Multitasking-Lernen für die Graphem-Phonem-Konvertierung (G2P) wird in Kapitel 7 vorgeschlagen, bei dem ein neuronales G2P-Modell, das auf der Sequenz-zu-Sequenz-Architektur basiert, auf mehreren Sprachen (Englisch und Deutsch) trainiert wird. Eine Fehlerauswertung zeigt, dass unregelmäßig ausgesprochene Wörter, wie z.B. Anglizismen, vom Modell oft falsch vorhergesagt werden. In Kapitel 8 wird ein Rezept zum Trainieren von TDNN-HMM ASR-Modellen für deutsche ASR auf 1.700 Stunden frei verfügbarer Audiodaten aus verschiedenen Quellen vorgestellt. Die Modelle werden auf dem Tuda-De Testset evaluiert, wobei das beste Modell sehr gute Ergebnisse bei der Wortfehlerrate liefert, die zuvor veröffentlichte End-to-End-Systeme übertreffen. Es wird auch mit dem Verbmobil-Testsatz evaluiert, was zeigt, dass auch bei Konversationssprache gute Ergebnisse erzielt werden können.

Die Anwendung des Modells auf die automatische Generierung von Untertiteln, bei der zusätzliche Aufgaben wie die Segmentierung von Untertiteln und die Rekonstruktion der Satzzeichen berücksichtigt werden müssen, verdeutlicht die praktische Anwendung. Die in Kapitel 7 gewonnenen Erkenntnisse werden genutzt, um ein frei verfügbares Lexikon um Anglizismen sowie um aktuellen deutschen Wortschatz zu erweitern. Die Wörter werden halbmanuell hinzugefügt, indem mehrere mögliche Aussprachen vorhergesagt werden und ein aktiver Lernansatz mit Sprachsynthese-Feedback verwendet wird. In einer weiteren Demoanwendung wird die Machbarkeit der Online-Dekodierung in einer Sprachanwendung demonstriert, die Meetings transkribiert, wobei die Anwendung auch in der Lage ist, Meetings zusammenzufassen.

In Kapitel 9 wird eine weitere Sprachanwendung vorgestellt, die in Konversationen in der Umgebung relevante Informationen recherchiert und relevante Dokumente dem Nutzer vorschlägt. Das System hört dem Sprachgeschehen passiv zu. Im Gegensatz zu anderen persönlichen Assistenten wird dieses System nicht gezielt ausgelöst, da es unauffällig im Hintergrund den Sprachströmen lauscht und implizit einen Dokumentenindex abfragt.

In Kapitel 10 kommt die Arbeit zu dem Schluss, dass Repräsentationslernen für eine Vielzahl von Sprachverarbeitungsaufgaben von zentraler Bedeutung ist und dass alternative maschinelle Lernparadigmen, die nicht rein überwacht sind, ungenutztes Potenzial versprechen.

Acknowledgements

Firstly, I would like to thank Prof. Dr. Chris Biemann for his excellent supervision and guidance. You are an exceptional mentor! To all my colleagues from the LT group at Universität Hamburg: I have fond memories of our time together and there was always some one there for discussing research ideas over a cup of coffee. Thank you all!

I also thank Prof. Dr. Chris Biemann, Prof. Dr.-Ing. Timo Gerkmann and Prof. Dr.-Ing. Florian Metze for reviewing my thesis and the suggestions of small editorial changes for this final version. I would like to thank Steffen Remus, Eugen Ruppert, Dr. Martin Riedl and Dr. Jinseok Nam for insightful discussions and comments on the topic of transfer learning in neural networks (Chapter 4). I would like to thank Prof. Dr. Timo Baumann for discussions on models I developed as well as other ASR-related topics. Thank you Tim Fischer and Steffen Remus for collaborating on MoM bot (Chapter 8). Furthermore, I would like to thank Dr. Arne Köhn and Prof. Dr. Timo Baumann for collaborating on the Spoken Wikipedia Corpus SWC integration into the German Kaldi ASR recipe (also Chapter 8). I also like to thank anyone who contributed to the uhh-lt/kaldi-tuda-de, german-asr-lm-tools and subtitle2go open source projects. I thank Michael Henretty and Mozilla for giving me access to Common Voice V1 speaker information (for Chapter 5).

My PhD thesis would not have been possible without funding. Again special thanks to Prof. Dr. Chris Biemann, for funding of the initial exploratory phases of this thesis at TU-Darmstadt as well as bridging any gaps between projects. I am also grateful to Telecooperation Lab at TU-Darmstadt, in particular Prof. Dr. Max Mühlhäuser and BMBF for funding research on ASR with Ambient Search. I would also like to thank Dr. Joachim Köhler and Dr. Chistoph Schmidt for making my research visit at Fraunhofer IAIS in St. Augustin possible and supervising me on different aspects of German broadcast ASR. I thank Universität Hamburg for funding my seminars and for giving me the opportunity to teach. I thank Deutsche Telekom AG for financially supporting the development of MoM bot. Also my grateful thank you to the Lecture2go team, particularly Martin Kriszat, for funding a project on automatic subtitling. It was really rewarding to see part of my work over the years being deployed to a production system and having it assist to make lectures at Universität Hamburg more accessible through subtitles.

I am grateful for the support of students and research assistants (HiWis). In particular Alexander Hendrich, as well as Jonas Wacker, Tim Fischer, Robert Geislinger, with whom I co-authored papers as well. I thank Sarah Milde and Sebastian Gonsior for proofreading. I profoundly thank Michelle Sandbrink, for always having my back and for never complaining in stressful periods or cancelled weekend plans due to paper deadlines. Furthermore, for also proofreading this thesis as well as helping to add countless German lexicon entries with the speech-lex-edit software and also with helping out with the annotation of relevance judgements in Chapter 9.

I have dedicated this thesis to my parents, who shaped my academic career and supported me through studies in Darmstadt and have always supported me and followed my scientific endeavours. I am truly grateful.

Of course, I would also like to thank the countless anonymous reviewers for their time to review my paper submissions and the valuable feedback they provided. Finally, there is a good chance that I missed naming you. To anybody not named above, thank you for helping to improve or shape this work in one way or another, even if it was just for mental support.

Contents

List of Figures

List of Tables

List of Abbreviations

ASR	**A**utomatic **S**peech **R**ecognition
ARI	**A**djusted **R**and **I**ndex
ARPA	**A**dvanced **R**esearch **P**rojects **A**gency
BLAS	**B**asic **L**inear **A**lgebra **S**ubprograms
CMVN	**C**epstral **M**ean and **V**ariance **N**ormalization
CNN	**C**onvolutional **N**eural **N**etwork
CPC	**C**ontrastive **P**redictive **C**oding
CTC	**C**onnectionist **T**emporal **C**lassification
CV	**C**ross **V**alidation
DBSCAN	**D**ensity **B**ased **S**patial **C**lustering of **A**pplications with **N**oise
DNN	**D**eep **N**eural **N**etwork
DPGMM	**D**irichlet **P**rocess **G**aussian **M**ixture **M**odel
DTW	**D**ynamic **T**ime **W**arping
FBANK	**F**ilter **Bank**
FIR	**F**inite **I**mpulse **R**esponse
FNN	**F**eed-forward **N**eural **N**etwork
FST	**F**inite **S**tate **T**ransducer
G2P	**G**rapheme **to** **P**honeme
GMM	**G**aussian **M**ixture **M**odel
GPU	**G**raphics **P**rocessing **U**nit
GRU	**G**ated **R**ecurrent **U**nit
HTML	**H**yper**T**ext **M**arkup **L**anguage
HMM	**H**idden **M**arkov **M**odel
IPA	**I**nternational **P**honetic **A**lphabet
JSON	**J**ava**S**cript **O**bject **N**otation
KL	**K**ullback **L**eibler
LDA	**L**atent **D**irichlet **A**llocation
LM	**L**anguage **M**odel
LPC	**L**inear **P**redictive **C**oding
LSTM	**L**ong **S**hort-**T**erm **M**emory
LVCSR	**L**arge **V**ocabulary **C**ontinuous **S**peech **R**ecognition
MFCC	**M**el **F**requency **C**epstral **C**oefficients
MLE	**M**aximum **L**ikelihood **E**stimation
MMI	**M**aximum **M**utual **I**nformation
MSE	**M**ean **S**quared **E**rror

NLP	**N**atural **L**anguage **P**rocessing
NLU	**N**atural **L**anguage **U**nderstanding
NMI	**N**ormalized **M**utual **I**nformation
NN	**N**eural **N**etwork
OOV	**O**ut **O**f **V**ocabulary
PCA	**P**rincipal **C**omponent **A**nalysis
PER	**P**honeme **E**rror **R**ate
PLP	**P**erceptual **L**inear **P**redictive
RNN	**R**ecurrent **N**eural **N**etwork
SAMPA	**S**peech **A**ssessment **M**ethods **P**honetic **A**lphabet
SGD	**S**tochastic **G**radient **D**escent
STFT	**S**hort-time **F**ourier **t**ransform
TDNN	**T**ime-**D**elayed **N**eural **N**etwork
TF	**T**ensor **F**low
TSNE	**T**-distributed **S**tochastic **N**eighbor **E**mbedding
TTS	**T**ext **T**o **S**peech
VAD	**V**oice **A**ctivity **D**etection
VGG	**V**isual **G**eometry **G**roup
VQ	**V**ector **Q**uantization
VTLN	**V**ocal **T**ract **L**ength **N**ormalization
VTLP	**V**ocal **T**ract **L**ength **P**erturbation
WER	**W**ord **E**rror **R**ate
WFST	**W**eighted **F**inite **S**tate **T**ransducer

Dedicated to my parents.

Chapter 1

Introduction

The ability to understand and transcribe spoken words is an elemental skill for humans and has been an active research field in computer science for the past decades. Even before the day we are born, we are surrounded by spoken language (Choi et al., 2017). It is one of the most important ways to communicate for us. And seemingly, understanding spoken words is easy for most people: we are effortlessly able to listen and understand our mother tongue on a daily basis. We quickly forget that this ability is shaped by years of training. Considering an average language development, we utter and understand our first 3-50 words when we are 18 months old (Lenneberg, 1967). What follows is a remarkable explosion of learning and language acquisition. Usually, after three years the speaking vocabulary has reached over a thousand words and a basic understanding of a languages' fundamental structure and grammar, e.g. forming questions, negations, past tenses, plurals is acquired. By age four, the essentials of one's native tongue are mastered (Lenneberg, 1967).

Learning a foreign languages later in life serves as a reminder of the difficulty and challenge associated with understanding spoken words. It takes a great deal of effort to learn to listen to it, let alone to learn to understand and speak it. In most cases, we will have to deal with a new inventory of the language: its vocabulary and grammar and in most cases even new phonetic sounds.

Teaching machines to understand and transcribe our spoken language is likewise faced with several difficulties. Today's computers record speech digitally as discretized samples at discretized time steps. A typical sampling rate for human speech in automatic speech processing is 16kHz, i.e. 16000 measurements per second of the oscillation of a microphone's membrane, measuring pressure differentials of the sound wave propagating through the air. In Figure 1.1, we show the oscillation of an example recording, from an utterance taken from the Common Voice corpus (release 1, train_valid: sample-055799.mp3), with the microphone's measurements normalized to values between -1.0 and 1.0.

A typical conversion of this one-dimensional signal is a decomposition into its frequency components, usually accomplished by a fast Fourier transformation (Cooley and Tukey, 1965). The result is a frequency domain representation in two dimensions that is easier to interpret for both machines and humans. This representation is frequently used to visualize speech and it is commonly referred to as the spectogram

of an utterance. Figure 1.2 shows such a spectogram of the same utterance.

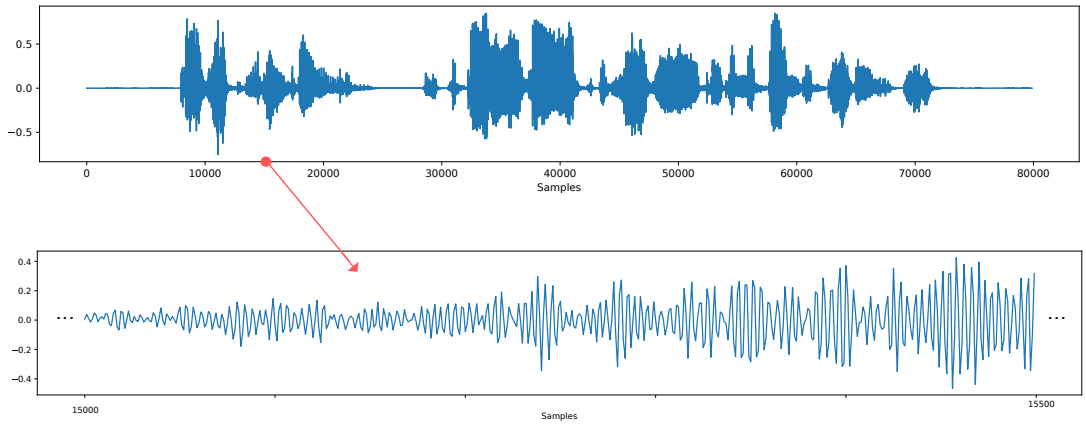

FIGURE 1.1: Waveform of the utterance ”dusk was falling as the boy arrived with his herd at an abandoned church” from the Common Voice V1 corpus (train_valid: sample-055799.mp3).

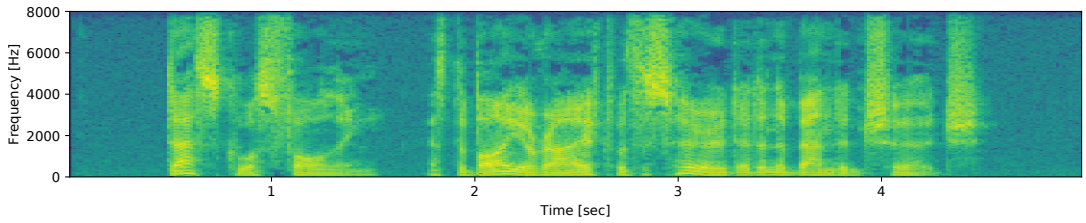

FIGURE 1.2: Corresponding spectrogram to the utterance ”dusk was falling as the boy arrived with his herd at an abandoned church”.

Spectograms of speech show a few unexpected occurrences – it is difficult to tell where individual words start and begin, even though we seem to perceive clear "separations" between words in our minds when we listen to speech. Recorded speech signals also show an enormous spectrum of variability, even though they could convey literally the same message. Many factors contribute to that variability, like characteristics of the speaker: e.g. gender, age and accent, but also the environment, how speech is recorded (what microphone is being used) and where speech is recorded. A central question in the field of automatic speech recognition is thus, how should speech be represented in order to aid its automatic recognition? Is there a transformation that could capture the space of variability, and ideally reduce it for further processing? Or should a speech recognition model be able to just learn to deal with this variability?

Over the past decades, a set of standard representations for representing speech for automatic speech processing has been established, most importantly filter bank (FBANK) representations and Mel-frequency cepstral coefficients (MFCC). These 2D representations are biologically inspired by the human ear, particularly FBANK features are also very similar to spectograms of speech. These common representations are discussed in more detail in Chapter 3.3.

However, over the past few years, the landscape of automatic speech processing has dramatically changed: characteristic of state-of-the-art models is the use of (deep) artificial neural networks that enabled dramatic error reductions in speech processing and recognition (see Section 2.7 for an introduction to artificial neural networks). Representations, or rather transformations over representations are individually trained for each network – within the network itself. This additional modelling capacity and an intrinsic capability to learn to deal with variability (instead of expecting it to be not present in the input representation) is what is commonly believed to be behind the recent rapid progress in reducing transcription errors for difficult large vocabulary speech recognition tasks.

Also, training acoustic models does now usually involve computations on powerful graphic processor units (GPUs), that evolved from being computing devices specific to computer graphics to general purpose accelerators. Through this heritage, processing capabilities of GPUs are ideal for vast parallelism and floating point operations. They allow to significantly speed up the matrix multiplications needed to optimize large and deep neural networks for speech processing and have been a contributing factor to their success.

New ways of modelling the temporal aspects of speech are also explored, challenging the Hidden Markov Model (HMM) framework (see Section 3.4 for a brief introduction to HMMs). End-to-end learning casts the problem of transcribing speech into a pure end-to-end machine learning problem, without any programmed signal processing steps. In some of these systems, no linguistic assumptions are made and there is no predefined set of linguistic building blocks of speech such as phonemes. This is a stark deviation from the status quo in automatic speech processing.

1.1 Challenges in Automatic Speech Processing and Speech Recognition

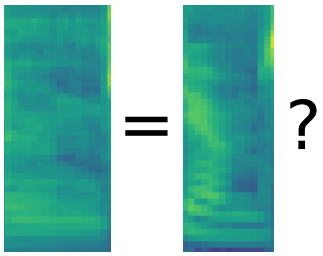

FIGURE 1.3: FBANK representations (similar to a spectogram) of two phonemes.

Figure 1.3 shows an example of a local phoneme comparision. It is quite difficult to tell from the two local spectograms if the same phoneme is being uttered or different ones. In Figure 1.4 we show that one of FBANK spectrograms is a recording of a woman taken from TIMIT (Garofolo et al., 1993), the other FBANK spectrum is for the same utterance, but spoken by a man. Recognizing the phoneme locally

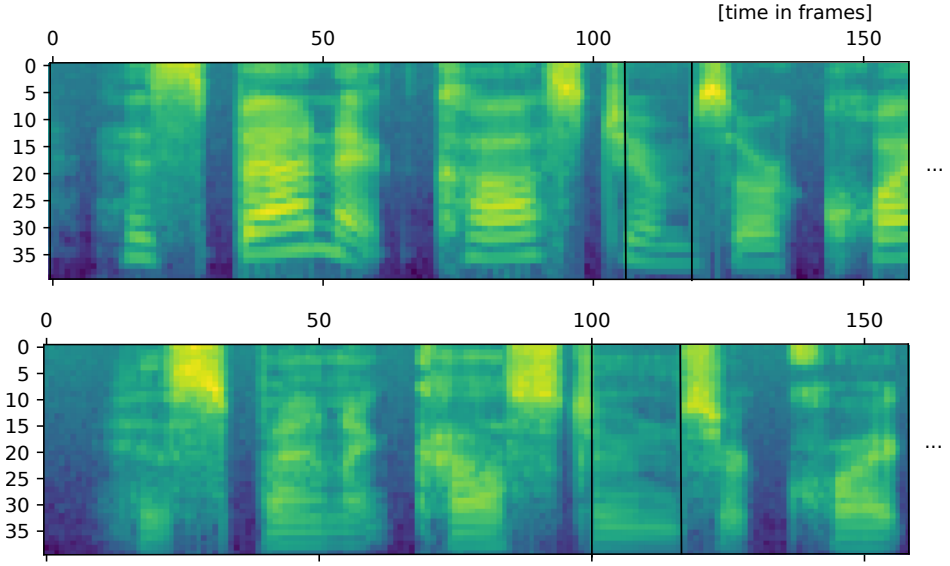

FIGURE 1.4: Local comparison of the same phoneme uttered by different speakers, together with its surrounding context. Example in the style of Jansen et al. (2013).

is difficult, as its (spectrogram) representations can differ drastically and contain a lot of variance. However, with the help of surrounding context, their similarity becomes much more apparent. In the remainder of the thesis, we describe such factors of variance also as the context of an utterance; speech sounds that occur close in time share similar contexts.

Recent acoustic models for automatic speech recognition (ASR) incorporate some form of (deep) neural network that can learn to deal with part of this variance by using supervised training data and the ability to learn representations as part of the model. It can also make sense to explicitly model some of these factors as dense speaker embeddings, so that these representations can also be used to compare similarity of speakers (Saon et al., 2013).

In the following sections we discuss the largest sources of variability in recorded speech signals. Making automatic speech systems robust and invariant towards these sources of variability is a fundamental problem.

1.1.1 Speaker Variability

Speaker variability is one of the biggest contributing factors of variance in speech signals. Physical shape and size of the vocal tract and nasal cavities directly results in large variations in speech characteristics (Mullennix et al., 1989). This also means that acoustic properties of individual phonemes change from speaker to speaker. The resulting speaker variability can be most substantially observed in vowels (Peterson and Barney, 1952).

Joos (1948) noted that vowels are most likely not perceived in isolation, but that they are perceived in the context of other vowels from the same speaker. Listeners

can compensate differences in speakers. There is a degree of habituation for familiar voices (e.g. family, friends) and these are more intelligible than voices of new persons (Nygaard and Pisoni, 1998). There is also further information, that gives a listener clues about a speakers' gender, age, cultural background and identity.

1.1.2 Pronunciation Variability

In a statistical analysis of speech variability, the second most important factor besides speaker variability is accent (Huang et al., 2001). It is also well known that foreign and accented speech is degrading speech recognition performance in a system trained on native speech (Huang et al., 2004). Accent deviation at the acoustic level and pronunciation errors are the two biggest sources of variation (Bouselmi et al., 2008). Note that humans of course face the same kind of variability in understanding speech, with unfamiliar accents being also problematic to understand (Mattys et al., 2012).

1.1.3 Background and Channel Variability

Background noises are usually present when speech is recorded in real-life scenarios. They are also usually undesirable in speech processing applications. Ambient noise, cars and other machine noise for instance may be overlayed in the signal. But even without any additive noise sources, variabilty due to room acoustics might be present. For example, reverberation from surface reflections in a room will also change and influence the recorded signal as well as spectral shaping from recording equipment itself (Acero and Stern, 1990). Another large source of variability of speech recordings is the distance to a microphone. Particular speech recorded at a distance makes automatic recognition more difficult (Wölfel and McDonough, 2009) and degrades recording quality.

1.1.4 Extra-Linguistic Variability

A person can influence his voice to be more quiet, even whisper, or to be louder, more tense or softer (Benzeghiba et al., 2007) or might speak at a faster or slower pace than usual. All these factors might again change the acoustic signal drastically, but do not change the transcription of an utterance. These changes in a recorded speech signal are known as extra-linguistic variability and can also depend on context, e.g. a speaker countering a particularly noisy background environment.

1.1.5 In Summary

To summarize, there are many highly non-linear interactions in spoken language and recorded speech signals. This makes static pattern matching and methods that do not account for variability prone to errors and challenging to build in a robust

manner. It is, in most cases, also difficult to anticipate all forms of variability programmatically. This makes a strong case for machine learning algorithms and representations that are learned from data, where robustness to speech variability has to be learned from the speech data itself.

1.2 Overview and research directions of this thesis

Humans and computers learn spoken language differently. Computers are usually "exposed" to parallel corpora of speech data, i.e. isolated utterances of speech with matching transcriptions, to learn to transcribe speech. Humans learn speech and language simultaneously, from direct to indirect supervision and feedback through imitation (Whitehurst and Vasta, 1975). Most strikingly, humans also learn from being exposed to language, and there is mounting evidence that infants learn the statistical distributions of speech through ambient language (Kuhl, 2004). In dataset terms, this calls for a mixture of different learning paradigms, such as supervised learning, unsupervised learning, semi-supervised learning, transfer learning and multi-task learning. This raises the following questions:

What can we do if there is little training data for a speech task? Models can be pre-trained on auxiliary tasks before they are applied to a target task with little labeled data. Chapter 4 explores this data scenario on a study on an isolated paralinguistic speech task with neural networks. An auxiliary task, in this case language identification, has lots of training examples that can be easily obtained in large quantities and the target paralinguistic speech task has a few hundert labeled examples. Traditionally, neural networks are known to overfit on such a small labeled data set, but recent regularization techniques such as dropout and the use of transfer learning counter this effectively. The speech representation learned on the auxiliary task proves to be useful for the target task as well.

What can we do with unlabeled speech data in automatic speech processing? Transcribing speech data to automatic speech processing systems is a tedious manual task. The 2004 Fisher effort to collect 2,000 hours of conversational English speech estimated 6 hours to 20 hours of manual labor per collected hour of speech (Cieri et al., 2004). The effort is dependent on the quality of the transcriptions. To collect 2,000 hours of high-quality transcriptions, up to 40,000 hours of manual labor might be necessary. On the other hand, unlabeled speech data can be collected with ease – vast amounts of audio data can be readily collected from the internet, as demonstrated in later chapters of this thesis (see e.g. Chapter 5) with dataset sizes of up to 10,000 hours.

Experiments with the proposed models *Unspeech* and *Sparsespeech* in Chapter 5 and Chapter 6, which are both trained on untranscribed audio data, target this research question. While *Unspeech* targets speaker and background characteristics,

Sparsespeech targets unsupervised acoustic unit discovery and unsupervised acoustic modeling. Furthermore, Sparsespeech uses the "feedback through imitation" paradigm to learn representations of speech, as Sparsespeech is a generative model as well.

What other parts of the ASR pipeline can benefit from alternatives to supervised learning? In Chapter 7, we use multi-task learning to learn an embedding representation with sequence encoder/decoder networks that can be used to predict pronunciations of unknown words (graphemes). Similarly to transcribing speech data, manual dictionary compilation is a tedious process as well. Grapheme-to-phoneme (G2P) models are an important tool for automatic dictionary expansion in ASR systems and also helpful for assisting semi-manual dictionary compilation as well. G2P is also used in current unsupervised ASR systems, to convert texts into sequences of phonetic units.

Finally, how do practical systems benefit from speech technology? In the final chapters of this thesis, application of speech at the intersection of NLP and ASR are explored. In Chapter 8 a system for German automatic subtitling is presented – an otherwise tedious manual task for human transcribers. In fully automatic subtitling, NLP tasks such as punctuation reconstruction and subtitle segmentation need to to be considered as well. A live subtitling application, *Meetingbot*, is presented as well that can transcribe meeting participants and can create an automatic summary, with on-the-fly speech decoding and visualization.

In Chapter 9, a speech application dubbed *Ambient Search* is presented. It is a system that ambiently researches relevant information, in the form of proposing relevant documents to users in conversations or users who passively listen to spoken language. In contrast to other personal assistants, this system is not specifically triggered, as it unobtrusively listens to speech streams in the background and implicitly queries an index of documents.

1.2.1 Thesis overview

Figure 1.5 provides an overview of the chapters of this thesis. The thesis can be roughly divided into three parts. The first part covers the introduction and two background chapters. Chapter 2 provides background information regarding machine learning, clustering and (deep) neural networks. Chapter 3 provides an introduction to automatic speech recognition.

In the second part, speech processing models are proposed and evaluated that are related to (mostly unsupervised) ASR. A model for speech embeddings is built in Chapter 5, a neural model for discrete units in Chapter 6 and a multi task model for G2P in Chapter 7 . Chapter 5, 6, 7 are models that are prerequisites to unsupervised ASR (see also Figure 3.11).

In the third part, practical speech systems at the intersection of ASR and NLP are proposed and evaluated. An ASR model for German speech recognition is developed and applied to automatic lecture subtitling. The applications also make use of

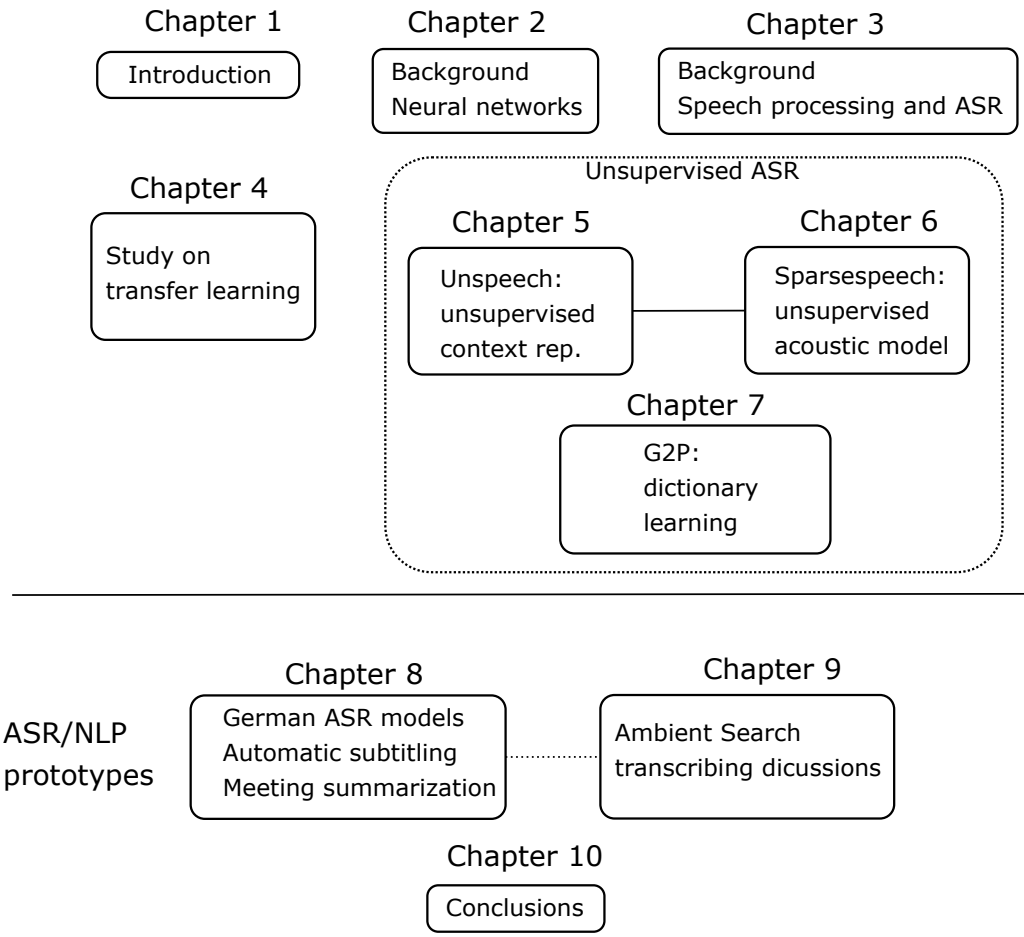

FIGURE 1.5: Overview of research directions and chapters in this thesis.

an active learning approach with G2P to extend the ASR system's lexicon as needed. A second speech application gives an example of an unobtrusive personal assistant that ambiently fetches relevant articles while passively listing to a discussion or lecture, for instance.

1.2.2 Publications forming the basis of this thesis

Parts of this thesis are based on previously published and co-authored peer-reviewed content. In the following, we list all contributing peer-reviewed publications, along with details to where and when they were published and comments to the degree of authorship if applicable.

1. Benjamin Milde and Chris Biemann (2015). "Using Representation Learning and Out-of-domain Data for a Paralinguistic Speech Task". In: Proceedings of Interspeech 2015. Dresden, Germany, pp. 904–908. (*)

2. Benjamin Milde, Jonas Wacker, Stefan Radomski, Max Mühlhäuser, and Chris Biemann (2016). "Demonstrating ambient search: Implicit document retrieval

for speech streams". In Proceedings of COLING 2016 (system demonstrations), Osaka, Japan, pp. 233–237.

3. Benjamin Milde, Jonas Wacker, Stefan Radomski, Max Mühlhäuser, and Chris Biemann (2016). "Ambient search: A document retrieval system for speech streams". In Proceedings of COLING 2016, Osaka, Japan, pp. 2082–2091.

4. Benjamin Milde, Christoph Schmidt, and Joachim Köhler (2017). "Multitask sequence-to- sequence models for grapheme-to-phoneme conversion." In: Proceedings of Interspeech 2017, Stockholm, Sweden, pp. 2536–2540.

5. Benjamin Milde and Arne Köhn (2018). "Open Source Automatic Speech Recognition for German". In: Proceedings of 13th ITG Conference on Speech Communication 2018, Oldenburg, pp. 251–255.

6. Benjamin Milde and Chris Biemann (2018). "Unspeech: Unsupervised Speech Context Embeddings". In: Proceedings of Interspeech 2018. Hyderabad, India, pp. 2693–2697.

7. Benjamin Milde and Chris Biemann (2019). "SparseSpeech: Unsupervised Acoustic Unit Discovery with Memory-Augmented Sequence Autoencoders". In: Proceedings of Interspeech 2019. Graz, Austria, pp. 256–260.

8. Benjamin Milde and Chris Biemann (2020). "Improving Unsupervised Sparsespeech Acoustic Models with Categorical Reparameterization". In: Proceedings of Interspeech 2020. Virtual Shanghai, China, pp. 2747–2751.

9. Benjamin Milde, Robert Geislinger, Irina Lindt, and Timo Baumann (2021). "Open Source Automatic Lecture Subtitling". In: Proceedings of ESSV 2021. Virtual Berlin, Germany, pp. 128–134.

10. Benjamin Milde, Tim Fischer, Steffen Remus, and Chris Biemann (2021). "MoM: Minutes of Meeting Bot". In: Proceedings of Interspeech 2021. Brno, Czech Republic, pp. 3311–3312.

11. Robert Geislinger, Benjamin Milde, Timo Baumann and Chris Biemann (2021). "Live Subtitling for BigBlueButton with Open-Source Software". In: Proceedings of Interspeech 2021. Brno, Czech Republic, pp. 3319–3320.

(*) Nominated for the ISCA best student paper award at Interspeech 2015.

Comments on the degree of authorship. In publications (1,6,7) and (8), I designed and carried out all experiments and wrote the paper, while Chris Biemann commented on the research and writing in his role as supervisor. In (1) Steffen Remus, Martin Riedl and Jinseok Nam were acknowledged for insightful discussions on the topic of transfer learning. In (2,3) I did the majority of the experiments, while Jonas Wacker helped with the implementation of the demo system, specifically the

keyword/keyphrase extraction. Stefan Radomski, Max Mühlhäuser, and Chris Biemann commented on the research and writing. Alexander Hendrich was acknowleged for contributing to improve the HTML5/JS display client and Michelle Sandbrink for helping out with the relevance judgements of the retrieved documents. In (4) I carried out all experiments and wrote the paper, while Christoph Schmidt and Joachim Köhler commented on the research and writing. In (5) I wrote the majority of the paper and did the experiments, while Arne Köhn gave me access to the Spoken Wikipedia Corpus (SWC) that he specifically filtered for ASR model training with Kaldi, also writing an exporter for the Kaldi corpus format. Arne Köhn wrote Section 2.1 about the SWC corpus in this paper and helped with the error analysis in Section 3.1. In (9) I wrote most of the paper and carried out most experiments, while Robert Geislinger and Irina Lindt helped with implementing the Software Subtitle2Go. Robert Geislinger evaluated Punctuator models for German texts to restore punctuation and wrote the section about it. Timo Baumann commented on the research in his role of supervising the Subtitle2Go project and helped to improve the writing style of the paper. In (10) I spearheaded the paper and programmed most of the backend modules. Tim Fischer implemented the frontend as well as the summarization and partially wrote the section on the interface and summarization. Steffen Remus wrote most of related works section and helped with containerizing Meetingbot for deployment. In (11), Robert Geislinger spearheaded the paper and carried out experiments and the evaluation, while I helped with kaldi-model-server and trouble shooting decoding problems, as well as improving and adding to the paper overall.

Relation to chapters of this thesis. Most chapters of this thesis are based on the listed publications. Specifically, Chapter 4 is based on (1), Chapter 5 is based on (6), while Chapter 6 is based on (7,8). Chapter 7 is based on (4) and Chapter 8 is based on (5,9,10) and includes a small text snippet and screenshot of (11). Chapter 9 is based on (2,3).

Chapter 2

Machine Learning and Neural Networks

In a procedural computer program, a programmer defines sequences of command statements. These statements govern how a program should react to a given input and how it should compute and output results of a computation. The programmer has to anticipate all possible scenarios and abstract logical rules that cover them with a procedural solution.

For some problems, determining such a procedural solution manually becomes a difficult task. Problems from the field of natural language processing and speech processing tend to fit this profile. Let us consider the following NLP problem: we want to determine the sentiment of a sentence. We could determine a set of procedural statements that lead to making decision A (positive) or decision B (negative), for example create lists of positive words and negative words. Then we count the number of positive and negative words and compute the sentiment according to what class of words was counted more often. We then look at example sentences and see how well this solves the problem. For example, we might discover that negations are not handled well with the current solution and create additional procedural rules to handle this problem. In a way, this is also executing a type of learning. Only that in this case, the adaption of the program happens manually by the programmer. In contrast, the field of machine learning deals with a fundamentally different class of programs:

> "The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience" (Mitchell, 1997).

Programs in machine learning usually interact with models. The program itself is a set of rules that define how the model is constructed, how it should be updated and adapted given some data and eventually (in most cases) how it is applied to new data. This also means a shift in the way the programmer approaches the problem. Instead, he does not have to anticipate every possible future scenario by creating fixed and static program rules. He is now concerned with questions such as: What is a good model for the given problem? How is this model trained and improved

with data? What is a good representation of the data, i.e. how is the data processed and presented to the model? How is the data and potentially its labeling created?

In speech processing and automatic speech recognition, solutions that rely on machine learning have produced the best results so far. For example, if we want to solve the sentiment problem with recordings of speech instead of texts: There is more to spoken utterances, than just the contents of words – if want to classify the sentiment of a spoken utterance the way how something is said will also play a role. But what exactly constitutes a friendly and positive tone in an utterance and how would the procedural statements look like?

There is simply no easy way to anticipate and account for the vast amount of variations in speaker voices, background noises and so on that a simple procedural solution could be easily formulated. A learning algorithm that discovers patterns in the speech data automatically and learns from examples makes the problem approachable. As our understanding of learning algorithms and the field of machine learning expands, as well as the amount of data that is used to train and learn from, it will undoubtedly continue to play a major role in computer science and fundamentally shape our experiences interacting with machines.

In the following, we categorize the main types of learning scenarios in machine learning:

2.1 Supervised Learning

In supervised learning, we are interested in computing a target label y for a data point x. In the context of a probabilistic model, we estimate y as $\mathbb{P}(y|x)$. We typically have training data, a collection of labeled data points, that we use to fit parameters of a model. All data points are annotated using a finite set of target labels. Ideally, given enough training examples, the model can be used and applied to the target task on unseen and new examples and is able to generalize from what it learned. This usually involves a trade-off; either the model will memorize the training examples well but overfits on them leading to issues with generalization, or some of the training examples will be misclassified, but overall the model will generalize better to unseen examples.

In automatic speech recognition, the most common way to introduce supervision is by having matching transcriptions for speech data. In this case, no single labels y are available as training data: instead $\mathbf{w}$ is usually a sequence of words matching $\mathbf{o}$, a sequence of observations. We then estimate the conditional probability as $\mathbb{P}(\mathbf{w}|\mathbf{o}) = \mathbb{P}(w_1, ..., w_n|o_1, ..., o_m)$ (Jurafsky and Martin, 2008). In other words, while there is supervised data, the precise alignment of which o_i matches what w_i is unknown. Appropriate models are then used, such as Hidden Markov Models (see Section 3.4) that can jointly infer alignment and classification from the training data. Because of this difference, sometimes, this form of supervision is categorized in its own category as a transcription task (Goodfellow et al., 2016).

2.2 Unsupervised Learning

In practice, collecting large amounts of unlabeled data is much easier than collecting labelled data. Large amounts of video, audio and texts can be readily downloaded from the Internet, video and speech data can also be obtained by recording broadcast media. Annotated data on the other hand is usually obtained from human annotators and is costly to create. Making use of large amounts of unannotated data in machine learning is thus of high practical interest.

In unsupervised learning, we are concerned with training scenarios where no labels y are present. We are usually interested in $p(x)$ or the latent structure, discovering patterns in $p(x)$ (Murphy, 2014) or distribution and interesting properties of $p(x)$ (Goodfellow et al., 2016). A popular method to discover structure in data is clustering, where similar data points x are grouped. This can also be understood as a form of representation learning: after all data points are clustered, each data point can be represented by a single discrete label. Another form of unsupervised learning is unsupervised feature and representation learning. An example is the projection of data points into an embedded space where distances then explicitly exhibit latent properties of the data. Such models and the features they learn can also be incorporated into supervised tasks.

The lines between supervised and unsupervised learning can sometimes be blurry, as it is possible to reformulate the $p(x)$ problem as a supervised task by decomposing the modelling of $p(x)$ into n supervised problems (Goodfellow et al., 2016):

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_i - 1)$$

That also means that supervised machine learning machinery can be applied to unsupervised learning. An example for this can be found in Chapter 5 of this thesis, where a discriminative model is used with unannotated data for unsupervised learning in speech processing.

2.2.1 Clustering

Categorizing data into clusters to infer interesting properties of $p(x)$ is one of the major diciplines in unsupervised learning. In clustering, we assign a categorical label y to every data point x. Note that clustering can be an ill-posed problem (Jain, 2010). There are many potential ways to categorize data and each could be equally valid. Finding a good clustering with a clustering algorithm is a matter of perspective – this problem becomes really apparent when we cluster audio and speech data. There are many interesting ways to partition raw speech data and all are equally desirable in different scenarios. We could for instance cluster and partition utterances according to speakers, speaker characteristics, microphone and quality of the recordings. Alternatively, if we are concerned with semantic properties of the speech data and would like to cluster utterances according to topics. If we want to find clusters of

words, syllables or phonemes – we might require an additional segmentation of the available data points x. Each of these problems has very different desired clusters of data and will need different approaches and possibly different cluster algorithms. Another general issue is how fine-grained the clustering should be, i.e. how many clusters there should be. While heuristics exist, this is typically a parameter of the algorithm or approach that needs to be tuned in one way or another. Ultimately, what constitutes a good clustering of a collection of data points will strongly be dependent on the intended application.

Another form of clustering is soft clustering. Compared to a hard decision of assigning a single cluster to each data point, we assign a weight vector to each data point x, expressing the degree of cluster membership to every cluster. A soft clustering can always be transformed into a hard clustering by selecting the categorical label belonging to the largest degree of cluster membership for each data point.

2.2.2 K-Means

In the k-means problem, datapoints $x \in \mathcal{X} \subset R^d$ must be assigned to k clusters $c_1, ..., c_k$ so that it minimizes (notation follows Arthur and Vassilvitskii (2007)):

$$\phi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} ||x - c||^2$$

where we minimize the sum of all distances from all x to their respective cluster centers c. Solving this problem exactly is NP hard, but a simple and fast algorithm exists to solve it approximately, known as k-means or Lloyd's algorithm (Lloyd, 1982):

1. Choose k cluster centers $c_1, ..., c_k$ randomly from $\mathcal{X}$.
2. Assign all data points to the closest cluster center c_k. The cluster C_i is the set of points assigned to it.
3. Recalculate the cluster centers as $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.
4. Repeat step 2 and 3 until the cluster assignment converges.

K-means can give poor results and large variations in the resulting clusters, depending on the randomly selected initial cluster centers. The following cluster center initialization algorithm has become popular as it is simple and fast, yet effective in selecting better initial cluster centers than choosing them uniformly at random. Let $D(x)$ denote the shortest distance from a data point to the closest center of all previously chosen centers c_i (Arthur and Vassilvitskii, 2007):

1. Take one center c_1, chosen uniformly at random from $\mathcal{X}$.
2. Take a new center c_i, choosing $x \in \mathcal{X}$ with probability $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$.

3. Repeat Step 2 until there are k centers.

Step 2 weights new centers according to their squared distance to already chosen cluster centers, spreading the initial cluster centers across the dataset. The combined algorithm using regular k-means on the output of this cluster center initialization is known as k-means++ and it outperforms regular k-means in speed and accurarcy, often by a large margin (Arthur and Vassilvitskii, 2007). Thus, k-means++ is used throughout this thesis in all experiments that use k-means clustering, such as in the initialization technique in Section 6.1.2.

2.2.3 DBSCAN

DBSCAN (Density Based Spatial Clustering of Applications with Noise) (Ester et al., 1996) is a popular density-based clustering algorithm that clusters data points according to their differences in density. The motivation for this clustering algorithm is the observation that regions with a higher density of points lying close together in a space typically form clusters, while those clusters are usually separated by regions of lower point densities.

The algorithm also has a notion of data points that are outliers, i.e. data points that do not belong to any cluster. Also, an advantage of a density-based clustering algorithm is that it can cluster data of arbitrary shape. We sketch the DBSCAN algorithm in the following. DBSCAN uses the notion of points that are density-reachable as an important mechanism to decide cluster membership:

A point p is directly density-reachable from another point q if it lies in its eps-neighborhood, i.e. its distance is less or equal than the eps parameter and it has at least a certain amount of points, as set by the MinPts threshold. DBSCAN distinguishes between two types of points: Core points that lie in the eps-neighborhood with a MinPts at or above threshold and border points in the eps-neighborhood with a MinPts below threshold.

1. Start with an arbitrary unclassified point p.
2. Query its eps-neighborhood and assign the same cluster ID to all points if it is a core point.
3. Assign the noise label if p is a border point.
4. Recursively iterate through unclassified points of the eps-neighborhood.
5. Go to step 1 if there are still unclassified points.

Note that points that are assigned to the noise label can be assigned a cluster ID at a later stage, if they are in the eps-neighborhood of another core point. The region query (finding points in the eps-neighborhood of a point p) is also known as a nearest neighbor search. Computing a region query exactly is a computationally expensive operation due to the curse of dimensionality (Beyer et al., 1999). There are however

efficient indexing structures that allow approximate nearest neighbor searches, with a trade-off between computation time (construction of the index and search time) and accuracy of the query. These can also be used to significantly speed-up DBSCAN in practice, allowing very large datasets to be clustered. While DBSCAN can be used with any distance function, most approximate nearest neighbor search methods are designed for the Euclidean distance between points.

Classic and popular algorithms include k-d trees (Bentley, 1975) and ball trees (Beyer et al., 1999), addressing efficiency problems of k-d trees in higher dimensions. R*-tree (Beckmann et al., 1990) was recommended in the original DBSCAN paper description (Ester et al., 1996). Recent approximate nearest neighbor methods such as IVFADC (Jegou et al., 2010) are known to scale well into very large datasets containing billions of vectors (Johnson et al., 2019).

The eps parameter in DBSCAN also needs tuning in practice, but in some cases, a good eps parameter can also be chosen by knowledge of special properties of the data. Otherwise, it needs to be fine tuned and it can often be very sensitive: bad eps parameters might give degenerate solutions such as putting everything into one cluster (eps too large) or putting every data point in its own cluster (eps too small).

HDBSCAN (McInnes et al., 2017) is a state-of-the-art extension to DBSCAN, effectively converting DBSCAN into a hierarchical clustering algorithm. The advantage of HDBSCAN over DBSCAN is that it works on datasets with clusters of varying densities and does not need tuning of the eps parameter. When an appropriate approximate nearest neighbor search is used, it also scales well to large amounts of data points like DBSCAN does. We make use of this advantage in Section 5.4.2 to cluster speech data.

2.3 Self-supervised Learning

In self-supervised learning, the input data itself is used for supervision. An artificial supervised learning task is created, for example by generating pseudo labels through an automatic process. Self-supervised learning is regarded as a form of unsupervised learning (Jing and Tian, 2020), as no manual labels are needed to train a model. It can be categorized into generative and contrastive self-supervised learning (Liu et al., 2021). In generative self-supervised learning, parts of the input data are usually masked, distorted or corrupted and need to be inferred from other data points. In contrastive self-supervised learning, two or more data points are contrasted with each other, for example by a binary classification between data points that are somehow similar or share a common trait and randomly chosen ones. Usually, co-occuring inputs and related information are used in self-supervised learning tasks (Liu et al., 2021), so that the model needs to learn how data points relate to each other in order to solve the task. The models and its learned representation can then often be used to facilitate downstream tasks that are supervised, improving classification performance through the self-supervised pre-training. Applications of

self-supervised learning can be found in Chapter 5 (contrastive) and Chapter 6 (generative).

2.4 Semi-supervised Learning

Another type of supervised machine learning is semi-supervised learning, which deals with a learning problem where only some of the data is labeled and some of it is unlabeled. Typically, the number of unlabeled data points is much larger than the number of labeled points. Self-training is one of the most well-known and straightforward techniques in semi-supervised learning (Zhu, 2005). For this technique, a bootstrap model is trained on the labeled data and is then used to predict unlabeled data points. All or some of the predicted data points are then added to the training data and the model is retrained. Data point selection, e.g. only adding high confidence predictions to the training data, is common. Self-training has been applied to many tasks, for example in word sense disambiguation (Yarowsky, 1995) and learning subjective nouns (Riloff et al., 2003). In Veselỳ et al. (2013), this has been applied to speech recognition with neural networks, yielding small improvements.

A variant of self-training is label propagation (Zhu and Ghahramani, 2002). The method uses a combination of random walk and clamping to assign labels to unlabeled data points. Ladder networks (Rasmus et al., 2015) is another recent technique, where supervised learning and unsupervised learning is combined in a single (deep) neural network.

2.5 Transfer Learning

In a transfer learning setting, a model is first trained on an auxiliary task. The model is then fully or partially transferred and trained on a target task (Caruana, 1997; Pan and Yang, 2009). This technique is often applied in a setting where a related auxiliary task has a lot of training data while the target task has limited training data. An example in speech processing would be language identification of an utterance as an auxiliary task (lots of training data), with dialect identification as the target task (likely much less training data). Pretraining on an auxiliary task with more data might then improve generalization of the model on the target task. Some of what can be learned from the auxiliary task might also be useful for the target task.

2.6 Evaluation

In the following, we list common evaluation metrics that are used in supervised and semi-supervised learning tasks, as well as metrics commonly used in unsupervised learning. These metrics are also used in machine learning experiments in this thesis.

2.6.1 In Supervised Learning

Accuracy is a simple performance metric for supervised machine learning. It is the fraction of correct classifications c in all tested classifications n:

$$\text{Accuracy} = \frac{c}{n}$$

If the labels of a dataset are imbalanced, then the following evaluation metrics are usually more useful:

Precision and recall are two popular metrics that are rooted in information retrieval and are defined in the context of relevance of retrieved documents, as judged by users of an information retrieval system. They are also applicable to machine learning problems (Mitchell, 1997; Witten et al., 2016) by categorizing classifications of a system compared to true labels into true positives (TP), false positives (FP), true negative (TN) and false negatives (FN):

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Another common metric is the F-measure, defined as

$$F_\beta = \frac{(1 + \beta)PR}{\beta(P + R)}$$

where β is a parameter that weights precision and recall. The F_1 measure ($\beta = 1$) weights precision and recall equally and is equivalent to the harmonic mean of precision (P) and recall (R):

$$F_1 = \frac{2PR}{P + R}$$

Typically, performance metrics are evaluated on one or more held out sets. A common setup is a split of a dataset into training data, a development set and a test set. Overfitting and over estimation of a performance metric can still occur when the held out sets are repeatedly used for testing. Cross-validation (Stone, 1974) may produce a more robust estimate of the classifier's performance than a held out set: in k-fold cross validation, a dataset is split k times into k distinct sets. The model is then trained and evaluated k times, using $k - 1$ sets to train a model and the remaining set to test its performance, until every set is evaluated once. The final performance metric is then averaged over all k runs.

2.6.2 In Unsupervised Learning

In the following, we list common and well-established metrics to compare a clustering to a gold clustering, typically in the form of manual labels or categories. In contrast to supervised evaluation metrics, cluster evaluation metrics need to consider

that the number of clusters is different in the two sets that are compared. Cluster IDs may also be in a different order. It is desirable for the evaluation metrics that a mapping between the best matching cluster IDs does not need to be provided or found.

Rand Index (RI) The Rand Index (RI) (Rand, 1971) is a measure based on how all $\binom{n}{2}$ possible pairs between all elements of X and Y can be partitioned as being:

1. type (I): pairs of elements that are in the same cluster in X and Y
2. type (II): pairs of elements that are in distinct clusters in X and Y
3. type (III): pairs of elements that are in the same cluster in X and in distinct clusters in Y
4. type (IV): pairs of elements that are in distinct clusters in X and in the same cluster in Y.

These are all possible variants of a pair of elements in X and Y, i.e. the number of occurrences of type (I) + type (II) + type (III) + type (IV) = all possible pairs = $\binom{n}{2}$. Intuitively, pairs of type (I) and type (II) can be interpreted as agreeing on a clustering, while type (III) and (IV) are data points where the partitions X and Y differ on the clustering. Let a be the number of pairs of type (1) and b the number of pairs of type (II) between a partitioning X and Y, then the Rand Index is defined as:

$$RI(X,Y) = \frac{a+b}{\binom{n}{2}}$$

Adjusted Rand Index (ARI) Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) is adjusted for chance and cluster labels with a perfect match that would yield a score of 1.0. If $RI(X,Y)$ is defined as above as the Rand Index and $E(RI(X,Y))$ the expectation of random mappings, then:

$$ARI(X,Y) = \frac{\text{Index} - \text{Expected Index}}{\text{Maximum Index} - \text{Expected Index}} = \frac{RI(X,Y) - E(RI(X,Y))}{1 - E(RI(X,Y))}$$

Mutual Information (MI) Mutual information (MI) between discrete random variables X and Y is a well-known measure, emerging from the foundations in information theory (Shannon, 1948).

Let $H(X)$ be the entropy of X and $H(X|Y)$ the conditional entropy of X given Y (notation follows (Kvålseth, 2017)), then:

$$I(X;Y) = H(X) - H(X|Y)$$

Using common definitions of the entropy and conditional entropy, we can also formulate this as:

$$H(X) - H(X|Y) = -\sum_{i=1}^{I} p(x_i) log p(x_i) + \sum_{i=1}^{I}\sum_{j=1}^{J} p(x_i, y_i) log p(x_i|y_j) \tag{2.1}$$

$$= \sum_{i=1}^{I}\sum_{j=1}^{J} p(x_i, y_i) log \left(\frac{p(x_i|y_j)}{p(x_i)} \right) \tag{2.2}$$

where $p(x_i)$ and $p(y_i)$ are the marginal probabilities and $p(x_i, y_i)$ the joint probability.

Normalized Mutual Information (NMI) While MI can be used to compare to clusterings X and Y, it is unbound. Normalized Mutual Information (NMI) (Strehl, 2002) is a popular normalized variant of MI. It is generally between zero and one, with random clusterings approaching zero and an identical clustering one. Thus, NMI provides an interpretable score that can be directly applied to clustering and its reference clustering, to provide a score of relative goodness between the clustering and its reference.

$$NMI(X;Y) = \frac{2 \cdot I(X;Y)}{H(X) + H(Y)}$$

No assumptions about what clusters are matching needs to be made, the measure also works if the number of classes are not balanced between a clustering and a reference clustering. However, in the unbalanced case, NMI is always smaller than one (Strehl, 2002).

V-measure The popular V-measure (Rosenberg and Hirschberg, 2007) is calculated differently, but yields a measure that is identical to NMI, c.f. proof in (Remus and Biemann, 2013). We prefer to refer to the measure as NMI throughout the thesis.

2.7 Artificial Neural Networks

"It is probably wise to include a random element in a learning machine."

- Alan Turing, 1950 (Turing, 1950)

Artificial neural networks (ANNs) are machine learning models that are inspired by the human brain. While they work very differently as compared to neurons in a human brain, the abstraction shares the idea of individual connected nodes that pass on information through outputs to inputs of other nodes. Perception tasks, where the underlying inputs are difficult to interpret directly, have recently benefited from the use of large artificial neural networks (see also Section 2.14). This is in part attributed to representation learning that is implicitly a part of the model, enabling

it to learn hierarchical features that are beneficial to perception tasks (Farabet et al., 2013). Also, linear models tend to underfit and under-utilize computing resources (Lipton et al., 2015). Back-propagation (see Section 2.10) and Stochastic Gradient Descent (see Section 2.12) is commonly used to train artificial neural networks. The following sections give an overview over past and current models, as well as their training methods.

2.8 Feed Forward Networks

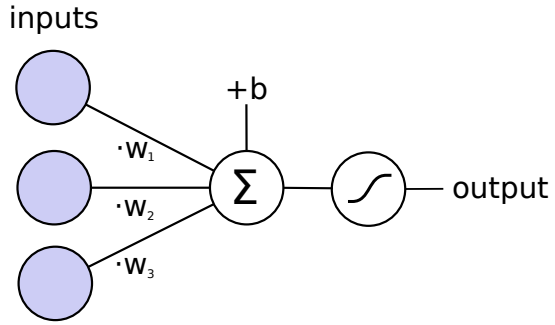

FIGURE 2.1: Example unit with three inputs in a Feed Forward Network.

Feed Forward Networks (FFN) are artificial neural networks, in which the interconnections of the network do not form any loops (Fiesler and Beale, 1996). FNNs are also known as multilayer perceptrons, mainly for historical reasons (Jurafsky and Martin, 2018). FNNs usually consist of units that are connected, Figure 2.1 depicts a single unit. It consists of a multiplication of a weight value with an input and a summation of all results (Rumelhart et al., 1986):

$$y = f_a(\sum_{i=1}^{n} w_i x_i + b)$$

where f_a is the activation function. A well known activation function is the sigmoid function σ (plotted in Figure 2.2). In machine learning, the sigmoid function usually refers to the logistic function (Mitchell, 1997):

$$\sigma = \frac{1}{1 + e^{-x}}$$

The output values of this function are bound between 0 and 1.

In the simplest form of a FNN, multiple of such units are organized in multiple layers, with connections between all units of consecutive layers (Rumelhart et al., 1986). A layer of this form is also called a fully connected layer. The computation of a fully connected layer can also be expressed compactly as a matrix multiplication and a vector addition:

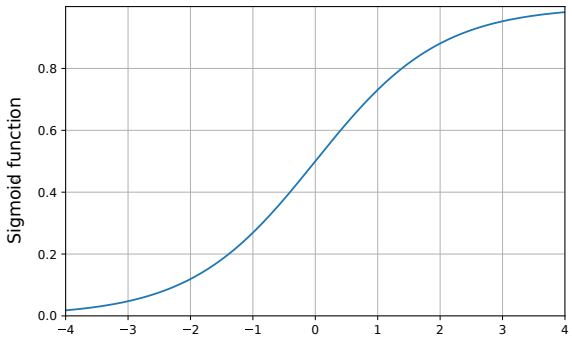

FIGURE 2.2: The sigmoid activation function

$$f_a(wx + b)$$

The activation function adds a non-linearity to the network that is important for stacking fully connected layers. Combining linear functions yields another linear function – the non-linearity however allows the network to learn and perform non-linear computations.

2.9 Loss Function

When training neural networks, we typically minimize an objective function. It is usually a loss function that associates a cost between the output of a neural network and the desired output for a particular data point. In the following, we list the most common loss functions:

Mean Squared Error (MSE) loss function:

$$MSE(Y, Y') = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2$$

where n is the number of outputs, Y the output vector of the network and Y' is the reference vector. MSE is often used in regression tasks, where the network should learn to output numeric values. A downside of MSE is that it gives a height weight to outliers. The Huber loss (Huber, 1964) is a piecewise function that can be used instead, giving less weight to outliers:

$$L_k(t) = \begin{cases} \frac{1}{2}t^2 & \text{for} \quad |t| < k \\ k|t| - \frac{1}{2}k^2 & \text{for} \quad |t| \geq k \end{cases}$$

where t refers to the difference between the output of the network and the reference value and k is a constant that can be set to control the influence of outliers. Figure 2.3 visualizes MSE and Huber loss. Huber loss is used in Chapter 6 and yielded better results than MSE on the self-supervised task in that Chapter.

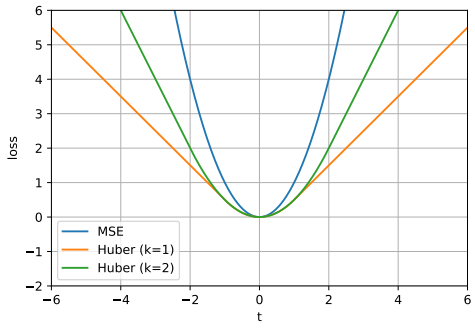

FIGURE 2.3: Mean squared error (MSE) and Huber loss

For classification problems, softmax (Bridle, 1990) is often applied to output of a network before the loss is computed. The softmax normalizes the network's output by squishing it so that all outputs sum up to one. The outputs $\sigma(x)_i$ can then be interpreted as pseudo-probabilities:

$$\sigma(x)_i = \frac{e_i^x}{\sum_{j=1}^{n} e_j^x}$$

The Kullback-Leibler divergence (Kullback and Leibler, 1951) defines a measure to compare two discrete probability functions P and Q. It is rooted in information theory:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

Cross entropy is a loss that is often used for classification in neural networks and it is closely related to Kullback-Leibler divergence:

$$H(P, Q) = -\sum_{x \in X} P(x) \log \left(Q(x) \right)$$

One of the first uses and mentions of cross entropy in neural networs appears in (Hinton et al., 1995). Cross entropy can be used in multi-label classification, the argmax of output vector yields the most probably classification according to the trained network for a data point.

$$H(P, y) = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log P_i) + (1 - y_i) \log (1 - P_i)$$

where y_i is the label for the i-th data point and P_i is the estimated (pseudo-)probability that the i-th label is positive (1).

Binary cross entropy is loss that is applied to classification tasks with one single label, which is either positive or negative. In this case, a single network output is trained to be in the range of 0 to 1. The decision boundary is then usually set to

0.5 (but can also be set to any other value) with values above the boundary being interpreted as a positive classification and values below as a negative classification.

2.10 Back-propagation

Back-propagation (backprop) is an algorithm to determine an analytical gradient function for a neural network. In a forward pass (also known as forward propagation) the output function of the network is computed. Back-propagation then determines a gradient function with respect to a given loss function and each weight of the network. While the term is often misused, back-propagation strictly refers to the computation of the gradient function (Goodfellow et al., 2016), while SGD (see 2.12) or a similar algorithm is used to optimize a neural network given a gradient function.

The first NN-specific application appeared in (Werbos, 1982) and the algorithm was popularized by Rumelhart et al. (1986). Back-propagation is also known as the reverse mode of automatic differentiation. See Griewank (2012) for an extended discussion on the origins of the reverse mode of automatic differentiation, some work in other fields is predating its first successful application to neural networks by a large margin (e.g. Hachtel et al. (1971) for using it in optimization of circuit design).

Back-propagation applies the chain rule of calculus recursively to obtain the gradient of a function composed of other functions whose gradients are known.

If $y = g(x)$ and $z = f(g(x)) = f(y)$, then the chain rule of calculus states (notation follows Goodfellow et al. (2016)):

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

This can be extended to vectors x and y with:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

For a more in depth introduction to the algorithm, see Goodfellow et al. (2016).

2.11 Gradient Descent

In Gradient Descent (GD), the weights w of a neural network are updated according to the following recursive function (Rumelhart et al., 1986; Bottou, 2010):

$$w_{t+1} = w_t \gamma \frac{1}{n} \sum_{i=1}^{n} \Delta_w l(f_w(x_i), y_i) \tag{2.3}$$

where γ is the learning rate, $f_w(x_i)$ the forward pass of the network for a data point x_i, l the loss function that quantifies how different $f_w(x_i)$ is to the desired

output y_i and $\Delta_w l$ is the gradient of the loss. In Gradient Descent, the update of the weights is calculated over the average of all data samples.

2.12 Stochastic Gradient Descent

In Stochastic Gradient Descent (SGD) (Robinds and Monro, 1951), the recursive update of the weights is modified to operate on single randomly drawn data points x_i:

$$w_{t+1} = w_t \gamma \Delta_w l(f_w(x_i), y_i) \tag{2.4}$$

Equation 2.4 simplifies and approximates the calculation of the gradient and weight updates can be calculated without passing through the whole dataset at once. However, when used to optimize neural networks, this also decreases the possibility for exploiting possible parallelism. In practice, mini-batch SGD is often used. This is a compromise between updating the weights with single data points and all data points at once, as it operates on a small number of randomly chosen data points those gradients are averaged (using Equation 2.3).

2.13 Vanishing Gradient Problem

The vanishing gradient problem can appear while training neural networks with gradient based training methods (Hochreiter, 1991; Bengio et al., 1994). For each weight in a neural network, a partial derivative is computed by back-propagation and the chain rule, chaining gradient computations from the loss function backwards through each layer of a network towards the inputs. In some network structures, particularly operations such as some activation functions those gradients are bound between 0 and 1, the gradient diminishes exponentially with each layer towards the input layer. In fact gradients of the front layers can than become so small that learning in those layers becomes impossible using standard floating-point arithmetic, as the weight changes are effectively zeroed out. While it is not an inherent problem of neural networks, gradient based optimization techniques are the most popular and usually the most efficient for training them, making successful training of neural networks more difficult as the number of operations and layers in a neural network increases (Goodfellow et al., 2016).

A related problem is the exploding gradient problem (Bengio et al., 1994) – typically encountered in chaining operations that grow the gradient computation exponentially towards the input layer instead of diminishing it, making computations with floating point arithmetic impossible once the numbers grow too large (and can only expressed as 'infinity').

Overcoming the vanishing and exploding gradient problem encountered with very deep neural structures is part of the success of recent advances in neural networks. The study of such deep neural networks is also known as deep learning (LeCun et al., 2015). Key methods and techniques are described in the following sections.

2.14 Deep Neural Networks

Deep Neural Networks (DNN) describe networks that are deep in their structure, e.g. have more than 3 layers. They are also characterized by training procedures and design decisions that permit training of the networks with many layers without encountering the vanishing gradient problem. Empirically, it has been demonstrated that deeper networks learn hierarchical features and that the network learns appropriate (non-linear) feature transformations for the problem (Farabet et al., 2013). Since representation learning is an integral part of deep neural networks, handcrafted features become optional for many problems – one appeal of DNNs is that they can often be applied directly to the underlying raw data.

In contrast to "shallow" standard feed forward networks (as described in Section 2.8), there are more changes than simply adding more layers to the network. They can be summarized as better initialization functions, a change of the non-linearities in the network, regularization and refined optimization algorithms. The following sections describe these solutions that were mainly developed in the past decade and help to train deep neural networks successfully. Today, the field concerned with training, understanding and applying deep neural networks is known as deep learning. Note that some early pioneering work on training such deep networks successfully predates the most recent literature by a wide margin (Fukushima and Miyake, 1982; LeCun and Bengio, 1995; Hochreiter and Schmidhuber, 1997). The following sections concentrate mainly on the advances of the past decade, but also mention some earlier work in a few places where appropriate.

2.14.1 Unsupervised pre-training

Hinton and Salakhutdinov (2006) proposed a practical solution to train deep neural networks. The idea is to split the training of the network into two phases: first, a good initialization is found by pre-training the network layer by layer using Restricted Boltzmann Machines (RBMs) (Rumelhart et al., 1987; Hinton, 2002) in an unsupervised fashion. In a second step, standard back propagation and SGD is applied to fine-tune the weights of the network.

The success of applying unsupervised pre-training in difficult supervised machine learning tasks, for example in allowing large reductions in word error rates in speech recognition (Dahl et al., 2012), helped to highlight an important realization:

"[...] the standard training schemes (based on random initialization) tend to place the parameters in regions of the parameters space that generalize poorly [...]" (Erhan et al., 2010)

In turn, this insight led to modern training recipes for DNNs and rekindled interest in (deep) neural networks. Nowadays, unsupervised pre-training of RBMs as originally proposed by Hinton et al. (2006) is rarely used anymore in supervised training tasks. It is largely superseded by employing better (static) initialization methods, advances in refining optimization algorithms, the choice of non-linearities and regularization methods that have shown fast convergence and better results in many practical tasks. But the idea of unsupervised pre-training ultimately led to key observations that facilitated practical solutions for optimizing deep neural networks.

2.14.2 Other types of non-linearities

In the following, we highlight a few non-linearities that are popular in modern deep neural networks and that typically replace the sigmoid activation function.

Rectifier / ReLU. The rectifier activation function is defined as:

$$f_r(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases}. \tag{2.5}$$

and is equivalent to $f_r(x) = max(x, 0)$.

Neurons (units) in a network that use the rectifier activation function are also called rectified linear units (ReLUs). This activation function was popularized by (Glorot et al., 2011). Currently, this is one of the most common choices for an activation function in dense and convolutional networks (see also some recent variants of ReLUs in the following sections). Using it instead of a sigmoid function has the advantage that it leads to exact sparsity in the internal representations of a network, since the activation function returns (exactly) zero for all negative inputs. In practice, with a uniform initialization around zero, over 50% of a network's ReLUs will be initialized to output exactly zero on average. It has been empirically shown that this rate rises while the network is trained and that sparsity is both advantageous to convergence and performance of the network (Glorot et al., 2011). It is not uncommon that after successful training only 20% of the networks neurons have a non-zero output, i.e. that the resulting final network is sparse.

While sparsity can also be achieved by imposing additional regularization constraints such as L_1 and L_2 regularization on the weights when optimizing a network with standard sigmoid activations, this will typically yield outputs that are close to zero, but not exactly zero. To achieve a value that is close to zero, the pre-activation output has to be a large negative number. Also, with a standard uniform initialization around zero, its outputs will be initialized to a non-zero mean, leading to poor learning dynamics (LeCun et al., 2012; Glorot and Bengio, 2010).

Empirically, ReLU also makes unsupervised pre-training obsolete and very deep neural networks can be successfully trained with standard back-propagation. This is usually attributed to the non-attenuating property of ReLUs, where positive values will be passed on without scaling. Since sigmoid activations would attenuate the input signal in every layer and this can be a contributing factor to the vanishing gradient problem.

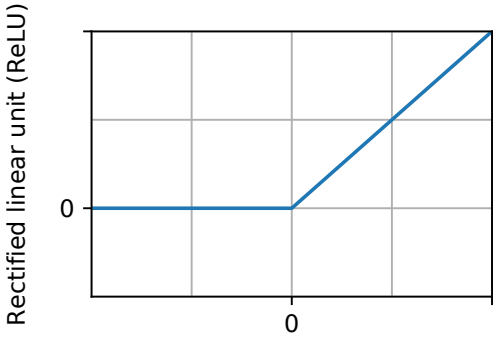

FIGURE 2.4: The ReLU activation function

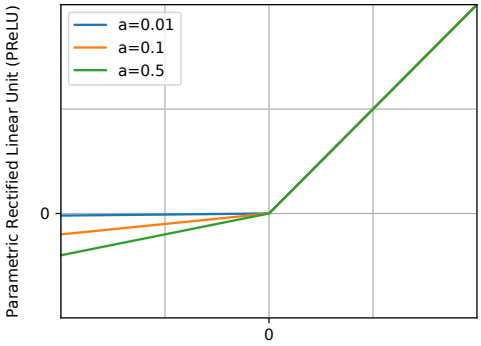

FIGURE 2.5: Example a plots for the PReLU activation function

PReLU. In Parametric rectified linear units (PReLU) (He et al., 2015), an additional parameter a is introduced (see Equation 2.6). This learnable parameter is also optimized with the rest of the networks parameters. PReLUs introduce a small number of extra learnable parameters in the network (either one per layer or one for every neuron), but the network can learn a fitting activation function according to the data it is trained on. Figure illustrates different parameters a

$$f_{PR}(y_i) = \begin{cases} x, & \text{if } x > 0 \\ ax, & \text{if } x \leq 0 \end{cases}. \tag{2.6}$$

This solves the dying neuron problem (He et al., 2015) that can often be observed with ReLUs.

LReLU. Leaky rectified linear units (LReLU) are similar to PReLUs, the only difference is that they use a fixed value for a (e.g. $a = 0.05$). They also counter the dying neuron problem, but they do not add additional trainable parameters to the network.

2.14.3 Weight Initialization

A commonly used heuristic prior to recent work on improving initialization methods in DNNs was the following random weight initialization:

$$W_{ij} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$$

where n is the width of a layer i. As demonstrated in (Jia et al., 2014), this random initialization is susceptible to the vanishing gradient problem with the sigmoid activation function. Assuming a flat layer structure of a neural network, Xavier initialization (Jia et al., 2014) is a proposed alternative initialization that mitigates this problem by setting the variance of weights W_i to:

$$Var(W_i) = \frac{2}{n_i + n_{i+1}}$$

More generally, $n_i = n_{in}$ the number of inputs to a layer and $n_{i+1} = n_{out}$ the number of outputs of a layer. The variance constraint can be satisfied by drawing weights W from the following normal distribution with zero mean:

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{in} + n_{out}}}\right)$$

He-Initialization: (He et al., 2015) adapted this strategy for the ReLU activation function (see Section 2.5):

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{in}}}\right)$$

and more generally:

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{(1 + a^2)n_{in}}}\right)$$

for any LReLU/PReLU activation function. When the negative slope parameter $a = 0$, the initialization and the activation function become ReLU as a special case.

2.14.4 Advances in gradient descent methods

Recent advances in gradient descent methods highlight the importance of adaptive learning rates: In Nesterov momentum (Nesterov, 1983; Sutskever et al., 2013), a velocity vector is accumulated to help accelerate gradient descent. AdaGrad (Duchi et al., 2011) and AdaDelta (Zeiler, 2012) introduce an individual learning rate for each

parameter of a network. The individual learning rates are estimated from past gradient updates. AdaGrad estimates from all past gradient updates, eventually leading to very small learning rates. AdaDelta and RMSProp with decay (Tieleman and Hinton, 2012) improve this by a decaying average over past squared gradients, giving more weight to recent gradient updates. While this expands the needed memory to train a network, this improves convergence and training speed. Adam (Kingma and Ba, 2014) also tracks momentum of the gradients, together with an adaptive learning rate from the decaying average over squared gradients as in AdaDelta and RMSProp. Adam and related optimizers are currently the preferred choice in most instances of DNN training (Liu et al., 2020). Adam is used in most neural network experiments in this thesis as well.

2.14.5 Dropout

Dropout is a simple yet effective technique to regularize a neural network and counter overfitting (Srivastava et al., 2014). While training the network, units and their connections are randomly dropped at each training step. At test time, the full network without any dropped units is used. The dropout rate is the probability for dropping a unit, a higher dropout rate will deactivate more units on average at training time.

A theoretical connection can be made to ensemble theory, in that the final network is an ensemble of all stochastically "thinned" networks at training time (Srivastava et al., 2014).

2.15 Convolutional Neural Networks

Convolutional neural networks (CNNs, also ConvNets) are neural networks that contain a convolution operation. One of the first succesful applications was hand written digit recognition (LeCun et al., 1989). A similar model structure, called the neocognitron was also earlier described by Fukushima and Miyake (1982). The structure is inspired by studies of the mammalian brain (Murphy, 2014). A pooling operation (max-pooling), still widely used in conjunction with convolutional layers, was described in (Zhou and Chellappa, 1988). In the following sections we first describe he convolution operator, then a convolutional layer and max pooling. The typical modern convolutional neural network structures, as also used in later chapters of this thesis are described next.

2.16 Convolution Layer

The 2D convolution operation ($*$) between an input matrix and a kernel of size (m, n) is:

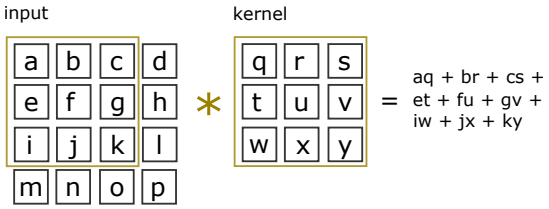

FIGURE 2.6: Example of a 3x3 convolution kernel applied to an 2D input matrix.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

See also Figure 2.6 for a visualization of this operation. Most machine-learning libraries implement convolution as cross correlation (Goodfellow et al., 2016). When used with SGD, the only difference is that the learned kernel would be flipped compared to the convolution operation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

A convolutional layer typically consists of multiple kernels, each with its own set of learnable weights. The input is usually much larger than the kernel size and all kernels are applied to all possible positions in the input (or some pattern). The resulting output of all kernels on all positions is the feature map of the convolutional layer and can be passed on as input to a subsequent layer as part of a larger neural network. The learned parameters of a convolutional layer are the weights of the kernels, hyperparameters are the size of kernels and the number of kernels, as well as the number of input and output channels. Multiple input channels are usually used for representing for instance RGB colors in an image, as red, green and blue would be separate input channels. In this thesis we will only use single-channel convolution, as standard feature representations of speech are represented with one channel for mono audio signals.

2.17 Pooling Layer

Max-pooling (Zhou and Chellappa, 1988) (also (Weng et al., 1993)) is a simple layer without learnable parameters. It is usually used following a convolutional layer and can also be applied to 2D spatial input. The output of a pooling operation is the maximum of its inputs. Like the kernels in a convolutional layer, the pooling operation operates on window sizes that are usually smaller than its inputs and the operation can be applied as a sliding window across its inputs. In average pooling, a similar operation, the output is the average of all inputs instead of the maximum.

2.18 CNN Network Structures

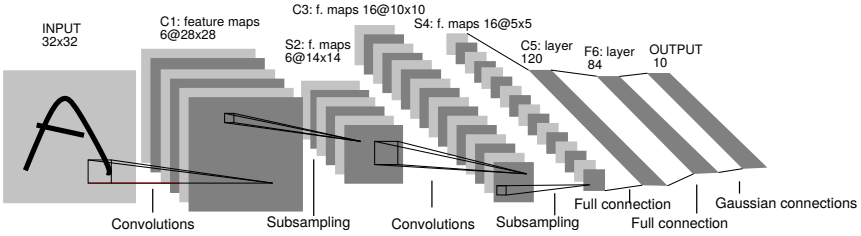

FIGURE 2.7: LeNet-5, a convolutional neural network for handwritten digit recognition. Figure taken from (LeCun et al., 1998).

LeNet-5 (LeCun et al., 1998) is a classic convolutional neural network structure and by current standards comparatively small. Figure 2.7 shows the topology of the network. A set of convolutional kernels (C1) follows a subsampling step (S2), another convolutional layer (C3) is then followed by subsampling (S4), with the final layers being a fully connected (C5, F6). The subsampling layer uses average pooling, with a kernel size of 2x2 and the convolutional kernels have a size of 5x5. All layers use the tanh activation function.

In the following, we list some newer and larger CNN network structures that build and extend the principles of LeNet-5:

AlexNet (Krizhevsky et al., 2012) was one of the first very deep convolutional neural networks, scaling up the LeNet architecture. The network design has a focus on GPU parallelization and consists of two 8 layer networks split onto two different GPUs. Each network consists of 5 convolutional layers (with 3x3 and 5x5 kernels), each followed by max pooling. Further architectural changes are the use of the ReLU activation function throughout the network and a larger number of kernels per convolutional layer, increasing from 48 kernels in the first convolutional layers to 192 convolutional layers in the last ones. The final fully connected layers of the network are also significantly larger with 2048 neurons. Dropout and image data augmentation was used to train the network.

VGGNet (Simonyan and Zisserman, 2015) is a series of differently sized CNNs. In Figure 2.8 we exemplary show VGGa and VGG16. VGGNets stack convolutional layers to increase the receptive window, while requiring less parameters than a single convolutional layer of the same receptive size. Stacking two 3x3 convolutional layers gives an effective receptive window of 5x5 and stacking three 3x3 convolutionals an effective window of 7x7. All pooling layers are max pooling. The activation function is also ReLU. ResNet (He et al., 2016) is another popular convolutional neural network, that improves ease of learning by adding residual connections. A residual component, a building block of ResNet, has the form (Equation 2.7):

$$y = \mathcal{F}(x, W_i) + x \tag{2.7}$$

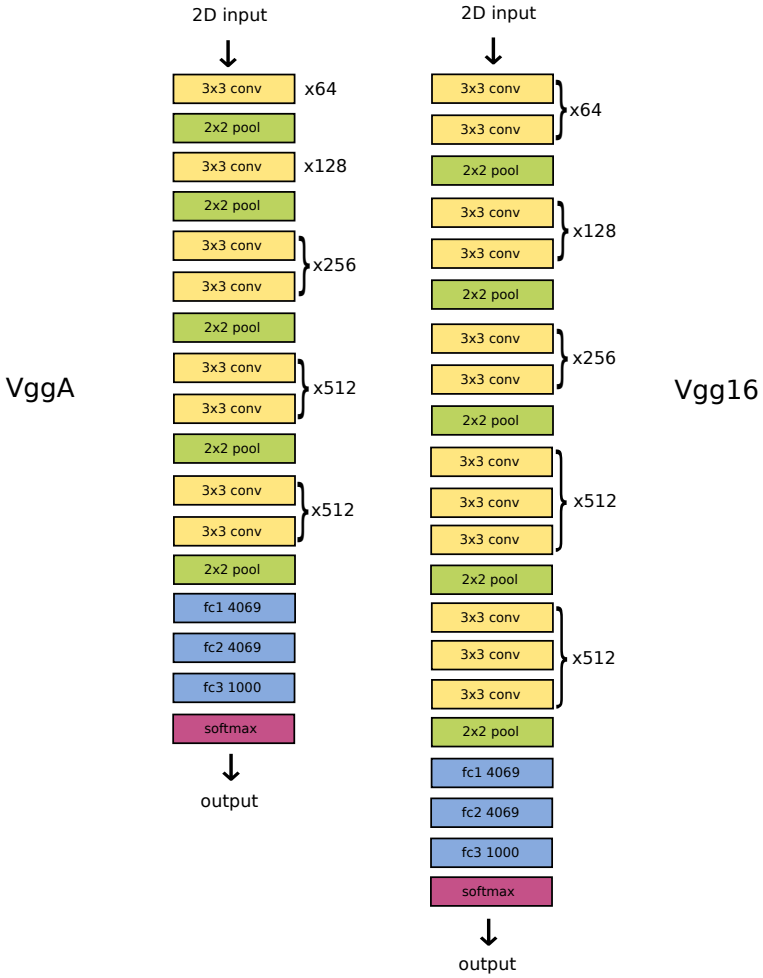

FIGURE 2.8: VGGA and VGG16, two networks of the VGGNet family

where x are inputs and y outputs. $\mathcal{F}$ represents the residual mapping that is being learned, usually spanning multiple layers, e.g. $\mathcal{F} = W_2\sigma(W_1x)$ (He et al., 2016). With a single layer residual component of the form $y = W_1x + x$ it becomes similar to a one layer neural network ($y = W_1x + b, b = x$) and no benefits were observed for this structure.

In Figure 2.9, ResNet-34 is illustrated and compared to VGG-19. The residual mappings are additional arrows in the network structure. Unlike VGGNet, ResNet can be scaled up to a large number of layers without degradation of learning performance, with the most popular configurations being ResNet-50, ResNet-101 and ResNet-152 (each named after the number of layers). The use of the PReLU activation function (see also Section 2.14.2) instead of ReLU further improves the network structure (He et al., 2015).

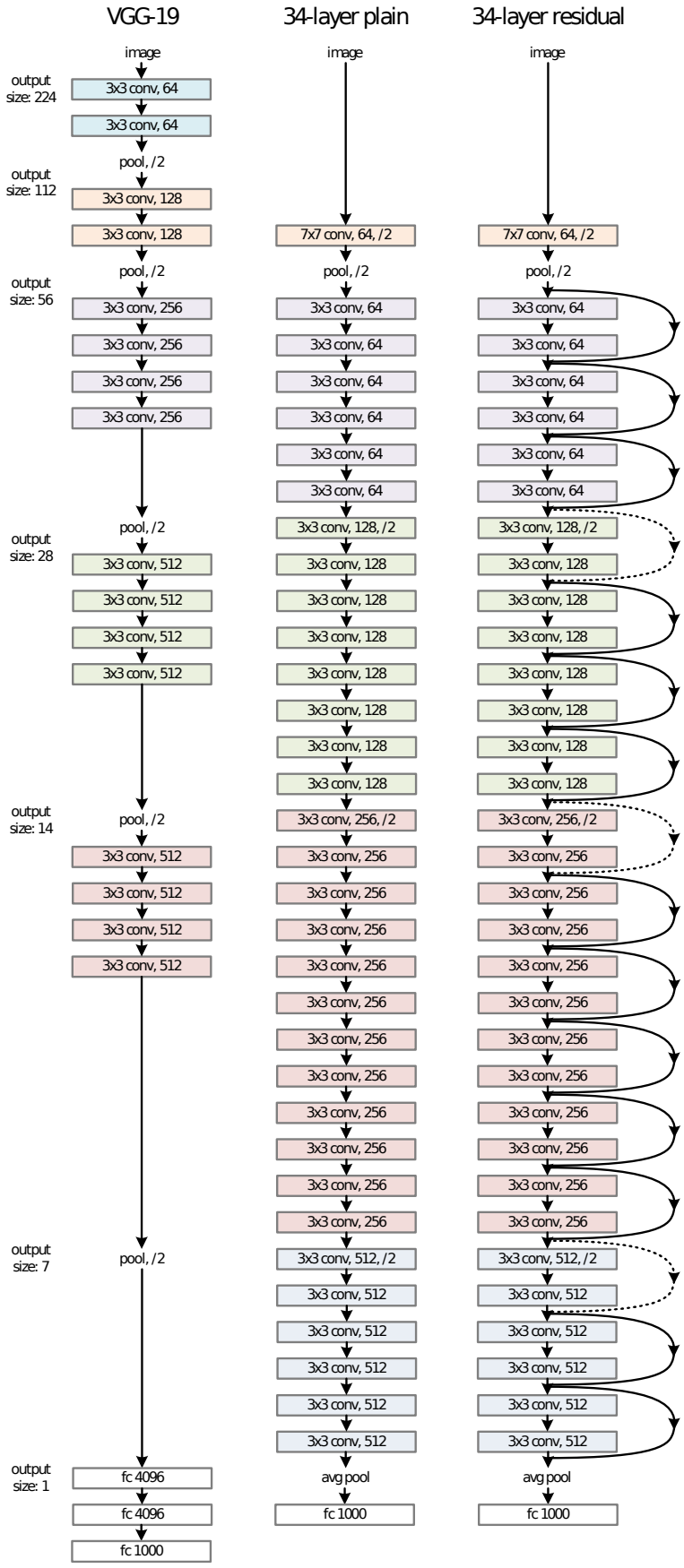

FIGURE 2.9: VGGNet-19 compared to ResNet-34 without residual connections and with them. Figure from (He et al., 2016).

2.19 Autoencoders

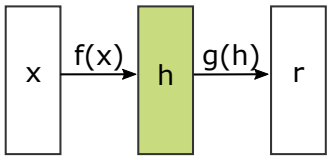

FIGURE 2.10: Generic structure of an autoencoder.

An autoencoder is a neural network that reconstructs its own inputs (Rumelhart et al., 1986; Baldi and Hornik, 1989), with the goal of learning useful properties of the underlying data. Figure 2.10 illustrates an autoencoder consisting of an encoder $f(x)$, mapping x to the internal representation h and $g(h)$ mapping h to the reconstructed output r. The encoder and the internal representation h is what is then typically used as a learned representation of the data.

2.20 Undercomplete Autoencoders

An undercomplete autoencoder learns a representation that has a smaller dimension than the input representation (Goodfellow et al., 2016). Figure 2.10 shows such an example autoencoder neural network structure. The idea of an undercomplete autoencoder is that by introducing a bottleneck with a smaller bandwidth than what is required to simply copy the inputs to the outputs, the autoencoder must learn to capture the most salient information in the data.

Typically, an autoencoder is trained with SGD, optimizing the error between the input representation x and $g(f(x))$(Goodfellow et al., 2016), see Equation 2.8.

$$L(x, g(f(x))) \tag{2.8}$$

L is the loss function for comparing these, usually a loss functions for regression tasks such as mean squared error (MSE) or the Huber loss (see also Section 2.9).

2.21 Regularized Autoencoders

Regularized autoencoders add a regularization term to the loss function, either encouraging network weights or representations with certain properties. One of the simplest forms of a regularized autoencoder is one with L2 regularization (Rifai et al., 2011), where as a regularization term all weights of the network are squared and summed up:

$$L(x, g(f(x))) + \lambda \sum_{ij} W_{ij}^2 \tag{2.9}$$

where λ controls the weight (strength) of the regularization, with larger values yielding smaller network weights.

Another example of regularized autoencoders are sparse autoencoders. They are able to train useful representation that are over-complete, i.e. with a dimensionality larger than the input dimension. In (Ng, 2011) and also in (Lee et al., 2008) a simple sparsity regularization loss is proposed, that is applied on the hidden units activations:

$$L(x, g(f(x))) + \sum_{j=1}^{n} \rho log\left(\frac{\rho}{\hat{\rho}_j}\right) + (1-\rho) log\left(\frac{1-\rho}{1-\hat{\rho}_j}\right) \tag{2.10}$$

where n is the number of neurons in the hidden layer, $\hat{\rho}_j$ is the average activation of hidden unit j and ρ is the sparsity parameter, usually a small value close to zero. This can also be interpreted as $\sum_{j=1}^{n} KL(\rho, \hat{\rho}_j)$ (Ng, 2011), comparing two Bernoulli random variables with mean ρ and $\hat{\rho}_j$ with KL divergence.

In contractive autoencoders (Rifai et al., 2011), the following regularization is proposed:

$$L(x, g(f(x))) + \lambda ||J_f(x)||_F^2 \tag{2.11}$$

where $||J_f(x)||_F^2$ is the Frobenius norm of the Jacobian of the encoder's activations with respect to the input x (Rifai et al., 2011). The hypothesis is that a good learned representation of x is one that is locally invariant in many directions of change in respect to x. This type of representation is what the penalty term in Equation 2.11 encourages.

2.22 Denoising Autoencoders

Denoising autoencoders (Vincent et al., 2010) corrupt the input representation x into $\tilde{x}$, the loss becomes (Equation 2.12, notation follows Goodfellow et al. (2016)):

$$L(x, g(f(\tilde{x}))) \tag{2.12}$$

In other words, the autoencoder must learn to reconstruct the original x from a noisy input $\tilde{x}$. Denoising autoencoders build on the idea that a good representation of the input data is one that can be generated from corrupted inputs to allow recovery of the original inputs (Vincent et al., 2010). Denoising is also a more challenging self-learning task than the reproduction of inputs of regular autoencoders.

2.23 Recurrent Neural Networks

The network of the architectures all require that the inputs are of the same size. Unless specifically preprocessed into a fixed vector representation, language is usually a sequence: input texts in NLP typically vary in length; in speech processing inputs typically vary in duration.

Recurrent neural networks (RNNs) are good models for sequential data, as they can be applied to inputs of varying length. With these networks, the inputs can be presented to the network one at a time. A distributed hidden state allows RNNs to store information about past time steps efficiently, while the state can be manipulated in complicated ways through non-linear interactions and transformations (Hinton, 2013).

Hopfield networks, proposed as an artificial neural associative memory, was one of the first architectures to propose the idea of recurrence in neural networks (Hopfield, 1982). Werbos (1988) and Williams and Zipser (1995) popularized recurrent neural networks and learning within the back propagation framework (calling it Back Propagation Through Time, BPTT).

A standard Recurrent Neural Network will have a state h, an input transformation function (i) and an output transformation (o):

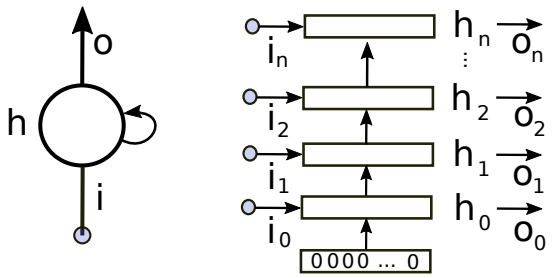

FIGURE 2.11: Schema of a Recurrent Neural Network (RNN) with a state h, input transformation function i and an output transformation o. Drawn in similar style as (Yan, 2015).

If we unroll the recurrence of a static RNN with a fixed number of time steps, as depicted in Figure 2.11, we can also interpret the unrolled network as a DNN that has as many layers as there are time steps. This means that RNNs are by design very deep networks, but unlike standard feed forward networks, there is an input at every layer. Furthermore, another major difference is that the weights of the layers are all tied. In other word the same transformation is applied at every time step (and accordingly optimized when trained). If we consider a classification problem, where we are only interested in the output of the last layer, then small changes to the weights will have potentiating effects throughout a sequence to the final output of the network. The same (matrix) multiplication is applied to the state as many times as there are time steps. This is why in practice, training such networks with SGD will be extremely difficult, as they encounter the vanishing gradient problem (Hochreiter, 1991) and also the opposite problem of exploding gradients very easily.

2.23.1 LSTM

The Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a solution to the vanishing gradient problem in vanilla RNNs. It introduces the concept of gating into the recurrence of the network. Gates control information flow. They are used within the recurrent cell to control how much the state should be changed given the current input at each step and also allowing to forget some of the old state, allowing adaptive changes to the state. This in turn solves the issue of potentiating changes throughout a sequence.

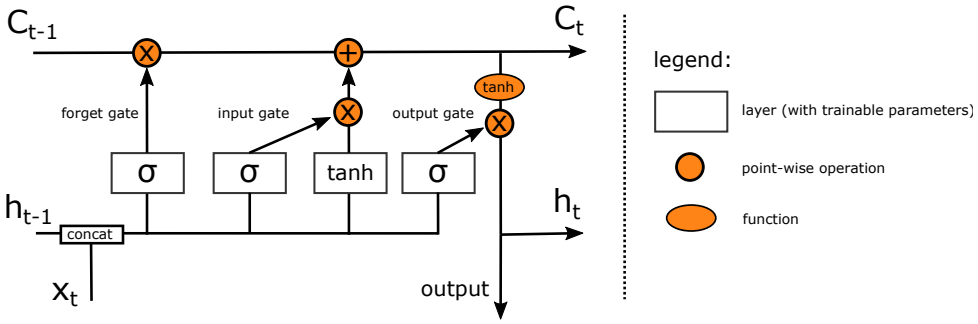

FIGURE 2.12: Schema of a Long short-term memory (LSTM) cell. Drawn in the style of Yan (2015).

In the following, we consider the canocial form of a standard LSTM with a forget gate from (Gers et al., 1999) and we follow the LSTM notation of (Goodfellow et al., 2016) for all equations. In Figure 2.12 we illustrate an LSTM cell, that is then recursively applied to a sequence. The gates are used in an LSTM, the forget, input and output gate. Each gate is a layer with trainable parameters followed by a sigmoid function, putting the output values of a gate in the range $[0, 1]$. When the output of a gate is element-wise multiplied with some other input, it can control the amount of information that is passing through it. In the forget gate unit, this is directly applied to the cell state, allowing certain cell state values to be reset ("forgotten"). Similarly, the input gate controls how much new input changes the cell state. Finally, the output gate then controls the flow and computation of an output from a current cell state for a particular time step.

The **forget gate** unit ${f_i}^{(t)}$ is formally defined as (Gers et al., 1999; Goodfellow et al., 2016):

$$ {f_i}^{(t)} = \sigma \left(b_i^f + \sum_j U_{i,j}^f {x_j}^{(t)} + \sum_j W_{i,j}^f {h_j}^{(t-1)} \right) \tag{2.13} $$

where b_f, U_f and W_f are trainable parameters (biases, input weights and recurrent weights) of the forget gate, $x^{(t)}$ is the input at time step t and $h(t)$ the hidden layer vector at time step t.

The internal cell state s is updated as follows:

$$s_i^{(t)} = f_i{}^{(t)} s_i{}^{(t-1)} + g_i^{(t)} \tanh \left(b_i + \sum_j U_{i,j} x_j{}^{(t)} + \sum_j W_{i,j} h_j{}^{(t-1)} \right) \tag{2.14}$$

where b, U and W are trainable parameters (biases, input weights and recurrent weights). The external **input gate** unit g is defined as:

$$g_i{}^{(t)} = \sigma \left(b_i^g + \sum_j U_{i,j}^g x_j{}^{(t)} + \sum_j W_{i,j}^g h_j{}^{(t-1)} \right) \tag{2.15}$$

with its own set of trainable parameters b^g, U^g and W^g. The output $h_i{}^t$ at time step t of the LSTM cell is computed as:

$$h_i{}^t = tanh \left(q_i^{(t)} s_i{}^{(t)} \right) \tag{2.16}$$

where the **output gate** $q_i{}^t$ is defined as:

$$q_i{}^{(t)} = \sigma \left(b_i{}^o + \sum_j U_{i,j}^o x_j(t) + \sum_j W_{i,j}^o x_j(t-1) \right) \tag{2.17}$$

Many alternative recurrent neural architectures and variants of the LSTM have also been proposed, e.g. (Cho et al., 2014b; Zhou et al., 2016; Wu et al., 2016b; Krause et al., 2017; Zilly et al., 2017). In Jozefowicz et al. (2015), a meta search with 10,000 different recurrent architectures was conducted. One of the findings of the ablative study is that the forget gate is one of the most important elements of the LSTM. It is not always easy to quantify which recurrent architecture is better than another, as the performance of a particular variant can be task-dependent.

2.23.2 Bi-directional

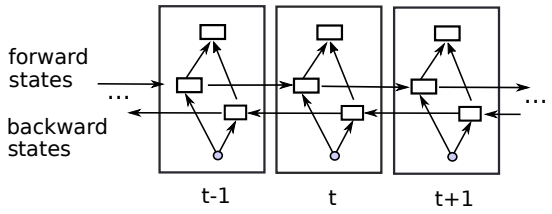

FIGURE 2.13: Schema of a bi-directional RNN, unfolded in three time steps $t-1$,t,$t+1$. Drawn in style of Schuster and Paliwal (1997).

A bi-directional RNNs allows outputs at each time step that are dependent on past and future time steps (Schuster and Paliwal, 1997). Figure 2.13 shows the general structure of a bi-directional RNN. It is also possible to use LSTM or GRU cells.

2.24 Neural Sequence-to-Sequence Models

Neural sequence-to-sequence models can learn a conditional distribution over a variable length sequence conditioned on another sequence $p(y_1, ..., y_T|x_1, ..., x_{T'})$, where T' can be different from T (Cho et al., 2014b).

Plain Seq2Seq models. A plain Seq2Seq model follows an encoder and decoder design, where the input sequence is encoded token by token and the output sequence is generated token by token. Typically, recurrent neural networks are used in both the decoder and encoder network. The RNN unit is usually one with a gating mechanism, e.g. a Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or a Gated Recurrent Unit (GRU) (Cho et al., 2014a), to counter the vanishing gradient problem inherent in RNNs (Hochreiter, 1991). The output sequence $y_1, ..., y_T$ is conditioned on a single vector that is generated with the encoder of the network, by initializing the RNN decoder state to the last RNN state of the encoder.

Encoder/decoder inputs. We use character embeddings for the encoder and phoneme embeddings for the decoder as inputs. For both we choose n=10 as the embedding size. The embeddings are trained as part of the model training. Following (Sutskever et al., 2014), we reverse the input character sequence.

Attention mechanism. A drawback of the plain Seq2Seq model is that all the source information has to pass through a single vector and needs to be carried forward in the decoder's state. In order to facilitate information flow from the source sequence to the target sequence and to relieve the decoder from having to remember the input sequence in its state, a context vector can be computed as a weighted sum over all encoder hidden states, dependent on the last decoder state (Bahdanau et al., 2015). The attention vector is either added or appended to the RNN decoder inputs. Different variations of this mechanism exist, which can be divided into local and global attention mechanisms (Luong et al., 2015). We use global attention, where an attention vector is generated at each phoneme generation step over the full character input sequence.

Stacked bi-directional LSTM. The encoder can also be extended to represent past and future dependencies, with two RNNs that read a word in opposite directions. Figure 7.1 depicts a Seq2Seq architecture for G2P with a bi-directional LSTM encoder using a decoder with global attention. Furthermore, LSTM cells can also be stacked vertically, by passing the output of a LSTM cell as input to another LSTM cell. We combine the outputs of the forward and backward encoder with a vector sum (an alternative would be to stack both vectors).

Residual connections. The residual network (He et al., 2016) provides skip-connections between layers of a network. The main idea is that learning the identity

function is simpler with a residual connection, because bypassing the layer's computation means that the output of a layer needs to produce a zero output and not the identity function. We make use of a simple residual connection between LSTM layers, where the output of an LSTM layer is added to its input before passing it to the next LSTM layer:

$$y = F(x; W) + x$$

where y is the output of a layer and x the input. $F(x; W)$ is a function with an internal parameter W (Kim et al., 2017). We use the LSTM unit directly as F when stacking LSTM units, i.e. we do not add skip connections between time steps.

Decoding. At each decoding step, the decoder generates a softmax distribution over the output vocabulary. The simplest method to generate the output sequence is to feed the argmax token to the next decoding step. However, a greedy decoder might not be able to find the sequence with the best joint probability. Beam decoding can be used to approximately search for the sequence with the best joint probability (as predicted by the model). It keeps n different best paths while decoding and explores only these in a further step, feeding and generating n new sequences per path.

Chapter 3

Automatic Speech Recognition

Automatic speech recognition (ASR) is a very active research area. General-purpose automatic speech recognition has evolved from a hopeless dream (Pierce, 1969) to systems that can reliably recognize conversational speech in the past 50 years[1]. In the following sections, we give an overview over the most common modern architectures and components. Two important strategies to approach ASR have had remarkable success. One is the statistical interpretation of the problem and a mathematically sound model that can be applied to it: Hidden Markov Models (HMMs, see Section 3.4) were instrumental in developing the first successful large-vocabulary recognition systems and also the first commercial systems (Juang and Rabiner, 2005). Since the 1980s, these models gained wide-spread traction in speech processing (Rabiner, 1989) they have been a dominant modeling technique and remain in widespread use today.

Another strategy, the integration of neural networks into the speech processing pipeline, has recently become prevalent and offers a large increase in recognition accuracy. This is currently also one of the most active research areas in speech processing. Neural models seem to scale well to large speech corpora and can effectively leverage today's increased computer power (Section 2.7). Some of these new architectures are hybrids, as they combine deep neural networks with HMMs. Some are deviating from the classical speech recognition pipeline and architecture and they do not use the HMM framework anymore (see also Sections 3.11 and 3.12). A search space is usually established for most probable transcription given an utterance according to the model, as detailed in the next section.

3.1 ASR search space

The search space for the likeliest transcription given some speech input is usually modelled using sequences of observations (O) and words (W):

$$O = o_1, o_2, o_3, ..., o_n$$

$$W = w_1, w_2, w_3, ..., w_n$$

[1] See also Juang and Rabiner (2005) for a historic overview of systems from the past century.

The likeliest transcription can then be defined as (Jurafsky and Martin, 2008):

$$\tilde{W} = \underset{W}{\operatorname{argmax}} P(W|O)$$

In other words, we are interested in the most probable sequence of words, given a sequence of observations. Using Bayes' rule (Bayes et al., 1763) this can also be transformed to:

$$\tilde{W} = \underset{W}{\operatorname{argmax}} \frac{P(O|W) \cdot P(W)}{P(O)} = \underset{W}{\operatorname{argmax}} P(O|W) \cdot P(W)$$

An interpretation of the formula above is that we model the likeliest transcription as the one being the likeliest explanation for how the observations were generated. Computationally, this also has several advantages: $P(W)$ can be independently modelled as the prior probability of a transcription. Language models (see Section 3.10) are used to estimate this probability and they can be effectively trained on large datasets of texts, without needing corresponding speech data. $P(O|W)$, also called the observation likelihood, can for instance be estimated with Gaussian Mixture Model – Hidden Markov Models (GMM-HMMs). For a more detailed description, see Section 3.5. If we are interested in the likeliest transcriptions, $P(O)$ does not need to be modelled since it is constant and scales all probabilities with the same factor. Because of the argmax, it can be omitted from the equation.

Using words for modelling the search space is the most common choice. However, this design decision is also language dependent. Morphologically productive languages such as German, Turkish, Estonian and Finnish might benefit from other choices such as subword-units or syllables (Smit et al., 2017). End-to-end models might also model the output sequence directly using letters, with the advantage that out of vocabulary (OOV) words are still potentially recognizable.

3.2 ASR components

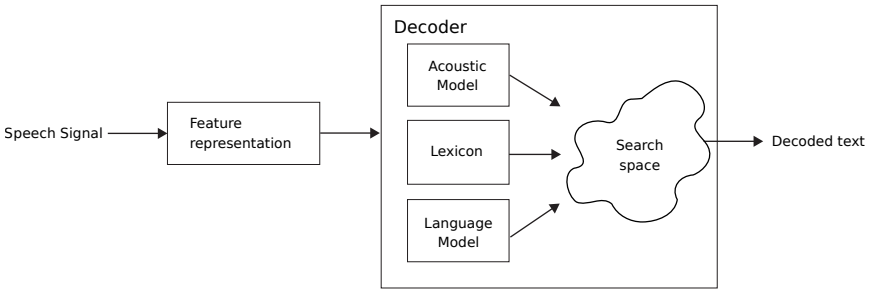

FIGURE 3.1: Overview of the components in a typical (traditional) ASR system. Figure in similar style as in (Milde, 2014).

In Figure 3.1 we illustrate a typical traditional ASR system. Note that some recent end-to-end ASR architectures deviate from this traditional architecture. A recorded signal is first transformed, often into some sort of representation derived

from spectral features (observation O). A decoder then searches for the most likely transcription according to the observations, using information derived from three components: the acoustic model, lexicon and language model. The acoustic model relates acoustic observations to internal states, for example using context dependent phonemes. The lexicon defines a closed vocabulary and usually relates words to sequences of phonemes. The language model models the likelihood of word sequences independently of the acoustic model. It is used by the decoder to weigh more likely word sequences higher than unlikely ones. The language model, dictionary and a translation between internal states and hypothesized word sequences can efficiently be combined and expressed as a single weighted finite state transducer (WFST), see Section 3.9. In the following, we describe these components and common modeling techniques in more detail, starting with common (hand-crafted) speech features.

3.3 Speech Features

Good speech features should preserve most of the desired content, while also offering a compact representation of the speech signal (Gales, Young, et al., 2008). In the following sections we describe typical speech features that are used in speech recognition and other speech processing tasks.

Most of the discriminative information in human speech is in a frequency range below 10 kHz (Jurafsky and Martin, 2008). Recorded digital speech is sampled, i.e. amplitudes are measured at sampled time steps. The sampling rate is the number of samples taken per second. Due to Nyquists theorem (Nyquist, 1928), the highest frequency that can be accurately measured is half the sampling rate. A typical sampling rate for speech recorded with microphones that is going to be used for automatic speech processing is 16 kHz.

Another typical preprocessing step is windowing, where speech is segmented into small overlapping windows of speech. While speech is non-stationary at large, we can make the approximation and assume that it is somewhat stationary in small windows (Jurafsky and Martin, 2008). This approximation also makes further computation easier. Such small windows of speech are also called speech frames.

3.3.1 FBANK

Figure 3.2 depicts a typical pipeline to compute FBANK vectors, along with example outputs at each stage. In the pre-emphasis step, higher frequencies are amplified (higher frequencies usually have smaller magnitudes). This can be done in time domain for an input $x[n]$ and $0.9 < \alpha < 1.0$ (Jurafsky and Martin, 2008):

$$y[n] = x[n] - \alpha x[n-1].$$

In the next step, windows are extracted from the signal to eventually compute features at discrete time steps. A typical resolution for FBANK features is to calculate

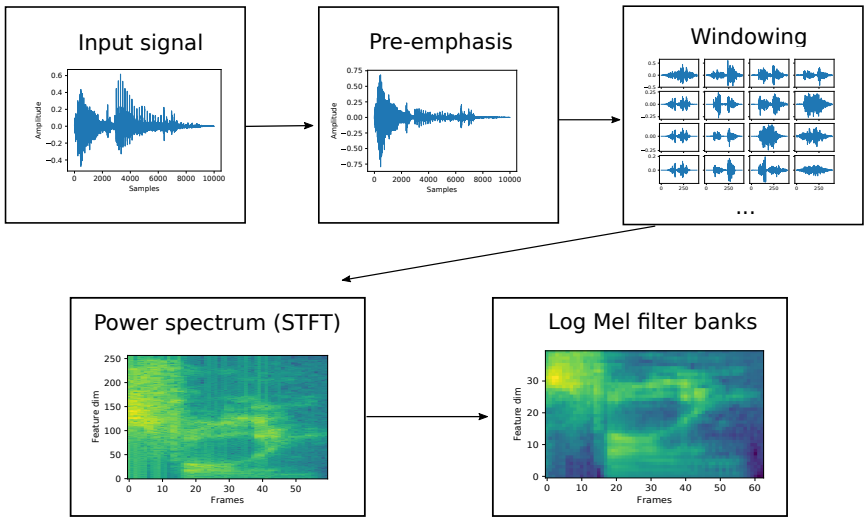

FIGURE 3.2: Pipeline steps to compute filter bank frames from speech. The output of these steps is partially computed for the utterance "dusk was falling as the boy arrived with his herd at an abandoned church" from the Common Voice V1 corpus (train_valid: sample-055799.mp3)

a frame for a window of 25 ms of speech, using overlapping speech frames every 10 ms. Discontinuities are problematic at edges of the windows, thus it is typically necessary to apply a window function. A common choice is the Hamming window, defined as:

$$w[n] = \alpha - \beta cos(\frac{2\pi n}{L}) \quad \text{with} \quad \alpha = \frac{25}{46} \approx 0.54 \quad \text{and} \quad \beta = 1 - \alpha \approx 0.46$$

There is a trade-off when the above window function is applied. The effects of discontinuities are diminished at the ends of the window, but at the same time the signal is changed. The Hamming window with the fixed parameters $\alpha = 0.54$ and $\beta = 0.46$ (rounded to two digits) is a common choice as the windowing function for extracting speech processing features. The exact value $\alpha = \frac{25}{46}$ minimizes spectral leakage of the first sidelobe optimally, c.f. (Harris, 1978).

After windowing the signal, Discrete Fourier Transforms (DFTs) are applied individually to all windows of the signal to extract spectral information (Jurafsky and Martin, 2008). This is also known as a short-time Fourier transform (STFT) (Allen, 1977; Griffin and Lim, 1984). Commonly Fast Fourier Transformations (FFTs) (Oppenheim and Schafer, 1989) are applied to a signal due to its computational efficiency. To compute the power spectrum, the absolute value of the complex output of the STFT is squared for each frequency bin (this is equivalent to sums of squaring the real and imaginary parts of the STFT).

Filters are then spaced linearly at lower frequencies (below 1000Hz) and logarithmically at higher frequencies (higher than 1000Hz) (Davis and Mermelstein, 1980). This mimics the sensitivity to frequencies of the human ear and its known variations

in critical bandwidths (Fletcher, 1940; Bullock and Evans, 1977; Davis and Mermelstein, 1980). This specific filter bank is also called a Mel filter bank (Jurafsky and Martin, 2008) due to its relation and similarity to the Mel scale (Stevens et al., 1937).

Finally, taking the logarithm of the output values of the filter banks compresses the dynamic range of the output and mimics the human perception of different signal levels (Jurafsky and Martin, 2008). The example outputs in Figure 3.2 only show logarithmic output, as the raw filter values are difficult to display otherwise due to large variations in magnitudes.

3.3.2 MFCC features

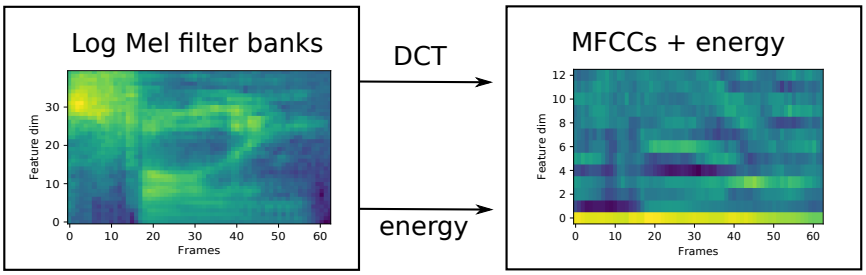

FIGURE 3.3: Final step to compute MFCCs from logarithmic Mel filter banks. The output of this steps is partially computed for the utterance "dusk was falling as the boy arrived with his herd at an abandoned church" from the Common Voice V1 corpus (train_valid: sample-055799.mp3)

Mel-frequency cepstral coefficients (MFCCs) (Mermelstein, 1976) are the most ubiquitous and well-known features in speech processing. To compute MFCC speech frames an additional discrete cosine transform (DCT) (Ahmed et al., 1974) is applied to the Mel-scaled filter banks from the previous section. The DCT smoothes the spectral estimate and decorrelates the feature elements (Gales, Young, et al., 2008). This is important when used in GMM-HMM acoustic models (see also Section 3.5). Figure 3.3 shows an example output for this computation. The most common MFCC representation is with 12 coefficients and an additional energy feature, totaling 13 dimensions per frame. The energy feature can be computed directly in the time domain by summing up the squares of all samples in the audio signals window.

3.3.3 Delta features

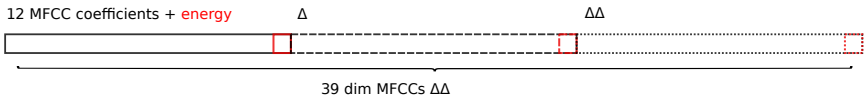

FIGURE 3.4: MFCC frame with delta features.

Δand ΔΔfeatures are commonly applied to MFCCs in many acoustic models recipes (Jurafsky and Martin, 2008). Δfeatures measure how MFCC cepstral components and the energy feature change (velocity). A simple way to compute this is

to calculate the (discrete) difference between neighboringing MFCC frames. $\Delta\Delta$features capture the rate of change of Δfeatures (acceleration). Figure 3.4 illustrates one MFCC speech frame with delta features, usually calculated on a window of 25ms every 10ms of a speech signal. The standard dimension for MFCCs with delta features is $(3 \cdot 13) = 39$.

3.3.4 CMVN

Cepstral mean and variance normalization (CMVN) (Liu et al., 1993; Viikki and Laurila, 1998), are simple and often effective normalization techniques for cepstral speech features. In cepstral mean normalization (CMN) (Liu et al., 1993), the mean of cepstral vectors is substracted from all frames in an utterance:

$$y[n] = x[n] - \frac{1}{N}\sum_{n=1}^{N} x[n] \tag{3.1}$$

for all frames n in an utterance. Additionally, variance can also be normalized by dividing by $\sigma(x[n])$ (CMVN).

3.3.5 PLP features

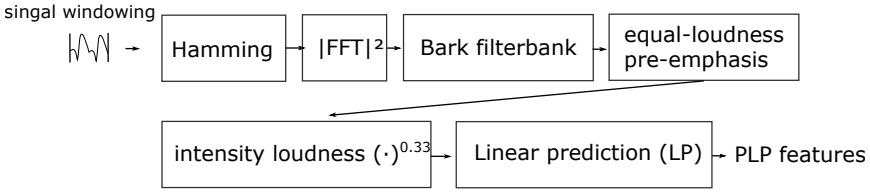

FIGURE 3.5: Processing steps for PLP speech features.

Perceptual linear predictive (PLP) (Hermansky, 1990) speech features are an alternative to MFCC features. Figure 3.5 shows a schematic view of all processing steps in PLP speech features. Both MFCC and PLP share several processing steps, such as applying a Hamming window, squaring the absolute values of the complex STFT and a filter bank computation. PLP uses the Bark scale (Schroeder, 1977) instead of the Mel-Scale for its filter banks. Compared to the Mel scale, Bark uses a different number and width of the filters and their shape is different (Hönig et al., 2005). While with MFCC pre-emphasis is applied in the time domain, with PLP features pre-emphasis is applied to the power spectrum, as each coefficient is multiplied with a weight that dependents on the frequency in Hertz. The intensity-to-loudness conversion used in PLP raises the power spectrum coefficients to the power of 0.33. This decreases dynamic variability and flattens peaks in the power spectrum (Hönig et al., 2005).

The final step in PLP is spectral linear prediction (LP) (Makhoul, 1975), by fitting the power spectrum coefficients onto an M th-order all-pole model (Hermansky, 1990). Additionally, the autoregressive coefficients are usually transformed into cepstral coefficients of the all-pole model.

3.4 Hidden Markov Models

A Hidden Markov Model (HMM) λ can be characterised by $\lambda = (A, B,\pi)$, where A are state transition probabilities, B observation symbol probability distribution for all states and π the initial state probabilities (Rabiner, 1989). Implicitly, λ also defines the alphabet $V = S_1, S_2, ..., S_M$, a set of discrete output symbols and a number of states $S = S_1, S_2, ..., S_N$.

HMMs are generative models. Given the parameters $\lambda = (A, B,\pi)$ of a model, a sequence of T observations can be generated as follows (Rabiner, 1989) with Algorithm 1:

Data: HMM $\lambda = (A, B,\pi)$, T the number of observations that should be generated.
Result: An observation sequence $O = o_1, o_2, ..., o_T$
$t = 1$;
From the initial state distribution π, choose a state q_t. ;
for t *in* $1, ..., T$ **do**
  Choose an observation o_t according to the observation symbol probability distribution of q_t. ;
  Transit to a new state q_{t+1} according to the transition probabilities of q_t ;
end

Algorithm 1: Generating a sequence with an HMM model.

There are three fundamental problems in HMMs (Rabiner, 1989): (1) Evaluation: compute the probability (likelihood) of an observation given a model λ (2) Recognition: find the state sequence $Q = q_1, q_2, ..., q_n$ that gives the best explanation for an observation sequence $O = o_1, o_2, ..., o_n$ (3) Training: find or adjust model parameters that maximize $P(O|\lambda)$ given an observation sequence O. In the following sections we briefly discuss these fundamental problems in HMMs and their relation to speech recognition.

3.4.1 Evaluation

In the evaluation problem, we seek to calculate the probability that a model λ generated a given observation sequence $O = o_1, o_2, ..., o_n$. The forward algorithm (Rabiner, 1989) can be used to calculate this probability.

3.4.2 Recognition

An HMM can be used for recognition by estimating the most likely explanation (path through the hidden states) for a sequence of observations $O = o_1, o_2, ..., o_n$. If for example the hidden states of the HMM correspond to phonemes, then the sequence of hidden states in the most probable path is the recognized output. A well-known algorithm that is used for decoding with HMMs is the Viterbi algorithm (Viterbi, 1967), see Section 3.7.

3.4.3 Training

When training an HMM, we are given an observation sequence $O = o_1, o_2, ..., o_n$ and we want to fit the model parameters $\lambda = (A, B, \pi)$ of the HMM. For maximum likelihood estimation, several algorithms exist, such as Baum-welch (Baum et al., 1970), Expectation-Maximization (Dempster et al., 1977) and Extended Baum-Welch (Gopalakrishnan et al., 1989). Gradient based optimization methods are also possible (Katagiri et al., 1991).

3.5 GMM-HMMs

In Gaussian Mixture Model – Hidden Markov Models (GMM-HMMs), HMM states and observation vectors are modelled as a mixture of multi-variate Gaussians. For each HMM state j with a mean vector μ_j, covariance Σ_j and an observation vector o_t (D-Dimensional), the multi-variate Gaussian probability can be computed as (notation follows (Jurafsky and Martin, 2008)):

$$b_j(o_t) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}(o_t-\mu_j)^T\Sigma_j^{-1}(o_t-\mu_j))\right)} \tag{3.2}$$

In a Gaussian mixture model, M multi-variate Gaussians are used to compute the output likelihood function $b_j(o_t)$:

$$b_j(o_t) = \sum_{k=1}^{M} c_{jm} \frac{1}{\sqrt{(2\pi)|\Sigma_{jm}|}} e^{\left((o_t-\mu_{jm})^T\Sigma_{jm}^{-1}(o_t-\mu_{jm}))\right)} \tag{3.3}$$

3.6 Context Dependent Phones

The realization of a phone depends strongly on its surrounding context. Phoneticians have been studying these effects (Oshika et al., 1975) from the perspective of human perception before automatic speech processing became popular. Most speech recognition systems account for this effect by modeling context dependent phones. A tri-phone model would make use of context information from a phone's

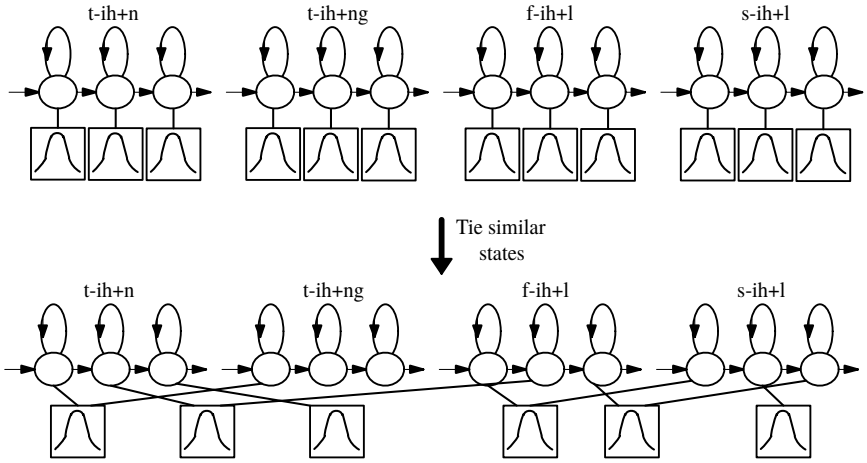

FIGURE 3.6: Example of state tying (taken from Gales, Young, et al. (2008)).

left and right neighboring phone. A model without context-dependent phones would be called a mono phone model (Gales, Young, et al., 2008).

However, if there were N phones, a tri-phone model would raise the number of states to potentially N^3, causing problems due to data sparsity (some tri-phone combinations would be very rare, some would never be seen in the training data). A typical techniques to mitigate this is to merge similar contexts. These can also be data driven, for example with the construction of binary decision trees and tree based clustering (Bahl et al., 1991).

In a tied-state HMM (Young et al., 1994), parameters are not tied at the state but rather at the model level. The aim of a tied state HMM is to ensure that there is enough training data for each set of state output distribution parameters. Figure 3.6 illustrates a tied-state HMM model.

3.7 Decoding

As noted in Section 3.1, the most likely transcription $\tilde{W}$ is part of a search space of all possible transcriptions W, explaining the observation sequence O. While an exhaustive search becomes computationally infeasible, the most likely transcription can be efficiently computed with the Viterbi algorithm (Viterbi, 1967):

Let $\phi_t^{(j)} = \max_\theta p(O_{1:t}, \theta_t = s_j; \lambda)$ be the maximum probability of observing the partial observation sequence $O_{1:t}$ with state s_j at time t and model parameters $\lambda = [\{a_{ij}\}, \{b_j()\}]$, where a_{ij} are transition probability parameters and $b_j()$ is the output likelihood function. $\phi_t^{(j)}$ can then be computed as (Gales, Young, et al., 2008):

$$\phi_t^{(j)} = \max_i \left\{ \phi_{t-1}^{(i)} \right\} b_j(o_t) \tag{3.4}$$

The recursive function is initialized by setting $\phi_0^{(j)}$ to 1 for the initial state and 0 for all others. Every maximization decision is recorded for $max_j \left\{\phi_T^{(j)}\right\}$, a traceback gives the best matching state sequence (Gales, Young, et al., 2008).

3.8 Evaluation

The de facto standard evaluation metric in speech recognition is word error rate (WER). It measures the number of insertions (I), deletions (D) and substitutions (S) necessary to match a reference transcript to a decoded transcript hypothesis:

$$WER = 100 \cdot \frac{I + D + S}{N}$$

where N is the number of words in the reference transcript. WER can be efficiently computed by dynamic programming algorithms, such as the Wagner-Fisher algorithm (Wagner and Fischer, 1974) or another improved variant (Navarro, 2001). Note that WER can be above 100%, as the number of necessary insertions can be higher than the sequence length of the reference transcript.

3.9 Speech Recognition with WFSTs

A finite-state transducer is an automaton that consumes an input sequence while also producing an output sequence. Transitions are weighted, i.e. there is weight encoded with each transition. The weights are used to accumulate an overall quantity along a path. For example, these can be probabilities, durations or penalties (Mohri et al., 2002).

Using weighted finite-state transducers in the decoding process of an ASR decoder allows all relevant auxillary information (phoneme lexicon, language model) to be represented by one compact network (Gales, Young, et al., 2008). The decoding FST is usually built and optimized a priori. This in turn allows very efficient decoding of utterances. FSTs are ubiquitous in speech processing, with popular speech recognition toolkits such as Kaldi (Povey et al., 2011) being based on FST decoding.

A decoding FST is usually constructed by combination ($\circ$) of its relevant sub FSTs. In an HMM based speech recognition system, it is usually combined in the following steps, where G is either a grammar or a language model encoded as WFST and L is the lexicon FST. Then:

$$L \circ G$$

is a transducer that maps sequences of phones to word sequences restricted to G (Mohri et al., 2002).

$$C \circ L \circ G$$

is a transducer that maps sequences of context-dependent phones to word sequences restricted to G. And finally,

$$H \circ C \circ L \circ G$$

is a transducer that maps sequences of HMM states to word sequences restricted to G.

3.10 Language Models

A language model can estimate the likelihood of a word sequence in natural language, one of its main uses in an ASR model. It can also be used as a generative model, i.e. new text can be sampled from the language model. N-gram language models are statistical models that calculate the probabilities, up to an order of n, from text collections.

N-gram language models approximate the conditional probability of a word that follows a word sequence with the estimate (Jurafsky and Martin, 2008):

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1}) \tag{3.5}$$

3.10.1 Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) (Jurafsky and Martin, 2008) is a simple technique to estimate n-gram probabilities:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \tag{3.6}$$

Where $C(x)$ is the count of x, i.e. the number of times x appears in the training text data.

3.10.2 Smoothing

Unfortunately, MLE estimation is not very robust for out-of-vocabulary words and infrequent words. For out-of-vocabulary words and n-grams, a zero probability is assigned, in consequence the model usually overestimates n-grams with non-zero count (Chen and Rosenfeld, 2000).

There are several techniques to combat this issue: discounting, interpolation and back-off. Discounting generally refers to lowering the probabilities of observed n-grams in order to distribute probability mass to unobserved ones (Jurafsky and Martin, 2008). A simple technique is to add one to all counts including unseen ones, this is also known as Laplace smoothing (Lidstone, 1920).

In interpolation models, several individual n-gram models with different n are computed and each n-gram estimate is interpolated from them. Examples of this technique include simple linear interpolation of all models or a recursive linear interpolation as in Jelinek and Mercer (1980).

In back-off models, n-grams are recursively estimated under certain conditions from lower order $n-1$ models. For example an n-gram being observed less than a specific number of times in the training data leads to a back off to a $(n-1)$-gram model, otherwise the $n-$gram is robust enough to provide the estimate on its own. Examples of well known back-off models are Katz smoothing (Katz, 1987) and Kneser-Ney (Ney et al., 1994). For the n-gram language models used in ASR models in experiments of this thesis, Kneser-Ney language models are used.

3.10.3 Neural Language Models

Neural probabilistic language models were first described in (Miikkulainen and Dyer, 1991; Bengio et al., 2003). In such a language model, a neural network estimates the probability distribution over a fixed vocabulary V with a fixed-sized context of n words. Most neural LMs use the next word prediction task for self-supervision, i.e. the network is trained to predict posterior probability for a word $w_n + 1$ following a history $h_i = w_1...w_n$ of previous words.

3.10.4 Recurrent LM

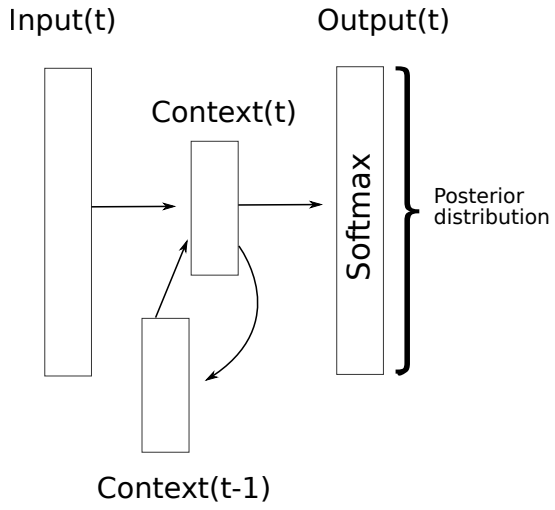

FIGURE 3.7: RNN language model

In RNN language models, a recurrent neural network is trained to predict the next word given a history $h_i = w_1...w_n$ of previous words:

$$P(w_1...w_n) = \prod_i = 1^n P(w_i|h_i) \tag{3.7}$$

The RNN is typically a LSTM (see Section 2.23.1) or GRU to combat the vanishing gradient problem. One of the advantages of a recurrent language model is that the context size is not fixed, since the posterior probability of the next word problem is estimated from the hidden state of the network. In practice, this allows a recurrent model to learn from much longer contexts than what is usually possible with n-gram models, since computation costs do not rise exponentially in n.

3.11 DNN-HMM Acoustic Model Hybrids

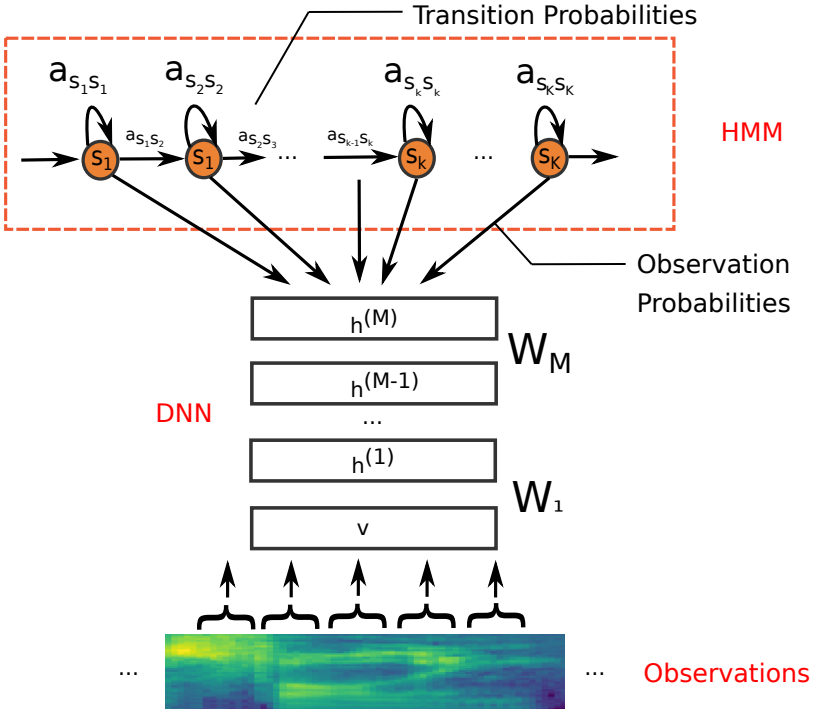

FIGURE 3.8: The DNN-HMM acoustic model. Drawn in the style of Dahl et al. (2012).

The DNN-HMM (Dahl et al., 2012) model is a hybrid HMMs acoustic model, based on DNNs that replace GMM modeling of the acoustic data in conventional GMM-HMMs. Some work on integrating NNs into an HMM predate DNN-HMMs, but the DNN-HMM is the first model to yield large performance gains in recognition accuracy. The architecture is illustrated in Figure 3.8. Input to the DNN are observations, usually in the form of multiple speech frames of speech features concatenated to a super vector. The DNN then computes the posterior probability over the (tied) states of the HMM.

The main training procedure of an DNN-HMM is as follows: First, a conventional GMM-HMM with state tying (see Section 3.6) is trained. Then, all tri-phone states are mapped to senone ids. Senones are tied tri-phone HMM states, see also Section 3.6. Now the training data is force-aligned with the GMM-HMM, i.e. we decode the training data, forcing the decoder to select the best path through the utterance matching the known transcription. The result is a mapping of each frame of

all the training data with a particular senone id. Then the DNN is trained separately on the force-aligned data, usually on a window of several speech frames, to give additional context to the neural network. Finally, the HMM structure is then borrowed and the transition probabilities are re-estimated. The resulting DNN-HMM can also be used to force-align the training data another time, repeating the process to train a new DNN, but this is rarely done in practice.

3.11.1 TDNN-HMMs

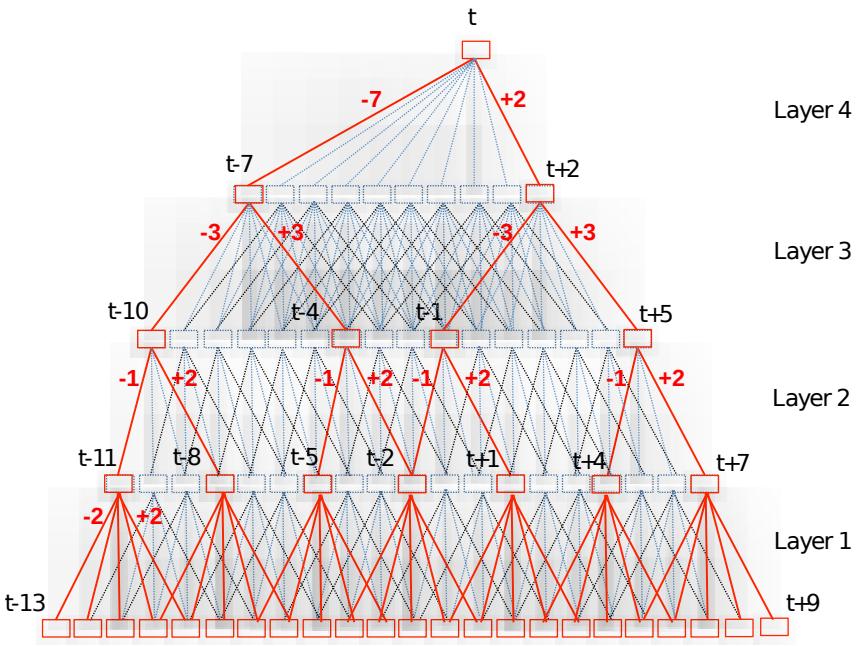

FIGURE 3.9: A TDNN with and without subsampling. Figure taken from Peddinti et al. (2015).

Time-delayed neural networks (TDNNs) have been originally introduced in (Waibel et al., 1990) for phoneme recognition, but were only recently combined with HMMs and trained on larger amounts of training data. In modern TDNN-HMMs, as popularized by Peddinti et al. (2015), subsampling is also used. Figure 3.9 shows a TDNN with and without subsampling. The TDNN is used as a replacement for the DNN-HMM and the training procedure is similar to bootstrapping from a GMM-HMM. Due to the hierarchical structure of a TDNN, the context to the local classification can be much larger, while subsampling keeps the number of parameters comparatively small. Since the weights of the TDNN are tied across time steps, they offer time invariance in speech signals, similar to translation invariance in CNNs. TDNNs are one-dimensional convolutional networks applied to time series (Murphy, 2014) and as such regarded as precursor to CNNs (Peddinti et al., 2015) (see Section 2.15).

3.12 Connectionist Temporal Classification

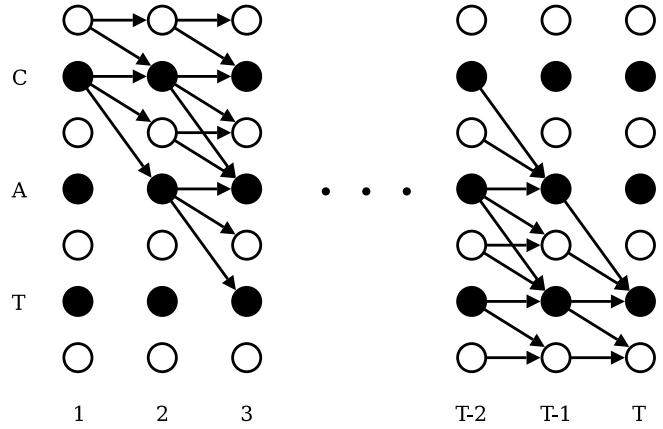

FIGURE 3.10: Possible paths aligning the word "cat" to T many acoustic observations. Black nodes are non-blank labels, while a white nodes are blank labels. The sum of all path probabilities can be efficiently computed with dynamic programming by visiting every node only once. Figure taken from Graves et al. (2006).

Connectionist Temporal Classification (CTC) (Graves et al., 2006) is an objective function that allows to compare a sequence of observation to a sequence of labels without any alignments, where the observation sequence is longer than the label sequence. In these models, no bootstrapping with conventional GMM-HMM is needed, as no frame-level labels are needed. CTC is fully differantiable and can be used with standard back propagation and SGD.

CTC introduces a blank label "_", i.e. to indicate no label. The shorter label sequence can be expanded with blanks and repetitions to match the observation length. There are an exponential number of ways (paths) to match a label sequence to an observation sequence in this manner and CTC sums over all possible paths:

$$p(l|x) = \sum_{\pi \in \mathscr{B}^{-1}(l)} p(\pi|x) \tag{3.8}$$

This defines a conditional probability of a given labeling l as the sum of the probabilities of all paths π corresponding to it. $\mathscr{B}$ is a mapping function that removes repetitions in sequences, e.g. $\mathscr{B}$(aa_bbb_b_a) = $\mathscr{B}$(aa_b__b_a) = a_b_b_a. $\mathscr{B}^{-1}$ is the corresponding inverse of this function. CTC then uses dynamic programming to compute the sum, with the CTC forward backward algorithm. See Figure 3.10 for an illustration of all possible paths aligning the word "cat" with acoustic observations. Summing all paths naively would otherwise be computationally unfeasible for longer sequences, but can be done efficiently with the CTC forward backward algorithm, c.f. (Graves et al., 2006).

3.13 Unsupervised ASR

In fully unsupervised ASR, the training process for an ASR system uses text data and audio without any alignments. That means there are two separate datasets of audio and text of the same language, with no transcriptions. Gathering such datasets is in most cases easily accomplished if enough text and audio resources of a language exist on the internet. In principle, this means that an alignment needs to be found between speech and text. In contrast to supervised ASR, the alignments have to be found not in short segments of parallel data, but between two large and independent collections of data. It can be approached by first learning self-supervised phonetic representations (see also Chapter 6) and then training another model to match these representations to symbolic sequences of real phonemes generated from text. In the following, we describe very recent advancements in making unsupervised ASR tractable in practice:

A precursor to fully unsupervised ASR is unsupervised phoneme recognition with known boundaries (Liu et al., 2018). Here, real phone boundaries are known in the training process, but the phone identity is inferred from the data. The main idea is to make use of a Generative Adversarial Network (GAN) (Goodfellow et al., 2014) to discriminate between imposter phoneme sequences and real ones. A generator takes as input discovered and unsupervised phone representations to output imposter phones. A discriminator learns to distinguish between imposter and real phone sequences. Over time, the generator might make use of the input representations to output good imposter phones. This in turn means a connection between the discovered and real phones is made.

In Unsupervised ASR, this principle is used in a full ASR system, so that a speech recognition system can be trained on audio and text data without matching transcriptions. Figure 3.11 illustrates the general training procedure. Through acoustic unit discovery, basic units of speech are discovered and embedded. A typical goal is to learn representations that attenuate speaker and channel differences and emphasizes phonetic content. Boundaries are not known a priori and need to be discovered from data as well, to enable unit discovery with varying unit lengths. Embeddings of these units are then the input to GAN training, effectively yielding a basic phone recognition model after training. For a full ASR system, regular FST-based decoding can be used to transcribe speech into words (see Section 3.9).

In (Baevski et al., 2021), a practical unsupervised ASR was proposed. It uses unsupervised speech representations that are learned with Wav2Vec (Schneider et al., 2019; Baevski et al., 2020b) to learn representations suitable for phone recognition and acoustic units discovery through clustering (K-Means) to obtain larger segments of varying size. These segments are then represented with average pooled dense vectors from the Wav2vec embeddings. Wav2vec is a self-supervised representation learning model based on Contrastive Predictive Coding (see below) that

learns embeddings of raw speech data in the time domain. Depending on the language, the proposed learning scheme uses a phonetic lexicon to translate texts into phoneme sequences and may use G2P on out-of-vocabulary words. The training process may be iteratively rerun.

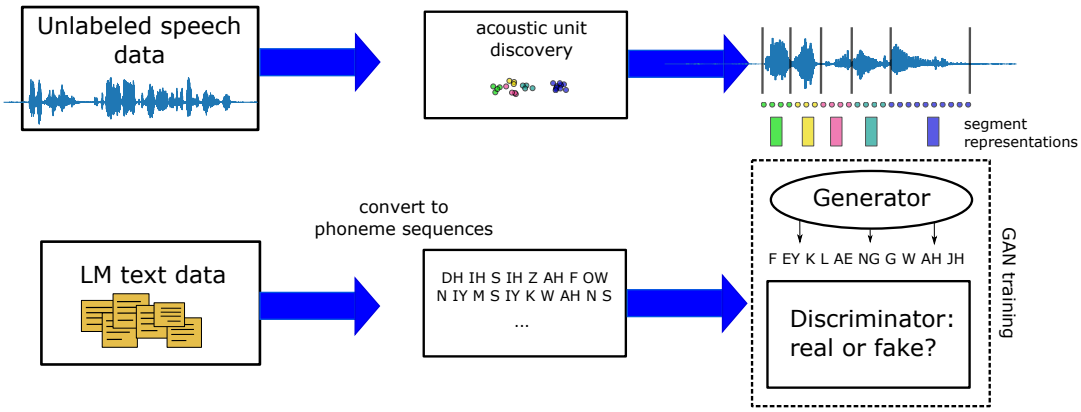

FIGURE 3.11: Unsupervised ASR training with generative adversarial networks. Drawn in the style of Baevski et al. (2021).

3.13.1 Contrastive Predictive Coding

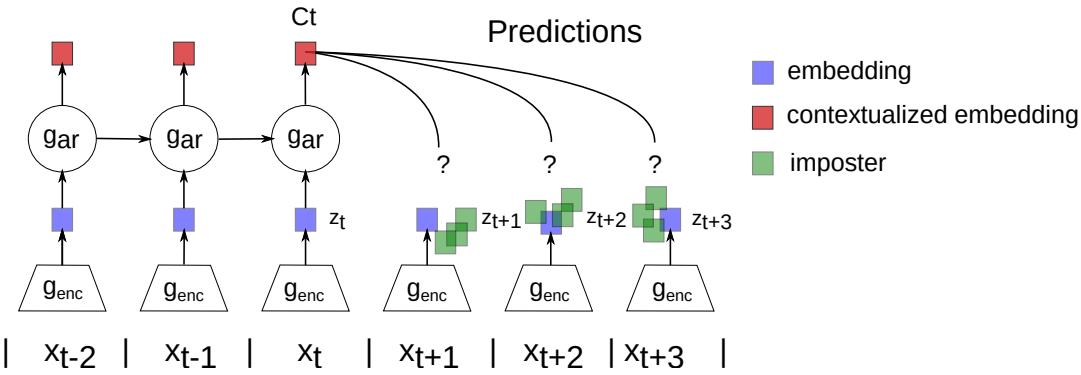

FIGURE 3.12: Contrastive Predictive Coding in a 1D time series. Drawn in the style of Oord et al. (2018).

Contrastive Predictive Coding (CPC) (Oord et al., 2018) is a universal method for unsupervised learning where individual data points are spatially connected or are from time series (or can be interpreted as such). It has been applied to modalities such images, text, speech and reinforcement learning. It has recently become popular in learning self-supervised representations for speech. Figure 3.12 depicts the CPC architecture for a 1D time series. Instead of predicting future data points, future representations are predicted. The main idea is that this forces the model to learn representations that are good for being predicted and disregards unpredictable information, such as noise. It is made tractable by using contrastive learning. The

main task is to discriminate a true representation of a future data point from imposter ones, as encoded by the model. Imposter representations are usually sampled from the dataset that the model is trained on.

Fully trained, the model can encode in two types of representations. Either the embeddings from the encoder g_{enc} can be used for downstream tasks, or the contextualized embeddings from g_{ar}. As context plays an important role in speech processing, typically g_{ar} is used for downstream tasks in the speech modality.

A typical choice for g_{ar} are neural networks that a frequently used in temporal learning, such as LSTM or GRUs (see Section 2.23). g_{enc} is usually convolutional (stacked 1D convolutions). It is common to use the raw 1D waveform time domain data in speech. In 16kHz recordings, speech is encoded in intervals of 160 data points, yielding 100 embeddings per second, as is typical for other speech features such as MFCCs. If the goal is to learn representations for speech recognition, negative samples are typically sampled from the same speaker (within speaker negative sampling).

3.14 Distributions in Spoken Language

One of the first empirical analyses of distributions in spoken language appears in (Zipf, 1929). Zipf's famous fundamental thesis of spoken language is:

> Principle of Frequency. The accent, or degree of conspicuousness, of any word, syllable, or sound, is inversely proportionate to the relative frequency of that word, syllable, or sound, among its fellow words, syllables, or sounds, in the stream of spoken language.

Zipf's law has been named after the author of this hypothesis. It can also be observed in collections of written text and has been observed in all natural languages, where word frequencies are inversely proportional to their rank when sorted by frequency. After the most common word in written English ("the"), the second-most common word "of" appears approximately 1/2 as often, the third most common word 1/3 as often and so on.

Zipf also observed that high-frequency words are usually more reduced and shorter than low-frequency words (to make the transmission of thoughts more efficient from speaker to speaker). As words enter common use and are used more frequently, they often get shorted: e.g. the word "gas" is a shortened form of "gasoline", "movies" was derived from "moving-pictures" and "good-bye" from "God be with you" (Zipf, 1929).

In the following, we empirically show that Zipf's law holds for spoken corpora used in this thesis and that it can be observed in phoneme frequency/rank relations as well as word frequency/rank relations.

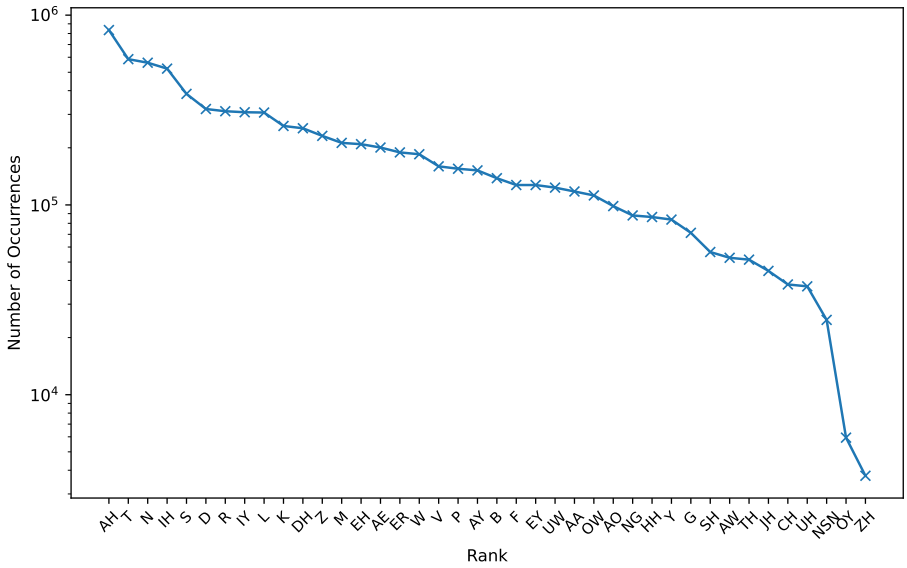

FIGURE 3.13: Most frequent phonemes (SAMPA notation) in English transcriptions of the TED-LIUM 2 corpus.

3.14.1 Phoneme frequencies

Figure 3.13 shows the most frequent phonemes in English, as observed on automatic phonetic alignment on the TED-LIUM 2 corpus (Rousseau et al., 2014). The ASR model to generate those alignments was trained with phonetic lexicon entries from cmudict (Weide, 2005) in SAMPA notation (Wells et al., 1992). In contrast to the International Phonetic Alphabet (IPA), SAMPA can be typed with standard ASCII letters and symbols. It is commonly used for phoneme lexicons in English ASR.

The two most frequent phonemes (as estimated on TED-LIUM 2) can unsurprisingly be found in very frequent words:

"AH" is in words such as "and" (AH N D), "than" (DH AH N) and "was" (W AH Z). "T" is part of the words "at" (AE T), "got" (G AA T) and "cat" (K AE T). The least frequent phonemes in English (as estimated on TED-LIUM 2) are OY and ZH. "OY" can be found in words such as oil (OY L) voice (V OY S) and royal (R OY AH L). "ZH" can be found in measure (M EH ZH ER), casual (K AE ZH AH W AH L) and occasion (AH K EY ZH AH N).

3.14.2 Average phoneme durations

In the following, we analyze average phoneme durations. We derive durations from automatic alignments of speech corpora. This allows to gain statistics over a vast number of occurrences, at the expense of accuracy as compared to manually annotated phoneme boundaries. The time durations are given as the number of frames aligned to a particular phoneme by the ASR system – this means that accuracy is also

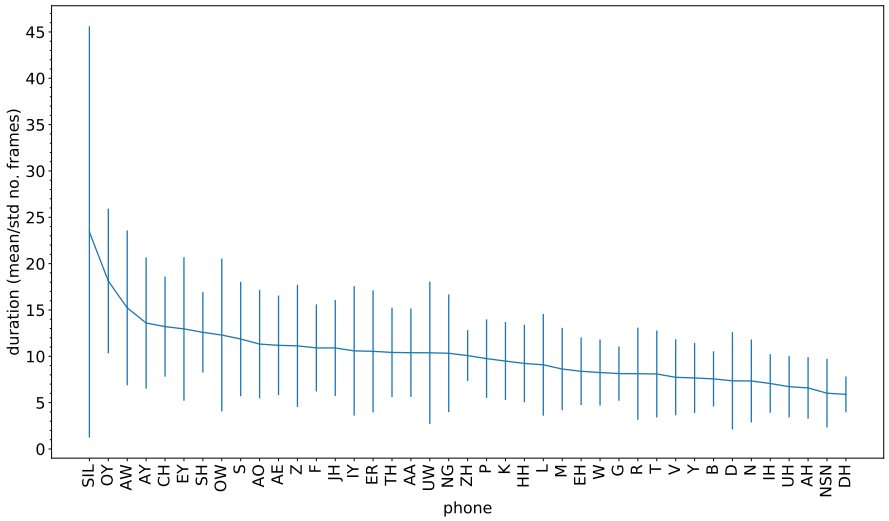

FIGURE 3.14: Average phone durations (with standard deviation) of each phoneme used in TED-LIUM, as automatically force-aligned by a GMM-HMM acoustic model over the full training set. The phonemes are sorted from longest average duration to smallest.

dependent on frame spacing being used with the acoustic model. In all of the following plots, the acoustic model used to align the phoneme boundaries was trained with a standard spacing of 10ms, thus a rough estimate of the actual duration in ms is the number of frames multiplied by 10ms. Figure 3.14 shows average durations for the 41 phonemes in the TED-LIUM corpus, as estimated on training corpus. In total 8,341,339 phoneme occurrences were counted, the average length of aligned frames per phoneme over all occurrences is 9.76 frames per phoneme.

3.14.3 Word frequencies

Figure 3.15 shows a logarithmic rank/frequency plot of all the tokens in the TED-LIUM 2 corpus and Figure 3.16 shows the same for the Switchboard corpus. Each of these corpora contain different modes of spoken language. TED-LIUM is based on TED talks[2], so the utterances are generally from a presentation speaking style. Switchboard is compiled of natural dyadic conversations recorded over the telephone. The most frequent words in TED-LIUM are function words: "the", "and", "to", "of" and "a". In contrast, in Switchboard the word ”I” is the most frequent word, followed by function words. ”You” is also very frequent, owing to the fact that these words are very frequent in conversations, but not in presentations.

[2] https://www.ted.com/

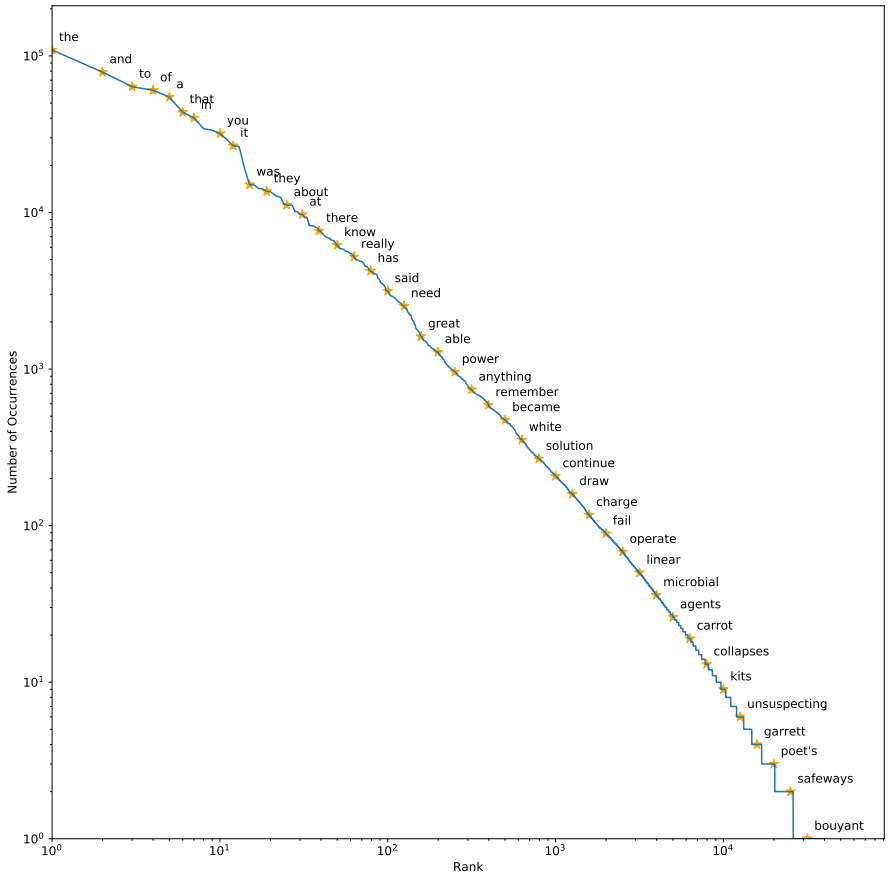

FIGURE 3.15: Frequency versus rank for all words occurring in the TED-LIUM 2 corpus. Both axes are plotted on a logarithmic scale.

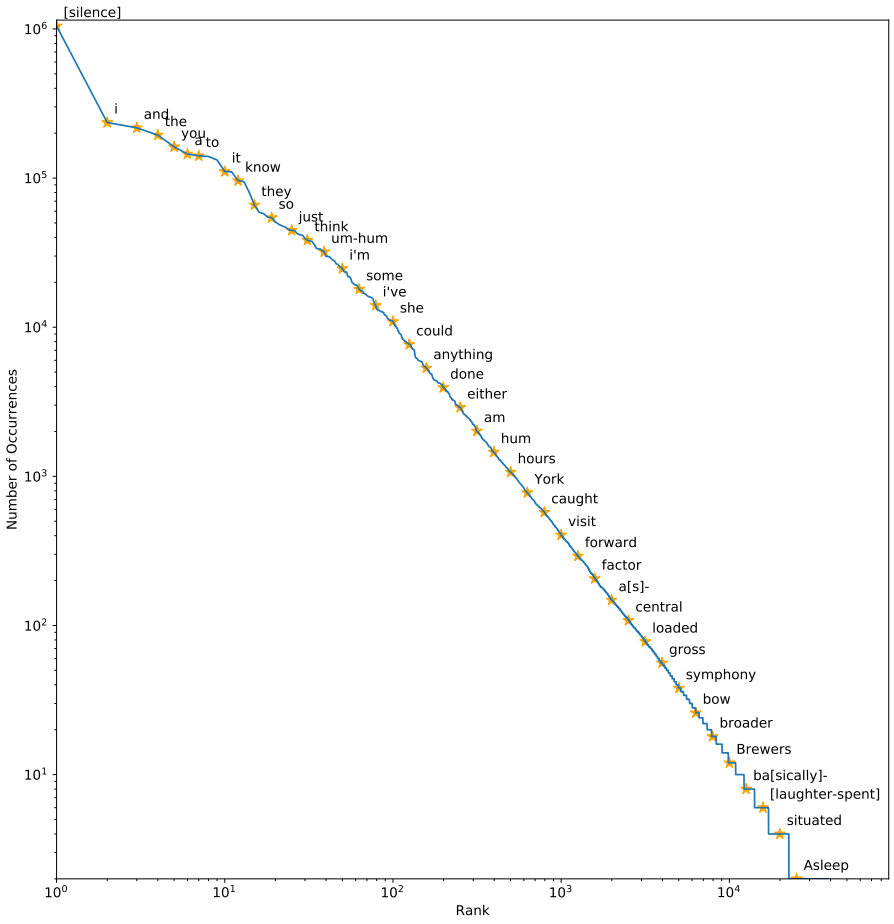

FIGURE 3.16: Frequency versus rank for all words and special tokens occurring in the Switchboard conversational corpus. Both axes are plotted on a logarithmic scale.

Chapter 4

Supervised Representation Learning for a Paralinguistic Speech Task

4.1 Introduction

In this chapter, we discuss representation learning on an isolated audio classification task. The study of paralinguistic speech involves perceptual speech phenomenons that lie beyond the pure transcriptional content of spoken speech (Schuller, 2012). The term paralanguage (Trager, 1958), literally 'alongside language', was coined by linguist Archibald Hill. With the advent of computational paralinguistics, such phenomena are attained by automatic means. Automated gender recognition, age estimation (Schuller et al., 2010) and emotional state classification (Schuller et al., 2009) are some examples with a range of diverse applications in e.g. health care, commerce (advertising) and education. Computational paralinguistic speech tasks can encounter data sparsity problems, as it can be more challenging to find or generate larger amounts of labelled training examples for the desired domain than in other areas of speech research.

It is also not easy to argue what constitutes good features for a specific paralinguistic regression or classification task and for some of these tasks, even human performance can be suboptimal (Schiel, 2011). However, a specific set of speech features developed in a particular paralinguistic domain can often be applied to another paralinguistic domain with success (Bone et al., 2011). It would certainly be desirable if features could adapt automatically to a new problem domain, instead of the need to manually engineer and tune features. Neural network approaches to image, audio and text classification problems enable such a learning paradigm (see also Section 2.7): promising feature representations can also be learned automatically from the data of the problem domain (Hinton, 2007).

In the following, we describe the methods we used for our entry in the Eating Condition (EC) challenge of the Interspeech 2015 Computational Paralinguistics ChallengE (ComParE2015) (Schuller et al., 2015). We use the iHEARu-EAT corpus (Hantke et al., 2015), which consists of 945 training utterances and 469 test utterances. The task is to predict the kind of food someone eats while s/he speaks and it is posed as a 7-class sequence classification problem with six specific kinds of food and one class for speech without food (No Food, Banana, Crisp, Nectarine, Haribo, Apple, Biscuit).

Nonetheless, we think the ideas should be transferable to other paralinguistic and also audio classification tasks. We particularly address the challenge of having only few labelled example sequences for each class and show how the training process makes meaningful use of out-of-domain data with transfer learning (Caruana, 1997; Pan and Yang, 2009) and data augmentation with synthetic training data examples. Our main idea is to train local classifiers that automatically learn representations on smaller windows of an audio sequence and to combine multiple classifications from such windows to obtain a classification result for a full sequence input. Alongside better classification performance when compared to systems trained only on per-sequence openSMILE (Eyben et al., 2010; Eyben et al., 2013) baseline

features, our window approach has the advantage that it also enables us to extract exact positions of an audio sequence that are most relevant for its assigned class.

4.2 Methods and approaches

Our approach bears similarity to the proposed deep learning architecture for language identification in (Lopez-Moreno et al., 2014). We have obtained good results by replacing Deep Neural Networks (DNNs) in this pipeline with Convolutional Neural Networks (CNNs) (LeCun and Bengio, 1995; LeCun et al., 1998; Krizhevsky et al., 2012), which have demonstrated recognition accuracy better than or comparable to humans in several visual recognition tasks (Taigman et al., 2014; He et al., 2015).

4.2.1 CNN-based audio sequence classification

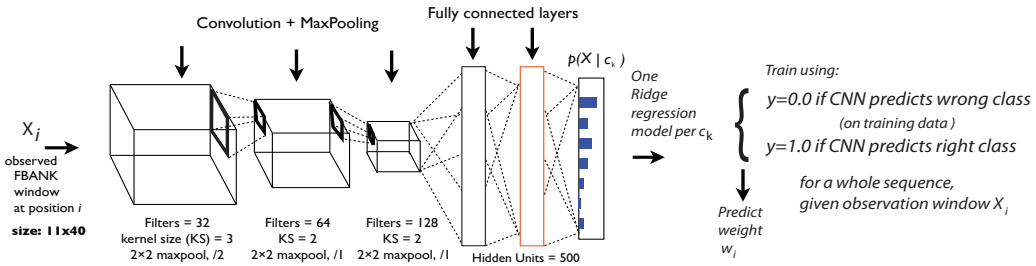

FIGURE 4.1: The CNN architecture used for learning on local FBANK patches x_i of a full sequence X. Contribution weights w_i are learned separately with linear models (one for each class), which assign a weight to the local classification's contribution to the full classification of a sequence.

Similarly to the recent work in applying CNNs to the audio and speech domain (Sainath et al., 2013; Lee et al., 2009), we interpret overlapping windows in a frequency spectrum representation as fixed-dimensional 2D input to the CNNs (comparable to grey scale images). We have used windows of 40-dimensional log mel filterbank features (FBANK) as input to the classifier. Each FBANK vector is calculated from a time frame of 25 ms of raw audio and we use a window of +/- 5 feature vectors around a central feature vector as input to the CNN. A window has thus a height of 40 and a width of 11 elements, totalling 440 inputs. We shift the FBANK window by 2 vectors through all FBANK frames of the signal.

Figure 4.1 illustrates our CNN architecture, along with a weight prediction scheme using linear models. We follow the standard CNN design of having alternating convolutional and maxpool layers followed by fully connected layers (Krizhevsky et al., 2012). Initially, we used three convolutional and maxpool layers followed by two fully connected layers. This is much smaller than proposed architectures for the

ImageNet classification task (Krizhevsky et al., 2012; He et al., 2015) to account for the smaller input size of the CNN and the smaller amount of training data available in iHEARu-EAT. We also use smaller convolution kernel sizes of 3x3 and 2x2, to achieve an implicit regularization effect (Simonyan and Zisserman, 2015). One particular challenge of audio sequence classification is that the length of unseen sequences is not fixed. We have implemented a simple majority voting strategy that selects the highest count of individual class predictions across all windows of the sequence and a weighted majority voting that learns contribution weights with linear regression models that we describe in the next subsection.

4.2.2 Weighted Majority Voting

We use one Ridge linear regression (Hoerl and Kennard, 1970) model per class c_i, to predict the weight of a window's prediction towards the full prediction of the sequence. A CNN model is trained on all windows X_i of all audio sequences $seqs$ = $seq_1, seq_2, ..., seq_j$, $X_i \in seq_j$. Each window is assigned the output class c_i of the sequence it belongs to. Then, for each X_i we let the model predict the most probable class c_i given its window of observation. We assign $y = 0.0$ for each wrong prediction and $y = 1.0$ for each successful prediction on the training data the model was trained on. The assumption here is that some windows of a sequence contain no discriminative information (e.g. only silence). A linear regression model m_{c_i} should then learn the connection between all predicted class probabilities of its class c_i given the window of observation X_i and weights $w_j = y$. Ideally, these weight should be closer to 0.0 if the predicted input class probabilities of the observation X_i are likely to contribute to an incorrect classification and 1.0 if it supports the full sequence's true class. Then, instead of a simple counting vote to aggregate individual classifications, we add the predicted weight value w_i towards a final vote of class c_i.

4.2.3 Regularization

We employ dropout (see also Section 2.14.5) as a first measure to counter overfitting in training our networks (Srivastava et al., 2014). It has been specifically proposed for cases where labelled data is scarce and thus where overfitting is likely to occur. It works by randomly omitting a certain percentage of nodes in the network at training time, while using the full network at test time.

4.2.4 Data Augmentation

Data augmentation has been widely used in neural network based pattern recognition tasks (LeCun et al., 1998; Simard et al., 2003; Krizhevsky et al., 2012). In image processing, data augmentation techniques are usually intuitive and transformations such as translation, deformation and reflection (Krizhevsky et al., 2012) have led

to significant improvements in recognition accuracy. Inspired by these image augmentation techniques, we employ a simple audio augmentation technique: We artificially generate pitch shifted versions of the original training files and add them to the training set. We make use of the Rubberband utility[1] to generate new files shifted by one semitone upwards and downwards, without changing the length of the recordings. We set the "crispness" level to 6, intended for drum tracks, as this produced the best sounding pitch-shifted speech output and was deemed somewhat fitting for smacking noises.

4.2.5 Transfer Learning

The idea of training a neural network on a related problem first and transferring its weight configuration for a target problem instead of random initialization is an old one (Pratt et al., 1991; Pratt, 1992). However, literal and naïve weight transfer has been shown to not only be ineffective but also detrimental to the classifiers performance (Pratt, 1992). These results have now been challenged on several visual recognition tasks with deep neural networks, where representation transfer has been shown to be astoundingly effective (Sharif Razavian et al., 2014). Azizpour et al. (2015) have recently systematically identified the list of factors that affect the transferability of CNN representations and direct transfer of weight configurations for visual recognition tasks, with exhaustive experimental evidence showing how these factors should be set.

An important factor is, unsurprisingly, how related the two problems are. It is then very important, dependent on the relatednesses of the tasks, how many of the first layers are transferred, while the remaining layers are randomly initialized as usual. We have chosen the Voxforge[2] corpus for transfer learning as it is freely available and easily accessible. We build a 7-class language classification problem, where the sequence should be classified as German, French, Spanish, Italian, Portuguese, Dutch or Russian. We selected 3000 utterances for each language from the Voxforge dataset, which is roughly 30 times larger than the iHEARu-EAT dataset. We then pretrained our CNN model on this problem first and then transferred the model's weight matrices to a new CNN model, which we trained on the iHEARu-EAT data. We have both tested the naïve transfer learning approach where we transfer all layers and one where we only transfer the weights of convolutional layers, but initialize the fully connected layers randomly.

4.2.6 Hyperparameters

For training DNNs and CNNs we use Nesterov's Accelerated Gradient Descent (NAG, Nesterov momentum) (Nesterov, 1983; Sutskever et al., 2013). For all random weight initializations, we have chosen He-initialization, as described in He

[1] `http://breakfastquay.com/rubberband/` (last accessed: December 2021)
[2] `http://www.voxforge.org/` (last accessed: December 2021)

Method	**UAR 7-class**	**UAR 2-class**
Baseline features (SVM)	56.0% (+/- 2.7%)	**94.3%** (+/- 2.3%)
Baseline features (RF)	53.6% (+/- 2.9%)	91.4% (+/- 4.3%)
net0: Baseline features (DNN)	**59.2%** (+/- 4.2%)	93.8% (+/- 3.6%)
net1: CNNs (simple maj. vote)	57.8% (+/- 3.6%)	94.4% (+/- 1.8%)
net2: net1+dropout	61.6% (+/- 6.1%)	94.1% (+/- 2.8%)
net3: net2+weighted maj.	63.1% (+/- 1.4%)	95.0% (+/- 2.2%)
net4: net3+naïve transfer learning	62.2% (+/- 3.6%)	95.4% (+/- 2.2%)
net5: net3+transfer learning	**65.5%** (+/- 2.2%)	**96.6%** (+/- 1.4%)
CNN ensemble (net0+net2+net3+net5)	**67.2%** (+/- 2.6%)	**96.3%** (+/- 2.0%)

TABLE 4.1: UAR mean and standard deviation with 5-fold speaker independent cross validation, using the training data of iHEARu-EAT. In each split the data of 16 speakers were used for training the classifier(s) and the utterances of the remaining 4 speakers were used to evaluate classification performance.

et al. (2015). We use categorical cross entropy as objective loss function, i.e. the measure of error between computed and desired target outputs of the training data, which we minimize. One of the downsides of gradient-based optimization methods, which are primarily used in DNNs and CNNs, is that they usually introduce additional hyperparameters that govern the behavior of the optimization techniques. We start with reasonable defaults and follow best practices: 0.02 is chosen as the start learning rate and 0.9 as the start momentum. For transfer learning, we slightly reduce the initial learning rate to 0.01 (Azizpour et al., 2015). Then, we linearly reduce it with each epoch, so that the learning rate for the last epoch is 0.0001. Similarly, momentum is linearly increased (Sutskever et al., 2013) to 0.999 in the last epoch. We track performance of the network for each epoch on 1% of all patches, which can be seen as held-out classification performance on known speakers. We train all our cross-validation models for 400 epochs. We set dropout to 0.1,0.2 and 0.3 for the first, second and third convolutional layers and 0.5 for all fully connected layers.

4.3 Results

We implemented the proposed approaches and methods in the previous section using Nolearn[3]+Lasagne[4] and Scikit-learn (Pedregosa et al., 2012) in Python. Since Lasagne is based on the Theano (Bergstra et al., 2010) architecture, we could use GPUs to speed up computations. The official development evaluation is unweighed average recall (UAR) with leave-one-speaker-out cross-validation (LOSO-CV). This proved to be computationally impractical for us, as computing one CNN model can take up to one day on a GPU (Nvidia Geforce 970), depending on the number of training epochs used. We opted to do 5-fold cross validation over the speakers instead, with 400 epochs of training. Table 4.1 shows our results for this cross validation method. While this allowed us to test different ideas quicker, it makes our results not comparable to the baseline LOSO-CV evaluation of the challenge (61.3% 7-class UAR) (Schuller et al., 2015). We thus recomputed the baseline results on exactly the same split with 3 different classifiers: SVM (C=0.001), Random Forests (RF) and a 3-layer DNN with dropout. CNN results were computed with the architecture shown in Figure 4.1. We performed mean normalization prior to training on the FBANK patches. For all results shown, we have kept the number of hidden units in the fully connected layers at 500. Preliminary experiments showed that setting hidden units to 1000 or 250 in the two last fully-connected layers of the CNN yielded worse results.

Without any regularization method, a CNN on FBANK features actually performs worse than a regularized DNN on per-sequence baseline features (net1). However, if we use dropout, we are able to out-perform the results obtained with the baseline features (net2). Replacing the simple majority voting with our weighted majority voting based on linear regression models also improves performance (net3), but using transfer learning naïvely by transferring all weights from the source task to the target task produces worse results (net4). However, when the last two fully connected layers are not transferred but initialized randomly, we found transfer learning to be surprisingly effective (net5), showing a large single-technique contribution. If we combine the class probability estimates of differently trained CNN models by their harmonic mean, we can further improve our score, beyond the performance of any single method (ensemble).

The first ensemble model for predicting on the test set was retrained on the full training set with the same parameters. With 400 training epochs, this produced a 70.0% 7-class UAR on the official test set, see Table 4.2. We proceeded to make some changes which extended training time considerably (> 5 days per model) and were thus only evaluated on the test set: We added one additional convolutional layer and pooling layer before the network's fully connected layers with the same parameters as its previous convolutional and pooling layer and extended the input window to 18

[3] `https://github.com/dnouri/nolearn` (last accessed: December 2021)
[4] `https://github.com/benanne/Lasagne` (last accessed: December 2021)

Submitted system	**UAR**	**Acc.**
8-layer CNN ensemble (400 epochs)	70.8%	70.0%
+ 10-layer CNN ensemble (3000 epochs)	73.6%	74.4%
+ 10-layer CNN with data aug. (3000 epochs)	75.4%	76.1%
+ Log. regression on CNN features (trained with SGD, 1000 epochs)	**75.9%**	**76.6%**

TABLE 4.2: 7-class UAR and accuracy scores on the official test set, as reported by the official challenge test submission system. Each new submission is an ensemble which also includes previous models. Five individual submissions were allowed.

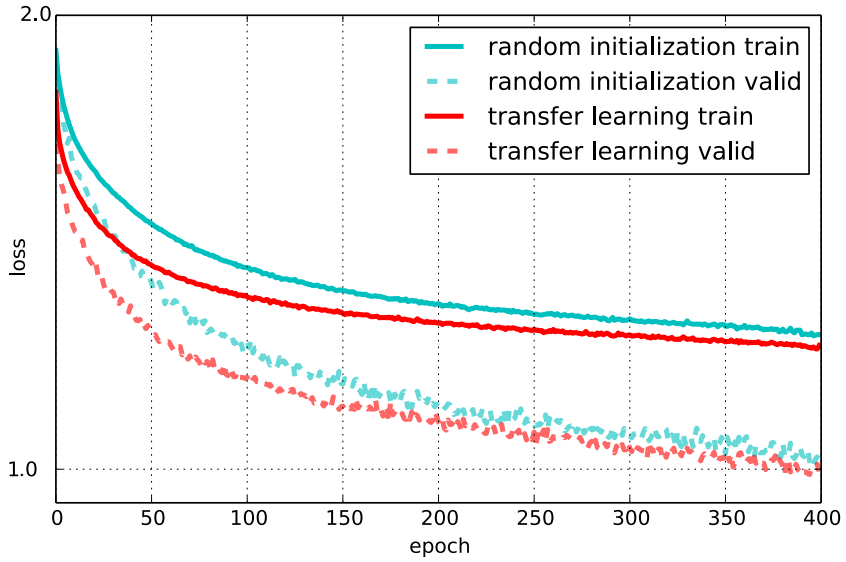

FIGURE 4.2: Minimizing the objective function: training loss per epoch for a CNN with completely random initialization and one with transferred weights. Validation loss is measured on held-out data from already known speakers.

frames with 60 dimensional FBANK features. We also trained the network for 3000 epochs instead of 400 epochs. An ensemble network trained with this larger network raised our test score to 73.6% UAR. Finally, adding a CNN to the ensemble that was additionally trained on augmented pitch-shifted training examples and logistic regression classifiers trained on CNN feature embeddings resulted in an UAR of **75.9%**, which is 15% better than the baseline UAR of 65.9% reported in (Schuller et al., 2015). Figure 4.2 compares validation loss and training loss per epoch for CNNs with completely random initialization and transferred convolutional kernel weights from the Voxforge language identification task, where the latter gives an advantage in all epochs.

4.3.1 Insights

Figure 4.3 shows an example classification on an at training time unseen speaker. The correct answer is 'Crisp', which the model assigned correctly. Note that the 'No

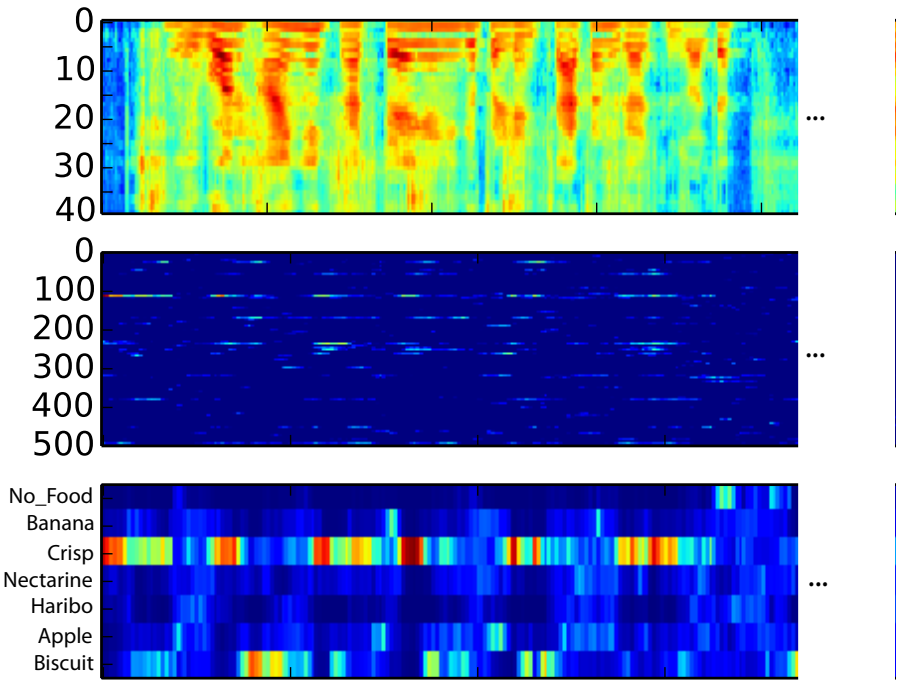

FIGURE 4.3: Classification of (yet unseen) train utterance `train_0112.wav` by a CNN model. The correct class is 'Crisp'. From top to bottom: FBANK features, CNN embedding and the 7 class probabilities for its corresponding window. Orange/red colours represent higher values, while dark blue represents zero.

food' class is relatively inactive for almost the entire utterance, which suggests that a small window of a whole utterance is sufficient to decide if someone eats while he speaks or not. The class 'Crisp' has several distinct spots where classification probabilities are very high, which correspond to characteristic crunch noises in the audio. Our classification pipeline is capable of producing insights, as it can point to sound examples and exact positions in an utterance which are most characteristic and discriminative of its class.

4.3.2 Challenge Results

The final system was submitted to the Interspeech 2015 Computational Paralinguistics Challenge (ComParE 2015). The resulting final UAR of 75.9% on the closed test set scored second place among 15 competing systems (including the baseline), see Table 4.3[5]. Note that there is unfortunately no further information than the UAR score for systems without an accompanying Interspeech 2015 paper. We still list these challenge entries in Table 4.3 and denote these systems with N/A.

Most systems made use of OpenSMILE features (3, 6, 7, 9). Otherwise the system in Wagner et al. (2015) computed filters on raw waveforms, Kaya et al. (2015) uses

[5]We thank Anton Batliner for sending us slides of the summary results talk at Interspeech 2015.

Rank	System	UAR
1.	MFCC+PLP, Speaker z-normalization applied after speaker clustering, Kernel Extreme Learning Machines (Kaya et al., 2015)	**83.1**
2.	**Our system**, raw FBANK features, CNN ensemble network with transfer learning (Milde and Biemann, 2015)	**75.9**
3.	Praat to extract HNR + OpenSMILE features + i-vector speaker normalization, SVM (Kim et al., 2015)	74.6
4.	N/A	73.6
5.	N/A	68.9
6.	OpenSMILE features, Shallow NN (Pellegrini, 2015)	68.4
7.	Wrapper-based feature group selection on OpenSMILE features, SVM (Pir and Brown, 2015)	67.9
8.	Hierarchical classification, signal processing (HP and EP filters) (Wagner et al., 2015)	67.6
9.	OpenSMILE features, feature selection + hierarchical classification, SVM (Prasad and Ghosh, 2015)	67.3
10.	N/A	66.1
11.	baseline, SVM on OpenSMILE features (Schuller et al., 2015)	65.9
12.	N/A	65.1
13.	N/A	64.4
14.	N/A	60.7
15.	N/A	45.5

TABLE 4.3: Overview of the submitted systems and methods used for the ComParE2015 iHEARu-EAT subtask challenge. Not all submissions to the challenge resulted in a paper at Interspeech. We listed these as N/A in the table, but neither the authors nor the methods used are known.

standard ASR MFCC+PLP features. We use standard ASR frequency features (40 dim FBANK), as these are closer to the raw audio signal.

Feature selection of the baseline OpenSMILE features and hierarchical classification were explored by some participants (7, 8, 9), with small improvements over the baseline scores in system 11. The systems by Kaya et al. (2015) and Kim et al. (2015) (1, 3) made use of explicit speaker adaptation and clustering. In both cases, despite very different machine learning models (SVM vs. Kernel Extreme Learning Machines (Huang and Siew, 2004; Huang et al., 2006)), a large increase in UAR was noted. The authors of the winning system noted an increase from 70.4% to 77.0% UAR solely due to speaker normalization for the non-ensemble system (Kaya et al., 2015), while Kim et al. (2015) observed that UAR increased from 65.3% to 75.7% due to speaker normalization (cross-validated on the training set).

Integrating some form of explicit speaker clustering and speaker adaptation might have helped to improve our system as well. In Chapter 5 of this thesis, an unsupervised speaker clustering method is developed and used in the context of ASR systems.

4.4 Conclusion

The eating challenge (EC) provides an interesting audio classification task, where relatively few samples per class are available, but good automated classification performance can still be attained. Prior to 2012, when neural networks were renowned for their propensity to overfit, they would have probably not been considered for such a learning task. Without techniques like dropout and transfer learning, it would have been difficult to beat the baseline results. On the cross-validation results, the use of additional speech data via transfer learning to initialize early layers of the CNN showed a large single-technique improvement and overall best performance was achieved with a voting scheme between several CNNs. At the expense of a longer training time, we could also further improve our test set result with data augmentation and a larger network, yielding a final improvement of 15% relative to the baseline score.

We have opted for a pipeline where individual classification results are averaged across a sequence, to produce a combined classification result for the full sequence. Examination of the feature representation of the last fully connected layer suggests that such an automatically learned representation is fairly sparse. Our results concerning the transferability of convolutional neural network weights are in line with newer results and insights obtained on visual object recognition tasks (Azizpour et al., 2015): It is paramount to only transfer the first layers to the target task and to initialize the last layers of the target network at random, since these are very specific to the target task, while the first layers of a CNN can encode more generic feature transformations. A different dataset than the Voxforge corpus would probably be a better fit for this kind of transfer learning, as literature and intuition suggests that the closeness of the source to the target task is related to how well the transfer learning works. But especially considering that the Voxforge corpus is recorded in very mixed and varying conditions and that its recording quality is varies due to the crowdsourced nature of the project, the obtained results are surprisingly good. We believe that the classification pipeline presented in this chapter could also be interesting to other – not only paralinguistic – audio classification tasks.[6]

4.5 Additional Remarks

This study on transfer learning was previously published in (Milde and Biemann, 2015), when transfer learning and the use of CNNs in speech and paralinguistic speech was very recent. It was one of the first such studies showing that even on rather small labeled paralinguistic speech datasets, CNNs can be trained effectively and learn useful representations that outperform standard paralinguistic features of speech signals by a large margin. This sparked a now well-known discussion in (Trigeorgis et al., 2016) about the relevance of manually designed features in the field

[6]Source code: `https://github.com/bmilde/deepschmatzing` (last accessed: April 2022)

of paralinguistic speech processing (this work also cites (Milde and Biemann, 2015)). The conclusion from this discussion going forward is that deep neural networks such as CNNs yield significantly better classification results than models trained on manually designed paralinguistic features and representations of speech.

Chapter 5

Unsupervised Representation Learning for Speech Contexts

5.1 Introduction

Unsupervised speech processing (or also zero resource speech processing) is a relatively new and growing field that deals with speech processing tasks where usually no transcriptions or labels are known. In many tasks, only raw audio data is available. One of the first successful applications is voice search by example, where a potentially large collection of audio data can be queried by an example audio query (Zhang and Glass, 2009). Just as with many other unsupervised methods (see also Section 2.2), one major advantage is that such systems are language-independent and speech data without transcriptions is usually easier to gather in large quantities.

Variance and variability in recordings of speech and its representations are a common problem in automatic speech processing tasks. For example speaker, environment characteristics and the type of microphone will cause large differences in typical speech representations (e.g. FBANK, MFCC), making direct similarity comparisons difficult. We can describe such factors of variance also as the context of an utterance; speech sounds that occur close in time share similar contexts. Based on this idea, we propose to learn representations of such contexts in an unsupervised way, without needing further speaker IDs, channel information or transcriptions of the data.

In this chapter we introduce "Unspeech" embeddings, which are based on unsupervised learning of context feature representations for spoken language. These embeddings can be trained on speech data without transcriptions or annonated speaker information and can be used to cluster speech data as well as augment ASR models with context information. The learning objective is based on context and non-context discrimination, where a model has to learn whether two speech segments appeared close in time or are unrelated.

Recent acoustic models for automatic speech recognition (ASR) incorporate some form of (deep) neural network that can learn to deal with part of this variance by using supervised training data in combination with the ability to learn representations as part of the model. A growing trend is to incorporate larger context views of the data explicitly into the neural network. In Deep Neural Network Hidden Markov Model (DNN-HMM) hybrids, fixed-length speaker embeddings like i-vectors are made available for the neural network as additional input features (Saon et al., 2013). Typically, larger temporal windows than single speech frames are also used as input to the neural network to make context available to local predictions. This can for instance be achieved by either stacking consecutive speech frames or by using Time-Delayed Neural Networks (TDNNs) (Waibel et al., 1990; Peddinti et al., 2015).

On the other hand, "Unspeech" embedding models embed a window of speech into a fixed-length vector so that *corresponding points are close, if they share similar contexts.* Unsupervised training of the embedding function is inspired by negative sampling in word2vec (Mikolov et al., 2013) – where words that share a similar meaning

are embedded in similar regions in a dense vector space. In this chapter, we demonstrate that the learned Unspeech context embeddings encode speaker characteristics and also can be used to cluster a speech corpus. As an additional context input feature, they can also improve supervised speech recognition tasks with TDNN-HMM acoustic models, in particular when adaptation to out-of-domain data is needed.

Building on the speech clustering that the Unspeech embedding provides, we also introduce a local-context variant of Unspeech embeddings in this chapter. This embedding can be used in speaker-independent tasks such as emotion and command recognition. In contrast to the previous chapter, the first task is employing self-supervised training on unlabeled speech data. Transfer learning then happens from this model to the target task, which is supervised but might only have a few annotated utterances.

5.2 Related Work

Speaker embeddings and phonetic embeddings are two major groups of proposed embeddings in speech: While speaker embeddings seek to model utterances from the same speaker so that they share similar regions in a dense vector space, in phonetic embeddings, the same or similar phonetic content is close.

I-vectors (Dehak et al., 2011) are well-known, popular speaker vectors. Recently, supervised neural network-based speaker embeddings also succeeded to show good speaker discriminative properties (Snyder et al., 2016; Snyder et al., 2017; Li et al., 2017), particularly on short utterances. Bengio and Heigold (2014) proposed supervised word embeddings for speech recognition, where words are nearby in the vector space if they sound alike. Kamper et al. (2015) showed that auto-encoders can also be used in conjunction with top-down information for unsupervised phonetic representation learning in speech. Chung et al. (2016) proposed audio word2vec, based on sequence-to-sequence auto-encoders trained on a dictionary of isolated spoken words. By analogy of auto-encoders, Pathak et al. (2016) introduced context encoders, a class of models that learn context embeddings in images. There is also growing interest in representation learning on non-speech audio by using learning objectives directly related to contexts. Jansen et al. (2018) encoded the notion that (non-speech) sounds occurring in context are more related.

Bromley et al. (1994) introduced Siamese neural networks: two (time-delayed) neural networks that embed digital signatures and a learning objective based on discriminating between true and false signatures. This idea has recently been revisited in the context of joint phoneme and speaker embeddigs learning in a weakly supervised setting, where speaker annotation, same word information and segmentation is available (Synnaeve and Dupoux, 2014; Zeghidour et al., 2016). Gutmann and Hyvärinen (2010) introduced Noise-Contrastive Estimation (NCE), an estimator based on discriminating between observed data and some artificially generated noise. Jati and Georgiou (2017) proposed Speaker2vec for speaker segmentation,

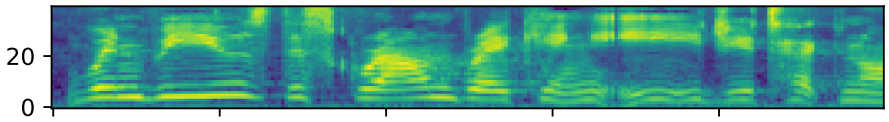

FIGURE 5.1: The initial sequence of FBANK vectors as generated by Kaldi (40-dimensional, with 100 frames per second).

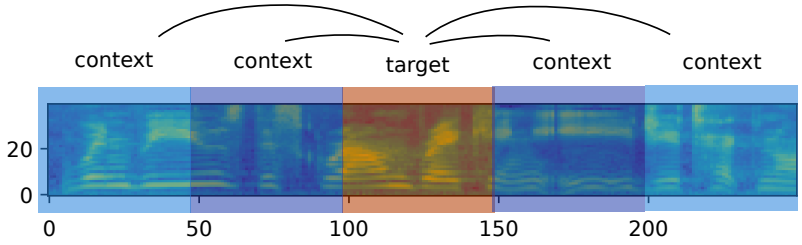

FIGURE 5.2: The initial sequence with unnormalized FBANK vectors: we choose one target window and two left and right contexts. All windows are of the same size.

with unsupervised training using a neural encoder/decoder. Very recently and in parallel to our efforts, Jati and Georgiou (2019) also proposed (unsupervised) neural predictive coding to learn speaker characteristics.

In (Jin et al., 1997) unsupervised speaker clustering was proposed to yield labels for speaker adaptation in acoustic models, based on the idea that consecutive windows of speech are likely from the same speaker. Several forms of context/speaker embeddings have also been used for (speaker) adaptation in state-of-the-art speech recognition acoustic models: i-vector speaker embeddings are by far the most popular (Saon et al., 2013; Senior and Lopez-Moreno, 2014; Miao et al., 2015). Veselỳ et al. (2016) proposed sequence summary neural networks for speaker adaptation, where utterance context vectors are averaged from the speech feature representation. Gupta et al. (2017) showed that visual features, in the form of activations from a pre-trained CNN for object detection on videos can also be used as context vectors in the acoustic model.

YAMnet (Plakal and Ellis, 2020) is a pretrained deep neural network that is trained on a dataset of 521 audio event classes (supervised). It can be used for transfer learning or for generating audio embeddings. Shor et al. (2020) proposed TRILL embeddings with an unsupervised triplet-loss objective that can also be used for fine tuning on audio/speech embeddings. Several down-stream benchmark classification tasks are proposed, where the unsupervised embeddings are fine-tuned on audio/speech classification tasks. We also make use of the proposed benchmarks in Section 5.5.3 to evaluate Unspeech embeddings. TRILL and YAMnet use raw audio waveforms as input, i.e. they are end-to-end networks.

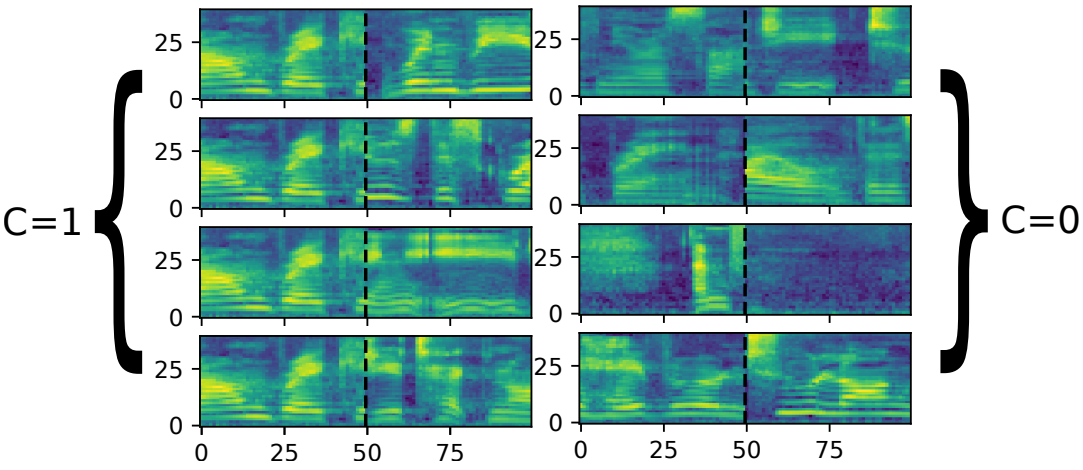

FIGURE 5.3: Sampling examples for two left contexts and two right contexts from the figure above. Positive example pairs are of class $C = 1$, negative sampled pairs are $C = 0$. In this example, each window has a size of 50 FBANK frames (0.5 seconds).

5.3 Proposed Models

Our goal is to train a model that learns a speech context embedding transformation. To train this model, we construct an artificial binary classification task with logistic regression, where two fixed-sized windows are compared. One target window can have multiple context windows, depending on the number of left and right contexts. For every left and right context, a pair with the target is created. Figure 5.1 and 5.2 illustrates this with two right and left contexts, yielding four positive contexts and four randomly sampled negative contexts. For the target window we denote emb_t as the target embedding transformation, taking a window of FBANK features and producing a fixed sized output vector, emb_c as the embedding of a true context and emb_{neg} as the embedding of a randomly sampled context. The pair of (embedded) speech windows is considered to be of class $C = 1$ if one window is the context of the other, or $C = 0$ if they are not. For $C = 1$, we sample the pairs from consecutive windows, for $C = 0$ we use negative sampling to construct a pair of speech windows that are unrelated with high probability: We uniformly sample a random utterance u and then uniformly a random position in u.

5.3.1 Objective Function

With the scalar x as the output of the model for a particular data point, σ the sigmoid function and C its true class $\in (0, 1)$, the logistic loss for a binary classification task is:

$$loss(x, C) = C(-log(\sigma(x))) + (1 - C)(-log(1 - \sigma(x))) \tag{5.1}$$

with $x = emb_t^T emb_c$, the dot product over target and context embedding transformations if $C = 1$ and $x = emb_{neg1_i}^T emb_{neg2_i}$, the dot product over two negative

sampled embedding transformations if $C = 0$, for $k =$ number of negative samples we can thus obtain:

$$NEG_{loss} = -k \cdot log(\sigma(emb_t^T emb_c)) - \sum_{i=1}^{k} log(1 - \sigma(emb_{neg1_i}^T emb_{neg2_i})) \tag{5.2}$$

Note that in the similar NCE loss formulation (Gutmann and Hyvärinen, 2010), $P(C = 1) = P(C = 0) = \frac{1}{2}$, i.e. the number of data points where $C = 1$ and $C = 0$ are the same, while we could have more negative than positive target/context embedding pairs, depending on k. Instead, for $C = 1$ we multiply with $k =$ the number of negative samples, to penalize errors on positive and negative context pairs equally. Another difference is that we sample two unrelated embedding windows instead of one.

5.3.2 Model Architecture

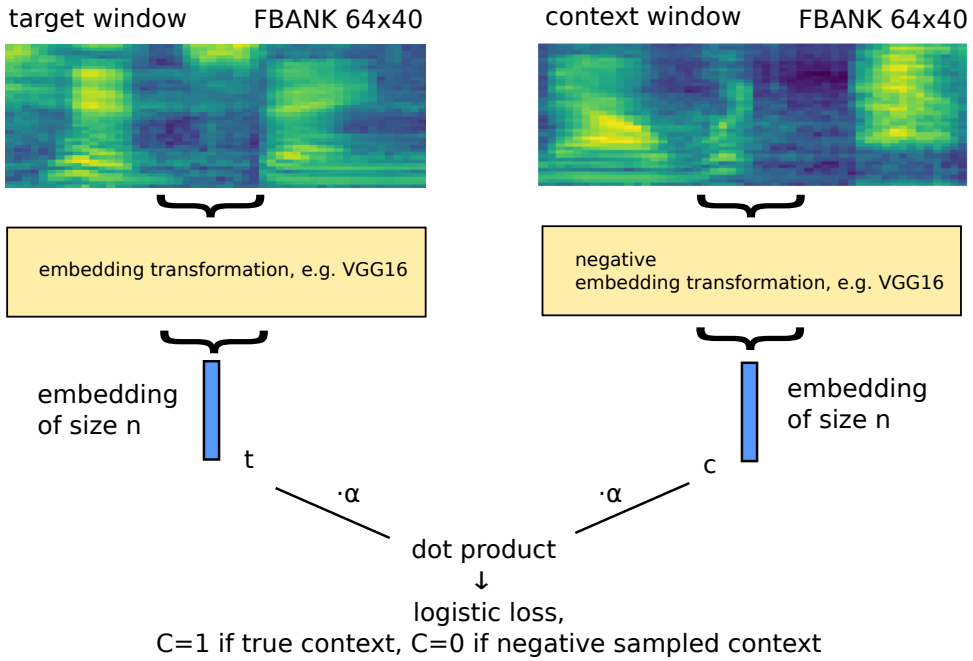

FIGURE 5.4: Unspeech embeddings are trained using a Siamese neural network architecture combined with a dot product. We use a VGG-like CNN network as embedding transformation in the yellow boxes (a convolutional neural network with 16 layers).

Figure 5.4 shows our architecture for FBANK input features. We project two dimensional input windows into two fixed sized embeddings, which are combined with a dot product. Since this is sensitive to the scaling of output embeddings, we multiply them with a single scalar parameter α, which is trained with the rest of the

network. Direct normalization to unit length (making the dot product equivalent to a cosine distance) hampers convergence of the loss and was discarded early on.

Many different architectures are possible for converting the input representation into a fixed sized embedding, but we mainly evaluated with a VGG-style CNN architecture (Model A in (Simonyan and Zisserman, 2015)), as it is well established and it can exploit the two dimensional structure in the FBANK signal. We share the weights of the convolutional layers of both embedding transformations, but keep the fully connected layers separate. Dropout is used for fully connected layers (0.1) and L2 regularization is used on the weights (0.0001), for all experiments we optimize with ADAM (Kingma and Ba, 2014). We make use of leaky ReLUs (Maas et al., 2013).

5.3.3 Implementation Details

We implemented all models in TensorFlow (Abadi et al., 2016) version 1.4 and Python 3.6. The CNN architectures VGG and ResNetv2 are adapted from the examples in the TF-slim package[1] FBANK input features are computed using Kaldi's (Povey et al., 2011) feature generation tools with the standard configuration for FBANK vectors: 40 banks, hamming windows of 25 ms, sampled every 10 ms.

For the loss function, we use the `sigmoid_cross_entropy_with_logits` function in TensorFlow, as it provides a stable reformulation of the logistic loss that ensures stability and avoids floating point overflow. Specifically[2], with $\sigma(x) = \frac{1}{1+e^{-x}}$ (standard sigmoid function) and $x =$ logits, $C =$ labels:

$$
\begin{aligned}
\text{loss(x,C)} &= C \cdot (-log(\sigma(x))) + (1 - C) \cdot (-log(1 - \sigma(x))) \\
&= C \cdot (-log(\frac{1}{1 + e^{-x}})) + (1 - C) \cdot (-log(\frac{e^{-x}}{1 + e^{-x}})) \\
&= C \cdot log(1 + e^{-x}) + (1 - C) \cdot (-log(e^{-x}) + log(1 + e^{-x})) \\
&= C \cdot log(1 + e^{-x}) + (1 - C) \cdot (x + log(1 + e^{-x}) \\
&= (1 - C) \cdot x + log(1 + e^{-x}) \\
&= x - x \cdot C + log(1 + e^{-x}) \\
&= log(e^{x}) - x \cdot C + log(1 + e^{-x}) \\
&= -x \cdot C + log(1 + e^{x})
\end{aligned}
$$

To avoid overflow and for numeric stability, TF uses this alternative formulation:

$$max(x, 0) - x \cdot C + log(1 + e^{-abs(x)})$$

[1] TF-Slim is a high-level library for common NN layers in TensorFlow.

[2] The formulas and derivations are taken from the TensorFlow v1.14 documentation: `https://github.com/tensorflow/docs/blob/r1.14/site/en/api_docs/python/tf/nn/sigmoid_cross_entropy_with_logits.md` (last accessed: December 2021)

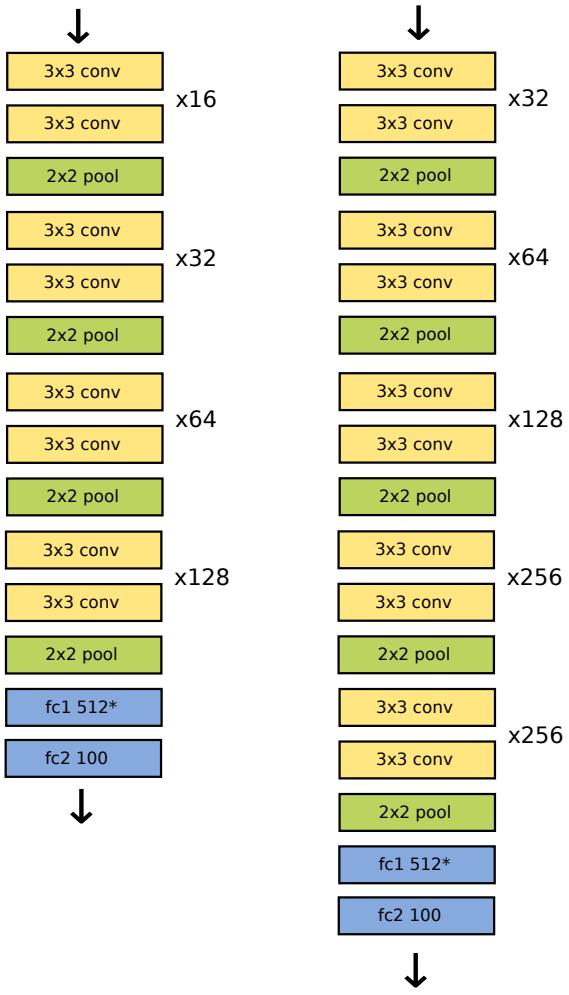

FIGURE 5.5: Hyperparameters and layer structure for two convolutional neural networks, in VGG-style with two successive convolutions followed by a pooling layer. "VGGsmall" is on the left, "VGG" is on the right.

Many different architectures are possible for converting the input representation into a fixed-sized embedding, but we mainly evaluated with a VGG style CNN architecture (Simonyan and Zisserman, 2015), as it is well established and it can exploit the two dimensional structure in the FBANK signal. We reduce the size of both CNN architectures so that two networks can fit better on the memory of one 12GB graphics card, so that training the complete network on a single GPU remained possible (in 2018). See Figure 5.5 for detailed hyperparameter choices of the convolution and pooling layers of the VGG net. Since the width of the window is reduced by half in every 2x2 pooling layers, the minimal input size to the network is 32 FBANK frames ($\approx$ 320 ms). If a smaller input size is needed, VGGsmall can also be used as shown in Figure, with a minimal input size of 16 frames. Dropout is used for fully connected layers and set to 0.1. L2 regularization is used on the weights and set to 0.0001. For all experiments we optimize the network with ADAM (Kingma and Ba, 2014).

5.4 Evaluation

TABLE 5.1: Comparison of English speech data sets used to train Unspeech embeddings.

	hours			speakers		
dataset	**train**	**dev**	**test**	**train**	**dev**	**test**
TED-LIUM V2	211	2	1	1,273+3	14+4	13+2
Common Voice V1	242	5	5	16,677	2,728	2,768
TEDx (crawled)	9,505			41,520 talks		

Table 5.1 characterizes the datasets we used in our evaluation. TED-LIUM V2 (Rousseau et al., 2014) has a comparatively small number of speakers, especially in the development and test set of the corpus. TED-LIUM and Common Voice (Ardila et al., 2020) are segmented at the utterance level, both are similar in the number of hours. In Common Voice, volunteers from all over the world recorded predefined prompts, in TED-LIUM utterances are segmented from and aligned to TED talks[3]. In order to explore large-scale training, we also downloaded all TEDx talks from 01-01-2016 until 03-01-2018 from YouTube, giving us 41,520 talks (0.5 TB compressed audio data) with a total of 9,505 hours of unannotated audio. While the majority of TEDx talks are in English, a very small number of them are in other languages or contain only music. We did not segment or clean the TEDx data in any way and kept all downloaded talks.

5.4.1 Same/Different Speaker Experiment

In the same/different speaker experiment, we evaluate a binary classification task: given two utterances, are they uttered from the same or different speakers? Our hypothesis is that Unspeech embeddings can be used for this task, because one strategy to discriminate samples of true contexts from negative sampled ones is modelling speaker traits. In Table 5.2 we show equal error rates (EER)[4] on same/different speaker comparisons of all utterance pairs, limiting the number of speakers to 100 in the train sets of TED-LIUM and Common Voice corpus in this experiment. The embedding dimension is 100 in all experiments. We train Unspeech models with different target window widths (32, 64, 128) and i-vectors are trained/extracted with Kaldi (Povey et al., 2011). For all experiments, we use two left and two right context windows and use the VGG structure as shown in Figure 5.5 on the right side.

The distance function $d_1(a, b) = \sigma(emb_t(a)^T emb_c(b))$ to compare two segments a and b correspond to the distance function in the Unspeech training process. The cosine distance, or equally after normalization to unit length, the Euclidean distance on vectors produced by emb_t also produces good comparison results, so that

[3] `https://www.ted.com/` (last accessed: December 2021)

[4] The error rate where the number of false positive and false negatives is the same, calculated using pyannote.metrics (Bredin, 2017)

TABLE 5.2: Equal error rates (EER) on TED-LIUM V2 – Unspeech embeddings correlate with speaker embeddings.

Embedding	**EER**		
TED-LIUM:	**train**	**dev**	**test**
(1) i-vector	7.59%	**0.46%**	1.09%
(2) i-vector-sp	**7.57%**	0.47%	**0.93%**
(3) unspeech-32-sp	13.84%	5.56%	3.73%
(4) unspeech-64	15.42%	5.35%	2.40%
(5) unspeech-64-sp	13.92%	3.4%	3.31%
(6) unspeech-64-tedx	19.56%	7.96%	4.96%
(7) unspeech-128-tedx	20.32%	5.56%	5.45%

TABLE 5.3: Comparing clustered utterances from TED-LIUM using i-vectors and (normalized) Unspeech embeddings with speaker labels from the corpus. "-sp" denotes embeddings trained with speed-perturbed training data.

Embedding	**Num. clusters**			**Outliers**			**ARI**			**NMI**		
	train	**dev**	**test**	**train**	**dev**	**test**	**train**	**dev**	**test**	**train**	**dev**	**test**
TED-LIUM IDs	1,273 (1,492)	14	13	3	4	2	1.0	1.0	1.0	1.0	1.0	1.0
i-vector	1,630	12	10	8,699	1	2	0.8713	**0.9717**	**0.9792**	0.9605	**0.9804**	**0.9598**
i-vector-sp	1,623	12	10	9,068	1	2	0.8641	**0.9717**	**0.9792**	0.9592	**0.9804**	**0.9598**
unspeech-32-sp	1,686	16	12	3235	22	32	**0.9313**	0.9456	0.9178	**0.9780**	0.9536	0.9146
unspeech-64	1,690	16	11	5,690	14	21	0.8130	0.9537	0.9458	0.9636	0.9636	0.9493
unspeech-64-sp	1,702	15	11	3,705	23	25	0.9205	0.9517	0.9340	0.9730	0.9633	0.9366

$d_2(a, b) = ||emb_t(a) - emb_t(b)||$ can be used for comparing two Unspeech segments. Sequences that are longer than the trained target window can be windowed and averaged to obtain a single vector for the whole sequence, since vectors that are close in time share contexts and correlate highly. However, EER on i-vectors trained with supervised speaker labels compared with the cosine distance (results with $d_2(a, b)$ are identical after normalization) are lower than on Unspeech embeddings with $d_2(a, b)$ (1,2 vs 3,4,5). Training Unspeech on TEDx talks instead of TED-LIUM also produces higher EER as a speaker embedding (6,7). "-sp" denotes training on speed-perturbed data: adding copies of the raw training data at 0.9 and 1.1 playing speed, as recommended in (Ko et al., 2015).

5.4.2 Clustering Utterances

We can also use the generated vectors to cluster a corpus of utterances to gain insight into what kind of utterances get clustered together. We use HDBSCAN (McInnes et al., 2017), a modern hierarchical density-based clustering algorithm for our experiments, since it scales very well to a large number of utterances and the number of clusters does not need to be known a priori. See Section 2.2.3 in Chapter 2 for details. It uses an approximate nearest neighbor search if the comparison metric is Euclidean, making it significantly faster on a large number of utterances as compared

to other speaker clustering methods that require distance computations of all utterance pairs (including greedy hierarchical clustering with BIC (Zhou and Hansen, 2000)).

We use Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) and Normalized Mutual Information (NMI) (Strehl, 2002) to compare the clusters to the speaker IDs provided by the TED-LIUM corpus in Table 5.3. We found that Unspeech embeddings and i-vectors give a sensible number of clusters, without much tweaking of HDBSCAN's two parameters (min. cluster size, min. samples)[5]. On the train set, Unspeech embeddings will provide slightly higher cluster scores, while i-vectors provide better scores for dev and test, which have a significantly smaller number of speakers. Unspeech-64 is slightly better than Unspeech-32 on the dev and test set. ARI is sensitive to the absolute number of outliers – we found NMI to be a better metric to compare the results on the train set.

Taking a closer look at the clustered Unspeech embeddings, we observed that different speakers in the same talk tend to get clustered into distinct clusters (making the clustered output very often more accurate than the train speaker IDs provided in TED-LIUM), while the same speaker across different talks and also the same speaker in one talk with significantly different background noises tends to be clustered into distinct clusters. This implies that Unspeech embeds more than just speaker traits.

5.4.3 Acoustic Models With Unspeech Cluster IDs

TABLE 5.4: Comparing the effect of two speaker division baselines (One speaker per talk, one speaker per utterance) and clustering with Unspeech on WER with GMM-HMM and TDNN-HMM chain acoustic models trained on TED-LIUM.

Acoustic model	**Spk. div.**	**Dev WER**		**Test WER**	
		plain	**resc.**	**plain**	**resc.**
GMM-HMM	per talk	19.2	18.2	17.6	16.7
TDNN-HMM		8.6	7.8	8.8	8.2
GMM-HMM	per utt.	19.6	18.7	20.1	19.2
TDNN-HMM		8.5	7.9	9.3	9.0
GMM-HMM	Unspeech	18.4	17.4	17.5	16.5
TDNN-HMM	64	8.6	7.8	**8.5**	**8.1**
GMM-HMM	Unspeech	18.4	17.5	17.2	16.4
TDNN-HMM	64-sp	**8.3**	**7.5**	8.6	8.2

We can also train acoustic models with the cluster IDs provided and use them in lieu of speaker IDs for HMM-GMM speaker adaptation and online i-vector training for the TDNN-HMM model. We use the TED-LIUM TDNN-HMM chain recipe

[5]We use 5/3 for all experiments shown in Table 5.3, but other parameters in the range 3-10 will give similar results.

(s5_r2) in Kaldi (Povey et al., 2016) and show WER before (plain) and after rescoring with the standard 4-gram Cantab TED-LIUM LM (resc.). Table 5.4 shows WER on different speaker separation strategies on the train set, with one speaker per talk being the default in the s5_r2 recipe. All models pre-trained online i-vectors based on the given speaker IDs and use those as additional input features. The standard recipe computes a fixed affine transform on the combined input features (40 dim hires MFCC + 100 dim i-vector), c.f. Appendix C.6 of (Povey et al., 2014). GMM-HMM bootstrap models will perform about 15% worse and TDNN-HMM trained on them will perform about 10% worse if no speaker information is available. Using the cluster IDs from clustering Unspeech embeddings of all utterances, the baseline WER can not only be recovered, but even slightly improved upon. For all TDNN-HMM models, we set the width of a layer to 1024.

TABLE 5.5: WER for TDNN-HMM chain models trained with Unspeech embeddings on TED-LIUM.

Context vector	**Dev WER**		**Test WER**	
	plain	**resc.**	**plain**	**resc.**
(1) none	9.1	8.5	9.5	9.1
(2) i-vector-sp-ted	**8.3**	**7.5**	8.6	8.2
(3) unspeech-64-sp-ted	9.1	8.3	9.6	9.0
(4) unspeech-64-sp-cv	9.1	8.3	9.5	9.1
(5) unspeech-64-sp-cv + (2)	8.4	7.6	**8.5**	**8.1**
(6) unspeech-64-tedx	9.0	8.2	9.4	8.7
(7) unspeech-128-tedx	8.9	8.2	9.4	8.9

We employed the proposed Unspeech adapation on GerTV1000h (Stadtschnitzer et al., 2014) as well. It is a large German speech dataset with 1005h of transcribed broadcast data. No speaker information is available for this corpus. Thus, we compare our clustering to a baseline utterance to speaker mapping, where every utterance ID is mapped to a new speaker. Table 5.6 shows our test results. We test the different approaches on DiSCo (Baum et al., n.d.), a test set for the German broadcast domain with finely annotated speech categories. We test WER separately on four different conditions in DiSCo: clean spontaneous speech, noisy spontaneous speech, clean planned speech and noisy planned speech.

We observe that the improvements resulting from our speaker adaptation (unspeech cluster IDs) are more effective for both noisy speech subsets. The GMM-HMM bootstrap models benefit the most from better speaker adaptation. TDNN-HMMs also benefit from a better assignment of utterances to (pseudo) speakers. Here, speaker information is incorporated into the speech feature layer with appended I-vectors to the input representation. A better GMM-HMM bootstrap model also potentially results in a better alignment. Language model rescoring via a neural Gated Convolutional Network (GCNN) improves these results by another 10% relative WER reduction.

TABLE 5.6: WER comparison on the DiSCo test corpus, with acoustic models trained with 1005h of German broadcast speech (GerTV1000h).

		DiSCo planned		DiSCo spontaneous	
Model	**dev set**	**clean**	**noisy**	**clean**	**noisy**
(1) GMM-HMM	22.55	16.14	25.92	20.93	41.86
(2) GMM-HMM unspeech	21.39	14.99	22.26	20.25	38.24
(3) TDNN chain	14.31	8.92	11.98	10.84	20.98
(4) TDNN chain unspeech	14.19	8.67	11.10	10.72	19.97
(3b) 3 + GCNN LM resc.	13.47	8.04	10.72	**9.65**	18.66
(4b) 4 + GCNN LM resc.	**13.25**	**7.79**	**9.98**	9.70	**17.86**

TABLE 5.7: Decoding Common Voice V1 utterances. Mozilla's open source dataset provides a challenging test set, which is out-of-domain for an acoustic model trained on TED-LIUM.

Context vector	**Dev WER**		**Test WER**	
	plain	resc.	plain	resc.
(1) none	31.2	29.6	29.9	28.5
(2) i-vector-sp-ted	30.3	29.0	29.9	28.2
(3) unspeech-64-sp-cv	**29.5**	**27.9**	**28.3**	**26.9**
(4) unspeech-64-sp-cv + (2)	29.6	28.2	28.9	27.4
(5) unspeech-64-tedx	30.2	28.8	29.2	27.5
(6) unspeech-128-tedx	30.1	28.7	29.5	28.0

5.4.4 Unspeech Context Vectors in TDNN-HMM Models

We can also replace the i-vector representation used in training the TDNN-HMM with the Unspeech context vector. In Table 5.5, we selected the strongest baseline from Table 5.4 according to the dev set (Unspeech 64-sp clusters) and show WERs on the TED-LIUM dev and test for different Unspeech context embeddings. We trained Unspeech models with different window sizes (64,128) on TED-LIUM (ted) and Common Voice V1 (cv) and computed them for every 10 frames, like the online i-vector baseline. While Unspeech embeddings can slightly improve a baseline model trained without any context vectors, with best results obtained when training on the 9,500 hours of TEDx data (6,7), using i-vectors (2) yields better WERs compared to Unspeech embeddings. Combining Unspeech embeddings trained on Common Voice and i-vectors in the input representation can yield slightly lower WERs than i-vectors alone (5).

In Table 5.7, we show WER on decoding utterances from the Common Voice V1 dev and test sets with TDNN-HMM acoustic models trained on TED-LIUM. Utterances from Common Voice are much harder to recognize, since a lot more noise and variability is present in the recordings and the recordings have a perceivably lower signal-to-noise ratio. Since they also contain over 2700 speakers each, using

an egregious range of microphones, they provide an excellent dev/test to test how robust the TDNN-HMM models are on out-of-domain data. Unsurprisingly, WERs are fairly high compared to the TED-LIUM test set with mostly clean and well pronounced speech. For Common Voice we observed that acoustic models trained with Unspeech embeddings consistently resulted in better WERs compared to the baselines, helping the model to adapt. Particularly pre-training Unspeech models on the Common Voice training data help a TDNN-HMM model trained on TED-LIUM to adapt to the style of Common Voice recordings. Embeddings from Unspeech models trained on TEDx will also perform better than the no context and i-vector baseline models. In contrast to the results in Table 5.5, in this decoding task, i-vectors in the acoustic model do not provide much of an improvement over the TDNN-HMM baseline model without context vectors.

5.5 Negative Sampling Methods

In plain random negative sampling, a pair of speech windows is uniformly sampled from a speech corpus. For example, we uniformly sample a random utterance u and then uniformly a random position in u for the sampled window. This is the original sampling method used in Section 5.3. If there are many different speakers recorded in different environments in the corpus, it will with high probability generate positive and negative samples from different speakers/environments. The effect is that the trained embeddings will cover speaker, channel and environment characteristics as context features, as those help to discriminate negative pairs from positive context pairs.

We propose a second form of negative sampling: *same context negative sampling*. With this sampling technique, the negative samples are coming from utterances inside clusters of contexts as the negative context pairs. Speaker, channel and environment characteristics can no longer be used by the classifier to differentiate positive from negative pairs. If we used a priori known speaker IDs, changes in the channel or environment (e.g. background noise changes) would still be easily discriminable and would then be contained in the resulting Unspeech embeddings. Using actual cluster IDs has the advantage that any obvious speaker, environment and channel context variance has already been picked up by the model. Of course, another obvious advantage is that it can be applied to speech data where no speaker information is available.

We first obtain standard Unspeech embeddings, then we cluster per utterance (average of all embeddings in the sequence). We use HDBSCAN (McInnes et al., 2017) to cluster the utterances, see also Section 2.2.3. In the remainder of the paper, we call embeddings from Unspeech models trained with same context negative sampling *local-context Unspeech embeddings* (lc-unspeech).

5.5.1 Word-Level Phonetic Information

With a window size of 32 frames, which is about 320ms of speech, we actually cover more than individual phonemes. Since the average length of a phoneme in TED-LIUM is about 10 frames, we expect to see at least 3-4 phonemes in such a window. Thus, if local-context Unspeech embeddings contain phonetic context information, we would expect to see similar phoneme sequences to have similar embeddings. Likewise, if global Unspeech embeddings only embed characteristics about speakers and channels, they are invariant to the phonetic content.

Calculated from over 4000 utterances from the TED-LIUM train set (292,125 phones), the ten most frequent phoneme 5-grams from the force-alignment are listed in Table 5.8.

TABLE 5.8: Most common 5-grams in a subset of TED-LIUM train utterances.

	Arpabet sequence	**As in**	**Count**
(0)	P IY P AH L	people	367
(1)	SIL AE N D DH	and	211
(2)	DH IH S IH Z	this is	199
(3)	B IH K AH Z	because	189
(4)	SIL AE N D SIL	and	183
(5)	EY SH AH N SIL	∼ation	141
(6)	Z SIL AE N D	and	136
(7)	S AH M TH IH	somethi(ng)	133
(8)	AH M TH IH NG	uhm thing	133
(9)	S T AA R T	start	110

For the purposes of visualization, this provides more than enough examples for commonly occurring phoneme 5-grams. We also expect that if 5 consecutive phonemes are slightly longer than 32 frames, their prefixes fit into the window. We further subsample these to 1000 random 5-grams. In Figure 5.6, we show a t-SNE (Maaten and Hinton, 2008) plot of these 5-grams with local-context Unspeech embeddings and in Figure 5.7 with global Unspeech embeddings.

5.5.2 Unspeech Embeddings in Acoustic Models for Out-Of-Domain Test Data

In the s5_s2 Kaldi training recipe for TED-LIUM, (online) i-vectors are calculated to capture speaker information. The default setting is to provide them as additional context input to the TDNN, to help the neural network with speaker adaptation, c.f. (Saon et al., 2013; Miao et al., 2015).

To see how well the acoustic models deal with domain adaptation, we also measured word error rates on Mozilla's new Common Voice dataset (Ardila et al., 2020). Table 5.9 shows our decoding results. We can verify that adding i-vectors is beneficial to WERs on TED-LIUMs test and dev set (1) vs. (2). However, on the Common

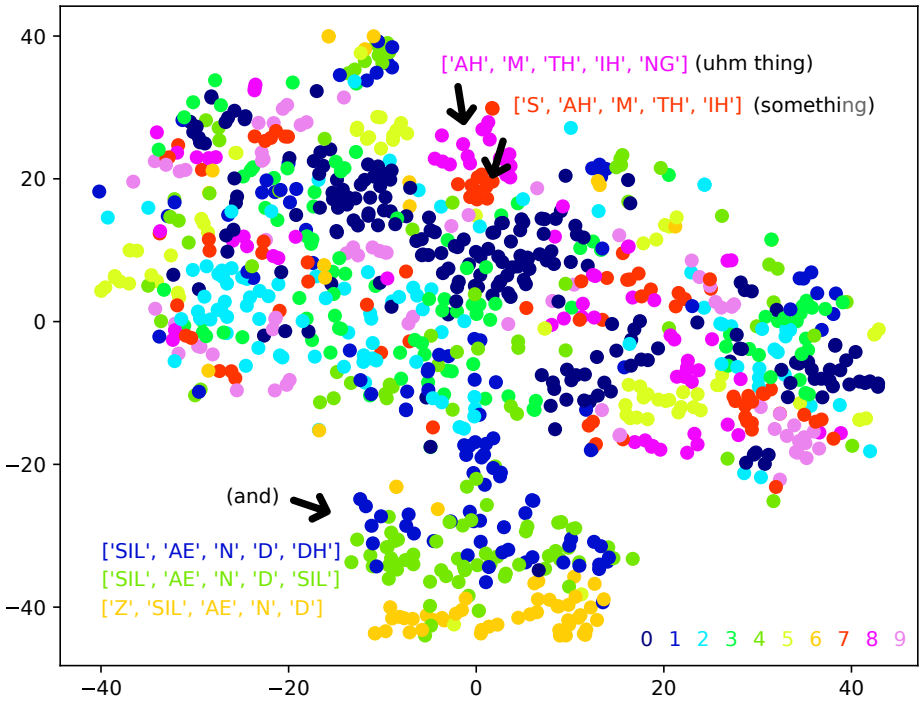

FIGURE 5.6: TSNE plot of *local-context Unspeech-32* embeddings over common phoneme 5-grams, see also Table 5.8. These embeddings have higher affinity towards phonetic content of the target window (32 frames = 320ms).

TABLE 5.9: WER for TDNN-HMM chain models trained with Unspeech embeddings on decoding TED-LIUM and Common Voice dev and test utterances. Unspeech models to provide context vectors were trained on different datasets, while the acoustic models are only trained on TED-LIUM speech data (+ context vector). Resc. means rescoring with a 5-gram Kneser-Ney (Kneser and Ney, 1995) language model, plain means only Kaldi's FST is used in decoding, but no further language model rescoring.

		TED-LIUM				**Common Voice**			
Context vector	**Trained on**	**Dev WER**		**Test WER**		**Dev WER**		**Test WER**	
		plain	resc.	plain	resc.	plain	resc.	plain	resc.
(1) none	-	9.1	8.5	9.5	9.1	31.2	29.6	29.9	28.5
(2) i-vector	TED-LIUM	**8.3**	**7.5**	**8.6**	**8.2**	30.3	29.0	29.9	28.2
(3) unspeech-32	TED-LIUM	8.9	8.1	9.3	8.9	31.2	29.7	30.0	28.2
(4) lc-unspeech-32	TED-LIUM + cl. (3)	9.4	8.5	9.4	8.9	29.9	28.4	28.5	26.9
(5) lc-unspeech-16	TED-LIUM + cl. (3)	9.1	8.4	9.4	8.9	**29.3**	**27.8**	**28.2**	26.7
(6) unspeech-32	Common Voice	9.1	8.5	9.4	8.9	**29.3**	28.0	28.3	**26.6**
(7) unspeech-64	Common Voice	9.1	8.3	9.5	9.1	29.5	27.9	28.3	26.9

Voice dev and test set, this benefit is much smaller. Utterances from Common Voice are much harder to recognize, since a lot more noise and variability is present in the recordings. There are also much more speakers in the dev and test sets and the recordings were made with a large range of different microphones. Thus, the Common Voice corpus provides an excellent dev and test to test how robust the TDNN-HMM models are on out-of-domain data. Unsurprisingly, WERs are fairly

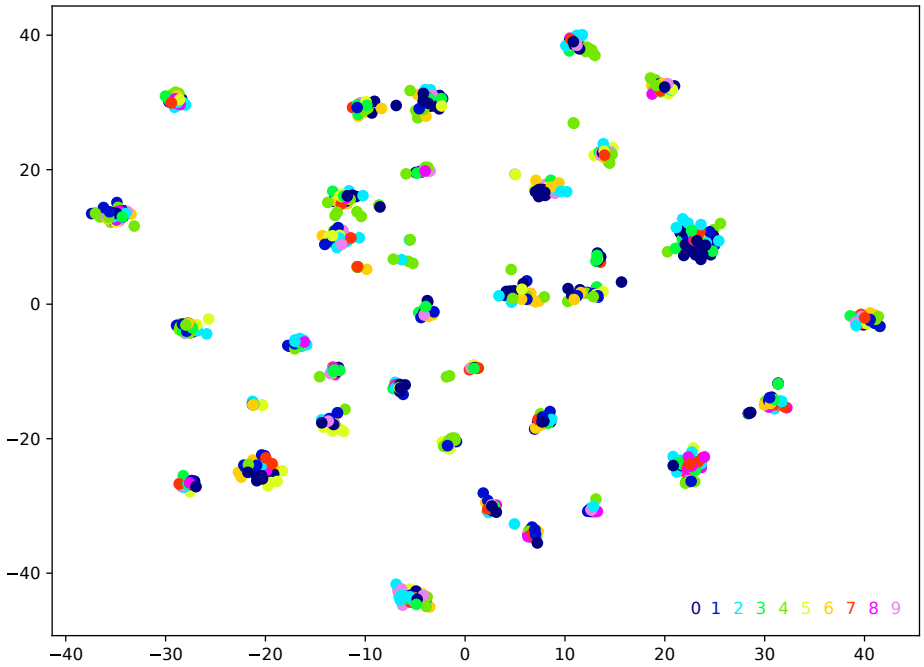

FIGURE 5.7: TSNE plot of *global Unspeech-32* embeddings over common phoneme 5-grams, see also Table 5.8. There is a strong affinity to preserve speaker and channel characteristics, while ignoring the phonetic content. The clusters largely correspond to the TED talks where the 5-grams have been taken from.

high in general compared to the TED-LIUM test set with mostly clean and well pronounced speech, as some of the utterances could also be very difficult to understand for humans.

We can replace the i-vector representation used in training the TDNN-HMM with Unspeech context embeddings. We trained Unspeech on different training data (TED-LIUM train or Common Voice train). For ic-unspeech (5) we additionally computed clusters from (3) with HDBSCAN[6] and used them in same context negative sampling.

While Unspeech embeddings can slightly improve upon a baseline model trained without any context vectors, using i-vectors (2) yields better WERs compared to Unspeech embeddings on the TED-LIUM dev and test set. With Common Voice, we observed that acoustic models trained with Unspeech embeddings resulted almost always in better WERs compared to the baselines, helping the model to adapt and to be more robust. In contrast to the results on TED-LIUM dev and test, in this decoding task, i-vectors do not provide much of an improvement over the TDNN-HMM baseline model without context vectors.

Using $emb_{neg1_i} = emb_t$ in the NEG loss function was slightly better on the Common Voice data than the standard Unspeech loss function, see (4) vs (3), but did not have much of an effect on the TED-LIUM test set data. Particularly local-context Unspeech embeddings (5, 6) also yielded very good WER results on Common Voice,

[6]setting the min_samples parameter of HDBSCAN to 3.

much better than the baseline Unspeech models and without seeing any of the out-of-domain Common Voice training data while training Unspeech. For local-context Unspeech embeddings, a smaller target window size of 16 worked better than 32. Pre-training Unspeech models on the Common Voice training data helped a TDNN-HMM model trained on TED-LIUM to adapt to the style of Common Voice recordings (7, 8).

5.5.3 Unspeech Embeddings in Down-Stream Classification Tasks

TABLE 5.10: Unspeech classification results (accuracy) for down stream speech classification tasks. MFCC, openSMILE, YAMNet, VGGish and TRILL results are taken from (Shor et al., 2020).

Embedding	**SAVEE**	**SAVEE (spk. dep.)**	**Speech Commands**
Number of classes	7	7	12
MFCC	55.6	77.2	47.7
OpenSmile	62.6	67.8	36.5
Random TRILL network	48.6	73.9	42.0
YAMNet top	45.4	64.4	40.7
YAMNet layer 10	62.3	79.2	73.1
VGGish top	49.8	68.7	28.3
VGGish FCN 1	57.7	77.9	52.7
TRILL top	53.7	83.9	60.4
TRILL layer 19	67.8	84.7	74.0
TRILL layer 19, MobileNet 2048d	60.0	(N/A)	74.9
TRILL finetuned	68.6	(N/A)	91.2
Unspeech top	30.8	65.2	61.7
Unspeech-lc top	55.5	70.6	66.6
Unspeech-lc pre-fc	56.7	71.3	75.6
Unspeech-lc pre-fc + 2 layer DNN	53.4	76.8	79.7

In Table 5.10, we evaluate pretrained Unspeech embeddings on the SAVEE (Haq et al., 2009) and Speech commands datasets (Warden, 2018). The SAVEE dataset is relatively small dataset with 480 utterances and 4 male speakers, the task is to identify one of 7 emotional categories. Originally, as proposed in (Haq et al., 2009), the SAVEE task was tested in a speaker-depended manner (randomized train/test splits with overlapping speakers). However, (Shor et al., 2020) also proposes a canonical split where speaker ”JK” and ”KL” are being used for training, ”JE” as a development set and results are reported on speaker ”DC”. We have tested both the speaker-dependent and the speaker-independent scenario on SAVEE with Unspeech embeddings and a simple linear regression classifier on top of the embeddings for classification.

The speech command dataset (Warden, 2018) is composed of crowd sourced recordings of short isolated commands, such as ”left”, ”right”, ”yes”, ”no”. This

dataset is much larger than SAVEE, with 105,829 utterances from 2,618 speakers in total. The task is proposed as a speaker-independent task, i.e. with no overlapping speakers in train and test. We compare Unspeech embeddings against several other embeddings and pretrained models: handcrafted openSMILE features (Eyben et al., 2010), YAMNet (Plakal and Ellis, 2020), VGGish (Hershey et al., 2017) with supervised training on Audio-Set (Gemmeke et al., 2017), TRILL and a MFCC baseline as reported in (Shor et al., 2020).

All reported Unspeech results in Table 5.10 are averaged over individual 10 runs. For the speaker-independent tasks, the same train/dev/test split as in (Shor et al., 2020) were used. Early stopping is applied and the training stops after there is no improvement on the dev set anymore. For the speaker-dependent tasks, the split is randomized on each training run with 80% train / 20% test, with no early stopping.

As somewhat expected, using regular Unspeech embeddings does not yield good results on speaker-independent tasks – as these embeddings mainly capture speaker related properties. However, we do observe a large gain in accuracy with Unspeech-lc compared to regular Unspeech on both downstream tasks. As noted in Shor et al. (2020), using intermediate layers of the model to extract embeddings yields stronger results on downstream tasks. We tested this hypothesis on Unspeech embeddigs as well and also observe an accuracy increase when we use the networks' activations right after the convolutional layers (denoted as pre-fc in Table 5.10).

On the commands dataset, using Unspeech-lc embeddings (pre-fc) yields strong results and is slightly better than TRILL (without fine tuning) and much better than all other baselines. On SAVEE (speaker-independent) the best unspeech results is similar to VGGish FCN 1, only slightly better than the openSMILE and MFCC baselines. Note that due to its small size, SAVEE is very sensitive to regularization of the linear network that is trained and we did not do a hyperparameter search. As expected, we observe higher accuracy when we test with the speaker-dependent SAVEE dataset as opposed to the speaker-independent one.

5.6 Conclusion

In this chapter, we proposed Unspeech speech embeddings and models to train these embeddings on unannoated speech data. These embeddings are trained in a self-supervised way and the models embed speech context information such as the speaker of an utterance, background noise and channel characteristics. The principle used for training these networks is that speech sounds that occur close in time share similar contexts.

The evaluation in this chapter showed that Unspeech context embeddings contain and embed speaker characteristics, but supervised speaker embeddings like i-vectors would be better suited for tasks like speaker recognition or authentication. However, clustering utterances according to Unspeech contexts and using the cluster IDs for speaker adaptation in HMM-GMM/TDNN-HMM models is a viable

alternative if no speaker information is available. While using Unspeech context embeddings as additional input features did not yield significant WER improvements compared to an i-vector baseline on TED-LIUM dev and test, we observed consistent WER reductions with out-of-domain data from the Common Voice corpus when we add Unspeech embeddings. This is a compelling use case of Unspeech context embedding for the adaptation of TDNN-HMM models. Better scores on the same/different speaker similarity task was not indicative of WER reduction – our TEDx Unspeech models scored higher EERs, but were at the same time better context vectors in the acoustic models. Training the model scales very well to large training sizes, as demonstrated by the scaling experiment.

We also proposed an embedding variant for local contexts, with a two stage training and clustering of Unspeech embeddings. The local context Unspeech embeddings can be used for speech tasks that require speaker-independence. We demonstrated that it can be used for speech classification tasks such as emotion recognition and short speech command recognition. Good results can already be obtained with a simple linear regression classifier on top of the trained embeddings. Transfer learning by fine-tuning Unspeech embeddings could also improve these results. We released our source code and offer pre-trained models.[7]

[7]See `https://unspeech.net` and `https://gitlab.com/milde/unspeech` (last accessed: April 2022), license: Apache 2.0

Chapter 6

Self-Supervised Discrete Representation Learning from Speech

6.1 Unsupervised Acoustic Unit Discovery

Unsupervised or zero resource speech processing is a relatively new and growing field that deals with speech processing setups where usually no transcriptions or labels are known. In many tasks, only raw audio data is available. One of the first successful applications is voice search by example, where a large collection of audio data can be queried by a voice query (Zhang and Glass, 2009).

Transcribed and labeled speech data is needed to train supervised speech recognition systems, but is usually costly to obtain. Unlabeled speech data on the other hand is much easier to obtain in larger quantities, even for languages for which much less resources are available as compared to e.g. English. We consider the problem of inducing an acoustic unit inventory (Siu et al., 2011; Badino et al., 2015) and perform self-labeling of untranscribed speech data. Such a system can be a building block for systems that learn lexical inventories (Kamper et al., 2014) from speech data alone, or could possibly aid in augmenting or replacing linguistically motivated phonemes in supervised automatic speech recognition (ASR). It is also one of the building blocks in current unsupervised ASR systems, such as Baevski et al. (2021).

In this chapter, we propose a novel neural architecture based on a sequence autoencoder and sequence masking of its internal representation. We force the encoder and decoder to develop a symbolic-like representation, with the goal of reconstructing the input speech representation with limited information. We aim to solve the problem of speech variability and speaker independence that the automatic units should be able to capture by using an utterance-level context embedding. The decoder can use this embedding to infer the "style" of the utterance in addition to its (sparse) input representation generated by the encoder. After the model is trained, speech can be represented as a sequence of units (or posteriograms) together with a context vector. See Figure 6.1 for an illustration of this decomposition.

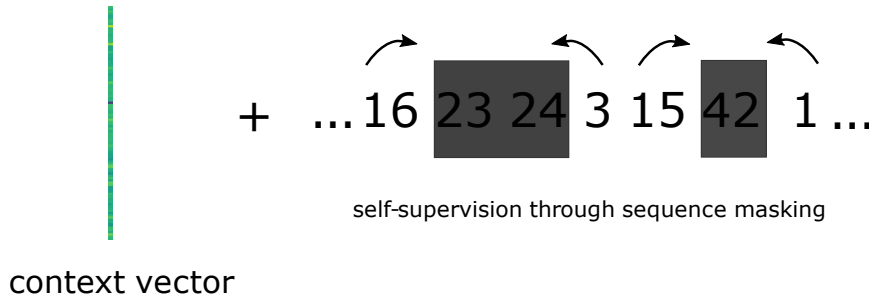

FIGURE 6.1: In Sparsespeech, we train a model that can represent an utterance as a sequence of discrete units together with a context embedding. The sequence is masked randomly and the missing elements must be inferred from context.

6.1.1 Related Work

Several approaches to unsupervised acoustic modeling and acoustic unit discovery have been proposed:

Segmental approaches: These approaches separate segmentation and clustering of acoustic units. Garcia and Gish (2006) developed one of the first of such systems. Segments are found based on spectral discontinuities in the signal and the clustering algorithm uses the polynomial trajectory of the cepstral features to compare speech units of varying length.

Autoencoder approaches: Badino et al. (2014) proposed k-means on framewise binarized autoencoder representations, where temporal smoothing of the frame level representations can be achieved with Hidden Markov Models (HMMs). Several autoencoder architectures are compared in (Renshaw et al., 2015), standard bottleneck autoencoders do not seem to produce representation with better minimal pair discriminability than MFCC features. Vector quantized variational autoencoders (Oord, Vinyals, et al., 2017) can also be used to learn discrete representations of speech, as demonstrated by the end-to-end system involving attention-based ASR and TTS (Tjandra et al., 2019) to encode and decode. Wang et al. (2020) proposed input masking in recurrent auto-encoders.

Bayesian non-parametric models: Lee and Glass (Lee and Glass, 2012) proposed a Dirichlet process mixture model, where each mixture is a HMM representing an acoustic unit. Chen et al. (2015) proposed Dirichlet process Gaussian mixture model (DPGMM) clustering for acoustic frame-level unit modeling in their winning entry for unsupervised acoustic unit modeling of the ZeroSpeech 2015 challenge on acoustic sub-word modeling (Versteegh et al., 2015). Heck et al. (2017) proposed DPGMM clustering on PLP features followed by obtaining unsupervised feature transformations by retraining with a speaker-independent GMM-HMM on the obtained labels. This was also the winning entry for automatic unit discovery of the ZeroSpeech challenge 2017 (Dunbar et al., 2017). Noteworthy is that while the re-training using a GMM-HMM with speaker adaptations improves on the DPGMM results, the DPGMM clustering on its own on PLP features on a frame-level basis already provides strong results.

The 2015, 2017 and 2019 ZeroSpeech challenges also targeted acoustic unit discovery (Versteegh et al., 2015; Dunbar et al., 2017; Dunbar et al., 2019) as part of the evaluation and have established the use of the ABX discriminability (Schatz et al., 2013; Schatz, 2016) to intrinsically compare how well semantically relevant sounds are mapped by a discovered representation.

Lightly supervised: Lightly supervised systems consider some other form of available information besides raw speech data. The systems in (Kamper et al., 2015) and (Kamper et al., 2016) use weak top-down constraints in the form of same-type annotations.

Self-supervised: Contrastive Predictive Coding (CPC) (Oord et al., 2018) is a representation learning model trained by predicting future hidden states, which can be

applied to raw speech in the time domain. This model has also become increasingly popular in acoustic unit discovery. There are various extensions such as VQ-CPC to combine CPC with Vector Quantisation (VQ) (Niekerk et al., 2020) and Aligned Contrastive Predictive Coding (A-CPC) (Chorowski et al., 2021), to relax exact time constraints for the predictions of future states. In Kamper and Niekerk (2021), such self-supervised voice-quantized neural networks are constrained to produce low-bitrate phone-like sequences.

Schneider et al. (2019) showed with wav2vec that pre-training a model on raw speech with a similar binary contrastive loss as word2vec (Mikolov et al., 2013) can be effective to improve supervised end-to-end acoustic models. The discrete variant vq-wav2vec (Baevski et al., 2020a; Baevski and Mohamed, 2020) of this model with vector quantization into several thousand units has also been successfully used to pretrain BERT (Devlin et al., 2019) on a sequence masking task, followed by using BERT representations in a wav2letter (Collobert et al., 2016) acoustic model for speech recognition.

Kahn et al. (2020) created a benchmark for ASR with no or limited supervision, based on English audiobooks and Librispeech data, also providing results for using CPC. We use the Libri-light corpus for training and evaluating our models, as it provides a good benchmark for unsupervised acoustic models that can scale well to large amounts of (untranscribed) speech data.

6.1.2 Proposed Sparsespeech Model

The proposed network in this chapter jointly infers a segmentation and a classification on an utterance level with an unsupervised training objective. Although the network outputs a label for each frame individually, in contrast to frame-level (or window-level) autoencoders, the network has access to the temporal structure of a full utterance. Figure 6.2 illustrates the proposed network. Encoder and decoder are bidirectional long short-term memories (LSTMs) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). Between encoder and decoder, we apply dropout at the sequence level, i.e. we randomly omit vectors in the sequence between them while training the network. Metaphorically speaking, this causes the encoder to never be sure that its outputs get passed on to the decoder, thereby forcing it to transmit the most salient information about the current and surrounding frames. The decoder must also learn to predict missing frames and corrects accordingly when new information is available from the encoder. Note that without this, the decoder could simply rely only on the inputs from the encoder at each time step and ignore its state, ultimately ignoring the temporal structure entirely and the network structure would be more similar to a regular autoencoder. We use mean squared error (MSE) for the reconstruction loss.

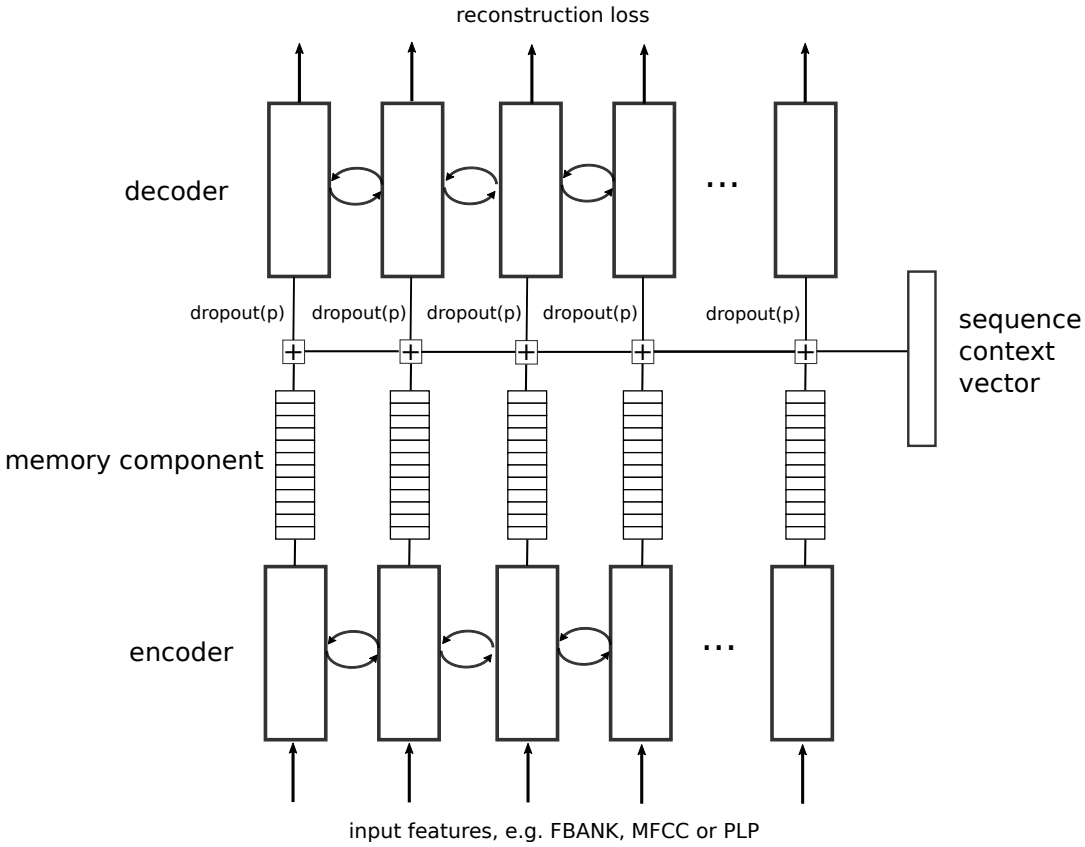

FIGURE 6.2: Sequence encoder / decoder with a memory component as sparsity bottleneck (Milde and Biemann, 2019).

Memory Component

Figure 6.3 illustrates the memory component. It is similar to the one used in (Sukhbaatar et al., 2015). For an input x we use $softmax(Wx + b)$, a standard single-layer network without an activation function for the key addressing. We set the dimensions of W such that the output of Wx has as many elements as there are value embeddings. After the softmax operation, we multiply each of the outputs with a corresponding value memory embedding and add up all vectors. This is the output vector of the memory module and it is a weighted sum over the value memory embeddings.

Note that the memory component that we used can also be understood as a form of key-value separated attention (Daniluk et al., 2017; Bahdanau et al., 2015; Luong et al., 2015), where we compare a query by attending to a key memory (instead of attending to RNN states) and output separate vectors from a value memory. The key memory, in our case, is the matrix W. The memory module is smooth and can be used and trained with back-propagation. As part of a bigger neural network, it can be placed between any layers and the input query dimension can be the same as the output vector dimension.

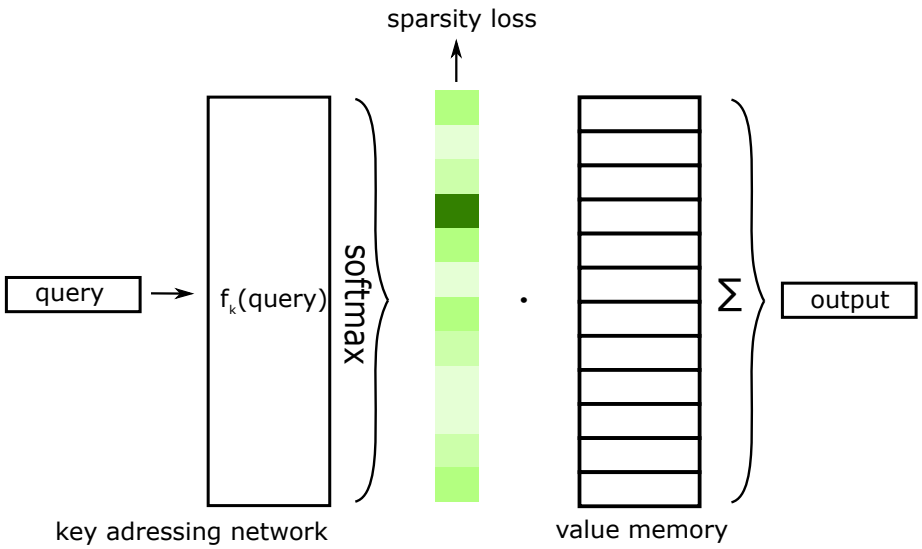

FIGURE 6.3: Memory module consisting of an input query key adressing network and a value memory bank containing fixed-sized embeddings.

The goal of the memory network, within the context of the sparse sequence autoencoder, is to sparsify the outputs of encoder network. However, without additional constraints, the output of the softmax layer will not necessarily be sparse and performs a soft-clustering over the output. We use the following constraint described in the next section on the softmax layer of the addressing network to achieve varying degrees of sparseness of this clustering, up to a quasi-hard clustering.

Enforcing Sparsity

We propose the following sparsity-inducing constraint for the softmax layer, based on L_∞ regularization. Let σ_1, ..., σ_n be the outputs of the softmax layer in the memory component, then:

$$\text{Softmax-}L_\infty = 1 - \sup_n \sigma_i \tag{6.1}$$

This regularization loss is zero, iff one of the softmax outputs is one and all other outputs are zero (perfect sparsity, one-hot encoding), because the softmax layer enforces the linear constraint that the sum of all outputs is one and all elements are positive. The regularization term is multiplied by a sparsity weight and is added to the loss of the full network.

Context Vector

We concatenate a context vector at each time step to the outputs of the encoder and memory component and use this combined vector as input to the decoder. We compare separately trained *Unspeech* context embeddings (Milde and Biemann, 2018) as

static additional input to the network and propose a simple integrated alternative inspired by sequence summary networks (Veselỳ et al., 2016) that does not require extra training: we calculate the mean vector over all encoder states.

Enforcing Diversity

A degenerate solution to minimize the sparsity constraint is to cluster all inputs into a single memory component. To discourage this, we also propose a diversity constraint calculated over m time steps of an utterance with n softmax outputs per timestep using Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) and $U(x) = \frac{1}{n}$:

$$\text{Diversity-}L = \frac{1}{m} \sum_{j=1}^{m} D_{KL}(\sigma_j || U) \tag{6.2}$$

Training Procedure

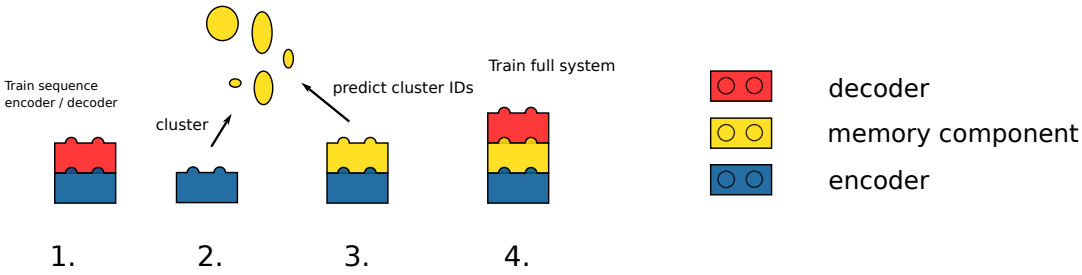

FIGURE 6.4: Sparsespeech training procedure illustrated as bricks that are separately trained until the full system is trained in the final training.

With a random initialization, we found it difficult to train the complete network together with the memory component directly from scratch, even with the diversity constraint. Most of the time, the training will be stuck in a degenerate solution of using either one memory component for all queries or two alternating memory value components for speech and silence.

We propose the following iterative training procedure, based on pre-training a network without the sparsity-inducing memory component first (also illustrated in Figure 6.4):

1. Train the network without the sparsity layer until the reconstruction loss converges.

2. Run a cluster algorithm, e.g. k-means++ (Arthur and Vassilvitskii, 2007) on a subset of the randomly selected bottleneck features obtained by running the encoder of the network on a couple of input sequences. Set k to the number of memory banks.

3. Initialize the value memory bank weights of the memory component to the cluster centers of Step (2). Then train the key addressing subnetwork on the cluster labels.

4. Connect the memory sub-module to the network by placing it between the encoder and decoder of the network trained in (1) and continue training the full network with added sparsity loss term to the loss function.

The proposed training procedure above fixes n (the number of units) for simplicity to a manually set value. By swapping the k-means initialization with a different cluster algorithm, e.g. a density-based one, n could also be inferred from the data.

6.1.3 Implementation

We implemented the model in TensorFlow (Abadi et al., 2016) and made the code publicly available[1]. Encoder and decoder are either 2-layer respectively 4-layer stacked LSTMs with a hidden size of 256. The bottleneck layer has a size of 32. We use ADAM (Kingma and Ba, 2014) for training the network with a learning rate of 0.001. For initializing the memory subnetwork we use scikit-learn's (Pedregosa et al., 2012) implementations for k-means++ (Arthur and Vassilvitskii, 2007) and DPGMM. The memory values are either initialized with the cluster centers from k-means, or the mean vectors of the DPGMM components. They are initialized from a subset of all training data, we randomly sample 1000 utterances and subsample them further to obtain 2 million vectors for the initial clustering. We use Unspeech vectors (see Chapter 5) trained on the same data with a size of 256 dimensions. Feature vectors (MFCC and PLP) are created with Kaldi (Povey et al., 2011). We also created our own phone and word alignments on Librispeech's 360 hour clean subset (Panayotov et al., 2015) using force-alignment with a strong speaker-independent GMM-HMM trained with Kaldi (7% WER on clean read speech).

6.1.4 Evaluation

Visualization of different sparsity values: Figure 6.5 shows an example input and reconstruction, along with pseudo-posteriors of training runs with different sparsity weights in the sparsity term of the loss function. The pseudo-posteriors are generated by running a forward pass on the encoder and memory component of the network and taking the output after the internal softmax. It also shows that the reconstruction is similar to the input, even though we zero out the connection between decoder and encoder with a probability of 66.6% (sequence dropout). The decoder seems to be able to guess missing feature vectors fairly well. Higher values for the sparsity weight yield sparser representations; at a sparsity weight of 10.0, the representations are mostly one-hot encoded.

[1] See `https://gitlab.com/milde/sparsespeech` (last accessed: April 2022)

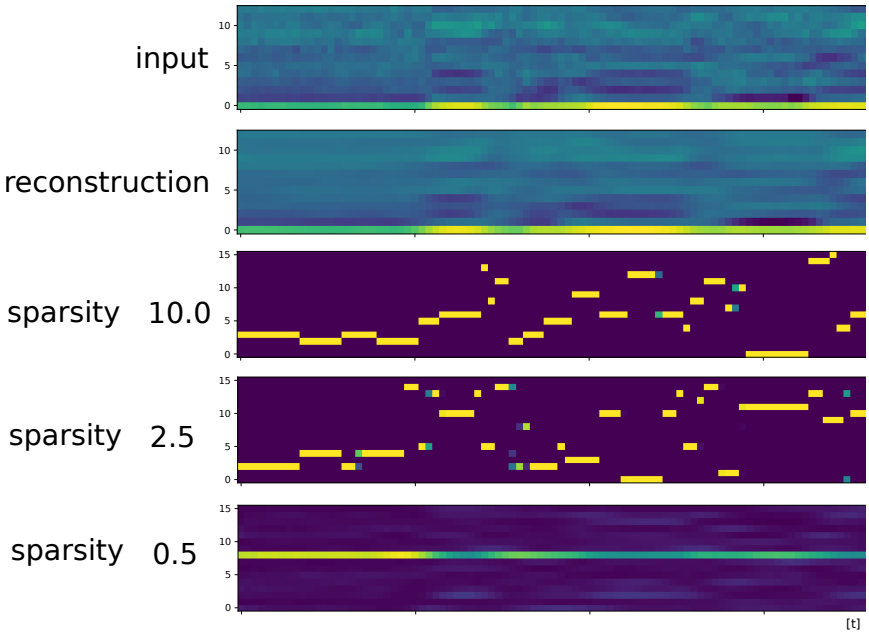

FIGURE 6.5: Training runs with varying sparsity weights (n=16).

TABLE 6.1: Baseline ABX values on English (Librispeech).

	ABX within		ABX across	
	word	triphone	word	triphone
MFCC	18.2	30.3	28.1	41.4
PLP	17.8	29.6	27.6	39.7
PLP-DPGMM	7.5	19.0	11.0	26.9

TABLE 6.2: Word and triphone minimal pair ABX error rates of the proposed model on 360 hours of English read speech (Librispeech).

context vector	stacked layers	memory init	sparsity	diversity	n	training epochs	ABX within speakers word	ABX within speakers triphn-MP	ABX across speakers word	ABX across speakers triphn-MP
Encoder mean	2	*k*-means++	10.0	0	10	10	7.8	18.4	9.2	22.4
Encoder mean	2	*k*-means++	10.0	0	16	10	6.9	17.2	8.5	22.0
Encoder mean	2	*k*-means++	10.0	0	20	10	6.7	16.8	9.2	22.5
Encoder mean	2	*k*-means++	10.0	0	20	1	8.0	18.4	11.1	25.2
Encoder mean	2	*k*-means++	10.0	0	42	10	7.8	17.9	12.6	26.5
Encoder mean	2	*k*-means++	10.0	0	80	10	9.4	19.3	15.2	29.0
Unspeech	2	*k*-means++	10.0	0	20	10	7.3	17.5	9.9	23.8
Unspeech	2	*k*-means++	10.0	0	42	10	8.5	18.8	13.4	27.7
Encoder mean	2	*k*-means++	0	0	16	10	8.8	18.0	16.2	29.5
Encoder mean	2	*k*-means++	0.5	0	16	10	9.2	18.5	17.1	30.6
Encoder mean	2	*k*-means++	5.0	0	16	10	6.7	16.4	8.2	20.1
Encoder mean	2	DPGMM	8.0	0	(max 80)	10	8.7	18.3	13.6	26.8
Encoder mean	2	DPGMM	10.0	0	(max 80)	10	8.5	18.1	13.4	26.6
Encoder mean	2	*k*-means++	10.0	10.0	16	10	6.9	16.9	8.6	21.5
Encoder mean	2	*k*-means++	2.0	10.0	16	10	5.5	14.5	6.7	18.2
Encoder mean	4	*k*-means++	2.0	10.0	16	10	5.5	**14.1**	**6.4**	17.5
Encoder mean	4	*k*-means++	2.0	100.0	16	10	**5.3**	**14.1**	**6.4**	**17.1**

ABX evaluation: In an ABX task, we test if stimulus A or B is closer to X. The minimal pair ABX discriminability (Schatz et al., 2013; Schatz, 2016) is an error measure that applies this scheme to speech sounds. We use two different forms of ABX tasks in our evaluation. In a word ABX task, A and B are different words, e.g. A=dog

TABLE 6.3: Triphone minimal pair ABX error rates on the ZeroSpeech 2017 test set (English), lower is better.

	MP-ABX within			**MP-ABX across**		
Features	1s	10s	120s	1s	10s	120s
(A) MFCC (Dunbar et al., 2017)	12.0	12.1	12.1	23.4	23.4	23.4
(B) Heck et al. (2017)	6.9	6.2	6.0	10.1	8.7	8.5
(C) Pellegrini et al. (2017)	9.8	8.1	8.2	17.6	16.2	16.3
(1) Bottleneck (dense)	9.7	9.7	9.7	23.4	23.4	23.4
(2) Bottleneck 360h (dense)	9.6	9.7	9.7	22.2	22.2	22.2
(3) n=20, KL	13.4	13.3	13.3	22.2	22.3	22.3
(4) n=20, 360h, KL	11.1	10.7	10.6	17.1	16.8	16.7
(5) n=42, 360h, KL	12.2	12.0	12.0	21.7	21.5	21.4
(6) n=16,+div, 360h, KL	9.4	8.9	9.0	14.1	13.3	13.2
(7) + 4-layer LSTM, KL	9.5	9.0	8.9	14.0	12.8	12.4
(B) + VQ argmax	22.6	11.5	11.8	30.1	16.2	16.7
(7) + VQ argmax	11.1	10.5	10.3	15.9	14.7	14.2

B=cat[1] X=cat[2] where B is closer to X. In a triphone minimal pair (MP) task, e.g. A=beg B=bag[1] X=bag[2], A and B are differentiated only by a center phone. The task includes all triphone phoneme sequences that appear in the data, not only the ones that form words. The ABX error measure discriminates between within-speaker and across-speaker tasks, in the former B and X are from the same speaker and in the latter they are uttered by different speakers. The distance metric is representation agnostic and uses Dynamic Time Warping (DTW) (Vintsyuk, 1968) to compare if A or B is closer (smaller distance) to X. For all posteriogram-like outputs, we use the Kullback-Leibler (KL) divergence as local comparison function, cosine distance otherwise.

The ZeroSpeech challenges in 2015 and 2017 used comparatively little speech data, 5 hours respectively 45 hours of English speech. We assume that neural models need more unlabeled data to learn effective representations and base our evaluation on training on the 360 hour subset of the Librispeech corpus (Panayotov et al., 2015) for English read speech. While our model can work with any feature representation, we follow the winning systems of the ZeroSpeech 2017 challenge (Dunbar et al., 2017) and train on perceptual linear predictive (PLP) features (Hermansky, 1990).

We use a randomized and unweighted version of the ABX measure (the random seed is fixed between all comparisons) to evaluate word and minimal-pair ABX. We sample 400 pairs of speakers from the corpus and select either all common words or triphone sequences for ABX tasks. We show baseline scores for common feature representations and a PLP-DPGMM in Table 6.1. Note that our ABX seems to be more difficult than the English subset in Task 1 of the ZeroSpeech 2017 challenge, as the baseline scores are higher on the triphone minimal-pair task. We computed the PLP-DPGMM with $1/10$ of the training data and restricted the maximum number of

components to 80, as we found it difficult to scale the DPGMM training to 360 hours.

In Table 6.2, we compare (sampled) word and triphone-MP ABX error rates on different sparsity and cluster settings on the 360 hour Librispeech set. DPGMM-based memory value initialization (effectively running DPGMM clustering on the encoder states) does not seem to be more effective than the much faster and simpler k-means-based initialization. Using the encoder-sum vector as context vector for the sequence is slightly better than using a fixed and pretrained Unspeech context embedding. With k-means, the best ABX scores across speakers are obtained with k=16, while k=20 yields slightly better within-speaker ABX scores. Training longer than 1 epoch after the initial cluster initialization improves ABX errors by a large margin. Disabling the sparsity constraint by setting it to zero yielded very high ABX error rates, reflecting its important role in our setup. The best ABX scores are obtained by additionally using the diversity constraint. Using 4-layer LSTM encoders and decoders is slightly more effective than 2 layers.

In Table 6.3 we use the evaluation scripts and English test data of the ZeroSpeech 2017 challenge. (1) and (3) are trained using the 45 hours train set of the challenge. The bottleneck features are dense and from the pretrained network without the memory component; they seem to provide improvements to within-speaker ABX scores over the baseline (A). Pre-training on more data does not seem to improve this significantly (2). System (3), trained with a sparsity weight of 10.0, outputs a quasi-symbolic representation. While this representation results in higher within-ABX error rates than MFCCs (A), it performs better across speakers. Systems (4) and (5), trained on an order of magnitude more data with 360 hours but the same parameters otherwise, drastically reduce within and across ABX error rates over System (3). Reference system (B) provides lower ABX error rates than our system (7), but the generated posteriors are 1144-dimensional and they are not as sparse as the ones generated by our system; for 10s the mean max. value is 0.386 vs. 0.986. In the last two rows, we compare (7) and (B) under the constraint that the representation must be completely sparse (symbolic) at each time step, i.e. we take the argmax and use indices in the comparison. In this case, our system provides significantly better ABX scores than the reference system (B).

6.1.5 Conclusion

We presented a novel neural approach to unsupervised acoustic unit discovery, based on a sequence autoencoder with a sparsity inducing memory component. The proposed sparsity constraint restricts the model to develop a quasi-symbolic representation. We propose an iterative training procedure, where the network is pre-trained without the sparsity memory component first. The architecture can be trained with standard back-propagation. This makes it easily possible to scale the training to larger training sizes. We were able to train the model on up to 360 hours of speech in two days on a single GPU. Extending this to much larger training sizes should

be easily possible. Our generated representations are very compact with few sub-word units and are mostly one-hot encoded symbolic-like representations, with better ABX discriminability than MFCC.

We can also confirm that a PLP-DPGMM model is a strong baseline for the ABX task, even though the training objective operates on the frame level. It often produces many hundreds of sub-word units when not restricted, which may be impractical in certain tasks, see (Wu et al., 2018). Also, we found it difficult to scale DPGMM to larger training sizes. Scaling our model to 360 hours of data provides representations with better ABX discriminability than PLP-DPGMM with speaker-independent transformations (Heck et al., 2017) under the constraint that the output representation needs to be completely sparse and symbolic.

6.2 Categorical Reparameterization

Discrete variables are difficult to train directly in a neural network, as the back-propagation algorithm (see also Section 2.10) cannot by applied to a non-differentable layer. In Section 6.1.2, we applied a sparsity loss to favor representations that approximate a one-hot encoding. One drawback of this original sparsity constraint is that it cannot be changed when doing inference, as it is a hyper-parameter at training time.

In the following sections, we use and evaluate categorical reparameterization (Jang et al., 2017) by Gumbel-Softmax (Gumbel, 1948) to implement approximate discrete inference within the Sparsespeech network. This allows us to model a discrete unit within Sparsespeech, while still allowing the network to be trained with standard back-propagation training. We introduce an additional parameter that can be used to control the sparseness of the pseudo-posteriorgram representations that our model generates after training. The approach is based on using Gumbel-softmax to address the memory component in Sparsespeech.

Also using what unsupervised acoustic models learn and transferring that knowledge in semi-supervised and transfer learning settings is of considerable practical interest. These learning settings hold the promise to boost performance of supervised systems, especially in low-resource settings. In this work, we extend and evaluate an unsupervised acoustic model originally proposed for acoustic unit discovery also in a semi-supervised setting. A large amount of untranscribed speech data (600h-6000h) and only a small amount (10h) of transcribed speech training data is available in this learning setting.

Using the masking technique for unsupervised modelling and then fine-tuning is reminiscent of masked language models such as BERT (Devlin et al., 2019), which are currently very popular on text data. As speech is continuous, using the idea of masking becomes a bit more difficult to transfer directly. In contrast to masked language models, we use the masking technique on the internal representation, i.e.

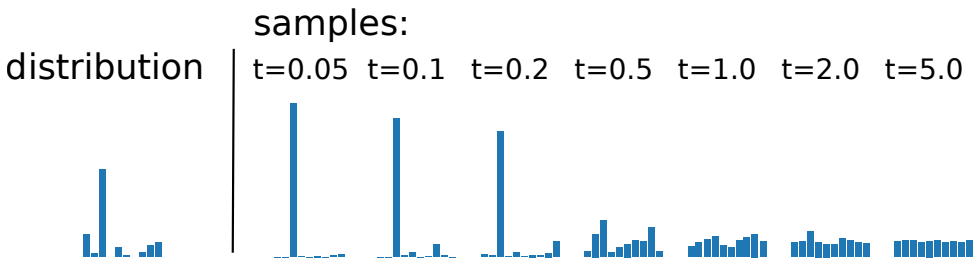

FIGURE 6.6: Drawing samples with different temperatures with the Gumbel-Softmax from a discrete distribution.

the pseudo-posteriograms used for reconstruction at each time step and not on the input representation.

We use the softmax function as a differentiable approximation to argmax as follows: We sample a noise vector $\boldsymbol{g} = g_1 \dots g_k$ from a Gumbel distribution (Gumbel, 1948) with a uniform random sampler U:

$$\boldsymbol{g} = -log(-log(U(0,1))) \tag{6.3}$$

where k is the number of elements in the softmax. We then compute the Gumbel-Softmax as:

$$softmax(\frac{logits + \omega \cdot \boldsymbol{g}}{\tau}) \tag{6.4}$$

where ω is a noise weight parameter. We set $\omega = 1$ while training the network and $\omega = 0$ after the training is completed to disable the Gumbel noise for inference. The temperature parameter τ controls the amount of sparsity of the sample drawn from the distribution provided by the (unscaled) input logits. We illustrate this in Figure 6.6 with example samples drawn from the same distribution with varying τ. Lower temperatures (0.05, 0.1, 0.2) tend to make the drawn samples sparse, approximating a one-hot vector, while higher temperatures (2.0, 5.0) increase density and approximate a uniform distribution.

While training the network we use annealing, starting with a higher temperature and slowly decreasing it to a cutoff value below 0.5, for example $\tau = 2 \rightarrow 0.2$. In the Sparsespeech model, the Gumbel-Softmax replaces the regular softmax with the sparsity constraint. Figure 6.7 illustrates the complete Sparsespeech model with the added Gumbel-Softmax.

6.2.1 Setup

For evaluation we use the supplied auxiliary scripts of the Libri-light corpus[2] with minor enhancements such as the possibility to use Kullback-Leibler (KL) (Kullback

[2] `https://github.com/facebookresearch/libri-light` (last accessed: December 2021)

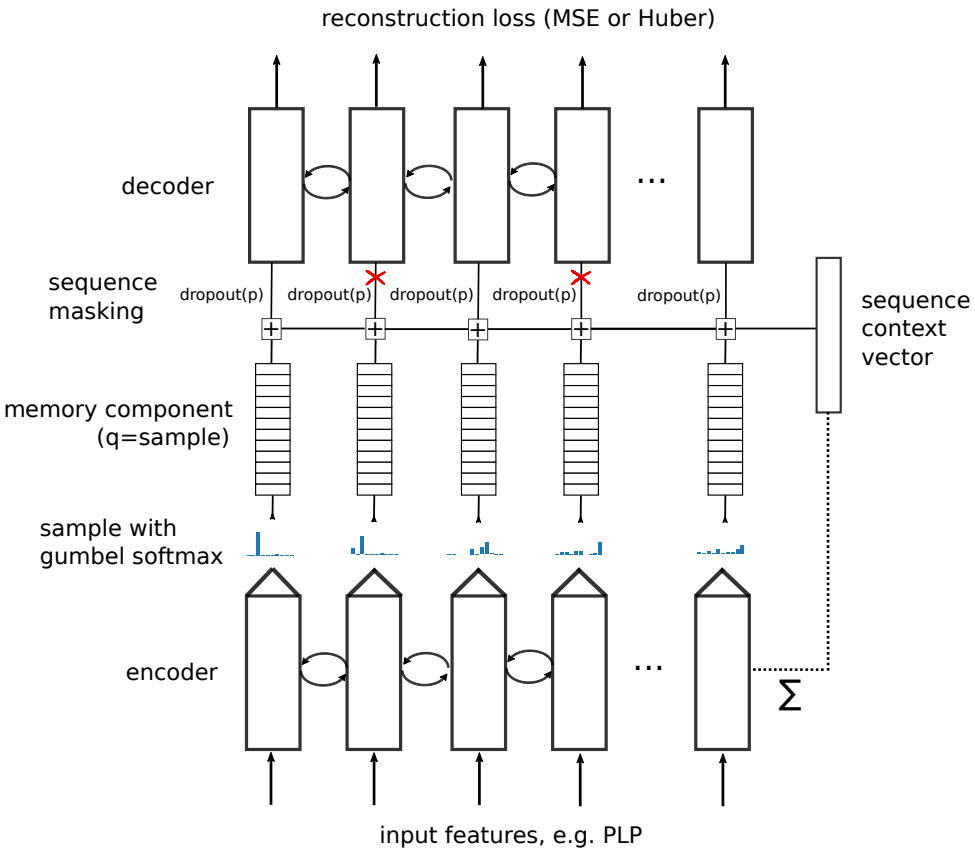

FIGURE 6.7: The Sparsespeech unsupervised acoustic model with Gumbel-Softmax (Milde and Biemann, 2020).

and Leibler, 1951) as a distance function in the ABX evaluation[3]. KL is a better metric to compare pseudo-spectograms such as the ones our model generates, while the default cosine distance function of the Libri-light scripts is better suited for comparing embedding representations.

We use 4-layer stacked BiLSTM decoder/encoders in our Sparsespeech models, with a width of 256 neurons for all experiments. Perceptual linear predictive (PLP) (Hermansky, 1990) input features are computed with Kaldi (Povey et al., 2011), using the standard settings of 13 dimensions and 100 frames per second on (downsampled if necessary) 16kHz audio. The sparsity constraint of Sparsespeech is disabled (sparsity weight set to 0) for all experiments with the new model, while the diversity constraint of the original model is kept and the diversity weight set to 100. We keep the multi-stage training approach of the original model (see Section 6.1.2) where the model is pre-trained without the memory component. For the second training stage, we use temperature annealing while training the network: the τ parameter for Gumbel-Softmax is set to 2.0 and then slowly decreases by multiplying with an annealing factor (0.9999) every x batches. A cut-off parameter, 0.1 or 0.2 in most experiments, is set after which the annealing scheme stops.

For the reconstruction loss we use MSE, as proposed in Section 6.1.2 and also

[3]This change has been merged into the Libri-light repository.

evaluate with Huber loss (Huber, 1964), as it gives less weight to outliers. See also Section 2.9 for a visual depiction of the Huber loss function.

6.3 Evaluation

We evaluate on two proposed evaluation tasks in Libri-light (Kahn et al., 2020): completely unsupervised and semi-supervised with limited supervision on English audiobook read speech. We currently focus on the 600h (small) and 6000h (medium) subsets of untranscribed speech to train our models. In the unsupervised evaluation, we measure ABX error rates (Schatz et al., 2013; Schatz, 2016), as also used in Section 6.1.4. This provides an error rate that measures how well the trained unsupervised representation can differentiate between same/different tri-phones within and across speakers, for example "bit" vs. "bat". We use the dev sets to calibrate parameters and test the best performing models on the test set. The ABX error measure uses DTW to compare two segments of different length and we use symmetric KL divergence as the local comparison function. This is the recommended distance function for posteriorgram-like representations (Dunbar et al., 2017; Dunbar et al., 2019). In the semi-supervised setting, we first train a Sparsespeech model on the unannotated data from Libri-light. We then follow (Kahn et al., 2020) and evaluate with a simple phoneme classifier with Connectionist Temporal Classification (CTC) loss (Graves et al., 2006) that is trained on the representation with 10h of limited-resource phone labels.

TABLE 6.4: ABX error rates on features/posteriograms generated by our model for the **Libri-light dev set**, with varying temperature τ.

Model or features	**Temp.**	**within speaker**		**across speaker**	
	τ	clean	other	clean	other
PLP Features	-	11.12	15.08	25.87	33.74
S6000h-n42-$\tau 2 \rightarrow 0.1$	0.2	12.66	15.52	18.86	24.84
"	0.8	11.04	13.65	17.02	23.01
"	1.0	10.66	13.25	16.34	22.55
"	2.0	9.57	12.15	14.73	20.68
"	**3.0**	**9.51**	**12.15**	**14.41**	**20.25**
"	5.0	10.48	12.94	15.28	20.87

In Table 6.4, we generate pseudo-posteriorgrams with different temperatures τ from the same model. This model has been trained on 6000h, with 42 components in the memory bank and output representation (n42) and a temperature annealing training scheme of $\tau 2 \rightarrow 0.1$. The sparseness of the output can also be controlled with τ after training. The ABX error measure is sensitive to too sparse representations, as a different scaling with a higher τ significantly reduces the error measure. Temperatures 2.0 and 3.0 produced the lowest ABX error rates.

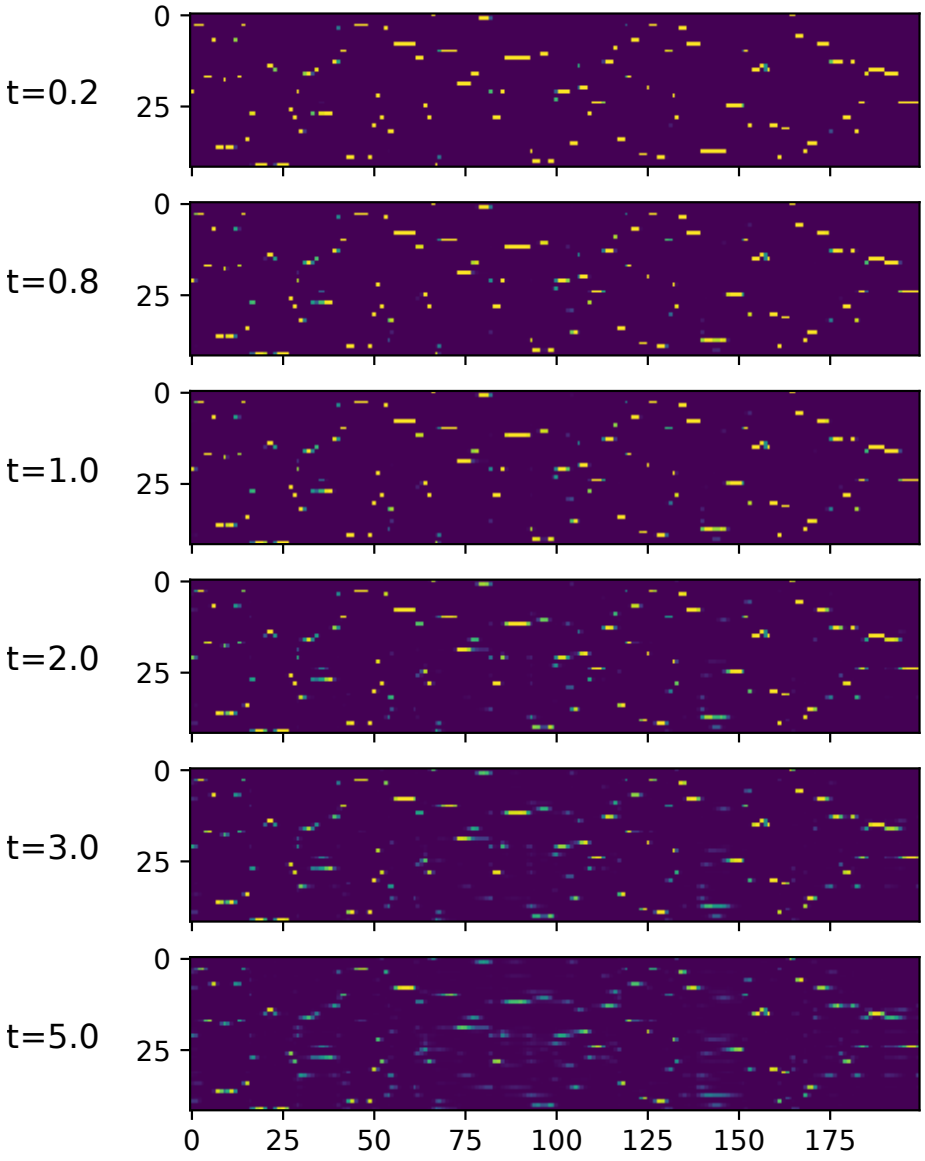

FIGURE 6.8: Example feature representations generated by the Sparsespeech model "S6000h-n42-$\tau 2 \rightarrow 0.1$" with varying temperature.

In Table 6.5, we mainly evaluate different n in the memory component of Sparsespeech, trained on the Libri-light small subset of 600h. Using $n = 100$ or $n = 128$ components produced good within-speakers results, $n = 42$ performed better on the across-speakers ABX error. All models have been trained for 3 epochs, with a training time ranging from 23.8h to 30.4h for the second stage of training (with the memory component) on a single Nvidia Titan XP GPU. We have experimented with different annealing schemes, but settled on $\tau = 2.0 \rightarrow 0.2$ for most experiments. Most models have been trained using MSE as reconstruction loss. For $n = 20$ and $n = 42$, we also trained models with Huber loss, which further improved ABX error rates. Additionally, we trained Sparsespeech models ranging from $n = 20$ to $n = 128$, using the original sparsity loss training method without Gumbel-Softmax.

TABLE 6.5: ABX error on features/posteriograms generated by our model for the **Libri-light dev set** with different n (components in the memory bank). Best results of each section in bold.

Model or features	**Temp.**	**within speaker**		**across speaker**	
	τ	clean	other	clean	other
PLP Features (n=13)	-	**11.12**	**15.08**	25.87	33.74
S600h-n20-sparsityloss-2.0	-	14.65	17.37	27.09	32.43
S600h-n20-sparsityloss-10.0	-	13.96	17.04	**21.48**	**26.09**
S600h-n42-sparsityloss-10.0	-	14.23	16.16	22.38	27.24
S600h-n100-sparsityloss-2.0	-	14.03	16.63	24.80	29.67
S600h-n128-sparsityloss-2.0	-	15.55	17.94	26.82	31.56
S600h-n20-$\tau 2 \rightarrow 0.5$	2.0	11.56	13.75	21.18	26.66
S600h-n20-$\tau 2 \rightarrow 0.2$	2.0	11.57	13.76	21.12	26.65
S600h-n42-$\tau 2 \rightarrow 0.2$	3.0	11.38	13.49	**17.64**	**22.46**
S600h-n100-$\tau 2 \rightarrow 0.2$	3.0	10.43	13.00	18.86	24.08
S600h-n128-$\tau 2 \rightarrow 0.2$	3.0	**10.00**	**12.46**	17.97	23.16
S600h-n256-$\tau 2 \rightarrow 0.2$	3.0	11.41	14.02	22.73	27.55
S600h-n20-$\tau 2 \rightarrow 0.2$-huber	2.0	11.11	13.45	16.37	**21.34**
S600h-n42-$\tau 2 \rightarrow 0.2$-huber	3.0	**10.30**	**12.82**	**16.34**	21.68

For $n = 20$, the sparsity loss constraint weight needed to be increased, as the recommended default (2.0) produced a degenerate solution. The original models did not show good ABX error rates on the Libri-light dev set. In fact, only accross-speaker ABX error improved over the PLP features baseline. The new models trained with Gumbel-Softmax show significant relative error rate improvements over the original Sparsespeech model, with nearly all tested representations better than PLP features in all settings.

In Table 6.6, we compare ABX error rates with some selected models on the Libri-light test set. We compare against baseline PLP features, a baseline Sparsespeech model trained with the original sparsity loss method without Gumbel-Softmax and representations from Contrastive Predictive Coding (CPC) (Oord et al., 2018) as reported in (Kahn et al., 2020). While the Sparsespeech models are trained on PLP features (n=13) as input, the CPC are trained on raw 16kHz speech in the time domain. Like on the dev set, there is a large reduction in error rates when training with Gumbel-Softmax. On the larger medium subset, a second training run using the Huber loss also further improved ABX error rates. However, the dense embedding representations trained with the CPC model show lower error rates than the best Sparsespeech model that we trained on the 6000h medium subset.

In Table 6.7, we compare the representations of our models in terms of how well a simple phoneme recognizer can classify phonemes with it as input. The phoneme recognizers are trained on 10h of representations with phone labels. They are trained without explicit alignments using the CTC loss. Using only a linear 1D-convolution

TABLE 6.6: ABX error on features/posteriograms generated by our model for the **Libri-light test set**. CPC results are from (Kahn et al., 2020).

Model or features	**Temp.**	**within speaker**		**across speaker**	
	τ	clean	other	clean	other
PLP Features (n=13)	-	**10.46**	**14.69**	23.78	34.15
S600h-n20-sparsityloss-10.0	-	13.66	16.83	**19.78**	**26.56**
S600h-n100-sparsityloss-2.0	-	14.12	16.97	22.86	30.53
S600h-n20-$\tau 2 \rightarrow 0.5$	2.0	10.92	14.06	18.86	27.16
S600h-n42-$\tau 2 \rightarrow 0.2$	3.0	10.59	13.78	15.68	23.16
S600h-n128-$\tau 2 \rightarrow 0.2$	3.0	9.39	12.42	16.01	23.49
S6000h-n42-$\tau 2 \rightarrow 0.1$	3.0	**9.33**	**12.05**	**13.53**	**20.60**
S600h-n20-$\tau 2 \rightarrow 0.2$-huber	2.0	10.4	13.82	15.32	22.43
S600h-n42-$\tau 2 \rightarrow 0.2$-huber	3.0	9.69	12.79	14.68	22.05
S6000h-n42-$\tau 2 \rightarrow 0.2$-huber	3.0	**8.79**	**11.62**	**12.55**	**19.84**
$\hookrightarrow$ +argmax		12.43	16.58	18.11	25.93
CPC-600h (n=256)	-	6.90	9.59	9.00	15.10
CPC-6000h (n=256)	-	6.22	8.55	8.05	13.81
CPC-60000h (n=256)	-	**5.83**	**8.14**	**7.56**	**13.42**

on posteriorgram-like representations as in (Kahn et al., 2020) proved to be challenging, as the most frequent emission symbol per timestep with the CTC loss is the blank label. Adding a simple 1-layer LSTM ensures that the network can learn when to emit a label other than the blank label and also keep track of context. The 1D convolution has a kernel size of 8 (default in the Libri-light evaluation script) and the number of output channels of the convolution is set to match the number of phones in the transcription plus the blank label (45). The 1-layer LSTM has a fixed hidden size of 100.

A simple decoder with beam search generates the hypothesized phone sequence. Phoneme error rate (PER) is then computed by comparing the sequence to the Libri-light transcriptions on the dev and test set (these sets are the same as the ones in Librispeech (Panayotov et al., 2015)). With the original Sparsespeech model we do not significantly surpass the PER results of the PLP baseline, but with the improved Sparsespeech model the phoneme recognizer can improve PER by 14.1% relative to the PLP baseline on test-clean and by 8.1% relative to test-other.

6.4 End-to-end Sparsespeech model

In the previous sections, the Sparsespeech model still relies on engineered features of the raw audio data in the time domain. In the following, we replace the engineered features with a neural network component that can be directly applied to raw audio in the time domain using 1D convolutions. For the encoder, we follow the construction in the CPC model, with multiple stacked 1D convolutions applied

TABLE 6.7: PER error for training a very simple phoneme recognizer with 10h of data on: PLP features, CPC model features or Sparsespeech model features.

Model or features	**Temp.**	**dev PER**		**test PER**	
	τ	clean	other	clean	other
PLP Features (n=13)	-	**52.44**	62.36	50.96	**63.13**
S600h-n100-sparsityloss-2.0	-	52.48	**61.65**	**50.48**	63.23
S600h-n20-sparsityloss-10.0	-	57.22	64.84	55.40	65.95
CPC-600h (n=256)	-	40.21	51.80	38.18	53.85
CPC-6000h (n=256)	-	34.40	47.60	34.44	49.40
CPC-60000h (n=256)	-	**31.16**	**46.67**	**32.67**	**48.93**
S600h-n100-$\tau 2 \rightarrow 0.2$	3.0	50.39	59.82	48.29	61.75
S600h-n128-$\tau 2 \rightarrow 0.2$	3.0	50.56	60.20	48.05	61.69
S600h-n42-$\tau 2 \rightarrow 0.2$-huber	3.0	49.80	59.09	47.16	60.36
S6000h-n42-$\tau 2 \rightarrow 0.1$	3.0	47.77	57.77	46.61	59.61
S6000h-n42-$\tau 2 \rightarrow 0.2$-huber	3.0	**45.18**	**56.31**	**43.77**	**58.02**

to the signal. Since the CPC model is not generative, it does not need a decoder generating the 1D signal in the time domain. On the other hand sparsespeech is a generative model and we construct a matching decoder using 1D-deconvolutions to reverse the 1D-convolutions and generate audio.

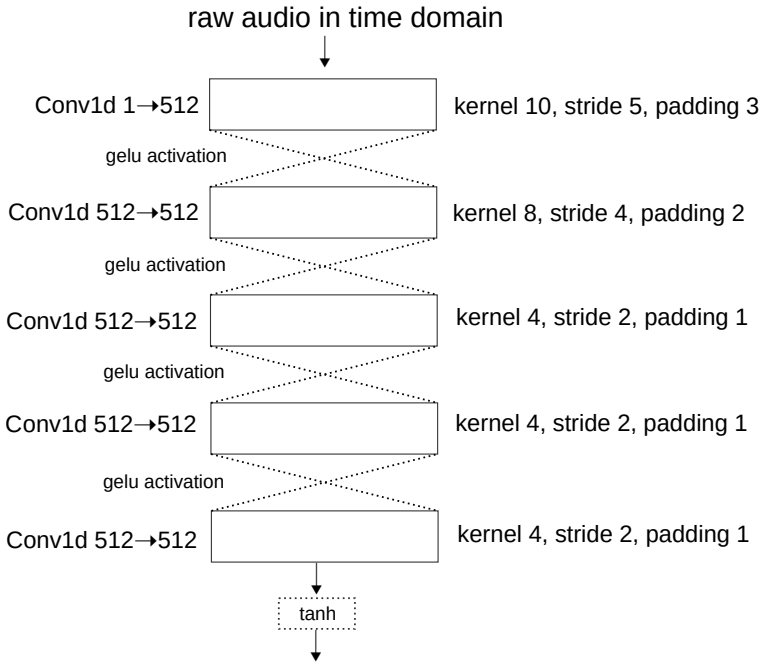

FIGURE 6.9: Raw encoder for the Sparsespeech model. Follows the implementation in (Oord et al., 2018) and (Kahn et al., 2020).

Figure 6.9 and Figure 6.10 illustrate the encoder and decoder. The raw encoder takes raw audio in the time domain (16kHz) as input, and outputs a sequence of 2D embeddings with similar shape as engineered speech features (FBANK, MFCC,

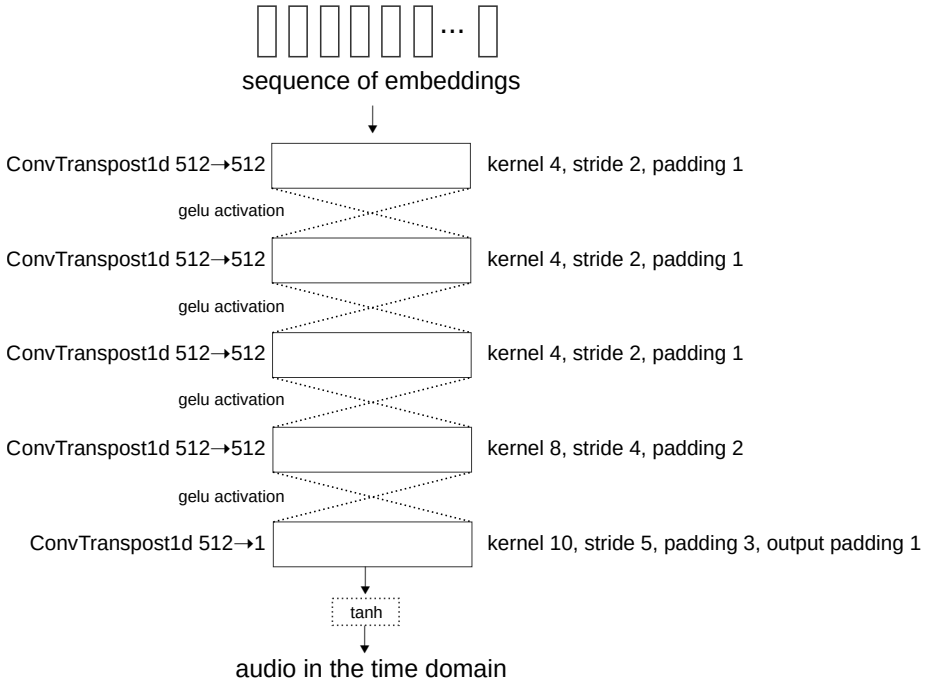

FIGURE 6.10: Raw decoder for the Sparsespeech model.

PLP...). The design of the encoder follows the implementation in (Oord et al., 2018) and (Kahn et al., 2020). A series of 1D conv is applied to the signal, denoted with the number of input and output channels in Figure 6.9, using GELU as activation function. Tanh is applied to the output to squish all output values to (-1,1). The raw decoder applies ConvTransposed1D, effectively reversing each 1D convolution with a matching deconvolution. The last ConvTransposed1D has only one output channel, so that it outputs a sequence of values in the time domain. Tanh is also applied to the all output values, so that the signal is squish to values in (-1,1).

An appropriate loss function is also needed to compare a generated audio sequence with the input audio sequence. Directly applying an error function in the time domain is too fine-grained and will not yield any useful learning, as SGD will get immediately stuck (we have verified this by testing it with an implementation). However, we can apply a Short-time Fourier transform (STFT), as it is a differentiable function as well. We can then compare the values in the frequency domain with Huber loss. We are basically comparing spectograms in the loss function. Additionally, we can also apply triangular filter function to further lower the dimensionality in the frequency domain and make the loss function a little more "forgiving" if frequencies are close, but do not match exactly. Like in MFCC features, we can also apply the Mel scale.

6.4.1 Implementation details

We have reimplemented[4] the Sparsespeech model in PyTorch (Paszke et al., 2019) and tested the end-to-end model with PyTorch version 1.10 and Python 3.8. For strides greater than 1 the output shapes for ConvTransposed1D can be ambiguous in PyTorch, the ambiguity can be resolved with the setting "output padding" parameter. This was necessary for the final ConvTransposed1D layer and is also denoted in Figure 6.10.

For the KL-based diversity loss we hit stability problems with PyTorch. For numerical stability, we used Equation 6.5 instead:

$$\text{kl\_diversity}(P, n) = -(max(P(x), 0) - P(x)^2 n + log(1 + e^{-|P(x)|})) \tag{6.5}$$

where $P(x) = \frac{1}{len(X)} \sum_{x \in X} x$, the mean vector of all pseudo-posteriogram representations x in a sequence X and n is its dimension. Recall that cross entropy $H(P, Q)$ and $KL(P||Q)$ are closely related (see also Section 2.9). KL can thus also be expressed in terms of $H(P, Q)$:

$$D_{KL}(P||Q) = -H(P, \frac{P}{Q}) \tag{6.6}$$

In Section 5.3.3, we show a numerically stable version of a sigmoid cross entropy that TensorFlow uses:

$$h(x, C) = max(x, 0) - x \cdot C + log(1 + e^{-abs(x)}) \tag{6.7}$$

With $x = P(x)$, $C = P(x)/\frac{1}{n} = P(x)n$, we get kl_diversity($P$,$n$) above.

6.4.2 ABX evaluation

TABLE 6.8: ABX error on features/posteriograms generated for the **Libri-light dev set** with the end-to-end Sparsespeech model.

Loss function	**with KL**	**within speaker**		**across speaker**	
		clean	other	clean	other
STFT (magnitude)	×	35.92	35.54	39.56	40.09
STFT (magnitude)	✓	34.74	34.94	37.68	38.60
Mel spectrum, n=40	✓	30.90	35.93	35.93	37.22
Mel spectrum, n=13	✓	27.81	34.76	34.76	35.52
PLP baseline	N/A	11.11	15.08	25.86	33.65

[4] `https://gitlab.com/milde/sparsespeech2` (last accessed: April 2022)

In Table 6.8, we list ABX scores for the end-to-end Sparsespeech model. Generally, ABX error are much higher than for the feature based model. The PLP baseline is better than all obtained ABX results with end-to-end model. The gap to the PLP features baseline is smallest for the across speaker ABX results. A significant influence on the results is the reconstruction loss function used in the end-to-end model. Processing steps in the loss function that mimic traditional speech features are more successful than directly comparing the magnitudes of the STFT coefficients.

6.5 Conclusion

Using Gumbel-Softmax and Huber loss in the Sparsespeech model is an effective improvement. On representations with $n = 20$, this yields a relative reduction of 22.5% in ABX error rates on the test set (with clean speech) across speakers compared to the original model (Milde and Biemann, 2019) (see also Section 6.1.2). The dimensionality of the learned representations of the new Sparsespeech model can now also be scaled up to representations with larger n. This also further improved ABX error rates. So far, $n = 100$ and $n = 128$ yielded the best results for within-speaker ABX when trained on 600h of untranscribed speech. Representations with bigger n also show a relative improvement of up to 31.3% on ABX error rates within speakers on the clean test set compared to the best original model. Our first results for training on the 6000h medium subset of the Libri-light corpus further improved error rates and show that the model is scalable. Currently, tuning the temperature parameter after a Sparsespeech model has been trained seems to be important to reduce ABX error rates, but higher temperatures when generating Sparsespeech features such as 2.0 and 3.0 seem to work well across models with different hyperparameters.

PER error rates also show an 14.1% improvement over a PLP baseline with the new model when a simple phoneme recognizer is trained on the representations. The generated representations from the new model are still relatively compact and sparse (see also Figure 6.8) with better phoneme discriminability as measured by ABX and PER than PLP features. However, when we compare ABX and PER error to unsupervised dense embedding representations such as the ones generated by CPC (n=256), there is still a relatively large gap in error rates on the Libri-light test set. One difference is the type of input features; CPC uses raw waveforms in the time domain while we have mainly used PLP features. Training Sparsespeech on raw waveforms does not seem to perform well, also feature engineering of speech features is still necessary in the reconstruction loss. CPC representations could however potentially also be used as input features to the Sparsespeech model.

Another major difference is structural in the type of the generated representations. There might be a trade-off in ABX error rates between low-bitrate sparse representations and higher bitrate dense representations. The results from last year's Zero Resource Challenge (Dunbar et al., 2019) support this hypothesis, with systems with higher ABX errors having lower bit rate representations. The organizers

concluded that "this suggests that discretizing learned speech embeddings well is hard". The pseudo-posteriorgrams that our Sparsespeech model can generate have the advantage over embeddings that they can directly be interpreted as a (soft) clustering of phoneme-like units. They can also be easily discretized and translated to symbolic pseudo transcriptions, where the ABX discriminability is still largely preserved.

Chapter 7

Multitask Grapheme-to-Phoneme Conversion

Another crucial component of most automatic speech recognition (ASR) systems is the phoneme lexicon, mapping words to their phonetic representation (e.g. Thursday → TH ER Z D EY). It is also being used as a resource in unsupervised ASR (see also Section 3.13), to generate large quantities of phoneme sequences out of texts (Baevski et al., 2021). Creating and maintaining phoneme lexicons manually is a tedious task and needs expert phoneticians. Typically, a seed lexicon is used to train a model that can automatically produce phonetic entries, to either aid expert phoneticians or (accepting a certain error rate) to generate new entries fully automatically.

Neural sequence-to-sequence models (Seq2Seq) have emerged as a generic approach to learn to translate between sequences of varying lengths (Sutskever et al., 2014). Initially applied to machine translation, Seq2Seq has been successfully applied to a wide range of various other tasks, including conversation modelling (Vinyals and Le, 2015), sentence-level grammatical error identification (Schmaltz et al., 2016), automatic regex generation (Locascio et al., 2016) and also recently grapheme-to-phoneme (G2P) conversion (Yao and Zweig, 2015). Seq2Seq models are generative language models, conditioned on an input sequence. After encoding the input sequence token by token, the output sequence is generated token by token. No explicit alignments between input and output sequences are necessary, as the system is trained in an end-to-end fashion.

We mainly target the use of the G2P model in automatic speech recognition. Training and using a G2P model is often directly integrated into the ASR training procedure, as phonetic out-of-vacabulary (OOV) words in the training set hamper the alignment of training data to its transcriptions. Common words as determined by the language model that do not have a pronunciation entry can also be candidates for G2P conversion and can be used to extend the vocabulary of a large vocabulary speech recognition system. Other uses for G2P conversion include generating OOV entries on the fly for text to speech systems and assisting humans in the generation of phoneme lexicons.

The next section covers related work on grapheme-to-phoneme conversion and

sequence-to-sequence models. We give an overview over neural grapheme-to-phoneme sequence-to-sequence models in Section 7.2, describe our evaluation in Section 7.3, compare Seq2Seq models with and without multitask learning in Section 7.4 and present the conclusion in Section 7.5.

7.1 Related Work

Joint-sequence n-gram models are both well performing and popular traditional models for grapheme-to-phoneme conversion (Galescu and Allen, 2001; Bisani and Ney, 2008). These models need to find a joint vocabulary of graphemes and phonemes (often called graphones) by aligning characters and phonemes. The output sequence is modelled as a sequence of graphones. Particularly Sequitur G2P (Bisani and Ney, 2008) is a well established G2P conversion tool using joint sequence modelling and is also commonly used in state-of-the-art speech recognition model recipes, e.g. (Panayotov et al., 2015). Joint n-gram models can also be efficiently represented as weighted finite state transducers (Novak et al., 2013; Novak, 2012).

Yao and Zweig described the first application of Seq2Seq to English G2P in (Yao and Zweig, 2015). The decoder/encoder network yielded 17% higher WERs than Sequitur G2P. However, they achieved state of the art results on three G2P datasets by using a fixed length recurrent architecture together with the HMM many-to-many alignment procedure from Jiampojamarn et al. (2007). Rao et al. (2015) proposes to use the connectist temporal classification (CTC) instead of Seq2Seq to jointly align and translate graphemes to phonemes in a single neural model. The proposed model on its own yields 5% higher word error rates than the Sequitur G2P baseline. However, when both models are combined with a finite state transducer (FST) n-gram model, they significantly outperform the baseline. Schnober et al. (2016) also reported a negative result for sequence-to-sequence models with an attention mechanism on a G2P task with 20k lexicon entries (among other monotone string translation tasks), compared to standard methods. Pritzen et al. (2021) approved upon the results of this chapter, by classifying anglicisms at inference time and using this classification as input to the Seq2Seq model.

Tsvetkov et al. (2016) trained phonetic language models in a multitask training setting. Perplexities of the joint model over all languages are lower than individual phonetic language models. State of the art models for translation use neural sequence-to-sequence models and incorporate an attention mechanism and residual learning (He et al., 2016; Wu et al., 2016a). This model can also be trained on multiple language pairs at once: an additional unique language identifier token at the beginning of the input sentence is added to specify the required target language to translate to Johnson et al. (2017). We follow this multitask learning (Caruana, 1997) approach to train multi-language and multi-alphabet grapheme-to-phoneme conversion models.

7.2 Neural Grapheme-to-Phoneme Models

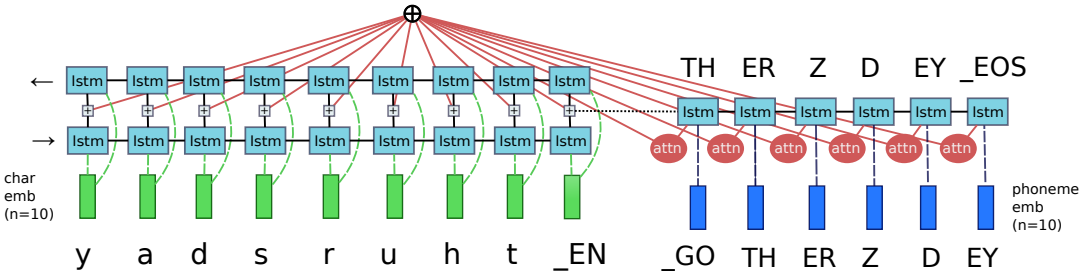

FIGURE 7.1: Seq2Seq model for G2P conversion with attention and character/phoneme embeddings, inputs are reversed. For multitask learning, we extend the source vocabulary with additional markers for the subtask that are placed at the beginning of each word.

Neural sequence-to-sequence models can learn a conditional distribution over a variable length sequence conditioned on another sequence $p(y_1, ..., y_T|x_1, ..., x_{T'})$, where T' can be different from T (Cho et al., 2014b). See also Section 2.24 for more details. When doing phoneme-to-grapheme conversion, we condition the output phoneme sequence $y_1, ..., y_T$ on the character sequence $x_1, ..., x_{T'}$. Figure 7.1 depicts a Seq2Seq architecture for G2P with a bidirectional LSTM encoder using a decoder with global attention. For multitask learning, we extend the source vocabulary with additional markers for the subtask that are placed at the beginning of each word. Since inputs are reversed, the marker will be placed at the end of the input sequence.

7.3 Evaluation

We base our evaluation on German and English G2P conversion, by comparing the model predictions on unseen data to manual entries by expert phoneticians. For German, we use Phonolex[1] core (66.8k entries) and the full Phonolex lexicon (1.4 million entries), that uses the SAMPA phonemes set. All entries in the core set are manually verified entries, while the source of the other entries is sometimes unclear, with most pronunciations entries of the full set likely to be already coming from rule based conversion systems (Schiel, 1997), automatic G2P conversions and data-driven alignments, e.g. BAS Maus (Beringer and Schiel, 2000). For English, we use CMUDICT 0.7b with about 138k manual pronunciation entries using the ARPAbet phoneme set. For all training scenarios, we preprocess the lexicon to remove stress markers, which are usually omitted in ASR acoustic modelling.

We evaluate phoneme error rate (PER) and phoneme word error rate (WER) by splitting the available data into training, development and test splits. For Phonolex (core/full), we only use manually verified entries as development and test data, i.e. we use 1,312 entries (2%) for each set of Phonolex core. For CMUDICT, we use

[1] `http://www.phonetik.uni-muenchen.de/Bas/BasPHONOLEXdeu.html` (last accessed: December 2021)

TABLE 7.1: Comparing Sequitur G2P and seq2seq-attn on the German Phonolex lexicon test data, with stress markers removed. (Best scores for single models and system combinations in bold.)

Model	Phonolex set	PER	WER	train time
(1) Sequitur G2P (model order 10)	core set	**1.98%**	11.54%	12h43
(2) Sequitur G2P (model order 6)	core set	**1.98%**	**11.30%**	3h40
(3) Sequitur G2P (model order 6)	full set	5.41%	29.86%	7.45 days
(4) seq2seq-attn (biLSTM 256x3, d=0.5)	full set	6.09%	32.69%	13h57
(5) seq2seq-attn (biLSTM 256x3, d=0.5)	core set	2.49%	13.64%	3h59
(6) seq2seq-attn (biLSTM residual 256x3, d=0.5)	core set	2.37%	12.75%	3h53
(7) seq2seq-attn (biLSTM residual 256x3, d=0.5) + multitask learning (de/en)	core set	2.57%	14.12%	5h51
(8) seq2seq-attn (biLSTM 512x3, d=0.5) + multitask learning (de/en)	core set	2.41%	13.32%	8h27
(9) seq2seq-attn (biLSTM res. 512x3, d=0.5) + multitask learning (Sampa/IPA)	core set	2.06%	**11.30%**	24h49
(10) System combination (2)+(6)	core set	1.88%	10.33%	(7h39)
(11) System combination (2)+(9)	core set	**1.70%**	**9.52%**	(28h29)

TABLE 7.2: WER and PER performance for different classes of words in the German Phonolex task. While regular German words can be phonetized with relatively small errors, loan words and named entities are particularly problematic in this G2P task.

	Abbreviation		English pronoun.		French pronoun.		Name / NE		Hyphen		Regular	
Count and % of all entries	23 (1.75%)		34 (2.59%)		8 (0.61%)		149 (11.36%)		29 (2.21%)		996 (75.91%)	
Model	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**	**PER**	**WER**
Sequitur G2P (model order 6)	**2.92%**	**8.70%**	**10.20%**	**38.24%**	35.00%	**62.50%**	**5.00%**	**23.49%**	2.43%	31.03%	1.13%	7.63%
seq2seq-attn (biLSTM 256x3)	10.59%	26.09%	16.93%	52.94%	32.81%	75.00%	5.80%	25.50%	**1.22%**	**13.79%**	1.18%	8.63%
+ multitask (de/en)	8.40%	26.09%	17.20%	61.76%	25.05%	**62.50%**	5.41%	26.10%	**1.22%**	17.24%	1.39%	9.24%
+ multitask (SAMPA/IPA)	6.72%	21.74%	12.60%	47.06%	**22.22%**	**62.50%**	5.58%	24.16%	2.22%	17.24%	**1.07%**	**7.33%**

the same train/dev/test splitting as proposed in the sequence-to-sequence example of the CNTK toolkit[2], i.e. 108,952 train entries, 5,447 dev entries and 12,855 test entries. While Phonolex core does not contain pronunciation variants, CMUDict does include them, and we also use them in the training process. Following Bisani and Ney (2008), the variant that minimizes the error is chosen for computing PER and WER.

To enable multi-alphabet training, we also generated our own German and English dictionaries using the International Phonetic Alphabet (IPA)[3]. We used recent dumps of the German and English Wiktionary project[4], containing IPA phoneme entries of voluntary contributers (not all lexicon entries have a phoneme entry). For German, we pick the top pronunciation entry. For English we include all variants, but not British ones (to favor American English, which CMUDICT also models). Otherwise, no attempt at normalizing notation styles is made. The English Wiktionary contains 4 million entries, but the largest fraction are non-English words,

[2] `https://github.com/Microsoft/CNTK/tree/master/Examples/SequenceToSequence/CMUDict/Data` (last accessed: December 2021)

[3] Scripts to reproduce the IPA dictionary generation are available at `https://github.com/bmilde/wiktionary_ipa_phoneme_lexicons` (last accessed: December 2021)

[4] dewiktionary-20170120 and enwiktionary-20170301

mostly with the exact pronunciation of the original language. Accordingly, by only using IPA entries that are clearly marked as German or English entries, we obtain an IPA lexicon of 330,473 words for German and a significantly smaller IPA lexicon of 34,943 words for English.

For Sequitur G2P, we tune the model order (n-gram) on the development set. It is also used in the training procedure to adjust the discount parameters of the joint grapheme/phoneme language model. While 6 is the recommended parameter of the training software, we found higher order models to give slightly better results on the development set.

For our sequence-to-sequence experiments we use seq2seq-attn[5], an open source neural translation system. It can be fitted to G2P conversion by preprocessing the dataset, so that single characters or phonemes are entire words. We also tune Seq2Seq models on the development set: we choose the width of the networks' LSTM layers from the set {64, 128, 256, 512}, number of layers from {2, 3, 4, 5} and dropout from {0.0, 0.2, 0.3, 0.5} and train with and without residual learning. We run all 128 configurations on a cluster with Nvidia Titan X GPUs and select promising candidate configurations on the development set. We omit hyperparameter optimization entirely for the full Phonolex set and leave out the 64 and 128 width configurations for multi-task training. We exclusively use Adam (Kingma and Ba, 2014) as optimization method for network training, with an initial learning rate of 0.001. For the first 3 epochs we use curriculum learning (sorting the sequences by length in an epoch), for all other epochs the order is randomized.

TABLE 7.3: Comparing Sequitur G2P and seq2seq-attn on a CMU-DICT test set, with stress markers removed. Results shown here are for a recent version of the lexicon (v0.7b).

Model	PER	WER	train time
(1) Sequitur G2P (model order 8)	**6.12%**	**25.71%**	34h49
(2) seq2seq-attn (biLSTM 256x4, d=0.3)	6.81%	29.51%	9h38
(3) seq2seq-attn (biLSTM 256x4 residual, d=0.3)	6.67%	28.72%	9h40
(4) seq2seq-attn (biLSTM 512x3) + multitask learning (en/de)	6.68%	29.28%	8h27
(5) seq2seq-attn (biLSTM 512x3) + multitask learning (ARPAbet/IPA)	6.50%	28.23%	6h30
(6) System combination (1) + (3)	5.91%	25.15%	(44h29)
(7) System combination (1) + (5)	**5.76%**	**24.88%**	(41h19)

In order to enable system combinations, we also let Sequitur G2P and seq2seq-attn generate n-best lists (n=10). We simply combine the n-best lists by adding normalized scores between 0.0 and 1.0 for entries that both models generated and append all other entries. After resorting, the output of the system combination is the highest scoring entry from the combined n-best list.

[5] `https://github.com/harvardnlp/seq2seq-attn` (last accessed: December 2021)

7.4 Results

In Table 7.1, we compare different G2P models on the Phonolex lexicon test set. Using the full lexicon as opposed to Phonolex core to train the models (the test set is the same in both cases) significantly reduces performances (3, 4). We attribute this to the problem that many entries of the full Phonolex set are already automatically generated in some way. We were able to spot several problematic non-core entries, mostly in entries marked as coming from automatic (and probably old) rule-based conversions. We thus focused on training with the core set after a few experiments. Seq2Seq models (5) and (6) do perform worse as a single model than the Sequitur G2P baselines (1) and (2). Training a multitask model on Phonolex and CMUDICT training data (7, 8) did not improve upon the residual network baseline (6), but is slightly better than a Seq2Seq model without residual learning (5). However, if we train a multitask model on our generated German IPA lexicon and Phonolex core (9), we can considerably lower error rates, as compared to all other Seq2Seq models. With this model, we can also match the WER score of the Sequitur G2P baseline model.

We observed that the multitask models generally need wider networks to account for the increased need of model capacity to model two G2P tasks. Residual learning can give a small performance improvement as in (5) vs. (6), and was also useful for (7) and (9), as tested on the dev set. Finally, if we combine the Sequitur G2P model (2) with our Seq2Seq models using n-best lists, we achieve the lowest error rates (10, 11).

To better understand the scores of the single models on the German Phonolex data, we annotated the Phonolex test set into the following irregular pronunciation cases: abbreviations (e.g. "GPA", "FFH", "MIDI-Dateien"), loan words and names with predominantly English pronunciation (e.g. "Toaster", "mousepad"), the same for French (e.g. "arrangierter", "Taille"), other names and named entities (e.g. "Dietmar", "Chemnitz") as well as hyphenated words (e.g. "Acht-Uhr-fünf-Flug"). The irregular pronunciation cases make up 24.09% of the data. We also discarded 73 (5.56%) of the test entries[6] from Phonolex, containing spelling errors of words, unknown words (e.g. "Beteginsdis") and transliterations of Bavarian dialects (e.g. "obgnomma", "ausaschauen" for standard German "abgenommen", "hinausschauen").

Table 7.2 shows PER and WER results separately for our word classes. As expected, there is a large increase in error rates for irregularly pronounced German words and other special forms across all models. Interestingly, Sequitur G2P struggles less than the Seq2Seq models with abbreviations, English and French loanwords and names[7]. Hyphenated words are considerably more problematic for Sequitur G2P than for Seq2Seq models. The multitask model trained on Sampa/IPA data mainly improves regular words compared to Sequitur G2P. Although still showing

[6]Discarded words are also discarded from the analysis in Table 7.1.

[7]The results for French loan words have to be taken with grain of salt though, as the sample size is very small (n=8).

slightly higher error rates on irregular cases than Sequitur G2P, it shows lower error rates for most classes of irregular words compared to the other Seq2Seq models.

In Table 7.3, we also compare similar models on English G2P conversion, as a control experiment. Similarly, Sequitur G2P (1) outperforms individual Seq2Seq models. Residual learning gives a small improvement (3) vs. (2) and the multi-task model trained on German and English data (4) degrades error rates very slightly, similarly to the German test scores in Table 7.1. The multitask model for English ARPAbet and IPA does slightly improvess the Seq2Seq scores (5), but fails to match the Sequitur G2P error rate on the CMUDICT test set (1). This can be attributed to the much smaller size of the auxiliary IPA lexicon, as it is 10 times smaller than in the German experiment and also smaller than CMUDICT itself. Again, if we use a simple combination of the model's n-best lists (6, 7), we observe lower error rates than from any individual model.

7.5 Conclusion

Our results still largely favor traditional joint sequence n-gram modelling (with Sequitur G2P) if individual models are compared, similarly to related work. Combining Seq2Seq G2P and n-gram modelling into an ensemble model improves upon the Sequitur G2P n-gram modelling baseline. Any further advances in either RNN architectures or the Seq2Seq architecture will likely also benefit Seq2Seq G2P. Simultaneous training on the German and English G2P task did no yield any benefits, but our multitask Sampa/IPA Seq2Seq model for German G2P performs on par with the Sequitur G2P baseline and shows significantly lower error rates than the other Seq2Seq models.

Our error analysis according to word classes shows that there is diversity in how the models learn to deal with regularly and irregularly pronounced German words, which are difficult to predict. Ultimately, this diversity allows us to combine different systems: combining the Sequitur G2P and Seq2Seq models without multitask learning (8.6% lower WER relative) and with multitask learning (15.8% lower WER relative) yields significant WER improvements for German G2P.

The results are similar for English G2P on CMUDICT, but since the auxiliary IPA lexicon is 10 times smaller than the German one, we can only observe a modest decrease in WER for multitask Seq2Seq models. We still think that multitask learning is a simple way to combine non-heterogeneous lexicons in Seq2Seq G2P models, without the need for an explicit translation between different phoneme sets and annotation styles.

Chapter 8

Automatic subtitling

Subtitles primarily display spoken content in videos and help to make the content accessible to people with hearing related problems, hearing aids and people who learn foreign languages. They can also help in situations where video content is consumed in a noisy environment. Translated subtitles can make videos accessible to people who do not speak the language used in the video.

However, video subtitling is a tedious manual task. Even trained transcribers need on average 13-18x more time relative to the speech time to subtitle videos without any automatic assistance (Roy and Roy, 2009). ASR can be used to transcribe speech automatically or to aid human transcribers. In fully automatic subtitling, further processing steps have to be considered as well, beyond simply running ASR decoding on the audio extracted from a video file. The output needs to be segmented into appropriate segments and interpunctuation must be added for better readability. Non-speech segments, such as silence, music and environmental sounds might also be present in the audio.

In this chapter and the following one, we propose practical systems for ASR applications. We introduce an automatic subtitling system for German with punctuation reconstruction and an appropriate segmentation strategy and demonstrate its capabilities on real world data. While our intended use case is the automatic subtitling of all video lectures published on the Lecture2Go e-learning platform[1] of Universität Hamburg, we expect that our open source subtitling solution will be a good starting point for general purpose subtitling as well.

8.1 German ASR

ASR models trained with open source software and freely available resources allow personal, academic and commercial use cases without licensing issues, lowering the barrier of entry. Having access to a locally running speech recognition software (or a private server instance) solves privacy issues of speech APIs from cloud providers. English speech recognition models for Kaldi are available as pretrained packages or freely available training recipes and these models are used in the wild for downstream NLP applications, e.g. (Oualil et al., 2017; Milde et al., 2016). We would

[1] `https://lecture2go.uni-hamburg.de/` (last accessed: April 2022)

like to establish the models presented in the following sections as go-to models for open source German speech recognition with Kaldi – with freely available training recipes, making it easily extensible, as well as offering pre-trained models. In the following, we discuss the freely available data resources for German and our recognition results.

8.1.1 Data Resources

Tuda-De. In (Radeck-Arneth et al., 2015), an open source corpus of German utterances was described and publicly released, with a focus on distant speech recognition. Sentences were sourced from different text genres: Wikipedia, parliament speeches and simple command and control prompts. Volunteers, mostly students, read the sentences into four different microphones, placed at a distance of one meter from the speaker. One of these microphones was a Microsoft Kinect. The corpus contains data from the beamformed signal of the Kinect, as well as mixed down single channel raw data from the microphone array (due to driver restrictions the raw multi channel data could not be recorded). Yamaha PSG-01S, a simple USB table microphone and a Samson C01U, a studio microphone, were also used to record audio simultaneously. A further simultaneous recording was made with a built-in laptop microphone (Realtek), at a different position in the room and next to a very noisy fan. For nearly every utterance the corpus contains five sound files, apart from a few where driver hiccups resulted in fewer recordings. Four of these streams are fairly clean and comprehensible, while the recordings from the Realtek microphone next to a noisy fan are very difficult to understand, even for humans. Female speakers make up about 30% of the data and most speakers are between 18 and 30 years old. We used version 2 of the corpus and also make use of its test and dev set in our evaluation. For the latest experiments in 2022, we used version 4.

SWC. Another resource for German ASR is automatically aligned found speech data from the German sub Spoken Wikipedia project (SWC) (Baumann et al., 2019). This corpus was created by aligning Wikipedia articles with publicly available recordings of them being read out. This is a very interesting resource, as new read speech data is consistently added to the project by volunteers and the training process can be extended form time to time with new data. The data is diverse and covers a lot of vocabulary due to the encyclopedic nature of the texts.

SWC does not have fixed utterance mapping, instead we use the SWC snippet extractor to generate training utterances along Voice Activity Detection (VAD) boundaries. In *conservative* pruning, we discard the following utterances: shorter than 0.6 seconds, more than 20% of unaligned data, more than two consecutive unaligned words, and an unaligned word at the beginning or end or pauses longer than 1.5 seconds.

In a second setting that we call *minimal* pruning, we extract utterances defined by VAD boundaries for which at least 65% of the words are aligned, with no other restrictions. The German Spoken Wikipedia corpus used in our experiments has 363

speakers who committed 349h of audio, of which 249h are aligned. Conservative pruning yields 141h of audio for ASR training and minimal pruning yields 285h of audio recordings.

M-AILABS. This dataset contains 233.70 hours of aligned read speech from German audiobooks of the Librivox project[2], with freely available texts from Project Gutenberg[3]. The dataset is distributed under a modified 2-clause BSD license for data. In this corpus, there are 117,856 unique German sentences and 118,521 utterances.

Common Voice. Ardila et al. (2020) created a web-based user platform to easily donate speech recordings in many languages to the Common Voice project. With the web app, users can read out sentences that are then made publicly available (CC-0 license). Most languages have sentences from Wikipedia as well as user-submitted sentences available. We used version 3 of the German Common Voice dataset, downloaded in October 2019 with 322.4 hours in total. In this version there are 19,489 unique German sentences and 281,184 utterances. In a series of new experiments, we updated Common Voice to version 8. In this version there are 451,731 unique German sentences and 746,482 utterances and total of 1050 hours of training data.

8.1.2 Lexicon

MARY-TTS (Schröder and Trouvain, 2003) is an open source Text-To-Speech (TTS) system. It also contains a manually created phoneme dictionary resource for German, containing 26,231 words and their phoneme transcriptions in a dialect of extended SAM-PA BAS (Bavarian Archive for Speech Signals, n.d.). We use Sequitur (Bisani and Ney, 2008) to train a grapheme-to-phoneme (G2P) model, to be able to add automatically generated entries for out-of-vocabulary (OOV) words to the lexicon as needed.

For the Tuda-De corpus the final lexicon size is 28,131 words; this includes all words from the MARY lexicon and automatically generated entries for all OOV words in the train set. When we combine the Tuda-De transcriptions with the SWC transcriptions, more OOV lexicon entries need to be automatically generated and the final lexicon size grows to 126,794 words using the conservatively pruned SWC data, or 182,784 words respectively with minimally pruned SWC data. To measure the effects of an even larger vocabulary size, we also computed the 300,000 most frequent words in the German Wikipedia (April 2018) and generated additional phonetic entries. We merged the vocabulary with the previous lexicon and obtained a larger lexicon containing 350,029 words. In later experiments, we expanded the vocabulary to 683,044 and 721,625 words in the same way.

Subsequently, in a later version of the LM and lexicon resources, we expanded the lexicon manually by 14,268 entries. We wrote the Software speech-lex-edit[4] to

[2] `https://librivox.org/` (last accessed: December 2021)
[3] `http://www.gutenberg.org/` (last accessed: December 2021)
[4] `https://github.com/uhh-lt/speech-lex-edit` (last accessed: December 2021)

facilitate an active learning approach, where Sequitur G2P is used to propose candidate pronunciation entries. The Sequitur G2P model was periodically retrained. Figure 8.1 shows a screenshot of the software. In the default setting, five pronunciation candidates with their confidence values are displayed to the user and can be listened to through a synthesized sound file produced by MARY TTS.

As candidates for the manual lexicon expansion, we choose words from the language model training texts that were both frequent and had low G2P likelihood / confidence scores and sorted the candidate vocabulary by $log(freq) \cdot G2Pconf$. Some candidates were also specified manually and some were hand picked from a list of anglicisms, as these are typically difficult for a German G2P to predict automatically, see also Chapter 7, Table 7.2 and (Milde et al., 2017).

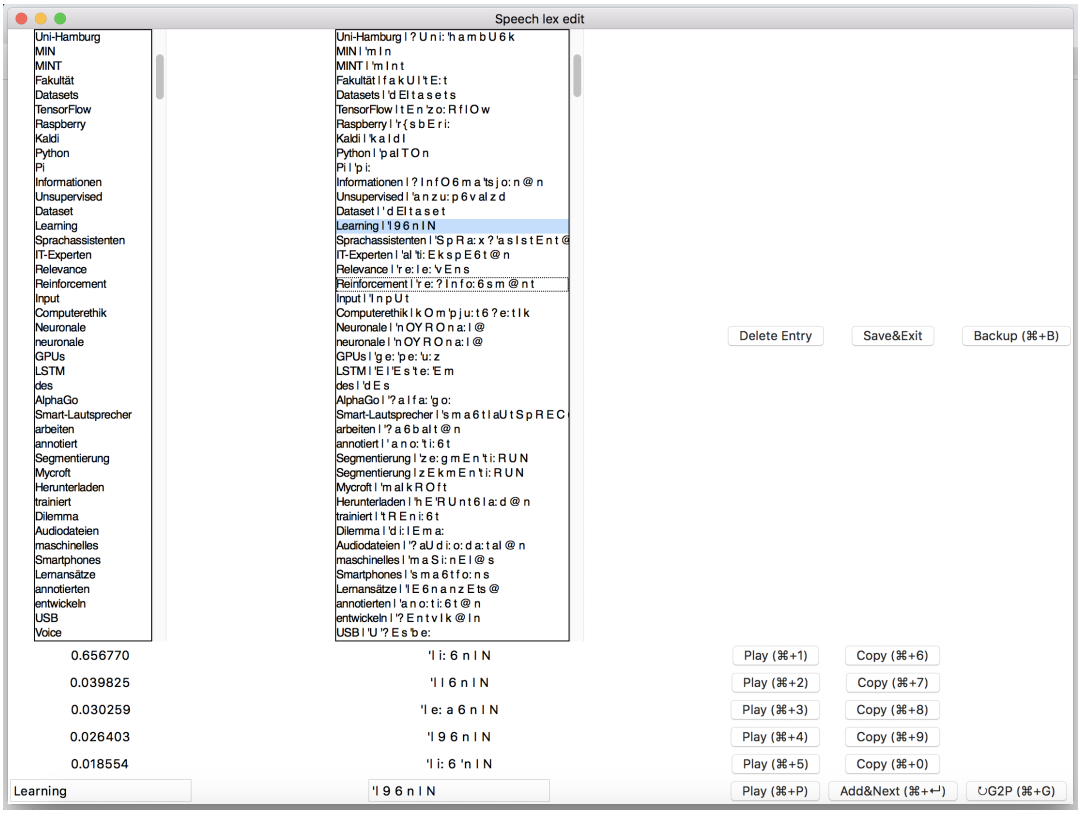

FIGURE 8.1: Screenshot of the lexicon editor using a G2P model for pronunciation suggestions.

8.1.3 Language Model Resources

We used the same text sources as in (Radeck-Arneth et al., 2015) and trained similar baseline language models. In particular, we trained 3-gram and 4-gram language models with Kneser-Ney smoothing (Kneser and Ney, 1995) and different vocabulary sizes on approximately 8 million German sentences. The sentences are selected from similar sources as the spoken data (Wikipedia, Parliament and some crawled sentences). Also, they are already filtered, so that sentences from the development

and test sets of the Tuda-De corpus are not included in LM training texts. All sentences were normalized using the frontend of the MARY TTS software (Schröder and Trouvain, 2003), similarly to the normalization process of the SWC corpus. We also use the newly released Kaldi-RNNLM (Xu et al., 2018b) to train a recurrent neural network based LM on the same text sources. We use the same parameters as in the Switchboard LSTM 1e example: two stacked LSTM layers with a cell width of 1024. We also tested four stacked LSTM layers.

In a second iteration of the LM and lexicon resources, we expanded the LM texts to 102.5 million sentences. We updated the German Wikipedia texts (May 2020 dump) and added crawled Tagesschau news[5] and subtitles crawled from Germany's public broadcasters online portals (Mediathek). This also adds new and recent vocabulary. All scripts to replicate the data gathering have been published as open source[6]. Instead of relying on MARY TTS text normalization, we developed our own regex based text normalization system that can handle text expansion of numbers, abbreviations, years, dates and time. All 102.5 million sentences were normalized.

8.1.4 Experiments and Evaluation

TABLE 8.1: WER results on the Tuda-De dev and test sets. The scores are for decoding combined data from Kinect (Beam and RAW), Samson and Yamaha microphones.

Model	Dataset	Vocabulary	LM	WER	
				dev	test
GMM-HMM	Tuda-De	28,131	3-gram KN	45.31	45.55
	Tuda-De	126,794	″	37.47	38.34
	Tuda-De + SWC (cons. pruned)	″	″	29.97	31.06
	Tuda-De + SWC (min. pruned)	″	″	29.79	30.99
	Tuda-De + SWC (min. pruned)	182,784	″	26.92	28.25
	Tuda-De + SWC (min. pruned)	350,029	4-gram KN	24.91	25.77
	+ M-AILABS + Common Voice V3	683,044	4-gram KN	23.06	24.38
	+ M-AILABS + Common Voice V8	721,625	4-gram KN	22.89	24.46
TDNN-HMM	Tuda-De	28,131	3-gram KN	35.53	36.32
	Tuda-De	126,794	″	28.08	28.96
	Tuda-De + SWC (cons. pruned)	″	″	20.91	22.22
	Tuda-De + SWC (min. pruned)	″	″	20.30	21.43
	Tuda-De + SWC (min. pruned)	182,784	″	18.39	19.60
	Tuda-De + SWC (min. pruned)	350,029	4-gram KN	15.32	16.49
	Tuda-De + SWC (min. pruned)	″	+ 2-layer LSTM LM	13.14	14.38
	+ M-AILABS + Common Voice V3	683,044	4-gram KN (102.5M sent.)	12.26	13.79
	+ M-AILABS + Common Voice V3	″	+ const ARPA rescoring	10.47	11.85
	+ M-AILABS + Common Voice V3	″	+ reformat numbers	8.61	9.85
	+ M-AILABS + Common Voice V8	721,625	4-gram KN (102.5M sent.)	10.94	12.09
	+ M-AILABS + Common Voice V8	″	+ const ARPA rescoring	9.25	10.17
	+ M-AILABS + Common Voice V8	″	+ reformat numbers	7.51	8.53
	+ M-AILABS + Common Voice V8	″	+ 2-layer LSTM LM	6.51	7.43
	+ M-AILABS + Common Voice V8	″	or 4-layer LSTM LM	**6.46**	**7.37**

[5] A German television and online news service from ARD, one of Germany's public TV broadcasters.
[6] `https://github.com/bmilde/german-asr-lm-tools` (last accessed: December 2021)

For our experiments, we use the kaldi-tuda-de[7] training scripts that we released as open source. We use Gaussian Mixture Model (GMM) - Hidden Markov Models (HMM) and Time-Delayed Neural Networks (TDNNs) (Waibel et al., 1990; Peddinti et al., 2015) as acoustic models following the chain-recipe (s5_r2) of the TED-LIUM corpus (Rousseau et al., 2014) example in Kaldi. The TDNNs have a width of 1024 neurons for training sizes of up to 412h and a width of 2048 neurons for all models trained with the additional Common Voice data. For GMM-HMM models, we adapted the Kaldi egs[8] for the Switchboard corpus (swbd s5c, model tri4). As input to the TDNN we also use online i-vectors (helping with speaker adaptation, c.f. (Saon et al., 2013; Senior and Lopez-Moreno, 2014; Miao et al., 2015)).

As the TDNN-HMM chain models are sequence-discriminatively trained on the utterances, they are more prone to overfitting and do not cope well with incorrect transcriptions (Povey et al., 2016). Since the SWC transcriptions are aligned from found data, we expect that some of the transcriptions could be problematic, particularly when we apply only minimal pruning to SWC. We follow the recipe used in the Kaldi TED-LIUM TDNN example and clean the training data by decoding it with an intermediate model using a reference-biased LM. Then we remove utterances which do not match their supposed transcriptions or resegment the utterance boundaries where applicable. While analyzing the cleaned utterances, we also noted that some of the Tuda-De training utterances are wrongly annotated, mostly because of hiccups in the recording software (Schnelle-Walka et al., 2014) resulting in (completely) wrongly assigned utterance transcriptions. The cleanup removes about 1.6% of the Tuda-De data and 6.9% of the combined Tuda-De and conservatively pruned SWC data (268.5h $\rightarrow$ 250h). With minimally pruned SWC data, 8.8% of the combined training data is removed.

We use the dev and test set from the Tuda-De corpus to measure word error rates (WER). The experiments in (Radeck-Arneth et al., 2015) defined a closed vocabulary task with no OOV words, as OOV words in test and dev were also added to the lexicon. This makes WER rates somewhat lower in comparison, but a bit unrealistic. In Table 8.1 we show results for a more realistic open domain setting, where the dev and test vocabulary is not known a priori. Using only a 28,131 word vocabulary yields very high WER for GMM-HMM and TDNN-HMM models alike, because of a high OOV rate. Extending the vocabulary to 126,794 words reduces both GMM-HMM and TDNN-HMM WER by about 20% relative. Adding SWC data to the Tuda-De utterances improves these TDNN-HMM results significantly, even when we use the same vocabulary size. Using a minimal pruning strategy with the SWC data and subsequently relying more on Kaldi's cleaning scripts gives slightly better results: 26% relative reduction vs. 23.3% relative reduction. However, we achieve our best WERs when we use a significantly larger vocabulary and a better LM. Our test score with an open domain vocabulary of 350,029 words is 16.49% WER and can be further

[7] `https://github.com/uhh-lt/kaldi-tuda-de` (last accessed: April 2022)

[8] "egs" is the name of a folder in Kaldi's repository containing examples of training recipes for many popular speech corpora.

improved by using lattice rescoring with an LSTM LM to 14.38% WER. Adding 322.4 hours from Common Voice V3 and expanding the vocabulary further to 683,044 as well as using the larger 102.5 million sentences LM with ARPA rescoring yields a test WER of 11.85%. The sentences we normalized for the LM split number literals into several tokens, to avoid vocabulary explosion (e.g. "drei und sechzig"). The correct orthographic spelling for numbers in German is single words. If we normalize numbers and years of the ASR output ("neunzehn hundert neun und neuzig" → "neunzehnhundertneunundneuzig"), test WER drops to 9.85%, since the dev and test sets use the correct orthographic spelling for numbers. Adding 700 additional hours of training data by updating the Common Voice dataset from version 3 to version 8 yields a WER of 8.5%. Additional rescoring with a LSTM LM further improves this to a WER of 7.43% with a 2-layer LSTM and 7.37% with a 4-layer LSTM. We use the pruned lattice rescoring described in (Xu et al., 2018a). This is a significant improvement over the 20.5 WER% (without OOVs, restricted vocabulary) reported in (Radeck-Arneth et al., 2015), even though we evaluate in an open vocabulary setting. It is also a large improvement over the previously published result of 14.38% WER in (Milde and Köhn, 2018).

Comparison to other systems

TABLE 8.2: WER comparison to other systems on the Tuda-De test set. All systems use additional training data, with varying amounts.

System	Model	Data	Year	test WER
Based on Deep Speech (Agarwal and Zesch, 2019)	E2E/CTC	302h	2019	15.1
IMS-Speech (Denisov and Vu, 2019)	E2E/Encoder-decoder	3797h	2019	12.0
CTC-Segmentation (Kürzinger et al., 2020)	E2E/CTC+Attention	1700h	2020	12.8
Scribosermo (Bermuth et al., 2021)	E2E/QuartzNet15x5DE	1000h (en) 836h (de)	2021	10.2
Conformer (Wirth and Peinl, 2022)	E2E/Conformer CTC	4520h	2022	7.8
Conformer (Wirth and Peinl, 2022)	E2E/Conformer T	4520h	2022	**5.8**
uhh-lt/kaldi-tuda-de (Milde and Köhn, 2018)	TDNN-HMM hybrid, FST	412h	2018	14.4
uhh-lt/kaldi-tuda-de (this chapter)	TDNN-HMM hybrid, FST	1720h	2022	**7.4**

The test set created in (Radeck-Arneth et al., 2015) has been adopted by the community for testing German ASR performance[9]. In Table 8.2 we compare how the WER results on the Tuda-De dataset compared to other systems with published results. Only the ASR system of this chapter uses a TDNN-HMM. All other systems

[9]An uptodate leaderboard can also be found at: `https://paperswithcode.com/sota/speech-recognition-on-tuda` (last accessed: December 2021)

use various forms of end-to-end (E2E) ASR, i.e. ASR systems that fully neural and do not use HMMs. However, the best test WER result of this chapter is better than five other end-to-end systems. A recent result from Bermuth et al. (2021) shows that pre-trained weights from an English E2E ASR model can be quite effectively used with transfer learning to train a German E2E ASR system. Until very recently, this was the best published E2E WER result on the Tuda-De test. The TDNN-HMM system presented in this chapter is about 27% better relative. Concurrently to the submission of this thesis, Wirth and Peinl (2022) presented a large-scale study on recent E2E modelling techniques for German ASR, using a lot more training data (4520h). Systems based on the conformer architecture (Gulati et al., 2020) gave the best results. The TDNN-HMM system in this chapter is still better than a Conformer model trained with CTC, although the Conformer model uses much more training data. However, a Conformer Transducer trained on 2.6x as much training data gives better results than the TDNN-HMM system (5.8% WER vs. 7.4% WER).

Conversational Speech

TABLE 8.3: WER results on the Verbmobil (VM1) dev and test data.

Model	**Vocabulary**	**Training dataset**	**WER**	
			dev	**test**
GMM-HMM	General purp. (~350k)	Tuda-De + SWC	46.42	50.56
TDNN-HMM	General purp. (~350k)	Tuda-De + SWC	33.69	38.23
TDNN-HMM	General purp. (~683k)	↪ + M-AI + CV3	24.95	27.79
TDNN-HMM	General purp. (~722k)	↪ + M-AI + CV8 +resc.	**19.35**	**22.34**
GMM-HMM	Domain specific (~7k)	Tuda-De + SWC	27.18	29.12
TDNN-HMM	Domain specific (~7k)	Tuda-De + SWC	**18.17**	**20.04**

In the Verbmobil project (1993-2000), the goal was to establish whether translation of spontaneous speech into other languages is possible (Wahlster, 2000). Conversational speech data was recorded in German, English and Japanese, in the limited domain of scheduling appointments. We used the dev and test data of the first revision of the German subset of the Verbmobil corpus (VM1). Since our acoustic models are trained exclusively on read speech, it provides a good test set showing how well our models cope with a more challenging conversational and spontaneous speaking style.

In Table 8.3, we show results for decoding VM1 utterances with our acoustic models. We decode with two different vocabularies and FSTs, a general purpose vocabulary (as also used for the results in Table 8.1) and a domain-specific vocabulary, using the lexicon words of the VM1 corpus (6851 words). For the latter we recomputed our LM with the reduced vocabulary. We do not use the manual lexicon entries of the VM1 corpus and instead use the same lexicon we use in the general

purpose case, reducing it and generating automatic OOV phoneme lexicon entries as needed.

The domain specific WER score with limited vocabulary is usually found in the literature for the Verbmobil corpus. A newer reference score for a DNN-HMM trained with Kaldi is 12.5% WER in (Gaida et al., 2014). Our score of 20.04% WER is probably due to not using the optimized and manually generated lexicon as well as due to a mismatch in the training data for the acoustic model (read speech vs. conversational speech). The model in (Gaida et al., 2014) is exclusively trained on in-domain audio data, while we excluded any proprietary VM1 speech training data and only used our freely available open source speech recordings. The WER gap to the domain specific result can be narrowed by training the acoustic model on additional data (M-AI + Common Voice V8), normalizing numbers in the output and rescoring with a better general purpose LM. Our best open vocabulary result for the VM1 test set is a WER of 22.34%.

8.2 Subtitling Pipeline

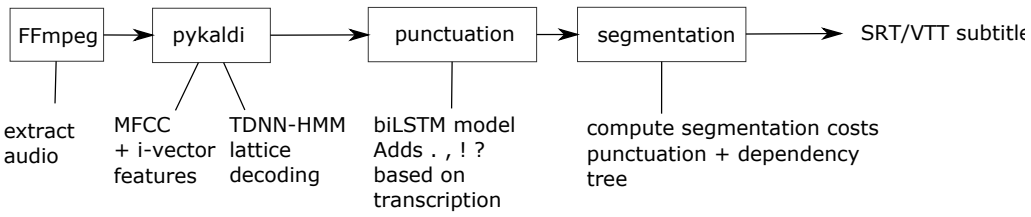

FIGURE 8.2: Subtitling pipeline and processing steps.

Figure 8.2 shows our subtitling pipeline. We first run FFmpeg/pyFFmpeg to extract 16 kHz pcm mono audio from any supported FFmpeg media file (all popular video formats are supported). We then use feature derivation and decoding with PyKaldi (Can et al., 2018), also mapping each word to the alignment information from the decoder. We then apply an LSTM-model that adds punctuation. Finally, we use a segment scoring function and apply beam search to search through different segmentation alternatives. Finally, based on the alignment information and the segment information, we write the complete subtitle file in either a SRT or WebVTT format (Pfeiffer, 2019).

8.2.1 Punctuation

Subtitles frequently feature text with punctuation, while ASR transcriptions are typically without any punctuation. We trained punctuation reconstruction models to reconstruct the punctuation on a textual basis from the ASR output. After transcriptions have been decoded, punctuation can then be reconstructed in a post processing step.

The punctuation models are trained on language model texts where the punctuation is removed from the training texts and used as labels in a sequence labeling task. The same language modelling resources as in Section 8.1.3 are used, where numbers, abbreviations etc. are normalized to better match ASR output.

For punctuation model training we used punctuator2 (Tilk and Alumäe, 2016), based on gated recurrent units (GRUs) (Cho et al., 2014b). In (Milde et al., 2021b), we trained a model to predict four different kinds of punctuations (.,?!), with a per token error rate of 2%. F_1 score for periods was 90.7%, for commas 89.4%, for question marks 74.2% and for exclamation marks 33.1%. The latter can be subjective, especially when predicted entirely from text, so recognition results are unsurprisingly low for exclamation marks. Commas, periods and question marks on the other hand where predicted quite reliably. This forms a basis for subtitle segmentation discussed in the next section, where punctuation is an important clue.

8.2.2 Subtitle Segmentation

We aim to segment the punctuated transcript to improve readability. The segmentation is based on a beam search with manually tuned weights. We want to balance a target average segment length with splitting at sentence punctuation (,.?!); all other potential splitting positions in a sentence are evaluated based on the shortest connecting path in a parse tree of that sentence. The intent is that we want to avoid splitting at words that are linguistically closely connected words/tokens, i.e. they should not be separated across two separate screens (or lines) if possible, so that the reading flow is not interrupted. Our beam search maximizes a performance function that sums up all individual segmentation decisions. In particular, we defined the performance f_s for sentence s that outputs a reward for segmenting at the position between the tokens t_1 and t_{i+1} as :

$$f_s(t_i, t_{i+1}) = r_l + \begin{cases} 0.9 \cdot \text{len}(s) & \text{if } t_i \in \{\text{'.'}, \text{'?'}, \text{'!'}\} \\ 0.7 \cdot \text{len}(s) & \text{if } t_i = \text{','} \\ \text{scp(parsetree}(s), t_i, t_i + 1) & \text{otherwise} \end{cases}$$

where scp is the shortest connecting path in the syntax tree (as parsed with spacy) and r_l the length reward for being as close as possible to the target token length for a particular segment.

$$r_l = 2.3 * (ttl - |ttl - j|)$$

where ttl is the target token length and j the resulting length of the segment, if a split were to be placed between t_i and t_{i+1}. In our experiments we set $ttl = 10$. We then maximize $\sum f_s(t_i, t_{i+1})$ over all chosen segmentation paths, expanding the beam up to a maximum number of lookahead tokens, evaluating each forward position from the current position i up to $i + 40$.

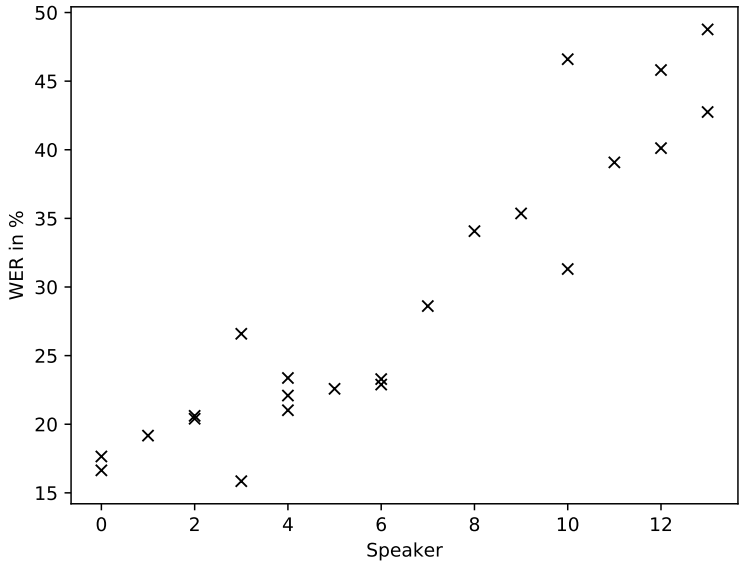

FIGURE 8.3: WER for 17 German lectures, with topics ranging from education science to computer science. The 14 anonymised speakers are sorted according to their average WER.

Other constraints could easily be integrated; e.g. a performance function that depends on the length in characters rather than tokens, or adding hard constrains on the maximum character length for a subtitle segment. Another improvement could be to include pausing information into the performance measure, or to differentiate between line and screen breaks.

8.2.3 Evaluation

Figure 8.3 shows WERs evaluated on 17 Universität Hamburg (UHH) lectures from 2009 to 2014 that were manually transcribed, totalling 12h of video recordings with subtitles. The average WER over all speakers and lectures on this test set is 26.3%. Many of the recorded lectures have challenging acoustics, as they were recorded in addition to students attending the lecture hall. We also tested Subtitle2Go on one newer video, a public speech by Dr. Frank-Walter Steinmeier from 2019, which yields a similar WER of 25.9%.

8.2.4 Decoding Speed

Decoding speed depends on many factors and for some there is a trade-off between speed and accuracy. In the following, we report results for tuning the beam size of the ASR decoder with the default model size of kaldi-tuda-de (3.5GB), while using the default options of Subtitle2Go otherwise. In Figure 8.4, we plot the complete computation time needed for a particular video length, including startup and loading times. The run times were measured on an Intel Xeon E5-2620 v4 CPU (2.10GHz)

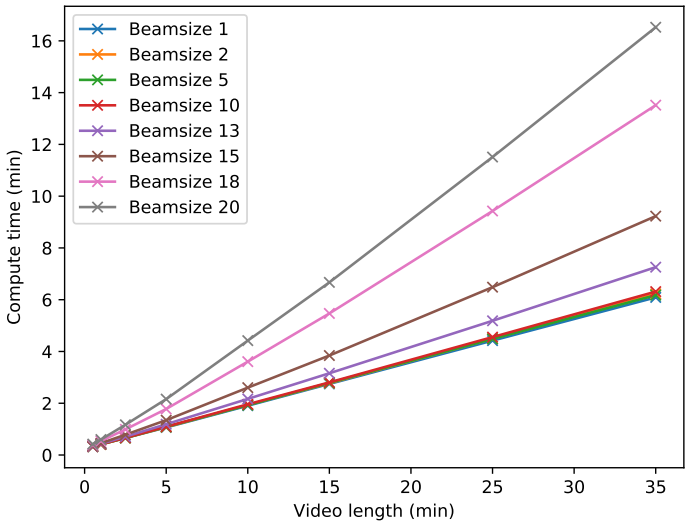

FIGURE 8.4: Computation time needed for all processing steps for different video and beam sizes. Timings were measured on one core of an Intel server CPU (Xeon E5-2620 v4).

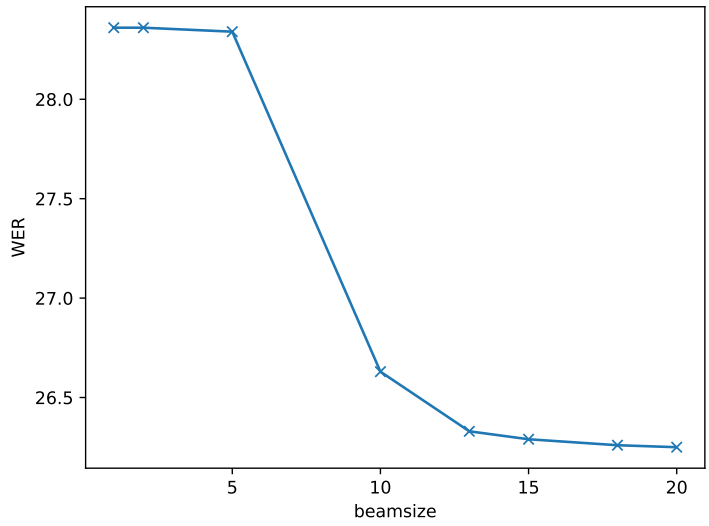

FIGURE 8.5: WER on the Lecture2Go test set, depending on the beamsize.

using one core, with Kaldi linked against Intel MKL. As can be seen in the figure, computation time grows linearly with video duration. In Figure 8.5, we show WER on the Lecture2Go test set with the same beam sizes. Since there is little WER improvement for beam sizes larger than 13, but computation time increases considerably, we settle on 13 as the default. Using this setting, a 35 minute video took 7m16s of computation time for all processing steps. This yields a real-time factor of 0.2 and a full 90-minute lecture could be subtitled in 20 minutes (i.e., before the next lecture

period starts). While there is no parallelization beyond the vectorization of the Intel MKL BLAS operations, multiple videos can be decoded at the same time with multiple cores.

8.2.5 Online Decoding

We used the German model for online decoding as well, enabling live transcriptions. We created kaldi-model-server[10] as an easy to use on-the-fly decoding server that support direct microphone access as well as speech input from byte streams over network. Kaldi-model-server uses PyKaldi (Can et al., 2018) with a custom nnet3 online decoder and can also be used with several microphones. When used with multi channel audio input, it decodes the most active audio channel.

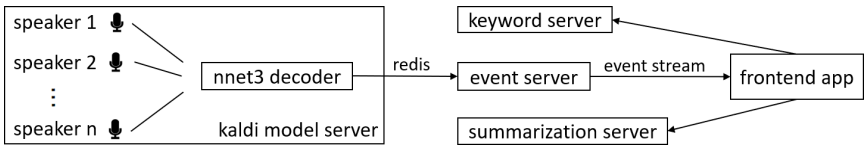

FIGURE 8.6: Architecture of MoM bot. The backend consists of a model server and various microservices, while the frontend is a VUE browser app.

In (Milde et al., 2021a), we presented meeting bot, a web based software that uses kaldi-model-server. Figure 8.6 shows a diagram of the architecture, organized as mircoservices. In our hardware setup, each speaker is given a (wireless) clip-on microphone, which also identifies the speaker. Multi channel data is directly recorded by kaldi-model-server and the most active audio channel is decoded. For German, we use the model presented in this chapter. For English, we trained a similar TDNN-HMM model on TED-LIUM 3 as in Chapter 5.

Using the online end pointing module in Kaldi, utterances are segmented on the fly with a rule-based system that takes speech pauses into account. At a hypothesized end point, we compute word confidences using Minimum Bayes Risk (MBR) decoding (Xu et al., 2011) given the decoding lattice of the current utterance. Kaldi model server then broadcasts partial and complete utterances with speaker information to the event server using redis[11]. The event server directly communicates with our VUE browser app and the current hypothesis is constantly updated through server side events that are sent to the browser app that displays the current hypothesis. The browser app uses a keyword microservice to extract relevant keywords on the fly. Once the meeting is finished, users can generate a PDF with a short summary and full transcription of the meeting, where the browser app interacts with the summarization server. For text summarization, we use TextRank. The resulting PDF is generated directly in the browser with javascript.

[10] `https://github.com/uhh-lt/kaldi-model-server` (last accessed: December 2021)

[11] Redis can be used as a message broker, with clients subscribing to channels and listing for events. See `https://redis.io` (last accessed: December 2021)

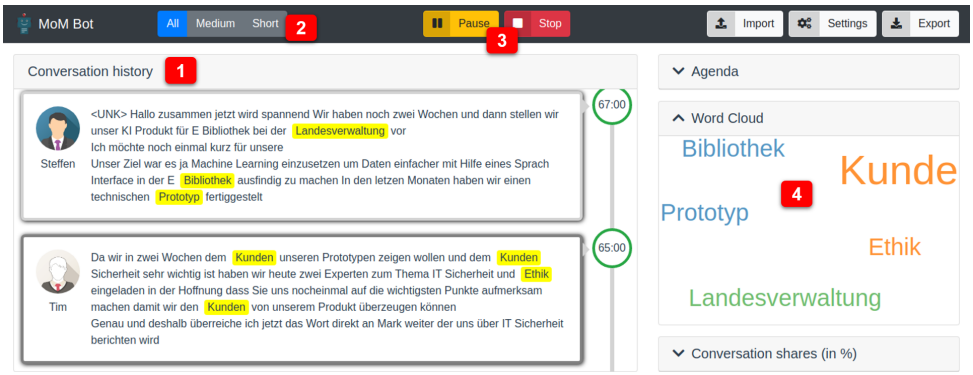

FIGURE 8.7: Example screenshot of the meeting bot system. Taken from (Milde et al., 2021a).

Figure 8.7 shows an example screenshot of our system. A timeline with separated boxes displays the transcribed text and conversation history in real time (1). We are showing the current hypothesis of the online model here as well. New boxes appear for each speaker change. Keywords are highlighted after each completed utterance. The text shown in the timeline can be shown in full, or in a reduced form (2), then only displaying keywords and some words of context around them. The system can be paused, resumed or stopped at any time (3). In (4), we show a word cloud generated from discovered keywords. Similar words are clustered and displayed in the same color. Newer keywords are weighted higher than old ones, so that the word cloud can gradually visualize changes in topics. The transcript can then also be edited in an editor that shows word and sentence level confidences of the recorded utterances, before the user can download it as a PDF. The final PDF contains the transcript of the meeting as well as a summarization.

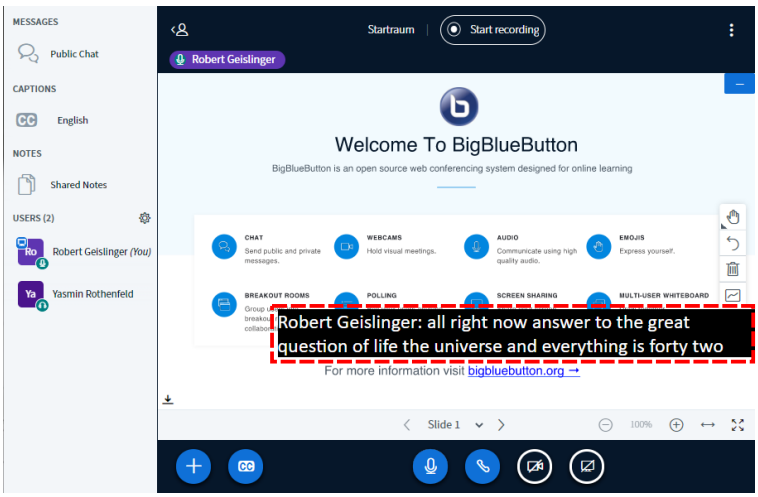

FIGURE 8.8: Example screenshot of video conferencing software BigBlueButton with live subtitling. Taken from Geislinger et al. (2021).

In (Geislinger et al., 2021), kaldi-model-server and the same German and English ASR models were used for creating a live subtitling plugin[12] for the popular open source video conferencing software BigBlueButton[13]. The screenshot in Figure 8.8 shows what BBB and the automatic subtitling looks like to the user. This is in contrast to Meetingbot, where all participants sit in the same room and speech is generally recorded separately. Thus this plugin decodes each speakers audio stream separately and in parallel, so that there is need for speaker diarization. Overlapping talk can also be handled seamlessly.

8.2.6 Conclusions and Outlook

We have introduced a freely available ASR model for German which improves the previously best one by a large margin, both due to improvements in algorithms and a significant increase of freely available data. Our models can be run locally and user-recorded speech data for ASR application does not have to be transferred to a 3rd party cloud provider for speech decoding, where privacy concerns will arise.

Our evaluation shows that the model performs well on new speakers and different microphones with around 7.4% WER for rescored TDNN-HMM models on the Tuda-De test set. This WER result is currently significantly better than other published results from various end-to-end systems on the same test set. The size of the general purpose vocabulary has a large effect on WERs - a large part of the remaining recognition errors are due to vocabulary problems and the underlying language model. We expect a subword unit or decompounding approach to work better than a fixed word approach for German read speech (Smit et al., 2017). A remaining challenge is conversational speech, as the training data is by and large read speech. As more and more articles are spoken and recorded by volunteers for the Spoken Wikipedia project, we also expect benefits for our acoustic models through the use of the additional data. Our final model can deal with different microphone and unknown speakers in an open vocabulary setting. The model can be used to decode very long audio files as well and we make use of this in automatic subtitling. Punctuation reconstruction is possible with reasonable accuracy on text data alone.

We proposed a segmentation algorithm for splitting the subtitle texts into reasonable chunks for display purposes, with a beam search to find a good global solution maximizing splitting rewards. The rewards are calculated based on the restored punctuation and shortest connecting paths of adjacent words in parse trees to avoid splitting at very closely connected adjacent words. The proposed solution is flexible and other constraints, such as a maximum character length of a displayed subtitle chunk, can easily be integrated into the segmentation search.

The model can also be used in online decoding to generate live subtitles and we demonstrated this capability in two applications: Meetingbot, a software for live

[12] Available at `https://github.com/uhh-lt/bbb-live-subtitles` (last accessed: April 2022)

[13] `https://bigbluebutton.org/` (last accessed: April 2022)

transcriptions as well as summarization of meetings and a plugin for the video conferencing software Big Blue Button.

A set of older and already transcribed lectures from 2009 to 2014 was also used for evaluation, yielding an average WER of 26.3%. Some of the recordings are also of sub-optimal quality and newer uploads tend to have better recording quality. Generally, there is a domain mismatch for lecture recordings, as the acoustic model is trained on read speech data.

As of October 2021, Lecture2Go[14], the university-wide media platform for lectures and presentations at Universität Hamburg included the open source German subtitling solution[15] presented in this chapter into their production system. Any lecturer uploading lecture videos to this platform can make use of the automatic subtitling feature and ASR model (currently only for German, but an English model will soon be available as well). The subtitles can then be further annotated and manually corrected on the Lecture2Go upload platform. Over time, these corrected transcriptions could also be used to retrain the acoustic and language models to reduce WER for lecture videos.

[14] `https://lecture2go.uni-hamburg.de/` (last accessed: April 2022)

[15] Available at `https://github.com/uhh-lt/subtitle2go` (last accessed: April 2022)

Chapter 9

A Document Retrieval System for Speech Streams

Recent advancements in Automated Speech Recognition (ASR) and Natural Language Understanding (NLU) have proliferated the use of personal assistants like Siri[1] or Google Now[2], with which people interact naturally with their voice. However, the activation of such systems has to be specifically triggered and they are targeted to an (ever–growing) set of anticipated question types and commands.

When taking part in a conversation or listening to a lecture, people may want to look up helpful information or check facts. Manually checking this information or interacting with a personal assistant would hamper the flow of the discussion, respectively distract from the lecture. In the following, we present *Ambient Search*, a system that ambiently researches relevant information in the form of proposing relevant documents to users in conversations or users who passively listen to spoken language. In contrast to other personal assistants, our system is not specifically triggered. Instead, it unobtrusively listens to speech streams in the background and implicitly queries an index of documents. We see the following utility in our approach: The assistant stays in the background and does not disturb the user. Access to the displayed snippets is on demand and the user can access the information in context without the need to formulate a specific query.

Of course, these advantages are fundamentally based on how well the system is able to retrieve relevant documents, as the system's utility diminishes when proposing a lot of irrelevant documents. In this chapter, we also evaluate how well the system is able to retrieve relevant Wikipedia articles in spite of average speech recognition word error rates (WER) of 15.6% on TED talks and show that it finds more relevant articles compared to another implicit information retrieval system on speech streams.

[1] `http://www.apple.com/ios/siri/` (last accessed: December 2021)
[2] `https://www.google.com/landing/now/` (last accessed: December 2021)

9.1 Related Work

The Remembrance Agent (Rhodes and Starner, 1996) is an early prototype of a continuously running automated information retrieval system, which was implemented as a plugin for the text editor Emacs[3]. Given a collection of the user's accumulated email, Usenet news articles, papers, saved HTML files and other text notes, it attempts to find those documents that are most relevant to the user's current context. Rhodes and Maes (2000) defined the term *just-in-time information retrieval agents* as "a class of software agents that proactively present information based on a person's context in an easily accessible and non-instrusive manner". A person's context can be a very broad term in this regard. E.g. Jimminy (Rhodes, 1997) represents a multimodal extension of the Remembrance Agent. Dumais et al. (2004) introduced an implicit query (IQ) system, which serves as a background system when writing emails. It also uses Term Frequency – Inverse Document Frequency (TF-IDF) scores for keyword extraction, like the Remembrance Agent.

Other systems cover a user's explicit information needs, e.g. Ada and Grace are virtual museum guides (Traum et al., 2012) that can suggest exhibitions and answer questions. The Mindmeld[4] commercial assistant can listen to conversations between people to improve the retrieval results by fusing the user's location information with from keywords extracted from transcripts. The FAME interactive space (Metze et al., 2005) is a multi-modal system that interacts with humans in multiple communication modes in order to suggest additional information to them. Although FAME supports speech recognition and voice commands, it only listens to conversations for a longer period of time when it guesses a conversation's topic and can suggest documents with explicit commands. Another class of systems try to record the content of a conversation or speech stream and visualize it using a network of terms: For example, SemanticTalk (Biemann et al., 2004) iteratively builds a structure similar to a mindmap and can also visualize conversation trails with respect to background documents.

The most similar approach to *Ambient Search* was presented by Habibi and Popescu-Belis (2015), extending earlier work of an Automatic Content Linking Device (ACLD) (Popescu-Belis et al., 2000). It uses an LDA topic model for the extraction of keywords and the formulation of topically separated search queries. The extracted set of keywords as well as the ultimately returned set of document recommendations fulfill a trade-off between topical coverage and topical diversity.

Since this system can be considered a state-of-the-art system of implicit information retrieval in speech streams, we compare our approach to this one in the evaluation in Section 9.3, alongside a TF-IDF-based baseline. A major difference to Habibi and Popescu-Belis, 2015, which operates on complete speech transcriptions only, is

[3] `https://www.gnu.org/software/emacs/` (last accessed: December 2021)
[4] `http://www.mindmeld.com` (last accessed: December 2021)

that our implementation is also able to retrieve relevant documents in real time, i.e. by processing live speech input.

9.2 Our Approach to Ambient Search

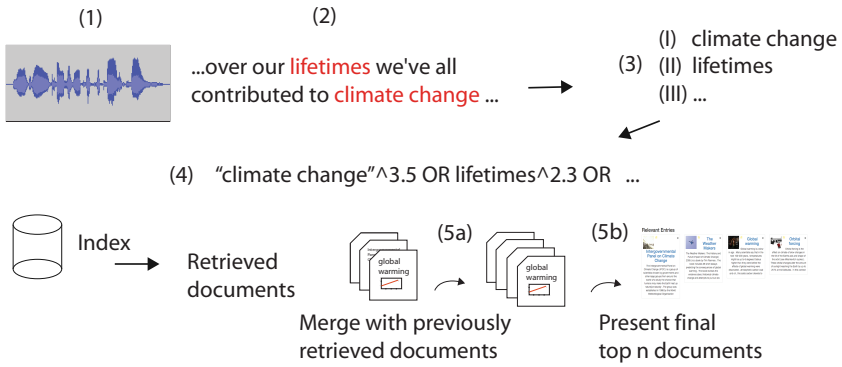

FIGURE 9.1: Processing steps of *Ambient search*

Our approach is based on five major processing steps, as depicted in Figure 9.1. These steps are carried out in real-time and in a streaming fashion, i.e. we make use of a new transcription hypothesis as soon as it is available.

At first, the speech signal is streamed into an ASR system (1). It emits the partial sentence hypothesis and also predicts sentence boundaries. Once a full sentence has been hypothesized, new keywords and keyphrases are extracted in the current sentence, if available (2). These terms are then ranked (3) and merged with the ones from previous sentences. A query is then composed, which is submitted to a precomputed index of documents (4).

Eventually, the returned documents are also aggregated (5a), i.e. previously found documents decay their score over time and newer documents are sorted into a list of n best documents. This list is then sorted by topical relevance of the documents and by time, with newer documents having precedence. Finally, the n best relevant documents are presented to the user (5b) and updated as soon as changes become available. Alongside the n best documents, a time line of previously suggested articles is also maintained and displayed. The next subsections provide further details on the individual major processing steps.

9.2.1 Speech Decoding

We use the popular Kaldi (Povey et al., 2011) open-source speech recognition framework and acoustic models based on the TED-LIUM corpus (Rousseau et al., 2014). We make use of online speech recognition, i.e. models that transcribe speech in real-time and emit a partial transcription hypothesis, as opposed to offline decoding that

operates on already recorded and complete utterances. The models were built using the standard recipe for online acoustic models based on a DNN-HMM acoustic model and i-vectors. We also make use of the TED-LIUM 4-gram language model (LM) from Cantab Research (Williams et al., 2015). The vocabulary of the speech recognizer is determined by its phoneme dictionary[5] and is confined to about 150k words. The online speech recognizer achieved an average WER of 15.6% on TED talks that we selected for the evaluation in Section 9.3.

We make use of kaldi-gstreamer-server[6], which wraps a Kaldi online model into a streaming server that can be accessed with websockets. This provides a bi-directional communication channel, where audio is streamed to the server application and partial and full sentence hypotheses and boundaries are simultaneously returned as JSON objects.

9.2.2 Keyphrase Extraction

A keyphrase, as opposed to a single keyword, can consist of one or more keywords that refer to one concept. We first precompute a DRUID (Riedl and Biemann, 2015) dictionary on a recent Wikipedia dump with scores for single adjectives or nouns and noun phrases. The restriction to only use adjectives and nouns is a common one in keyword detection, c.f. (Liu et al., 2010). DRUID is an unsupervised measure for multiword expressions using distributional semantics. Intuitively, DRUID finds multiword expressions by combining a uniqueness measure for phrases with a measure for their incompleteness. Uniqueness in this context is based on the assumption that multiword expressions (MWEs) can often be substituted by a single word without considerably changing the meaning of a sentence.

The uniqueness measure $uq(t)$ is computed with a distributional thesaurus, as the ratio of all similar unigrams of a term t divided by the number of n-grams similar to t. The incompleteness (ic) measure serves to punish incomplete terms, in that it counts the number of times that the same words appear next to a term. The final DRUID measure for any term t is the subtraction of the incompleteness measure from the uniqueness measure: $DRUID(t) = uq(t) - ic(t)$. This helps to rank incomplete multiwords lower than their complete counterparts, e.g. 'red blood' is ranked lower than 'red blood cell'.

DRUID is implemented as a JoBimText (Biemann and Riedl, 2013) component, which can be downloaded from the JoBimText project website[7] alongside precomputed dictionaries for English.

[5]The Kaldi TED-LIUM recipe uses CMUdict (`https://github.com/cmusphinx/cmudict` (last accessed: December 2021)) plus a few automatically generated entries

[6]`https://github.com/alumae/kaldi-gstreamer-server` (last accessed: December 2021)

[7]`http://ltmaggie.informatik.uni-hamburg.de/jobimtext/components/druid/` (last accessed: December 2021)

9.2.3 Term Ranking

We first precompute IDF and Word2Vec (Mikolov et al., 2013) lookup tables for all unique words in the Simple English Wikipedia and for all multiword terms in our DRUID dictionary. Word2Vec (CBOW) is our source for semantic similarity. We train it on stemmed text and treat multiwords as found with DRUID as opaque units. The final word embedding lookup table maps (in our case stemmed) word and phrase IDs into a 100 dimensional continuous vector space. The model exploits the distributional properties of raw text for semantic similarity and the distance between embeddings can be used as a word and phrase similarity measure.

Using the lookup tables, we build a term ranking measure as follows. We extract all keyphrases from the last 10 sentences with a DRUID score of $\geq c$ and filter all stop words and any word that is not an adjective or noun, as determined by an off-the-shelf part of speech (POS)-tagger[8]. The cutoff constant c can be used to tune the amount of generated multiword candidates, with useful values ranging from 0.3 to 0.7 (see also Section 9.3). All multiwords and any single word remaining after filtering is proposed as a candidate. We then compute the average Word2Vec vector over all candidate terms. Finally, we score each candidate term according to the cosine distance of each term word vector to the average term word vector of the last 10 sentences and multiply this with the TF-IDF score of the given term:

$$tr(term_k, trans) = d_{cos}\left(w2v(term_k), \frac{1}{|terms|}\sum_{i=1}^{|terms|} w2v(term_i)\right) \cdot TFIDF(term_k, trans) \tag{9.1}$$

where tr is the term ranking function that ranks a $term_k$ out of the set of all candidate $terms$ to a given transcript $trans$. $w2v$ yields the embedding of the given term, TFIDF yields the TFIDF score of the given term and transcript (IDF computed on the background Wikipedia corpus) and d_{cos} is the standard cosine similarity.

This ranking measure tr can be interpreted as a combination of the distance to the core topic (Word2Vec) and the general importance of the term for the text window (TF-IDF). We use the measure to extract up to 10 highest ranked candidate keywords and keyphrases in the text window. For instance, for the first third of Alice Bows Larkin's TED talk on climate change[9], the system would rank the terms as: "climate change", future, emissions, "negative impacts", potential, profound, workplaces, behaviors, gas.

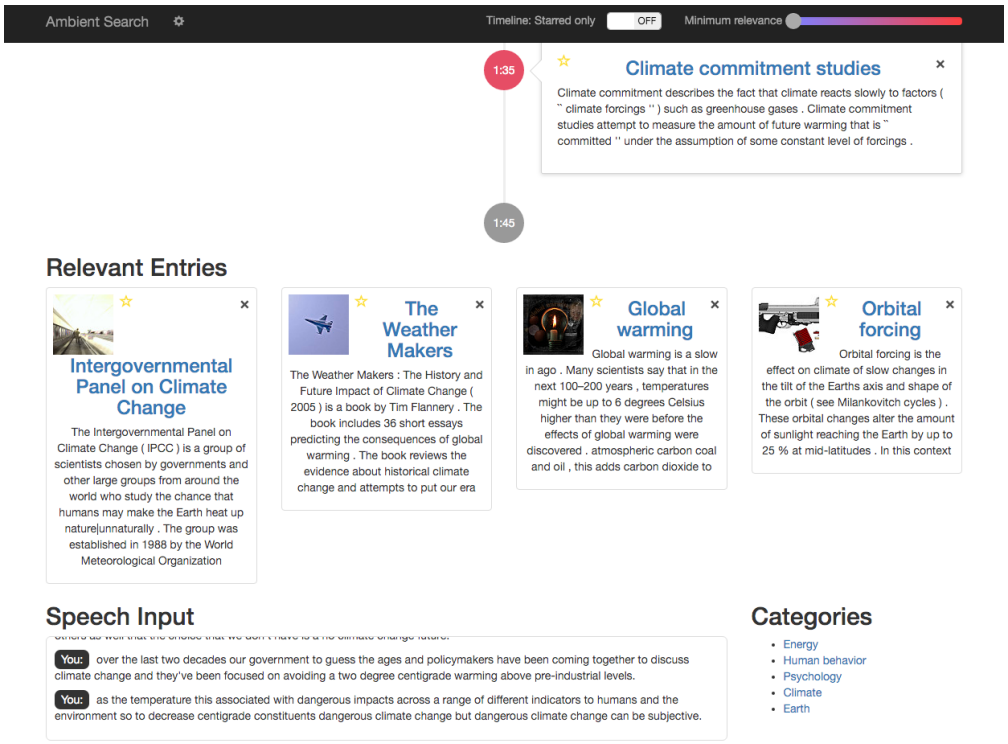

FIGURE 9.2: Screenshot of the system after listening to the first minutes of the TED talk "We're too late to prevent climate change - here is how we adapt" by Alice Bows Larkin

9.2.4 Index Queries

We use Elastic Search[10] and stream2es[11] to build an index of the Simple English Wikipedia[12]. We index all articles, including special pages and disambiguation pages and use a query filter to obtain only regular articles when querying the index. We build an OR query where at least 25% of the query terms should match (by setting the "minimum_should_match" parameter), also assigning the scores obtained in the last section to the individual terms in the query.

With the example ranking from the previous section, the query would be:

```
"climate change"^23.111 future^13.537 emissions^9.431 "negative
    impacts"^3.120 potential^2.985 profound^2.679 workplaces^2.562
    behaviors^2.368 gas^1.925
```

It would return the following Wikipedia articles (ranked by Elastic Search in that order):

[8] The POS tagger we use is from the spacy library (`http://spacy.io`) (last accessed: December 2021)

[9] `https://www.ted.com/talks/alice_bows_larkin_we_re_too_late_to_prevent_climate_change_here_s_how_we_adapt` (last accessed: December 2021)

[10] `https://www.elastic.co/` (last accessed: December 2021)

[11] `https://github.com/elastic/stream2es` (last accessed: December 2021)

[12] `https://simple.wikipedia.org` (last accessed: December 2021)

```
"Intergovernmental Panel on Climate Change", "Global warming", "
   Climate change", "Global dimming", "The Weather Makers", "
   Greenhouse effect", "United Kingdom Climate Change Programme", "
   Ocean acidification", ...
```

This process is repeated for every new sentence and the scores of older retrieved documents decay (are multiplied with $d = 0.9$), to allow newer documents to rank higher.

9.2.5 Visual Presentation

Figure 9.2 gives a visual impression of our system, after it has listened for a few minutes to Alice Bows Larkin's TED talk on climate change. We show excerpts of up to four relevant Wikipedia documents to the user. Clicking on such a document opens up a modal view to read the Wikipedia article. Articles are either retrieved online or using an offline version of the Simple English Wikipedia using XOWA[13]. Articles can be starred to quickly retrieve them later and also removed, to signal the system that the article was irrelevant. When newer and more relevant articles are retrieved, older articles move into a timeline, which is constructed above the currently retrieved articles. The newest articles are at the bottom of the page and the page keeps automatically scrolling to the end, like a terminal, if the user does not scroll up. In the timeline, the relevance of a document is also visually displayed with different coloring of an element's circular anchor. The user can also regulate the threshold for minimum document relevance.

9.2.6 Implementation Details

We encapsulate the processing steps outlined in Section 9.2 into the following Python programs:

(1) A *Kaldi client program* that either uses the system's microphone or an audio file, streaming it in real time to obtain partial and full transcription hypothesis. (2) A *relevant event generator program* that searches for new keywords and keyphrases and queries the elastic search index to obtain relevant documents. (3) The *Ambient Search server*, which sends out appropriate events to the browser view to display the current top n relevant documents and to move older documents into a timeline.

We make use of message passing to communicate inputs and results of the individual programs using a redis-server[14]. Word2Vec and TF-IDF vectors are computed with the Gensim (Řehůřek and Sojka, 2010) package, while DRUID is precomputed as a list with JoBimText[15]. The Ambient Search web page is using HTML5/JS and Bootstrap[16] and connects to an ambient server instance running on the Python

[13] `https://gnosygnu.github.io/xowa/` (last accessed: December 2021)

[14] `http://redis.io/` (last accessed: December 2021)

[15] `http://ltmaggie.informatik.uni-hamburg.de/jobimtext/components/druid/` (last accessed: December 2021)

[16] `http://getbootstrap.com/` (last accessed: December 2021)

micro-framework Flask[17]. The web page is updated using Server Sent Events (SSE) or Long Polling as a browser fallback. This enables a reversed information channel, where the server pushes descriptions of new relevant documents to the browser client as it becomes available.

9.3 Evaluation

We base our evaluation on 30 fragments of 10 recent TED talks, which we downloaded as mp3 files from the TED.com website. These talks are not part of the TED-LIUM training dataset. In the following, we evaluate the proposed keywords and keyphrases, as well as the proposed documents from the audio file transcribed in real-time.

9.3.1 Keyphrase and Document Retrieval

We had two annotators manually pick terms (keywords and keyphrases) that are central to the topic of the talk and those that would cover a user's potential additional information needs. What should be included as a term can be very subjective, the inter-annotator agreement is $\kappa = 0.45$, with one annotator choosing 292 terms in total and the other 580. The overlapping set we use in our evaluation consists of 206 terms and 460 other terms were chosen by only one of the annotators.

Finally, we also measure directly how relevant the retrieved documents are: We focus on an evaluation of the top-ranked documents returned by our ambient IR system for a particular TED talk fragment, since only top documents are suggested to the user of *Ambient Search*. The Normalized Discounted Cumulative Gain (NDCG) measure (Järvelin and Kekäläinen, 2002) is a popular choice to evaluate search engines and also takes into account the ranking of the proposed documents.

We evaluate on the top-5 returned documents of the complete system. We had two annotators that used the standard relevance scale from 0-3, where 0 means irrelevant and 3 very relevant. NDCG directly measures how relevant the returned documents are. While the effort is considerably higher, since different system outputs have to be judged, NDCG measures the end-to-end performance of the system. For computing NDCG, we pool all judgments across systems, obtaining an average of 27.7 relevance judgments per fragment, following standard practices for IR evaluations (Clarke et al., 2012). We use the standard NDCG measure with $k = 5$:

$$NDCG_k = \frac{DCG_k}{IDCG_k} \tag{9.2}$$

[17] `http://flask.pocoo.org/` (last accessed: December 2021)

$$DCG_k = (rel_1 + \sum_{i=2}^{k} \frac{rel_i}{log_2 i}) \tag{9.3}$$

where rel_i is a document's average relevance score in respect to the speech input. The Ideal Discounted Cumulative Gain (IDCG) assumes the best ranking of all possible relevant documents found in the set of all pooled judgments of a given transcript. The DCG on this optimal ranking, with respect to the set of documents retrieved by all systems for a particular transcript, is then used to compute IDCG.

9.3.2 Results

TABLE 9.1: Comparison of TF-IDF baseline keyword and keyphrase extraction methods, the proposed LDA based keyword extraction method by Habibi and Popescu-Belis, 2015 and our proposed method based on DRUID, Word2vec and TF-IDF. The comparison is based on the same Kaldi transcriptions and the same training resources (Simple English Wikipedia from May 2016).

Keyword/keyphrase extraction method	Mean Recall (Std. Dev. in %)	Mean Precision (Std. Dev. in %)	Mean NDCG (Std. Dev. in %)
(1) TF-IDF baseline no MWEs, no filtering	26.97% (16.74%)	24.33% (15.42%)	0.188 (20.0%)
(2) TF-IDF baseline no MWEs, stopword filtering	39.24% (15.36%)	34.33% (10.86%)	0.387 (27.8%)
(3) TF-IDF baseline no MWEs, full filtering	40.91% (13.55%)	36.42% (10.99%)	0.426 (27.8%)
(4) TF-IDF baseline with MWEs (c=0.3), full filtering	43.22% (18.22%)	37.09% (16.61%)	0.392 (27.4%)
(5) Habibi and PB original implementation	36.68% (15.37%)	32.00% % (11.66%)	0.427 (28.0%)
(6) Habibi and PB our prep., without MWEs	43.76% (16.78%)	39.24% % (12.75%)	0.465 (24.1%)
(7) Our proposed method with MWEs (c=0.3)	**48.52% (21.55%)**	41.89 % (15.82%)	0.453 (25.7%)
(8) Our proposed method with MWEs (c=0.5)	48.08% (17.63%)	**42.42 % (13.20%)**	0.469 (26.9%)
(9) Our proposed method with MWEs (c=0.7)	48.48% (19.15%)	**42.42 % (13.45%)**	0.471 (26.1%)
(10) Our proposed method without MWEs	44.87% (17.24%)	40.08% (14.03%)	**0.481 (26.8%)**

In Table 9.1, we show a comparison of different methods for automatic keyword extraction on TED talk transcriptions (as produced by kaldi-gstreamer-server / the Kaldi online model). All methods use the same resources, i.e. they are all pretrained on the same Simple English Wikipedia dump from May 2016. However, our proposed method and the TF-IDF baseline can also produce terms that are DRUID multiwords, while the original implementation of Habibi and Popescu-Belis (2015) can

TABLE 9.2: Comparison of the proposed LDA based keyword extraction method by (Habibi and Popescu-Belis, 2015) and our proposed method based on DRUID, Word2vec and TF-IDF on manual TED talk transcripts.

Keyword/keyphrase extraction method	Mean Recall (Std. Dev. in %)	Mean Precision (Std. Dev. in %)	Mean NDCG (Std. Dev. in %)
(11) Habibi and PB our prep., without MWEs	43.99% (15.26%)	39.33 % (12.63%)	0.476 (21.7%)
(12) Our proposed method with MWEs (c=0.3)	51.75% (20.43%)	45.67 % (16.47%)	0.518 (24.8%)
(13) Our proposed method with MWEs (c=0.5)	52.19% (19.09%)	46.19 % (15.27%)	0.574 (22.1%)
(14) Our proposed method with MWEs (c=0.7)	**52.68% (17.20%)**	**46.85 % (14.76%)**	**0.602 (22.1%)**
(15) Our proposed method without MWEs	47.81% (17.28%)	43.52 % (16.09%)	0.578 (25.2%)

only produce single keywords. All methods where allowed to produce maximally 10 words in the keyword evaluation. Partially covered keyphrases where also counted as a hit for the direct keyword evaluation and a multiword term was counted as multiple words. In the NDCG evaluation, we allow each system to produce an equal number of 10 terms.

For the TF-IDF baselines (1-4), preprocessing is the most important performance factor, with the best results being obtained by filtering stop words and any words that are not adjectives and nouns. However, while DRUID multiwords help to gain much better keyword recognition scores, it did not achieve a better NDCG score on speech transcripts. We also saw good results using the method proposed by Habibi and Popescu-Belis (2015), with the diversity constraint (λ) set to the default value of 0.75, which was the optimal value in the domain of meeting transcripts. However, we noticed that the publicly available Matlab implementation of this method[18] only removed stopwords as part of its preprocessing (5). When we use our preprocessing as input (6), we can improve both keyword and NDCG evaluation scores significantly.

Our proposed methods (7-9) with enabled multiword keyphrases seem to better represent the content of fragments, as shown by the keyword judgements. Again, DRUID further improved keyword recognition scores, but it did not achieve a better NDCG score on speech transcripts. The best NDCG score using speech transcripts was obtained with our proposed method *without* using multiwords (10). We experimented with different values of c: 0.3, 0.5 and 0.7, which all lowered NDCG scores. On average, this translates to 2.16, 1.56 and 0.53 multiword terms per query, respectively. The numbers are slightly lower if we use manual transcripts (1.9, 1.4 , 0.5).

We also evaluated our methods on manual transcriptions (11-15), see Table 9.2.

[18] `https://github.com/idiap/DocRec` (last accessed: December 2021)

Here the picture is different, as using DRUID can improve NDCG scores. However, only the highest cutoff factor of 0.7 (producing the smallest number of multiword candidates) yielded the best performance, suggesting that the number of added multiword candidates is an important parameter in the query generation. The scores on manual transcriptions can also be understood as the theoretically best possible scores for each method, assuming a perfect speech transcription system. If we compare them, we find that imperfect transcriptions have a high impact on system performance for all methods, as NDCGs are considerably higher with manual transcripts. If we correlate WER with our method in (10), we only observe a weak negative correlation of -0.193. If we use multiword terms, the negative correlation is higher with a coefficient of -0.293. The comparison system from Habibi and Popescu-Belis in (6) has the lowest negative correlation of -0.118 and it does not seem to gain as much in the NDCG evaluation on perfect transcriptions as our system.

9.3.3 Error Analysis

If we look at fragments individually and compare our method (10) to Habibi and Popescu-Belis (2015), we find that in 15 transcription fragments their system has a higher NDCG score and in 14 our system scores higher, with one equal score. On average, in cases where our system scored higher, WER was 14.8%, and where it scored lower WER was 16.8%. For example, for all 3 fragments of the talk "Kids Science" by Cesar Harada[19], where the accent of the speaker deteriorates WER to 33.4%, our system returns much more irrelevant documents. Our average NDCG for the talk is 0.180, while Habibi and Popescu-Belis' system scores 0.454. Among the articles found by our system are "National School Lunch Program", "Arctic Ocean", "Fresh water", "Water", "Coal preparation plant", "Coal mining" and "Plant", with most of the articles being irrelevant to the talk.

Word errors and resulting erroneous search terms are responsible for most irrelevant documents. For example, the phrase "in coal power plant" appears in the transcript instead of "nuclear power plant". In this example, Habibi and Popescu-Belis' system finds better matching articles, like "Microscope", "Light Microscope", "Mangrove" , "Fresh water", "River delta", "Fishing" which can be attributed to finding the keyword "microscope" and otherwise picking simpler keywords like "ocean", "sea", "fishing" and "river", which our system entirely misses. This changes when we run the systems on the manually transcribed texts, as our system with enabled multiword terms (9) then finds "nuclear power plant", which helps to retrieve very relevant documents ("Nuclear reaction", "Nuclear chemistry", "Nuclear power plants").

Moreover, if we enable the use of multiword terms in our method with c=0.7, we observe that NDCG was improved by the keyphrase enabled method in 9 cases, but also decreased in 11, with the other 10 transcripts remaining unchanged. If WER

[19] `http://www.ted.com/talks/cesar_harada_how_i_teach_kids_to_love_science` (last accessed: December 2021)

is poor, the keyphrase enabled methods do not seem to contribute to improving NDCG performance and tend to lower it. For example, in the 5 transcripts with the highest WER (19.8-40.9%, average: 26.4%), 3 scores are lowered and 2 unchanged. If we group all cases where the NDCG performance drops, we observe an average WER of 16.9% vs. 12.3% for the cases where they help to improve the NDCG score (average of all transcripts is 15.6%). This further suggests that a query generation with multiword terms helps more in cases where word error rates are low.

Interestingly, in 5 out of 30 transcripts, no multiword terms are found with c=0.7 but NDCG values were still slightly lower in all cases compared to our single word method. While the set of terms in all queries were nearly unchanged, their ranking was affected. This might be attributed to how we build IDF and Word2Vec models: multiwords are opaque units in the models. This can change the dense vectors and IDF values for the constituents of multiwords compared to training on single words and thus affect ranking scores. However, in some of the automatic transcriptions, only constituents of the correct multiwords can be found because of transcription errors, so that our method has to rank the constituent instead of the full multiword.

9.4 Conclusion

We presented *Ambient Search*, an approach that can show and retrieve relevant documents for speech streams. Our current prototype uses Wikipedia pages, as this provides a large document collection for testing purposes with a universal coverage of different topics.

Our method compares favorably over previous methods of topic discovery and keyword extraction in speech transcriptions. We explored the use of multiword terms as keyphrases, alongside single word terms. Our proposed extraction method using Word2Vec (CBOW) embeddings and TF-IDF is fairly simple to implement and can also be adapted quickly to other languages as it does not need any labelled training data. The only barrier of entry can be the availability of a speech recognition system in the target language.

We also plan to evaluate a more dynamic approach to query generation, where the number of terms is dynamically chosen and not simply capped at a maximum number of term candidates after ranking. As the proposed use of multiword terms seems to be somewhat dependent on the quality of the transcription, it might also make sense to include likelihood information of the speech recognition system. Our evaluation on manual transcriptions also suggests that there is quite a large headroom for our system to benefit from any future reductions in WER of the online speech recognition component.

In actual live deployment and usage in discussions, lectures or business meetings, confidential information can be present in the speech streams. A privacy aspect has already been addressed by *Ambient Search*: the speech recognition is not carried out "in the cloud" and can be deployed on one's own infrastructure. Similarly, an

offline version of the Simple English Wikipedia and a corresponding search index is used to retrieve and find articles. The transmission of personal data through the internet can be entirely omitted – a vital aspect for the acceptance of such an application.

We have published the source code of *Ambient Search* under a permissive license on Github[20], along with all pretrained models, a demonstration video, evaluation files and scripts that are necessary to repeat and reproduce the results presented in this chapter.

[20] `https://github.com/bmilde/ambientsearch` (last accessed: April 2022)

Chapter 10

Conclusion

In this thesis, we demonstrated that effective speech representations can be learned from the speech data itself, mostly without any further annotations. To that end, deep neural networks trained with stochastic gradient decent are a powerful tool to train models that learn representations of speech through context. Several different training paradigms ranging from transfer learning, unsupervised learning, self-supervised learning, multitask learning to supervised learning were applied to speech processing tasks.

Transfer learning can be useful when neural networks are trained on speech processing tasks with little training data. A model that is pretrained on an auxiliary task may generalize better to a target task. We used it effectively together with neural network based representation learning in a paralinguistic classification task with raw spectogram features, beating the classification results of models based on handcrafted features by a large margin.

The Unspeech model proposed in this thesis can explicitly embed speech contexts into fixed-sized dense vectors. The model is trained on unannotated speech using unsupervised contrastive self-supervision. Due to a wide range of variability because of channel, speaker and environment factors, context is important in speech processing tasks. We showed that depending on the negative sampling method, such context embeddings can have speaker properties, or be used to embed speaker-independent speech information. The embeddings can be used to cluster speech segments. Using the labels yields gains in ASR application, when no speaker labels are available in the training data. The embeddings can also directly be appended to speech representations commonly used as input to acoustic models such as TDNN-HMMs to improve speech recognition tasks. Results on further downstream task applications such as speaker recognition, short command recognition and emotion recognition show a wide range of other possible applications. Unspeech embeddings can thus also be used in tasks that require speaker independence. They can be used to rapidly tackle speech tasks with simple linear classifiers, leveraging pretrained Unspeech models to generate the embeddings.

In automatic unit discovery, we aim to find units that behave similarly to linguistically motivated phones, in the sense that the learned representations are able to encode differences in minimal pairs of a language. Until recently, this task was

a niche subject in speech processing, but has received continued research interest over the past years. Recent works of (Liu et al., 2018) and (Baevski et al., 2021) show that acoustic unit discovery is currently key to solving unsupervised ASR (see also Chapter 3.13). In most cases, the task of acoustic unit discovery is using untranscribed speech as input data. The representations can then be used as input to train unsupervised ASR systems that only need collections of text and audio data without any alignments to iteratively train a full ASR system. ABX error rate as a direct proxy can be used to assess the quality of either and can directly measure if the learned representations can distinguish minimal pairs of a language as well as distinguish between speaker properties of the signal. However, clustering discretized units of speech well seems to be difficult. Nonetheless, the discretized units scored lower ABX error rates than standard speech features such as PLP and MFCC.

While ABX is representation agnostic, i.e. it can be used to compare discretized units as well sequences of embeddings (as well as standard speech features such as MFCC and PLP), it seems to be difficult to fairly compare ABX errors on discretized units vs. dense embeddings. However, when dense embeddings from other systems are discretized for a fairer comparison, Sparsespeech units compare favorably. Gumbel softmax is a powerful tool to integrate discrete inference into neural networks. The approximation is also an elegant solution to the otherwise non-differentiability of discretization in neural networks. It allows discrete inference to be used in models that can then still be trained with standard back-propagation. The resulting output can be interpreted as an unsupervised posteriogram over an inventory of units.

Automatic G2P conversion is also needed for unsupervised speech recognition and is used to extend the lexicon in many supervised ASR models as well. Outliers and loan words are difficult corner cases for automatic G2P, even when multi-task learning is applied and G2P models are trained on multiple languages. Manual annotation for these difficult cases is unfortunately still necessary, but confidence values can be used to select frequent candidates where G2P is likely to fail.

In the more practical-oriented part of this thesis, a recipe to train German ASR models was developed to leverage freely available audio sources and insights gained on lexicon extension with automatic G2P conversion for German. A TDNN-HMM model trained on 1,700 hours of freely available audio data yields strong WER results on the Tuda-De test set and beats other published WER results on this dataset by a large margin, including various end-to-end models. Only a very recent result, concurrent to the submission of this thesis, shows that a Conformer Transducer trained on much more training data (4,520 hours) is able to exceed this result.

The ASR model was also successfully applied to automatic subtitling for lectures, yielding a practical system that is used by the lecture media platform of Universität Hamburg (Lecture2Go). Any lecturer uploading videos to Lecture2Go can now use this automatic subtitling service. Furthermore, at the intersection of NLP and speech applications, several speech applications have been proposed, such as a live meeting transcription system with automatic summarization and simultaneous visualization

and an ambient search application that retrieves relevant Wikipedia articles on the subject of a discussion. Privacy is important in such applications, thus processing speech locally is a necessity.

Representation learning is key to solving the problem of high variance and complexity of spoken language in speech signals. Alternative machine learning paradigms, other than purely supervised, are going to be central in improving speech recognition and speech processing tasks further. In turn, they allow more flexible data scenarios, in which for example speech data is only partially labeled, or where a G2P model is trained on multiple related languages or lexicon sources with different phoneme sets. Pre-trained speech models with self-supervision allow to rapidly build a system for speech classification tasks.

While this thesis focuses on German and English ASR, there is undoubtedly untapped potential in languages that do not enjoy such a high amount of available resources as English and German, when alternative machine learning paradigms are applied to them. Going forward, particularly models that can learn from unlabeled speech data, such as the ones proposed in this thesis, will be of high interest. Applying them effectively to under-resourced languages with unsupervised speech recognition will undoubtedly make it easier to unlock practical speech systems for many other languages.

Bibliography

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. (2016). "TensorFlow: A system for large-scale machine learning". In: *Proc. USENIX Symposium on Operating Systems Design and Implementation*. Savannah, GA, USA, pp. 265–283 (cit. on pp. 83, 104).

Acero, Alejandro and Richard Stern (1990). "Environmental robustness in automatic speech recognition". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Albuquerque, NM, USA, pp. 849–852 (cit. on p. 5).

Agarwal, Aashish and Torsten Zesch (2019). "German End-to-end Speech Recognition based on DeepSpeech". In: *Proc. KONVENS*. Erlangen, Germany, pp. 111–119 (cit. on p. 135).

Ahmed, Nasir, T Raj Natarajan, and Kamisetty Rao (1974). "Discrete cosine transform". In: *IEEE transactions on Computers* 100.1, pp. 90–93 (cit. on p. 47).

Allen, Jonathan (1977). "Short term spectral analysis, synthesis, and modification by discrete Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25.3, pp. 235–238 (cit. on p. 46).

Ardila, Rosana, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber (2020). "Common Voice: A Massively-Multilingual Speech Corpus". In: *Proc. LREC*. Virtual, pp. 4211–4215 (cit. on pp. 85, 91, 131).

Arthur, David and Sergei Vassilvitskii (2007). "k-means++: The advantages of careful seeding". In: *Proc. ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. New Orleans, LA, USA, pp. 1027–1035 (cit. on pp. 14, 15, 103, 104).

Azizpour, Hossein, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson (2015). "From generic to specific deep representations for visual recognition". In: *Proc. IEEE conference on computer vision and pattern recognition workshops*. Boston, MA, USA, pp. 36–45 (cit. on pp. 69, 70, 75).

Badino, Leonardo, Claudia Canevari, Luciano Fadiga, and Giorgio Metta (2014). "An auto-encoder based approach to unsupervised learning of subword units". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy, pp. 7634–7638 (cit. on p. 99).

Badino, Leonardo, Alessio Mereta, and Lorenzo Rosasco (2015). "Discovering discrete subword units with binarized autoencoders and hidden-markov-model encoders". In: *Proc. Interspeech*. Dresden, Germany, pp. 3174–3178 (cit. on p. 98).

Baevski, Alexei, Wei-Ning Hsu, Alexis Conneau, and Michael Auli (2021). "Unsupervised Speech Recognition". In: *arXiv preprint arXiv:2105.11084* (cit. on pp. 58, 59, 98, 121, 160).

Baevski, Alexei and Abdelrahman Mohamed (2020). "Effectiveness of Self-Supervised Pre-Training for ASR". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain, pp. 7694–7698 (cit. on p. 100).

Baevski, Alexei, Steffen Schneider, and Michael Auli (2020a). "vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations". In: *Proc. International Conference on Learning Representations (ICLR)*. Virtual Addis Ababa, Ethopia (cit. on p. 100).

Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli (2020b). "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *Proc. Advances in Neural Information Processing Systems 33*, pp. 12449–12460 (cit. on p. 58).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. International Conference on Learning Representations (ICLR)*. San Diego, CA, USA (cit. on pp. 40, 101).

Bahl, Lalit, Peter de Souza, Ponani Gopalakrishnan, David Nahamoo, and Michael Picheny (1991). "Context dependent modeling of phones in continuous speech using decision trees". In: *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, pp. 264–270 (cit. on p. 51).

Baldi, Pierre and Kurt Hornik (1989). "Neural networks and principal component analysis: Learning from examples without local minima". In: *Neural networks* 2.1, pp. 53–58 (cit. on p. 35).

Baum, Doris, Daniel Schneider, Jochen Schwenninger, Barbara Samlowski, Thomas Winkler, and Joachim Köhler (n.d.). "DiSCo-A german evaluation corpus for challenging problems in the broadcast domain". In: *Proc. LREC*, pp. 1695–1699 (cit. on p. 88).

Baum, Leonard E., Ted Petrie, George Soules, and Norman Weiss (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *The annals of mathematical statistics* 41.1, pp. 164–171 (cit. on p. 50).

Baumann, Timo, Arne Köhn, and Felix Hennig (2019). "The Spoken Wikipedia Corpus collection: Harvesting, alignment and an application to hyperlistening". In: *Language Resources and Evaluation* 53.2, pp. 303–329 (cit. on p. 130).

Bavarian Archive for Speech Signals (n.d.). *Extended SAM-PA*. `http://www.bas.uni-muenchen.de/forschung/Bas/BasSAMPA` (cit. on p. 131).

Bayes, Thomas, Richard Price, and John Canton (1763). "An essay towards solving a problem in the doctrine of chances". In: *Philosophical Transactions of the Royal Society of London* 53, pp. 370–418 (cit. on p. 44).

Beckmann, Norbert, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger (1990). "The R*-tree: an efficient and robust access method for points and rectangles". In: *Proc. ACM SIGMOID*. Vol. 19. 2. Atlantic City, NJ, USA, pp. 322–331 (cit. on p. 16).

Bengio, Samy and Georg Heigold (2014). "Word embeddings for speech recognition". In: *Proc. Interspeech*. Singapore, pp. 1053–1057 (cit. on p. 79).

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin (2003). "A neural probabilistic language model". In: *Journal of machine learning research* 3, pp. 1137–1155 (cit. on p. 54).

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2, pp. 157–166 (cit. on p. 25).

Bentley, Jon Louis (1975). "Multidimensional binary search trees used for associative searching". In: *Communications of the ACM* 18.9, pp. 509–517 (cit. on p. 16).

Benzeghiba, Mohamed, Renato De Mori, Olivier Deroo, Stephane Dupont, Teodora Erbes, Denis Jouvet, Luciano Fissore, Pietro Laface, Alfred Mertins, Christophe Ris, et al. (2007). "Automatic speech recognition and speech variability: A review". In: *Speech communication* 49.10-11, pp. 763–786 (cit. on p. 5).

Bergstra, James, Olivier Breuleux, Frederic Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio (2010). "Theano: a CPU and GPU math compiler in Python". In: *9th Python in Science Conference (SCIPY)*. Austin, TX, USA, pp. 18–24 (cit. on p. 71).

Beringer, Nicole and Florian Schiel (2000). "The Quality of Multilingual Automatic Segmentation using German MAUS". In: *Proc. Interspeech*. Beijing, China, pp. 728–731 (cit. on p. 123).

Bermuth, Daniel, Alexander Poeppel, and Wolfgang Reif (2021). "Scribosermo: Fast Speech-to-Text models for German and other Languages". In: *arXiv preprint arXiv :2110.07982* (cit. on pp. 135, 136).

Beyer, Kevin, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft (1999). "When is "nearest neighbor" meaningful?" In: *Proceedings international conference on database theory*. Jerusalem, Israel, pp. 217–235 (cit. on pp. 15, 16).

Biemann, Chris, Karsten Böhm, Gerhard Heyer, and Ronny Melz (2004). "SemanticTalk: Software for Visualizing Brainstorming Sessions and Thematic Concept Trails on Document Collections". In: *Proc. ECML/PKDD*. Pisa, Italy, pp. 534–536 (cit. on p. 146).

Biemann, Chris and Martin Riedl (2013). "Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity". In: *Journal of Language Modelling* 1.1, pp. 55–95 (cit. on p. 148).

Bisani, Maximilian and Hermann Ney (2008). "Joint-sequence models for grapheme-to-phoneme conversion". In: *Speech communication* 50.5, pp. 434–451 (cit. on pp. 122, 124, 131).

Bone, Daniel, Matthew Black, Ming Li, Angeliki Metallinou, Sungbok Lee, and Shrikanth S Narayanan (2011). "Intoxicated Speech Detection by Fusion of Speaker

Normalized Hierarchical Features and GMM Supervectors." In: *Proc. Interspeech*. Florence, Italy, pp. 3217–3220 (cit. on p. 66).

Bottou, Léon (2010). "Large-scale machine learning with stochastic gradient descent". In: *Proc. COMPSTAT*. Paris, France, pp. 177–186 (cit. on p. 24).

Bouselmi, Ghazi, Dominique Fohr, and Irina Illina (2008). "Multi-accent and accent-independent non-native speech recognition". In: *Proc. Interspeech*. Brisbane, Australia, pp. 2703–2706 (cit. on p. 5).

Bredin, Hervé (2017). "pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems". In: *Proc. Interspeech*. Stockholm, Sweden, pp. 3587–3591 (cit. on p. 85).

Bridle, John (1990). "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition". In: *Neurocomputing* 68, pp. 227–236 (cit. on p. 23).

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah (1994). "Signature verification using a "Siamese" time delay neural network". In: *Proc. Advances in Neural Information Processing Systems*. Denver, CO, USA, pp. 737–744 (cit. on p. 79).

Bullock, Theodore Holmes and Edward Frank Evans (1977). "Recognition of complex acoustic signals". In: *Dahlem Workshop on Recognition of Complex Acoustic Signals* (cit. on p. 47).

Can, Dogan, Victor Martinez, Pavlos Papadopoulos, and Shrikanth Narayanan (2018). "PyKaldi: A Python Wrapper for Kaldi". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, Canada, pp. 5889–5893 (cit. on pp. 137, 141).

Caruana, Rich (1997). "Multitask Learning". In: *Machine Learning* 28, pp. 41–75 (cit. on pp. 17, 66, 122).

Chen, Hongjie, Cheung-Chi Leung, Lei Xie, Bin Ma, and Haizhou Li (2015). "Parallel inference of Dirichlet process Gaussian mixture models for unsupervised acoustic modeling: A feasibility study". In: *Proc. Interspeech*. Dresden, Germany, pp. 3189–3193 (cit. on p. 99).

Chen, Stanley and Ronald Rosenfeld (2000). "A survey of smoothing techniques for ME models". In: *IEEE transactions on Speech and Audio Processing* 8.1, pp. 37–50 (cit. on p. 53).

Cho, Kyunghyun, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio (2014a). "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *Proc. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*. Doha, Qatar, pp. 103–111 (cit. on p. 40).

Cho, Kyunghyun, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014b). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proc. EMNLP*. Doha, Qatar, pp. 1724–1734 (cit. on pp. 39, 40, 123, 138).

Choi, Jiyoun, Anne Cutler, and Mirjam Broersma (2017). "Early development of abstract language knowledge: evidence from perception–production transfer of

birth-language memory". In: *Royal Society open science* 4.1. Article ID 160660 (cit. on p. 1).

Chorowski, Jan, Grzegorz Ciesielski, Jarosław Dzikowski, Adrian Łacucki, Ricard Marxer, Mateusz Opala, Piotr Pusz, Paweł Rychlikowski, and Michał Stypułkowski (2021). "Aligned Contrastive Predictive Coding". In: *Proc. Interspeech*. Brno, Czech Republic, pp. 976–980 (cit. on p. 100).

Chung, Yu-An, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee (2016). "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder". In: *Proc. Interspeech*. San Francisco, CA, USA, pp. 765–769 (cit. on p. 79).

Cieri, Christopher, David Miller, and Kevin Walker (2004). "The Fisher Corpus: a Resource for the Next Generations of Speech-to-Text." In: *Proc. LREC*. Lisbon, Portugal, pp. 69–71 (cit. on p. 6).

Clarke, Charles, Nick Craswell, and Ellen Voorhees (2012). "Overview of the TREC 2012 Web Track". In: *Proc. TREC*. Gaithersburg, MD, USA (cit. on p. 152).

Collobert, Ronan, Christian Puhrsch, and Gabriel Synnaeve (2016). "Wav2letter: an end-to-end convnet-based speech recognition system". In: *arXiv preprint arXiv: 1609.03193* (cit. on p. 100).

Cooley, James W and John W Tukey (1965). "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of computation* 19.90, pp. 297–301 (cit. on p. 1).

Dahl, George, Dong Yu, Li Deng, and Alex Acero (2012). "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1, pp. 30–42 (cit. on pp. 26, 55).

Daniluk, Michal, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel (2017). "Frustratingly Short Attention Spans in Neural Language Modeling". In: *Proc. International Conference on Learning Representations (ICLR)*. Toulon, France (cit. on p. 101).

Davis, Steven and Paul Mermelstein (1980). "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *IEEE transactions on Acoustics, Speech, and Signal Processing* 28.4, pp. 357–366 (cit. on pp. 46, 47).

Dehak, Najim, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet (2011). "Front-end factor analysis for speaker verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4, pp. 788–798 (cit. on p. 79).

Dempster, Arthur, Nan Laird, and Donald Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22 (cit. on p. 50).

Denisov, Pavel and Ngoc Thang Vu (2019). "IMS-speech: A speech to text tool". In: *Elektronische Sprachsignalverarbeitung 2019*. Dresden, Germany, pp. 170–177 (cit. on p. 135).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, MN, USA, pp. 4171–4186 (cit. on pp. 100, 108).

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul, pp. 2121–2159 (cit. on p. 29).

Dumais, Susan, Edward Cutrell, Raman Sarin, and Eric Horvitz (2004). "Implicit Queries (IQ) for Contextualized Search". In: *Proc. SIGIR*. Sheffield, UK, p. 594 (cit. on p. 146).

Dunbar, Ewan, Robin Algayres, Julien Karadayi, Mathieu Bernard, Juan Benjumea, Xuan-Nga Cao, Lucie Miskic, Charlotte Dugrain, Lucas Ondel, Alan W Black, et al. (2019). "The zero resource speech challenge 2019: TTS without T". In: *Proc. Interspeech*. Graz, Austria, pp. 1088–1092 (cit. on pp. 99, 111, 118).

Dunbar, Ewan, Xuan Nga Cao, Juan Benjumea, Julien Karadayi, Mathieu Bernard, Laurent Besacier, Xavier Anguera, and Emmanuel Dupoux (2017). "The Zero Resource Speech Challenge 2017". In: *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 323–330 (cit. on pp. 99, 106, 111).

Erhan, Dumitru, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio (2010). "Why does unsupervised pre-training help deep learning?" In: *Journal of Machine Learning Research* 11.Feb, pp. 625–660 (cit. on p. 27).

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Proceedings KDD*. Vol. 96. 34, pp. 226–231 (cit. on pp. 15, 16).

Eyben, Florian, Felix Weninger, Florian Gross, and Björn Schuller (2013). "Recent developments in openSMILE, the munich open-source multimedia feature extractor". In: *Proc. ACM International conference on Multimedia*. Barcelona, Spain, pp. 835–838 (cit. on p. 66).

Eyben, Florian, Martin Wöllmer, and Björn Schuller (2010). "openSMILE: The Munich versatile and fast open-source audio feature extractor". In: *Proc. ACM international conference on Multimedia*. Firenze, Italy, pp. 1459–1462 (cit. on pp. 66, 95).

Farabet, Clement, Camille Couprie, Laurent Najman, and Yann LeCun (2013). "Learning hierarchical features for scene labeling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8, pp. 1915–1929 (cit. on pp. 21, 26).

Fiesler, Emile and Russell Beale (1996). *Handbook of neural computation*. ISBN: 978-0128113189 (cit. on p. 21).

Fletcher, Harvey (1940). "Auditory patterns". In: *Reviews of modern physics* 12.1, p. 47 (cit. on p. 47).

Fukushima, Kunihiko and Sei Miyake (1982). "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Proceedings Competition and cooperation in neural nets*. Kyoto, Japan, pp. 267–285 (cit. on pp. 26, 30).

Gaida, Christian, Patrick Lange, Rico Petrick, Patrick Proba, Ahmed Malatawy, and David Suendermann-Oeft (2014). "Comparing open-source speech recognition toolkits". In: *Tech. Rep., DHBW Stuttgart* (cit. on p. 137).

Gales, Mark, Steve Young, et al. (2008). "The application of hidden Markov models in speech recognition". In: *Foundations and Trends in Signal Processing* 1.3, pp. 195–304 (cit. on pp. 45, 47, 51, 52).

Galescu, Lucian and James Allen (2001). "Bi-directional Conversion Between Graphemes and Phonemes using a Joint N-gram Model". In: *Proc. 4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*. Paper 131. Perthshire, Scotland (cit. on p. 122).

Garcia, Alvin and Herbert Gish (2006). "Keyword spotting of arbitrary words using minimal speech resources". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Toulouse, France, pp. 949–952 (cit. on p. 99).

Garofolo, John, Lori Lamel, William Fisher, Jonathan Fiscus, and David Pallett (1993). "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1". In: *STIN* 93. 27403 (cit. on p. 3).

Geislinger, Robert, Benjamin Milde, Timo Baumann, and Chris Biemann (2021). "Live Subtitling for BigBlueButton with Open-Source Software". In: *Proc. Interspeech*. Brno, Czech Republic, pp. 3319–3320 (cit. on pp. 142, 143).

Gemmeke, Jort, Daniel Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, Channing Moore, Manoj Plakal, and Marvin Ritter (2017). "Audio Set: An ontology and human-labeled dataset for audio events". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA, USA, pp. 776–780 (cit. on p. 95).

Gers, Felix, Jürgen Schmidhuber, and Fred Cummins (1999). "Learning to forget: Continual prediction with LSTM". In: *Neural Computation* 12.10, pp. 2451–2471 (cit. on p. 38).

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proc. AISTATS*. Sardinia, Italy, pp. 249–256 (cit. on p. 27).

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Deep sparse rectifier neural networks". In: *Proc. AISTATS*. Ft. Lauderdale, FL, USA, pp. 315–323 (cit. on p. 27).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press. ISBN: 0262035618 (cit. on pp. 12, 13, 24, 25, 31, 35, 36, 38).

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Proc. Advances in Neural Information Processing Systems*. Vol. 27. Montréal, Canada, pp. 2672–2680 (cit. on p. 58).

Gopalakrishnan, Ponani, Dimitri Kanevsky, Arthur Nadas, and David Nahamoo (1989). "A generalization of the Baum algorithm to rational objective functions". In: *Proc. Acoustics, Speech, and Signal Processing (ICASSP)*. Glasgow, Scottland, pp. 631–634 (cit. on p. 50).

Graves, Alex, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber (2006). "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proc. ICML*. Pittsburgh, PA, pp. 369–376 (cit. on pp. 57, 111).

Griewank, Andreas (2012). "Who Invented the Reverse Mode of Differentiation?" In: *Documenta Mathematica, Extra Volume ISMP*, pp. 389–400 (cit. on p. 24).

Griffin, Daniel and Jae Lim (1984). "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2, pp. 236–243 (cit. on p. 46).

Gulati, Anmol, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang (2020). "Conformer: Convolution-augmented Transformer for Speech Recognition". In: *Proc. Interspeech*. Virtual Shanghai, China, pp. 5036–5040 (cit. on p. 136).

Gumbel, Emil Julius (1948). *Statistical theory of extreme values and some practical applications: a series of lectures*. Vol. 33. US Government Printing Office (cit. on pp. 108, 109).

Gupta, Abhinav, Yajie Miao, Leonardo Neves, and Florian Metze (2017). "Visual features for context-aware speech recognition". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA, USA, pp. 5020–5024 (cit. on p. 80).

Gutmann, Michael and Aapo Hyvärinen (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proc. International Conference on Artificial Intelligence and Statistics*. Sardinia, Italy, pp. 297–304 (cit. on pp. 79, 82).

Habibi, Maryam and Andrei Popescu-Belis (2015). "Keyword Extraction and Clustering for Document Recommendation in Conversations". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.4, pp. 746–759 (cit. on pp. 146, 153–155).

Hachtel, Gary, R Brayton, and Fred Gustavson (1971). "The sparse tableau approach to network analysis and design". In: *IEEE Transactions on circuit theory* 18.1, pp. 101–113 (cit. on p. 24).

Hantke, Simone, Felix Weninger, Richard Kurle, Fabien Ringeval, Anton Batliner, Amr El-Desoky Mousa, and Björn Schuller (2015). "I hear you eat and speak: automatic recognition of eating condition and food type". In: *PloS one* 11.5. e0154486 (cit. on p. 66).

Haq, Sanaul, Philip JB Jackson, and J Edge (2009). "Speaker-dependent audio-visual emotion recognition." In: *Auditory-Visual Speech Processing (AVSP)*. University of East Anglia, Norwich, UK, pp. 53–58 (cit. on p. 94).

Harris, Fredric (1978). "On the use of windows for harmonic analysis with the discrete Fourier transform". In: *Proceedings of the IEEE* 66.1, pp. 51–83 (cit. on p. 46).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile (cit. on pp. 28, 29, 33, 67–69).

— (2016). "Deep Residual Learning for Image Recognition". In: *Proc. CVPR*. Las Vegas, NV, USA, pp. 770–778 (cit. on pp. 32–34, 40, 122).

Heck, Michael, Sakriani Sakti, and Satoshi Nakamura (2017). "Feature optimized DPGMM clustering for unsupervised subword modeling: A contribution to ZeroSpeech 2017". In: *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*. Okinawa, Japan, pp. 740–746 (cit. on pp. 99, 106, 108).

Hermansky, Hynek (1990). "Perceptual linear predictive (PLP) analysis of speech". In: *the Journal of the Acoustical Society of America* 87.4, pp. 1738–1752 (cit. on pp. 48, 49, 106, 110).

Hershey, Shawn, Sourish Chaudhuri, Daniel Ellis, Jort Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson (2017). "CNN architectures for large-scale audio classification". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA, USA, pp. 131–135 (cit. on p. 95).

Hinton, Geoffrey (2002). "Training products of experts by minimizing contrastive divergence". In: *Neural computation* 14.8, pp. 1771–1800 (cit. on p. 26).

— (2007). "To recognize shapes, first learn to generate images". In: *Progress in Brain Research* 165, pp. 535–547 (cit. on p. 66).

— (2013). "Lecture 10-Recurrent Neural Networks, Coursera: Neural networks for machine learning". In: *University of Toronto, Technical Report* (cit. on p. 37).

Hinton, Geoffrey, Peter Dayan, Brendan Frey, and Radford Neal (1995). "The "wake-sleep" algorithm for unsupervised neural networks". In: *Science* 268.5214, pp. 1158–1161 (cit. on p. 23).

Hinton, Geoffrey, Simon Osindero, and Yee-Whye Teh (2006). "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7, pp. 1527–1554 (cit. on p. 27).

Hinton, Geoffrey and Ruslan Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786, pp. 504–507 (cit. on p. 26).

Hochreiter, Sepp (1991). "Untersuchungen zu dynamischen neuronalen Netzen". In: *Master's thesis, Institut für Informatik, Technische Universitat, München* (cit. on pp. 25, 37, 40).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-term Memory". In: *Neural computation* 9.8, pp. 1735–1780 (cit. on pp. 26, 38, 40, 100).

Hoerl, Arthur and Robert Kennard (1970). "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* 12.1, pp. 55–67 (cit. on p. 68).

Hönig, Florian, Georg Stemmer, Christian Hacker, and Fabio Brugnara (2005). "Revising perceptual linear prediction (PLP)". In: *Proc. Interspeech*. Lisbon, Portugal, pp. 2997–3000 (cit. on p. 48).

Hopfield, John J (1982). "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8, pp. 2554–2558 (cit. on p. 37).

Huang, Chao, Tao Chen, and Eric Chang (2004). "Accent issues in large vocabulary continuous speech recognition". In: *International Journal of Speech Technology* 7.2-3, pp. 141–153 (cit. on p. 5).

Huang, Chao, Tao Chen, Stan Li, Eric Chang, and Jianlai Zhou (2001). "Analysis of speaker variability". In: *Proc. Eurospeech*. Aalborg, Denmark, pp. 1377–1380 (cit. on p. 5).

Huang, Guang-Bin and Chee-Kheong Siew (2004). "Extreme learning machine: RBF network case". In: *Proc. ICARCV 2004 8th Control, Automation, Robotics and Vision Conference*. Vol. 2. Kunming, China, pp. 1029–1036 (cit. on p. 74).

Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew (2006). "Extreme learning machine: theory and applications". In: *Neurocomputing* 70.1-3, pp. 489–501 (cit. on p. 74).

Huber, Peter (1964). "Robust estimation of a location parameter". In: *Breakthroughs in statistics* 53.1, pp. 73–101 (cit. on pp. 22, 111).

Hubert, Lawrence and Phipps Arabie (1985). "Comparing partitions". In: *Journal of classification* 2.1, pp. 193–218 (cit. on pp. 19, 87).

Jain, Anil (2010). "Data clustering: 50 years beyond K-means". In: *Pattern recognition letters* 31.8, pp. 651–666 (cit. on p. 13).

Jang, Eric, Shixiang Gu, and Ben Poole (2017). "Categorical reparameterization with gumbel-softmax". In: *Proc. International Conference on Learning Representations (ICLR)*. Toulon, France (cit. on p. 108).

Jansen, Aren, Manoj Plakal, Ratheet Pandya, Daniel PW Ellis, Shawn Hershey, Jiayang Liu, R Channing Moore, and Rif A Saurous (2018). "Unsupervised Learning of Semantic Audio Representations". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, AB, Canada, pp. 126–130 (cit. on p. 79).

Jansen, Aren, Samuel Thomas, and Hynek Hermansky (2013). "Weak top-down constraints for unsupervised acoustic model training." In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada, pp. 8091–8095 (cit. on p. 4).

Järvelin, Kalervo and Janaa Kekäläinen (2002). "Cumulated Gain-based Evaluation of IR Techniques". In: *ACM Transactions on Information Systems* 20.4, pp. 422–446 (cit. on p. 152).

Jati, Arindam and Panayiotis Georgiou (2017). "Speaker2Vec: Unsupervised Learning and Adaptation of a Speaker Manifold using Deep Neural Networks with an

Evaluation on Speaker Segmentation". In: *Proc. Interspeech*, pp. 3567–3571 (cit. on p. 79).

— (2019). "Neural predictive coding using convolutional neural networks toward unsupervised learning of speaker characteristics". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.10, pp. 1577–1589 (cit. on p. 80).

Jegou, Herve, Matthijs Douze, and Cordelia Schmid (2010). "Product quantization for nearest neighbor search". In: *IEEE transactions on pattern analysis and machine intelligence* 33.1, pp. 117–128 (cit. on p. 16).

Jelinek, Frederick and Robert L Mercer (1980). "Interpolated estimation of Markov source parameters from sparse data". In: *Proc. Workshop on Pattern Recognition in Practice, 1980*, pp. 381–397 (cit. on p. 54).

Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell (2014). "Caffe: Convolutional architecture for fast feature embedding". In: *Proc. 22nd ACM international conference on Multimedia*. ACM, pp. 675–678 (cit. on p. 29).

Jiampojamarn, Sittichai, Grzegorz Kondrak, and Tarek Sherif (2007). "Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion". In: *Proc. NAACL-HLT*. Rochester, New York, pp. 372–379 (cit. on p. 122).

Jin, Hubert, Francis Kubala, and Rich Schwartz (1997). "Automatic speaker clustering". In: *Proc. the DARPA speech recognition workshop*, pp. 108–111 (cit. on p. 80).

Jing, Longlong and Yingli Tian (2020). "Self-supervised visual feature learning with deep neural networks: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 43.11, pp. 4037–4058 (cit. on p. 16).

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2019). "Billion-scale similarity search with gpus". In: *IEEE Transactions on Big Data* 7.3, pp. 535–547 (cit. on p. 16).

Johnson, Melvin, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. (2017). "Googles multilingual neural machine translation system: Enabling zero-shot translation". In: *Transactions of the Association for Computational Linguistics* 5, pp. 339–351 (cit. on p. 122).

Joos, Martin (1948). "Acoustic phonetics". In: *Language* 24.2, pp. 5–136 (cit. on p. 4).

Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever (2015). "An empirical exploration of recurrent network architectures". In: *International Conference on Machine Learning*. Lille, France, pp. 2342–2350 (cit. on p. 39).

Juang, Biing-Hwang and Lawrence Rabiner (2005). "Automatic speech recognition–a brief history of the technology development". In: *Georgia Institute of Technology* 1, p. 67 (cit. on p. 43).

Jurafsky, Dan and James H Martin (2008). *Speech and language processing*. 2nd ed. ISBN: 0131873210 (cit. on pp. 12, 44–47, 50, 53).

— (2018). *Speech and language processing*. 3rd ed. draft (cit. on p. 21).

Kahn, Jacob, Morgane Rivière, Weiyi Zheng, Evgeny Kharitonov, Qiantong Xu, Pierre-Emmanuel Mazaré, Julien Karadayi, Vitaliy Liptchinsky, Ronan Collobert, Christian Fuegen, Tatiana Likhomanenko, Gabriel Synnaeve, Armand Joulin, Abdelrahman Mohamed, and Emmanuel Dupoux (2020). "Libri-Light: A Benchmark for ASR with Limited or No Supervision". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Virtual Barcelona, Spain, pp. 7669–7673 (cit. on pp. 100, 111, 113–116).

Kamper, Herman, Micha Elsner, Aren Jansen, and Sharon Goldwater (2015). "Unsupervised neural network based feature extraction using weak top-down constraints". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. South Brisbane, QL, Australia, pp. 5818–5822 (cit. on pp. 79, 99).

Kamper, Herman, Aren Jansen, Simon King, and Sharon Goldwater (2014). "Unsupervised lexical clustering of speech segments using fixed-dimensional acoustic embeddings". In: *Proc. Spoken Language Technology Workshop (SLT)*. South Lake Tahoe, NV, USA, pp. 100–105 (cit. on p. 98).

Kamper, Herman and Benjamin van Niekerk (2021). "Towards Unsupervised Phone and Word Segmentation Using Self-Supervised Vector-Quantized Neural Networks". In: *Proc. Interspeech*. Brno, Czech Republic, pp. 1539–1543 (cit. on p. 100).

Kamper, Herman, Weiran Wang, and Karen Livescu (2016). "Deep convolutional acoustic word embeddings using word-pair side information". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China, pp. 4950–4954 (cit. on p. 99).

Katagiri, Shigeru, Chin-hui Lee, and Biing-Hwang Juang (1991). "New discriminative training algorithms based on the generalized probabilistic descent method". In: *Proc. 1991 IEEE Workshop Neural Networks for Signal Processing*. Princeton, NJ, USA, pp. 299–308 (cit. on p. 50).

Katz, Slava (1987). "Estimation of probabilities from sparse data for the language model component of a speech recognizer". In: *IEEE transactions on acoustics, speech, and signal processing* 35.3, pp. 400–401 (cit. on p. 54).

Kaya, Heysem, Alexey Karpov, and Albert Ali Salah (2015). "Fisher vectors with cascaded normalization for paralinguistic analysis". In: *Proc. Interspeech*. Dresden, Germany, pp. 909–913 (cit. on pp. 73, 74).

Kim, Jaeyoung, Mostafa El-Khamy, and Jungwon Lee (2017). "Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition". In: *Proc. Interspeech*. Stockholm, Sweden, pp. 1591–1595 (cit. on p. 41).

Kim, Jangwon, Md Nasir, Rahul Gupta, Maarten Van Segbroeck, Daniel Bone, Matthew Black, Zisis Iason Skordilis, Zhaojun Yang, Panayiotis Georgiou, and Shrikanth Narayanan (2015). "Automatic estimation of Parkinson's disease severity from diverse speech tasks". In: *Proc. Interspeech*. Dresden, Germany, pp. 889–893 (cit. on p. 74).

Kingma, Diederik and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6 (cit. on pp. 30, 83, 84, 104, 125).

Kneser, Reinhard and Hermann Ney (1995). "Improved backing-off for m-gram language modeling". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Detroit, MI, USA, pp. 181–184 (cit. on pp. 92, 132).

Ko, Tom, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur (2015). "Audio augmentation for speech recognition". In: *Proc. Interspeech*. Dresden, Germany, pp. 3586–3589 (cit. on p. 86).

Krause, Ben, Iain Murray, Steve Renals, and Liang Lu (2017). "Multiplicative LSTM for sequence modelling". In: *Proc. ICLR Workshop track* (cit. on p. 39).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Hinton (2012). "ImageNet classification with deep convolutional neural networks". In: *Proc. Advances in Neural Information Processing Systems*. Lake Tahoe, CA, USA, pp. 1097–1105 (cit. on pp. 32, 67, 68).

Kuhl, Patricia (2004). "Early language acquisition: cracking the speech code". In: *Nature reviews neuroscience* 5.11, pp. 831–843 (cit. on p. 6).

Kullback, Solomon and Richard A Leibler (1951). "On information and sufficiency". In: *The annals of mathematical statistics* 22.1, pp. 79–86 (cit. on pp. 23, 103, 109).

Kürzinger, Ludwig, Dominik Winkelbauer, Lujun Li, Tobias Watzel, and Gerhard Rigoll (2020). "CTC-Segmentation of Large Corpora for German End-to-End Speech Recognition". In: *Proc. Speech and Computer: 22nd International Conference (SPECOM 2020)*. St. Petersburg, Russia, pp. 267–278 (cit. on p. 135).

Kvålseth, Tarald (2017). "On normalized mutual information: measure derivations and properties". In: *Entropy* 19.11, p. 631 (cit. on p. 19).

LeCun, Yann and Yoshua Bengio (1995). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361, pp. 255–258 (cit. on pp. 26, 67).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444 (cit. on p. 26).

LeCun, Yann, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel (1989). "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4, pp. 541–551 (cit. on p. 30).

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2323 (cit. on pp. 32, 67, 68).

LeCun, Yann, Léon Bottou, Genevieve Orr, and Klaus-Robert Müller (2012). "Efficient backprop". In: *Neural networks: Tricks of the trade*, pp. 9–48. ISBN: 978-3-642-35289-8 (cit. on p. 27).

Lee, Chia-Ying and James Glass (2012). "A nonparametric Bayesian approach to acoustic model discovery". In: *Proc. ACL*. Jeju Island, South Korea, pp. 40–49 (cit. on p. 99).

Lee, Honglak, Chaitanya Ekanadham, and Andrew Ng (2008). "Sparse deep belief net model for visual area V2". In: *Proc. Advances in Neural Information Processing Systems*. Vancouver, BC, Canada, pp. 873–880 (cit. on p. 36).

Lee, Honglak, Yan Largman, Peter Pham, and Andrew Ng (2009). "Unsupervised feature learning for audio classification using convolutional deep belief networks". In: *Proc. Advances in Neural Information Processing Systems*. Vancouver, BC, Canada, pp. 1096–1104 (cit. on p. 67).

Lenneberg, Eric (1967). "The biological foundations of language". In: *Hospital Practice* 2.12, pp. 59–67 (cit. on p. 1).

Li, Lantian, Yixiang Chen, Ying Shi, Zhiyuan Tang, and Dong Wang (2017). "Deep speaker feature learning for text-independent speaker verification". In: *Proc. Interspeech*. Stockholm, Sweden, pp. 1542–1545 (cit. on p. 79).

Lidstone, George James (1920). "Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities". In: *Transactions of the Faculty of Actuaries* 8.182-192, p. 13 (cit. on p. 53).

Lipton, Zachary, John Berkowitz, and Charles Elkan (2015). "A critical review of recurrent neural networks for sequence learning". In: *arXiv preprint arXiv:1506.00019* (cit. on p. 21).

Liu, Da-Rong, Kuan-Yu Chen, Hung yi Lee, and Lin shan Lee (2018). "Completely Unsupervised Phoneme Recognition by Adversarially Learning Mapping Relationships from Audio Embeddings". In: *Proc. Interspeech*. Hyderabad, India, pp. 3748–3752 (cit. on pp. 58, 160).

Liu, Fu-Hua, Richard M Stern, Xuedong Huang, and Alejandro Acero (1993). "Efficient cepstral normalization for robust speech recognition". In: *Proc. Workshop on Human Language Technology*. Princeton, NJ, USA, pp. 69–74 (cit. on p. 48).

Liu, Liyuan, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han (2020). "On the variance of the adaptive learning rate and beyond". In: *Proc. International Conference on Learning Representations (ICLR)*. Virtual, Paper 164 (cit. on p. 30).

Liu, Xiao, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang (2021). "Self-supervised learning: Generative or contrastive". In: *IEEE Transactions on Knowledge and Data Engineering (Early access)* (cit. on p. 16).

Liu, Zhiyuan, Wenyi Huang, Yabin Zheng, and Maosong Sun (2010). "Automatic Keyphrase Extraction via Topic Decomposition". In: *Proc. EMNLP*. Cambridge, MA, USA, pp. 366–376 (cit. on p. 148).

Lloyd, Stuart (1982). "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2, pp. 129–137 (cit. on p. 14).

Locascio, Nicholas, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay (2016). "Neural Generation of Regular Expressions from Natural Language with Minimal Domain Knowledge". In: *Proc. EMNLP*. Austin, Texas, pp. 1918–1923 (cit. on p. 121).

Lopez-Moreno, Ignacio, Javier Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pedro Moreno (2014). "Automatic language identification using deep neural networks". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. Florence, Italy, pp. 5337–5341 (cit. on p. 67).

Luong, Minh-Thang, Hieu Pham, and Christopher Manning (2015). "Effective approaches to attention-based neural machine translation". In: *Proc. Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal, pp. 1412–1421 (cit. on pp. 40, 101).

Maas, Andrew, Awni Hannun, and Andrew Ng (2013). "Rectifier nonlinearities improve neural network acoustic models". In: *ICML Workshop on Deep Learning for Audio, Speech and Language*. Vol. 30. Atlanta, GA, USA (cit. on p. 83).

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.11, pp. 2579–2605 (cit. on p. 91).

Makhoul, John (1975). "Spectral linear prediction: Properties and applications". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 23.3, pp. 283–296 (cit. on p. 49).

Mattys, Sven, Matthew Davis, Ann Bradlow, and Sophie Scott (2012). "Speech recognition in adverse conditions: A review". In: *Language and Cognitive Processes* 27.7-8, pp. 953–978 (cit. on p. 5).

McInnes, Leland, John Healy, and Steve Astels (2017). "hdbscan: Hierarchical density based clustering". In: *The Journal of Open Source Software* 2.11, p. 205 (cit. on pp. 16, 86, 90).

Mermelstein, Paul (1976). "Distance measures for speech recognition, psychological and instrumental". In: *Pattern recognition and artificial intelligence* 116, pp. 374–388 (cit. on p. 47).

Metze, Florian, Petra Gieselmann, Hartwig Holzapfel, Thomas Kluge, Ivica Rogina, Alex Waibel, Matthias Wölfel, James Crowley, Patrick Reignier, Dominique Vaufreydaz, François Bérard, Bérangère Cohen, Joelle Coutaz, Sylvie Rouillard, Victoria Arranz, Manuel Bertrán, and Horacio Rodríguez (2005). "The FAME Interactive Space". In: *Proc. International Workshop on Machine Learning for Multimodal Interaction*. Edinburgh, United Kingdom, pp. 126–137 (cit. on p. 146).

Miao, Yajie, Hao Zhang, and Florian Metze (2015). "Speaker adaptive training of deep neural network acoustic models using i-vectors". In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 23.11, pp. 1938–1949 (cit. on pp. 80, 91, 134).

Miikkulainen, Risto and Michael Dyer (1991). "Natural language processing with modular PDP networks and distributed lexicon". In: *Cognitive Science* 15.3, pp. 343–399 (cit. on p. 54).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Proc. Advances in Neural Information Processing Systems*. Lake Tahoe, NV, USA, pp. 3111–3119 (cit. on pp. 78, 100, 149).

Milde, Benjamin (2014). "Unsupervised acquisition of acoustic models for speech-to-text alignment". MA thesis. Technische Universität Darmstadt, Germany (cit. on p. 44).

Milde, Benjamin and Chris Biemann (2015). "Using Representation Learning and Out-of-domain Data for a Paralinguistic Speech Task". In: *Proc. Interspeech*. Dresden, Germany, pp. 904–908 (cit. on pp. 74–76).

— (2018). "Unspeech: Unsupervised Speech Context Embeddings". In: *Proc. Interspeech*. Hyderabad, India, pp. 2693–2697 (cit. on p. 102).

— (2019). "SparseSpeech: Unsupervised Acoustic Unit Discovery with Memory-Augmented Sequence Autoencoders". In: *Proc. Interspeech*. Graz, Austria, pp. 256–260 (cit. on pp. 101, 118).

— (2020). "Improving Unsupervised Sparsespeech Acoustic Models with Categorical Reparameterization". In: *Proc. Interspeech*. Virtual Shanghai, China, pp. 2747–2751 (cit. on p. 110).

Milde, Benjamin, Tim Fischer, Steffen Remus, and Chris Biemann (2021a). "MoM: Minutes of Meeting Bot". In: *Proc. Interspeech*. Brno, Czech Republic, pp. 3311–3312 (cit. on pp. 141, 142).

Milde, Benjamin, Robert Geislinger, Irina Lindt, and Timo Baumann (2021b). "Open source automatic lecture subtitling". In: *Proc. 32nd Conference on Electronical Speech Signal Processing (ESSV)*. Virtual Berlin, pp. 128–134 (cit. on p. 138).

Milde, Benjamin and Arne Köhn (2018). "Open Source Automatic Speech Recognition for German". In: *Proc. ITG 2018*. Oldenburg, pp. 251–255 (cit. on p. 135).

Milde, Benjamin, Christoph Schmidt, and Joachim Köhler (2017). "Multitask Sequence-to-Sequence Models for Grapheme-to-Phoneme Conversion". In: *Proc. Interspeech*. Stockholm, Sweden, pp. 2536–2540 (cit. on p. 132).

Milde, Benjamin, Jonas Wacker, Stefan Radomski, Max Mühlhäuser, and Chris Biemann (2016). "Ambient Search: A Document Retrieval System for Speech Streams". In: *Proc. COLING 2016*. Osaka, Japan, pp. 2082–2091 (cit. on p. 129).

Mitchell, Tom M. (1997). *Machine Learning*. MIT Press and the McGraw-Hill Company. ISBN: 978-0071154673 (cit. on pp. 11, 18, 21).

Mohri, Mehryar, Fernando Pereira, and Michael Riley (2002). "Weighted finite-state transducers in speech recognition". In: *Computer Speech & Language* 16.1, pp. 69–88 (cit. on p. 52).

Mullennix, John W, David B Pisoni, and Christopher S Martin (1989). "Some effects of talker variability on spoken word recognition". In: *The Journal of the Acoustical Society of America* 85.1, pp. 365–378 (cit. on p. 4).

Murphy, Kevin P. (2014). *Machine learning, a probabilistic perspective* (cit. on pp. 13, 30, 56).

Navarro, Gonzalo (2001). "A guided tour to approximate string matching". In: *ACM computing surveys (CSUR)* 33.1, pp. 31–88 (cit. on p. 52).

Nesterov, Yurii (1983). "A method of solving a convex programming problem with convergence rate O (1/k2)". In: *Soviet Mathematics Doklady* 27, pp. 372–376 (cit. on pp. 29, 69).

Ney, Hermann, Ute Essen, and Reinhard Kneser (1994). "On structuring probabilistic dependences in stochastic language modelling". In: *Computer Speech and Language* 8.1, pp. 1–38 (cit. on p. 54).

Ng, Andrew (2011). "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011, pp. 1–19 (cit. on p. 36).

Niekerk, Benjamin van, Leanne Nortje, and Herman Kamper (2020). "Vector-Quantized Neural Networks for Acoustic Unit Discovery in the ZeroSpeech 2020 Challenge". In: *Proc. Interspeech*. Virtual Shanghai, China, pp. 4836–4840 (cit. on p. 100).

Novak, Josef (2012). *Phonetisaurus*. URL: `https://github.com/AdolfVonKleist/Phonetisaurus` (cit. on p. 122).

Novak, Josef, Nobuaki Minematsu, and Keikichi Hirose (2013). "Failure Transitions for Joint N-gram Models and G2P Conversion." In: *Proc. Interspeech*. Lyon, France, pp. 1821–1825 (cit. on p. 122).

Nygaard, Lynne and David Pisoni (1998). "Talker-specific learning in speech perception". In: *Perception & psychophysics* 60.3, pp. 355–376 (cit. on p. 5).

Nyquist, Harry (1928). "Certain topics in telegraph transmission theory". In: *Transactions of the American Institute of Electrical Engineers* 47.2, pp. 617–644 (cit. on p. 45).

Oord, Aaron van den, Oriol Vinyals, et al. (2017). "Neural discrete representation learning". In: *Proc. Advances in Neural Information Processing Systems*. Long Beach, CA, USA, pp. 6306–6315 (cit. on p. 99).

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (cit. on pp. 59, 99, 113, 115, 116).

Oppenheim, Alan and Ronald Schafer (1989). *Discrete-time signal processing*. 2nd ed. Prentice-Hall. ISBN: 978-0-13-754920-7 (cit. on p. 46).

Oshika, Beatrice, Victor Zue, Rollin Weeks, Helene Neu, and Joseph Aurbach (1975). "The role of phonological rules in speech understanding research". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 23.1, pp. 104–112 (cit. on p. 50).

Oualil, Youssef, Dietrich Klakow, Gyorgy Szaszák, Ajay Srinivasamurthy, Hartmut Helmke, and Petr Motlicek (2017). "A context-aware speech recognition and understanding system for air traffic control domain". In: *Proc. ASRU*. Okinawa, Japan, pp. 404–408 (cit. on p. 129).

Pan, Sinno Jialin and Qiang Yang (2009). "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359 (cit. on pp. 17, 66).

Panayotov, Vassil, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur (2015). "Librispeech: An ASR Corpus Based On Public Domain Audio Books". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, QL, Australia, pp. 5206–5210 (cit. on pp. 104, 106, 114, 122).

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Proc. Advances in Neural Information Processing Systems*. Vol. 32. Vancouver, BC, Canada (cit. on p. 117).

Pathak, Deepak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros (2016). "Context encoders: Feature learning by inpainting". In: *Proc. Computer Vision and Pattern Recognition (CVPR)*. Caesars Palace, NV, USA, pp. 2536–2544 (cit. on p. 79).

Peddinti, Vijayaditya, Daniel Povey, and Sanjeev Khudanpur (2015). "A time delay neural network architecture for efficient modeling of long temporal contexts". In: *Proc. Interspeech*. Dresden, Germany, pp. 3214–3218 (cit. on pp. 56, 78, 134).

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay (2012). "Scikit-learn: Machine Learning in Python". In: *The Journal of Machine Learning Research* 12, pp. 2825–2830 (cit. on pp. 71, 104).

Pellegrini, Thomas (2015). "Comparing SVM, Softmax, and shallow neural networks for eating condition classification". In: *Proc. Interspeech*. Dresden, Germany, pp. 899–903 (cit. on p. 74).

Pellegrini, Thomas, Céline Manenti, and Julien Pinquier (2017). *Technical Report: The IRIT-UPS system at ZeroSpeech 2017 Track1: Unsupervised Subword Modeling*. IRIT, Université de Toulouse (cit. on p. 106).

Peterson, Gordon E and Harold L Barney (1952). "Control methods used in a study of the vowels". In: *The Journal of the acoustical society of America* 24.2, pp. 175–184 (cit. on p. 4).

Pfeiffer, Silvia (Apr. 2019). *WebVTT: The Web Video Text Tracks Format*. Candidate Recommendation. W3C. URL: `https://www.w3.org/TR/2019/CR-webvtt1-20190404/` (cit. on p. 137).

Pierce, John (1969). "Whither speech recognition?" In: *The journal of the acoustical society of america* 46.4B, pp. 1049–1051 (cit. on p. 43).

Pir, Dara and Theodore Brown (2015). "Acoustic group feature selection using wrapper method for automatic eating condition recognition". In: *Proc. Interspeech*. Dresden, Germany, pp. 894–898 (cit. on p. 74).

Plakal, Manoj and Dan Ellis (2020). *YAMNet*. `https://github.com/tensorflow/models/tree/master/research/audioset/yamnet` (cit. on pp. 80, 95).

Popescu-Belis, Andrei, Jonathan Kilgour, Peter Poller, Alexandre Nanchen, Erik Boertjes, and Joost de De Wit (2000). "Automatic Content Linking: Speech-based Just-in-time Retrieval for Multimedia Archives". In: *Proc. SIGIR*. Athens, Greece, p. 703 (cit. on p. 146).

Povey, Daniel, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely (2011). "The Kaldi speech recognition toolkit". In: *Proc. ASRU*. Hilton Waikoloa Village, Big Island, HI, US (cit. on pp. 52, 83, 85, 104, 110, 147).

Povey, Daniel, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur (2016). "Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI." In: *Proc. Interspeech*. San Fransisco, CA, USA, pp. 2751–2755 (cit. on pp. 88, 134).

Povey, Daniel, Xiaohui Zhang, and Sanjeev Khudanpur (2014). "Parallel training of DNNs with natural gradient and parameter averaging". In: *arXiv preprint arXiv:1410.7455* (cit. on p. 88).

Prasad, Abhay and Prasanta Kumar Ghosh (2015). "Automatic classification of eating conditions from speech using acoustic feature selection and a set of hierarchical support vector machine classifiers". In: *Proc. Interspeech*. Dresden, Germany, pp. 884–888 (cit. on p. 74).

Pratt, Lorien (1992). "Discriminability-based transfer between neural networks". In: *Proc. Advances in Neural Information Processing Systems*. Denver, CO, USA, pp. 204–211 (cit. on p. 69).

Pratt, Lorien, Jack Mostow, Candace Kamm, and Ace Kamm (1991). "Direct Transfer of Learned Information Among Neural Networks". In: *AAAI*. Vol. 91, pp. 584–589 (cit. on p. 69).

Pritzen, Julia, Michael Gref, Dietlind Zühlke, and Christoph Schmidt (2021). "Multitask Learning for Grapheme-to-Phoneme Conversion of Anglicisms in German Speech Recognition". In: *arXiv preprint arXiv:2105.12708* (cit. on p. 122).

Rabiner, Lawrence (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2, pp. 257–286 (cit. on pp. 43, 49).

Radeck-Arneth, Stephan, Benjamin Milde, Arvid Lange, Evandro Gouvêa, Stefan Radomski, Max Mühlhäuser, and Chris Biemann (2015). "Open source german distant speech recognition: Corpus and acoustic model". In: *Proc. Text, Speech, and Dialogue (TSD)*. Pilsen, Czech Republic, pp. 480–488 (cit. on pp. 130, 132, 134, 135).

Rand, William (1971). "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical association* 66.336, pp. 846–850 (cit. on p. 19).

Rao, Kanishka, Fuchun Peng, Hasim Sak, and Françoise Beaufays (2015). "Grapheme-to-phoneme Conversion using Long Short-term Memory Recurrent Neural Networks". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, QL, Australia, pp. 4225–4229 (cit. on p. 122).

Rasmus, Antti, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko (2015). "Semi-supervised learning with ladder networks". In: *Proc. Advances in Neural Information Processing Systems*, pp. 3546–3554 (cit. on p. 17).

Řehůřek, Radim and Petr Sojka (2010). "Software Framework for Topic Modelling with Large Corpora". In: *Proc. LREC*. Valletta, Malta, pp. 45–50 (cit. on p. 151).

Remus, Steffen and Chris Biemann (2013). "Three knowledge-free methods for automatic lexical chain extraction". In: *Proc. 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, GE, USA, pp. 989–999 (cit. on p. 20).

Renshaw, Daniel, Herman Kamper, Aren Jansen, and Sharon Goldwater (2015). "A comparison of neural network methods for unsupervised representation learning on the zero resource speech challenge". In: *Proc. Interspeech*. Dresden, Germany, pp. 3199–3203 (cit. on p. 99).

Rhodes, Bradley (1997). "The Wearable Remembrance Agent: A System for Augmented Memory". In: *Personal and Ubiquitous Computing* 1.4, pp. 218–224 (cit. on p. 146).

Rhodes, Bradley and Pattie Maes (2000). "Just-in-time Information Retrieval Agents". In: *IBM Systems Journal* 39.3, p. 686 (cit. on p. 146).

Rhodes, Bradley and Thad Starner (1996). "Remembrance Agent: A Continuously Running Automated Information Retrieval System". In: *Proc. Practical Application Of Intelligent Agents and Multi Agent Technology*, pp. 487–495 (cit. on p. 146).

Riedl, Martin and Chris Biemann (2015). "A Single Word is not Enough: Ranking Multiword Expressions Using Distributional Semantics". In: *Proc. EMNLP*. Lisbon, Portugal, pp. 2430–2440 (cit. on p. 148).

Rifai, Salah, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio (2011). "Contractive auto-encoders: Explicit invariance during feature extraction". In: *The International Conference on Machine Learning (ICML)*. Bellevue, Washington, USA, pp. 833–840 (cit. on pp. 35, 36).

Riloff, Ellen, Janyce Wiebe, and Theresa Wilson (2003). "Learning subjective nouns using extraction pattern bootstrapping". In: *Proc. ACL*. Sapporo, Japan, pp. 25–32 (cit. on p. 17).

Robinds, Herbert and Sutton Monro (1951). "A stochastic approximation method". In: *Annals of Mathematical Statistics* 22, pp. 400–407 (cit. on p. 25).

Rosenberg, Andrew and Julia Hirschberg (2007). "V-measure: A conditional entropy-based external cluster evaluation measure". In: *Proc. EMNLP-CoNLL*. Prague, Czech Republic, pp. 410–420 (cit. on p. 20).

Rousseau, Anthony, Paul Deléglise, and Yannick Estève (2014). "Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks". In: *Proc. LREC*. Reykjavik, Iceland, pp. 3935–3939 (cit. on pp. 61, 85, 134, 147).

Roy, Brandon Cain and Deb K Roy (2009). "Fast transcription of unstructured audio recordings". In: *Proc. Interspeech*. Brighton, UK, pp. 1647–1650 (cit. on p. 129).

Rumelhart, David, Geoffrey Hinton, and Ronald Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, p. 533 (cit. on pp. 21, 24, 35).

Rumelhart, David, James McClelland, and the PDP Research Group (1987). *Parallel distributed processing*. Vol. 1. MIT Press Cambridge, MA. ISBN: 9780262181204 (cit. on p. 26).

Sainath, Tara, Abdel Rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran (2013). "Deep convolutional neural networks for LVCSR". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada, pp. 8614–8618 (cit. on p. 67).

Saon, George, Hagen Soltau, David Nahamoo, and Michael Picheny (2013). "Speaker adaptation of neural network acoustic models using i-vectors." In: *ASRU*. Olomouc, Czech Republic, pp. 55–59 (cit. on pp. 4, 78, 80, 91, 134).

Schatz, Thomas (2016). "ABX-discriminability measures and applications". PhD thesis. Université Paris 6 (UPMC) (cit. on pp. 99, 105, 111).

Schatz, Thomas, Vijayaditya Peddinti, Francis Bach, Aren Jansen, Hynek Hermansky, and Emmanuel Dupoux (2013). "Evaluating speech features with the minimal-pair ABX task: Analysis of the classical MFC/PLP pipeline". In: *Proc. Interspeech*. Lyon, France, pp. 1781–1785 (cit. on pp. 99, 105, 111).

Schiel, Florian (1997). "The Bavarian Archive for Speech Signals: Resources for the Speech Community". In: *Proc. Eurospeech*. Rhodes, Greece, pp. 1687–1690 (cit. on p. 123).

— (2011). "Perception of alcoholic intoxication in speech". In: *Proc. Interspeech*. Florence, Italy, pp. 3281–3284 (cit. on p. 66).

Schmaltz, Allen, Yoon Kim, Alexander M Rush, and Stuart M Shieber (2016). "Sentence-level Grammatical Error Identification as Sequence-to-sequence Correction". In: *NAACL HLT*. San Diego, CA, USA, pp. 242–251 (cit. on p. 121).

Schneider, Steffen, Alexei Baevski, Ronan Collobert, and Michael Auli (2019). "wav2vec: Unsupervised Pre-Training for Speech Recognition". In: *Proc. Interspeech*. Graz, Austria, pp. 3465–3469 (cit. on pp. 58, 100).

Schnelle-Walka, Dirk, Stephan Radeck-Arneth, Chris Biemann, and Stefan Radomski (2014). "An open source corpus and recording software for distant speech recognition with the microsoft kinect". In: *Proc. ITG*. Erlangen, Germany (cit. on p. 134).

Schnober, Carsten, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych (2016). "Still not there? Comparing Traditional Sequence-to-Sequence Models to Encoder-Decoder Neural Networks on Monotone String Translation Tasks". In: *Proc. COLING*. Osaka, Japan, pp. 1703–1714 (cit. on p. 122).

Schröder, Marc and Jürgen Trouvain (2003). "The German text-to-speech synthesis system MARY: A tool for research, development and teaching". In: *International Journal of Speech Technology* 6.4, pp. 365–377 (cit. on pp. 131, 133).

Schroeder, Manfred (1977). "Recognition of Complex Acoustic Signals, Life Sciences Research Report 5". In: *Dahlem Konferenzen*, pp. 323–328 (cit. on p. 48).

Schuller, Björn (2012). "The Computational Paralinguistics Challenge [Social Sciences]". In: *Signal Processing Magazine, IEEE* 29.4, pp. 97–101 (cit. on p. 66).

Schuller, Björn, Stefan Steidl, and Anton Batliner (2009). "The INTERSPEECH 2009 emotion challenge". In: *Proc. Interspeech*. Brighton, UK, pp. 312–315 (cit. on p. 66).

Schuller, Björn, Stefan Steidl, Anton Batliner, Felix Burkhardt, Laurence Devillers, Christian Müller, and Shrikanth Narayanan (2010). "The INTERSPEECH 2010 Paralinguistic Challenge". In: *Proc. Interspeech*. Makuhari, Chiba, Japan, pp. 2794–2797 (cit. on p. 66).

Schuller, Björn, Stefan Steidl, Anton Batliner, Simone Hantke, Florian Hönig, Juan Rafael Orozco-Arroyave, Elmar Nöth, Yue Zhang, and Felix Weniger (2015). "The INTERSPEECH 2015 Computional Paralinguistics Challenge: Nativeness, Parkinson's and Eating Condition". In: *Proc. Interspeech*. Dresden, Germany, pp. 478–482 (cit. on pp. 66, 71, 72, 74).

Schuster, Mike and Kuldip K Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681 (cit. on pp. 39, 40, 100).

Senior, Andrew and Ignacio Lopez-Moreno (2014). "Improving DNN speaker independence with i-vector inputs". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy, pp. 225–229 (cit. on pp. 80, 134).

Shannon, Claude Elwood (1948). "A mathematical theory of communication". In: *Bell system technical journal* 27.3, pp. 379–423 (cit. on p. 19).

Sharif Razavian, Ali, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson (2014). "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proc. IEEE conference on computer vision and pattern recognition workshops*. Columbus, OH, USA, pp. 806–813 (cit. on p. 69).

Shor, Joel, Aren Jansen, Ronnie Maor, Oran Lang, Omry Tuval, Félix de Chaumont Quitry, Marco Tagliasacchi, Ira Shavitt, Dotan Emanuel, and Yinnon Haviv (2020). "Towards Learning a Universal Non-Semantic Representation of Speech". In: *Proc. Interspeech*. Virtual Shangai, China, pp. 140–144 (cit. on pp. 80, 94, 95).

Simard, Patrice, Dave Steinkraus, and John Platt (2003). "Best practices for convolutional neural networks applied to visual document analysis". In: *Seventh International Conference on Document Analysis and Recognition*, p. 958 (cit. on p. 68).

Simonyan, Karen and Andrew Zisserman (2015). "Very deep convolutional networks for large-scale image recognition". In: *Proc. International Conference on Learning Representations (ICLR)*. San Diego, CA, USA (cit. on pp. 32, 68, 83, 84).

Siu, Man-hung, Herbert Gish, Steve Lowe, and Arthur Chan (2011). "Unsupervised audio patterns discovery using HMM-based self-organized units". In: *Proc. Interspeech*. Florence, Italy, pp. 2333–2336 (cit. on p. 98).

Smit, Peter, Sami Virpioja, Mikko Kurimo, et al. (2017). "Improved Subword Modeling for WFST-Based Speech Recognition." In: *Proc. Interspeech*. Stockholm, Sweden, pp. 2551–2555 (cit. on pp. 44, 143).

Snyder, David, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur (2017). "Deep Neural Network Embeddings for Text-Independent Speaker Verification". In: *Proc. Interspeech*. Stockholm, Sweden, pp. 999–1003 (cit. on p. 79).

Snyder, David, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur (2016). "Deep neural network-based speaker embeddings for end-to-end speaker verification". In: *Proc. Spoken Language Technology Workshop (SLT)*. San Diego, CA, USA, pp. 165–170 (cit. on p. 79).

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1, pp. 1929–1958 (cit. on pp. 30, 68).

Stadtschnitzer, Michael, Jochen Schwenninger, Daniel Stein, and Joachim Köhler (2014). "Exploiting the large-scale German Broadcast Corpus to boost the Fraunhofer IAIS Speech Recognition System." In: *Proc. LREC*. Reykjavik, Iceland, pp. 3887–3890 (cit. on p. 88).

Stevens, Stanley Smith, John Volkmann, and Edwin Newman (1937). "A scale for the measurement of the psychological magnitude pitch". In: *The Journal of the Acoustical Society of America* 8.3, pp. 185–190 (cit. on p. 47).

Stone, Mervyn (1974). "Cross-validatory choice and assessment of statistical predictions". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 36.2, pp. 111–133 (cit. on p. 18).

Strehl, Alexander (2002). "Relationship-based clustering and cluster ensembles for high-dimensional data mining". PhD thesis. University Of Texas at Austin (cit. on pp. 20, 87).

Sukhbaatar, Sainbayar, Arthur Szlam, Jason Weston, and Rob Fergus (2015). "End-to-end memory networks". In: *Proc. Advances in Neural Information Processing Systems*. Montreal, Canada, pp. 2440–2448 (cit. on p. 101).

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton (2013). "On the importance of initialization and momentum in deep learning". In: *30th International Conference on Machine Learning (ICML)*. Atlanta, GA, USA, pp. 1139–1147 (cit. on pp. 29, 69, 70).

Sutskever, Ilya, Oriol Vinyals, and Quoc Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Proc. Advances in Neural Information Processing Systems*. Montreal, Canada, pp. 3104–3112 (cit. on pp. 40, 121).

Synnaeve, Gabriel and Emmanuel Dupoux (2014). "Weakly supervised multi-embeddings learning of acoustic models". In: *arXiv preprint arXiv:1412.6645* (cit. on p. 79).

Taigman, Yaniv, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf (2014). "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. Columbus, OH, USA, p. 8 (cit. on p. 67).

Tieleman, Tijmen and Geoffrey Hinton (2012). "Lecture 6.5-RMSProp, Coursera: Neural networks for machine learning". In: *University of Toronto, Technical Report* (cit. on p. 30).

Tilk, Ottokar and Tanel Alumäe (2016). "Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration". In: *Proc. Interspeech*. San Francisco, California, USA, pp. 3047–3051 (cit. on p. 138).

Tjandra, Andros, Berrak Sisman, Mingyang Zhang, Sakriani Sakti, Haizhou Li, and Satoshi Nakamura (2019). "VQVAE Unsupervised Unit Discovery and Multi-Scale Code2Spec Inverter for Zerospeech Challenge 2019". In: *Proc. Interspeech*. Graz, Austria, pp. 1118–1122 (cit. on p. 99).

Trager, George (1958). "Paralanguage: A first approximation". In: *Studies in linguistics* 13.1-2, pp. 1–12 (cit. on p. 66).

Traum, David, Priti Aggarwal, Ron Artstein, Susan Foutz, Jillian Gerten, Athanasios Katsamanis, Anton Leuski, Dan Noren, and William Swartout (2012). "Ada and Grace: Direct interaction with museum visitors". In: *Proc. International conference on intelligent virtual agents*. Santa Cruz, CA, USA, pp. 245–251 (cit. on p. 146).

Trigeorgis, George, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A Nicolaou, Björn Schuller, and Stefanos Zafeiriou (2016). "Adieu features? End-to-end speech emotion recognition using a deep convolutional recurrent network". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China, pp. 5200–5204 (cit. on p. 75).

Tsvetkov, Yulia, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer (2016). "Polyglot Neural Language Models: A Case Study in Cross-Lingual Phonetic Representation Learning". In: *Proc. NAACL*. San Diego, CA, USA, pp. 1357–1366 (cit. on p. 122).

Turing, Alan (1950). "Computing Machinery and Intelligence". In: *Mind* 59.236, p. 433 (cit. on p. 20).

Versteegh, Maarten, Roland Thiolliere, Thomas Schatz, Xuan Nga Cao, Xavier Anguera, Aren Jansen, and Emmanuel Dupoux (2015). "The Zero Resource Speech Challenge 2015". In: *Proc. Interspeech*. Dresden, Germany, pp. 3169–3173 (cit. on p. 99).

Veselỳ, Karel, Mirko Hannemann, and Lukáš Burget (2013). "Semi-supervised training of deep neural networks". In: *Proc. Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Olomouc, Czech Republic, pp. 267–272 (cit. on p. 17).

Veselỳ, Karel, Shinji Watanabe, Katerina Žmolíková, Martin Karafiát, Lukáš Burget, and Jan Honza Černockỳ (2016). "Sequence summarizing neural network for speaker adaptation". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China, pp. 5315–5319 (cit. on pp. 80, 103).

Viikki, Olli and Kari Laurila (1998). "Cepstral domain segmental feature vector normalization for noise robust speech recognition". In: *Speech Communication* 25.1-3, pp. 133–147 (cit. on p. 48).

Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: *Journal of machine learning research* 11.12 (cit. on p. 36).

Vintsyuk, Taras K (1968). "Speech discrimination by dynamic programming". In: *Cybernetics and Systems Analysis* 4.1, pp. 52–57 (cit. on p. 106).

Vinyals, Oriol and Quoc Le (2015). "A Neural Conversation Model". In: *Proc. of ICML Deep Learning Workshop*. Lille, France (cit. on p. 121).

Viterbi, Andrew (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE transactions on Information Theory* 13.2, pp. 260–269 (cit. on pp. 50, 51).

Wagner, Johannes, Andreas Seiderer, Florian Lingenfelser, and Elisabeth André (2015). "Combining hierarchical classification with frequency weighting for the recognition of eating conditions". In: *Proc. Interspeech*. Dresden, Germany, pp. 889–893 (cit. on pp. 73, 74).

Wagner, Robert and Michael Fischer (1974). "The string-to-string correction problem". In: *Journal of the ACM (JACM)* 21.1, pp. 168–173 (cit. on p. 52).

Wahlster, Wolfgang (2000). *Verbmobil: foundations of speech-to-speech translation*. Springer-Verlag Berlin/Heidelberg. ISBN: 9783642087301 (cit. on p. 136).

Waibel, Alexander, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin Lang (1990). "Phoneme recognition using time-delay neural networks". In: *Readings in speech recognition*, pp. 393–404 (cit. on pp. 56, 78, 134).

Wang, Weiran, Qingming Tang, and Karen Livescu (2020). "Unsupervised Pre-training of Bidirectional Speech Encoders via Masked Reconstruction". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Virtual Barcelona, Spain, pp. 6889–6893 (cit. on p. 99).

Warden, Pete (2018). "Speech commands: A dataset for limited-vocabulary speech recognition". In: *arXiv preprint arXiv:1804.03209* (cit. on p. 94).

Weide, Robert (2005). *The Carnegie Mellon Pronouncing Dictionary [cmudict. 0.7b]* (cit. on p. 61).

Wells, John, William Barry, Martine Grice, Adrian Fourcin, and Dafydd Gibbon (1992). "Standard computer-compatible transcription". In: *Esprit project 2589 (SAM), Doc. no. SAM-UCL 37* (cit. on p. 61).

Weng, John, Narendra Ahuja, and Thomas Huang (1993). "Learning recognition and segmentation of 3-D objects from 2-D images". In: *Proc. Fourth International Conference on Computer Vision*. Berlin, Germany, pp. 121–128 (cit. on p. 31).

Werbos, Paul (1982). "Applications of advances in nonlinear sensitivity analysis". In: pp. 762–770 (cit. on p. 24).

— (1988). "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural networks* 1.4, pp. 339–356 (cit. on p. 37).

Whitehurst, Grover and Ross Vasta (1975). "Is language acquired through imitation?" In: *Journal of Psycholinguistic Research* 4.1, pp. 37–59 (cit. on p. 6).

Williams, Ronald J and David Zipser (1995). "Gradient-based learning algorithms for recurrent networks and their computational complexity". In: *Backpropagation: Theory, architectures, and applications* 1, pp. 433–486 (cit. on p. 37).

Williams, Will, Niranjani Prasad, David Mrva, Tom Ash, and Tony Robinson (2015). "Scaling Recurrent Neural Network Language Models". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Brisbane, QL, Australia, pp. 5391–5395 (cit. on p. 148).

Wirth, Johannes and Rene Peinl (2022). "ASR in German: A Detailed Error Analysis". In: *arXiv preprint arXiv:2204.05617* (cit. on pp. 135, 136).

Witten, Ian, Eibe Frank, Mark Hall, and Christopher Pal (2016). *Data Mining: Practical machine learning tools and techniques*. 3rd ed. Morgan Kaufmann. ISBN: 978-0123748560 (cit. on p. 18).

Wölfel, Matthias and John McDonough (2009). *Distant speech recognition*. John Wiley & Sons. ISBN: 978-0-470-51704-8 (cit. on p. 5).

Wu, Bin, Sakriani Sakti, Jinsong Zhang, and Satoshi Nakamura (2018). "Optimizing DPGMM Clustering in Zero-Resource Setting based on Functional Load". In: *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages*. Gurugram, India, pp. 1–5 (cit. on p. 108).

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean (2016a). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *arXiv preprint arXiv:1609.08144* (cit. on p. 122).

Wu, Yuhuai, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Russ R Salakhutdinov (2016b). "On multiplicative integration with recurrent neural networks". In: *Proc. Advances in Neural Information Processing Systems*. Barcelona, spain, pp. 2856–2864 (cit. on p. 39).

Xu, Haihua, Daniel Povey, Lidia Mangu, and Jie Zhu (2011). "Minimum Bayes risk decoding and system combination based on a recursion for edit distance". In: *Computer Speech & Language* 25.4, pp. 802–828 (cit. on p. 141).

Xu, Hainan, Tongfei Chen, Dongji Gao, Yiming Wang, Ke Li, Nagendra Goel, Yishay Carmiel, Daniel Povey, and Sanjeev Khudanpur (2018a). "A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition". In: *Proc. acoustics, speech and signal processing (ICASSP)*. Calgary, AB, Canada, pp. 5929–5933 (cit. on p. 135).

Xu, Hainan, Ke Li, Yiming Wang, Jian Wang, Shiyin Kang, Xie Chen, Daniel Povey, and Sanjeev Khudanpur (2018b). "Neural network language modeling with letter-based features and importance sampling". In: *Proc. Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, Alberta, Canada (cit. on p. 133).

Yan, S (2015). "Understanding LSTM networks". In: *Online. Accessed on November 2020.* URL: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/` (cit. on pp. 37, 38).

Yao, Kaisheng and Geoffrey Zweig (2015). "Sequence-to-sequence Neural Net Models for Grapheme-to-phoneme Conversion". In: *arXiv preprint arXiv:1506.00196* (cit. on pp. 121, 122).

Yarowsky, David (1995). "Unsupervised word sense disambiguation rivaling supervised methods". In: *Proc. ACL*. MIT, Cambridge, Massachusetts, pp. 189–196 (cit. on p. 17).

Young, Steve, Julian Odell, and Phil Woodland (1994). "Tree-based state tying for high accuracy modelling". In: *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, pp. 307–312 (cit. on p. 51).

Zeghidour, Neil, Gabriel Synnaeve, Nicolas Usunier, and Emmanuel Dupoux (2016). "Joint Learning of Speaker and Phonetic Similarities with Siamese Networks". In: *Proc. Interspeech*. San Francisco, CA, USA, pp. 1295–1299 (cit. on p. 79).

Zeiler, Matthew (2012). "ADADELTA: An Adaptive Learning Rate Method". In: *arXiv preprint arXiv:1212.5701* (cit. on p. 29).

Zhang, Yaodong and James R Glass (2009). "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams". In: *Proc. Automatic Speech Recognition & Understanding (ASRU)*. Merano/Meran, Italy, pp. 398–403 (cit. on pp. 78, 98).

Zhou, Bowen and John Hansen (2000). "Unsupervised audio stream segmentation and clustering via the Bayesian information criterion". In: *International Conference on Spoken Language Processing (ICSLP)*. Beijing, China, pp. 714–717 (cit. on p. 87).

Zhou, Guo-Bing, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou (2016). "Minimal gated unit for recurrent neural networks". In: *International Journal of Automation and Computing* 13.3, pp. 226–234 (cit. on p. 39).

Zhou, Yi-Tong and Rama Chellappa (1988). "Computation of optical flow using a neural network". In: *Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000)*, pp. 71–78 (cit. on pp. 30, 31).

Zhu, Xiaojin and Zoubin Ghahramani (2002). "Learning from labeled and unlabeled data with label propagation". In: (cit. on p. 17).

Zhu, Xiaojin Jerry (2005). *Semi-supervised learning literature survey*. Tech. rep. 1530. Computer Sciences, University of Wisconsin-Madison (cit. on p. 17).

Zilly, Julian Georg, Rupesh Kumar Srivastava, Jan Koutnık, and Jürgen Schmidhuber (2017). "Recurrent highway networks". In: *International Conference on Machine Learning (ICML)*. Sydney, Australia, pp. 4189–4198 (cit. on p. 39).

Zipf, George Kingsley (1929). "Relative frequency as a determinant of phonetic change". In: *Harvard studies in classical philology* 40, pp. 1–95 (cit. on p. 60).

Appendix

Several model training scripts and speech applications where developed as part of this thesis. They have been released as open source software, to make the experiments conducted in this thesis reproducible. This appendix provides an overview over the project repositories:

Chapter 4: Source code to replicate the results of eating condition classification can be found at `https://github.com/bmilde/deepschmatzing`. This system was submitted to the Interspeech 2015 Computational Paralinguistics challenge.

Chapter 5: Training code for Unspeech - Unsupervised Speech Context Embeddings can be found at `https://gitlab.com/milde/unspeech`. For further usage instructions and pre-trained models see also `https://unspeech.net/`.

Chapter 6: Training code for the Sparsespeech model can be found at `https://gitlab.com/milde/sparsespeech` (TensorFlow reference implementation) and `https://gitlab.com/milde/sparsespeech2` (end-to-end PyTorch reimplementation). Furthermore, contributions were also made to `https://github.com/facebookresearch/libri-light`, to evaluate Sparsespeech models with Kullback-Leibler divergence as local distance function in the ABX error measure.

Chapter 7: Scripts to reproduce the IPA dictionary generation are available at `https://github.com/bmilde/wiktionary_ipa_phoneme_lexicons`.

Chapter 8: At `https://github.com/uhh-lt/kaldi-tuda-de` we published all scripts and resources to train German Kaldi ASR models on about 1,720h of speech data as well as pre trained models.

`https://github.com/bmilde/german-asr-lm-tools` is a sub repository, used to clean and normalize German texts crawled from the internet for language model training for ASR. Kaldi-model-server can be found at `https://github.com/uhh-lt/kaldi-model-server` and can be used for online decoding with these models. Contributions were also made to the PyKaldi project at `https://github.com/pykaldi/pykaldi`.

The automatic subtitling software for German discussed in this chapter is available at `https://github.com/uhh-lt/subtitle2go`. The prototype for automatic meeting transcription and summarization can be found at `https://github.`

com/uhh-lt/MeetingBot with instructions on how to install and run the software.

Bbb-live-subtitles, a plugin for automatic subtitling in BBB briefly discussed in this Chapter as well, can be found at:
https://github.com/uhh-lt/bbb-live-subtitles.

Chapter 9: Ambient search, the open source system for displaying and retrieving relevant documents in real time for speech input can be found at:
https://github.com/bmilde/ambientsearch.

All links have been last accessed in April 2022.