Deep Neural Networks for Automatic Lyrics Transcription

Emir Demirel

Submitted in fulfillment of the requirements

of the Degree of Doctor of Philosophy

School of Electronic Engineering and Computer Science

Queen Mary University of London

May 2022

Statement of originality

I, Emir Demirel, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature:

Date: 17.02.2021

Details of collaboration and publications:
See Section 1.4

This dissertation has been carried out at the Centre for Digital Music research lab in Queen Mary University of London, UK, from November 2018 to May, 2022. It has been supervised by Prof. Simon Dixon and Prof. Sven Ahlbäck. This research was funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068.

Abstract

Applications that aim to transcribe singing performances automatically into music notation are mostly concerned with representing the singing voice in terms of pitch over time. Little research has focused on the automatic transcription of lyrics and their representation alongside music notes on staff notation. Song lyrics are one of the core building blocks of singing performances and an essential part of the music listening experience. Thus, the automatic retrieval of song lyrics has a huge potential for impact across many applications such as songwriting tools, audio/video captioning, karaoke applications, music catalogue creation, music recommendation, playlist generation and royalty forecasting. This study formalises lyrics transcription by means of Large Vocabulary Continuous Speech Recognition (LVCSR) from the singing voice and aims to develop an automatic lyrics transcription system that has a robust and scalable performance across varying domains. In particular, challenges and opportunities within two major paradigms of Deep Neural Network (DNN)-based LVCSR systems are investigated: the first one is the hybrid DNN - Hidden Markov Model (DNN-HMM) trained on the Lattice-free Maximum Mutual Information (LFMMI) objective. For the DNN-HMM framework, a number of novel methods are proposed: using a lyrics-specific corpus for building the language model, a singing-adapted pronunciation dictionary for modelling common pronunciation variants in singing, a compact multistream neural network architecture to enhance performance against noisy environments, and a cross-domain acoustic model with music informed silence modelling. The second approach to lyrics transcription focuses on the end-to-end models, where transfer learning is applied via fine tuning a pretrained speech recognition model on singing data. Specifically, the end-to-end model has a transformer architecture and is trained on the hybrid CTC / Attention objective.

Some of the key findings of this study can be summarised as follows: song lyrics have diverse lexical and prosodic characteristics to spontaneous or read speech, thus a lyrics-specific language model often performs better than one trained on speech transcriptions. Secondly, adapting a pronunciation dictionary to singing data leads to consistent but modest improvements. The multistream neural network architecture for the acoustic model leads to an improved word error rate and faster inference performance compared to its single stream counterpart. Domain invariant transcription performance can be achieved by including both monophonic and polyphonic recordings during training. Using distinct target class labels for non-vocal silent and music instances helps improve lyrics transcription rates on polyphonic recordings. The final DNN-HMM model achieves the state-of-the-art by a considerable margin through combining the aforementioned methods. The experiments on the end-to-end approach show that in the presence of low data resources, end-to-end models can achieve comparable transcription results to the DNN-HMM models on solo singing through speech-to-singing transfer. However, the word error rates of end-to-end models on polyphonic recordings are still much higher than those of the state-of-the-art DNN-HMM-based ALT. Furthermore, a new evaluation metric is introduced which measures a model's performance drop across multiple datasets. As a conclusion, this study elaborates the opportunities and challenges of the DNN-HMM and end-to-end approaches, provides benchmark results, and contributes to reducing the research and literature gaps between lyrics transcription and speech recognition with the goal of setting a point of reference for the next generation of lyrics transcription systems. For reproducibility, the codebase of this thesis is shared publicly and a tutorial is provided to retrieve the data used in the experiments.

Acknowledgements

Since day zero of my journey in the academy, my essential goals have been enabling technologies that would contribute to musicians around the world in enhancing their creativity and music listening experience, and removing the barriers for non-musicians to experience the joy of music creation. Personally, being surrounded by such brilliant people and immersed in such a creative environment within the Music Information Retrieval research community, and especially in the Centre for Digital Music, helped me greatly to achieve my research goals. I would like to give equal credit to all in the open-source community whose work I used in my project. As an advocate for open science, I feel honored to be a part of the community by following the principles for reproducibility. No doubt doing a Ph.D. is challenging, but a life-changing experience for many scholars as it was to me. This experience would not be the same without the people one has walked along together or encountered during the journey. Here, I would like to mention a few names that have taken part in mine.

I would like to start with my supervisors Simon Dixon and Sven Ahlbäck. Having such great supervision was invaluable regarding my personal and professional development, and in achieving my goals. Thanks greatly for always being supportive, patient, and open-minded, giving constructive feedback, and helping me consider my research from a multidimensional perspective. With your supervision, I was able to improve my personal, and especially critical and strategic thinking skills. The feedback I received from you always allowed me to understand the bigger picture and consider new ways to approach a problem. I would also like to express my gratitude to the independent assessor of my project, Emmanouil Benetos, for giving great insights and feedback related to my research direction.

Special regards to the Doremir team. I would like to especially express my gratitude to Sven Emtell and Patrik Ohlsson who enabled the research product we developed as an outcome of this dissertation to be deployed in the industry. Working with them helped me to gain industrial experience while being in the academy.

Although this document covers my Ph.D. research, my journey as a researcher started before. I cannot appreciate more about my experience during my Master's degree programme at the Music Technology Group in Barcelona. Working with and being supervised by Xavier Serra and Barış Bozkurt helped me to get a comprehensive understanding of the challenges and opportunities in the domain of Music Information Retrieval research.

Thanks to all colleagues who enriched my vision through the scientific and intellectual talks we had, especially: Saurjya Sarkar, Brendan O'Connor, Sertan Şentürk, Adrien Ycart, Daniel Stoller, Gerardo Roa Dabike, Jiawen Huang, and Furkan Yeşiler. Moreover, special thanks to Yukun Li, Polina Proutskova, Charles Brazier, and Alejandro Delgado who showed me the possibilities for leveraging a lyrics transcription system in other Music Information Retrieval tasks. I would also like to give my regards to Georgi Dzhambazov, who helped me become a member of the MIREX community.

It was a great privilege to be a part of the New Frontiers in Music Information Processing project under the European Union's Horizon 2020 programme and to have the opportunity to work with such great researchers. It has always been a pleasure to be a part of this team of great colleagues, Carlos Lordelo, Ruchit Agrawal, Vinod Subramanian, Karim Ibrahim, Javier Nistal, Kilian Schulze-Foster, Georgia Cantisani, Ondřej Cífka, Philip Tovstogan, Antonio Ramirez, Luís Carvalho; researchers Daniel Wolff, Stefan Lattner, Andreas Arzt, Geoffroy Peeters, Gaël Richard and Gerhard Widmer. Last but not least, many thanks to Alvaro Bort, the coordinator of the project, for all the great organisational work despite the pandemic.

I would like to express my special gratitude to Johann Sundberg, who showed kindness to share his experience and gave invaluable insights on singing voice at the very beginning of my research. It is an honor to have met him personally.

As a final remark, the globe witnessed the COVID-19 pandemic in 2020 and 2021, which is when this dissertation was produced. The years of the pandemic were extremely challenging for all. As someone far from home and his family during those days, I consider doing a Ph.D. kept me driven. I would like to give my regards to all my friends who were with me during the pandemic.

Finally, I would like to thank my family for always being my biggest supporters both on sunny and rainy days.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Music and lyrics co-occur through singing in most human populations around the globe (Clarke, 1952; Levitin, 2006; Donald et al., 2012). In particular, singing with lyrics is a medium of communication where two of the most complex auditory cognitive abilities of humans coexist: speech and music (Slevc, 2012). Contrary to spontaneous speech, the intended information is conveyed with singing lyrics through the complex interaction between musical and lyrical structure in addition to the linguistic content. In this regard, automatic lyrics transcription (ALT) is a computational task that is located at the junction of music, speech and language processing. This dissertation aims to tackle this problem using deep neural networks (DNN) and explores a number of methods for an improved and generalisable lyrics transcription performance.

1.1 Motivation

Previous psychological studies that examined the function of song lyrics showed that in certain situations, lyrics can be used to express, explore, and discuss emotions (Ali and Peynircioğlu, 2006; Barradas and Sakka, 2021), feelings, problems, personal ideas (MacDonald et al., 2002; Gardstrom, 1987) and political views (Van Sickel, 2005), help deal with everyday problems (Gibson et al., 2000), and influence and reflect listener's daily actions and decisions as well their music listening behaviour (Ballard et al., 1999; North and Hargreaves, 2008). For instance, Stratton and Zalanowski (1994) argued that listeners may prefer richer, more thoughtful, persuasive, and emotional lyrics. The authors also found evidence for the addition of lyrics to instrumental songs invoked stronger emotions on listeners. Furthermore, the same study argued that when the music and the lyrics have contradictory *affection* information, the lyrics would be more influential on the induced mood. Song lyrics may give clues about listeners' socioeconomic status as well (Pettijohn and Sacco Jr, 2009a,b). In particular, the recent study by Putter et al. (2021) found evidence from lyrics that people in the UK and US with lower socioeconomic status listened to lyrics that reflect social isolation and lower satisfaction, during the COVID-19 pandemic. In this regard, lyrics constitute one of the core elements of music that establish a communication ground between the singer/composer and the audience.

With the digital media revolution, song lyrics have also been exploited extensively to organise and navigate music collections. While song lyrics can be directly processed from text, these are often not available or not correctly paired with their music recordings. From this point of view, the automatic transcription of lyric information has a great potential to enhance the creation, listening and distribution of music. Such automatic transcription tools are generally referred to as *Automatic Lyrics Transcription* (ALT) systems, which are specifically designed to transcribe the sung lyrics from audio into text.

At first sight, the above described systems can be considered to be similar to speech-to-text machines. Although this perspective would not be terribly invalid as the inputs to both systems are human voice and the expected output is their orthographic transcriptions, singing has specific attributes compared to natural speech, which stand out as challenges for the industry-standard speech-to-text machines in transcribing sung lyrics accurately. Similar to such machines, transcribing sung lyrics is already challenging for human listeners. The experiments held by Collister and Huron (2008) showed that the word intelligibility is lower for singing than speech by human listeners. One of the most obvious reasons for this is the way that vowels are uttered in singing. Collister and Huron (2008) suspected that confusions due to the timbral and the temporal changes of vowels (which also lead to confusion in the perception of the surrounding consonants) can be a candidate cause for the lower word intelligibility. Sundberg and Rossing (1990) considered word intelligibility of sung vowels as a function of the fundamental frequency of vowel syllables. Palmer and Kelly (1992) showed that vowel duration variances and extensions change the rate at which syllables are uttered, hence affecting the prosody. Leanderson et al. (1987) highlighted the physiological differences between natural speech and singing, and observed higher glottal pressure variance for the latter. The authors argued that this is related to the greater emotional expression in singing. (Sundberg, 1995) showed that vocal styles like vibrato also influence word intelligibility. A number of studies focused on the cluster of powerful formants in the spectrum observed for singing performances, i.e. the singer's formant (Smith and Scott, 1980; Gregg and Scherer, 2006; Sundberg and Romedahl, 2009), which is mostly observed in Western opera. These arguments are mostly concerned with performer (singer) related factors affecting word intelligibility, though listener and environment related factors are not to be omitted. Due to the hearing effects (like masking), the listeners' understanding of sung words might be affected by the background noise, especially when there is are instruments accompanying the vocals (Di Carlo, 2007). The acoustics and the reverberation of the auditory environment (Sato and Prodi, 2009), and vocal effects (like artificial reverberation, chorus, etc.) are some other major factors that influence the word intelligibility. Furthermore, the pilot study by Ibrahim et al. (2017) highlighted the importance of music genres. Above mentioned research outlines the major contradictory elements between natural speech and singing, and must be taken into account when attempting to build a lyrics transcriber.

The automatic retrieval of sung lyrics and transcribing natural speech do not only differ by their physical attributes. The semantics of both domains are also distinct. Lyrics are somehow closer to poetry, and natural speech is to prose. Song lyrics are often temporally and structurally aligned with the underlying musical piece. Non-linguistic words are often neglected during the automatic transcription of natural speech, however they contribute to the sung melody, hence should be included in the transcription output. More about these differences between speech and singing are scrutinised further through a few examples in Section 2.1 - Problem Definition.

The research in ASR has been developing since long before its applications found place in the industry. The efforts of numerous researchers and research groups has helped establishing standardised methods, open-source data resources, and training / testing schemes for model evaluation. ALT is an emerging field, and hence lacks the research and literature that ASR research has been developing. To reduce this research and literature gap is one of the main driving factors that motivated the progression of this research.

Until recently, studies in ALT had used private datasets for model evaluation (Fujihara et al., 2006; Mesaros and Virtanen, 2010a; Hansen, 2012; Kruspe, 2016), which made the model comparison across different publications difficult. Secondly, the availability of large-scale training data resources has been among the major bottlenecks for ALT research. For instance, a benchmark training set in ASR, namely Librispeech (Panayotov et al., 2015) has nearly 1000 hours of transcribed natural speech. The size of the training sets used previously for ALT are around 150 hours (Meseguer-Brocal et al., 2019), which weren't available until recently. The curation and distribution of large-scale datasets is not only a bottleneck for ALT, but also for music information retrieval (MIR) research, due to copyright issues that emerge when sharing music recordings. From this perspective, this dissertation provides experiments and results exploiting the available training and evaluation data resources commonly used in research, with the goal of establishing benchmark ALT results.

Some of ALT's direct use cases include music video captioning/subtitling (e.g. for Karaoke), music recognition, and query by singing (Watanabe and Goto, 2019). For the task of audio-to-lyrics alignment - the automatic retrieval of word timings in music signals - the best performing approaches include a pretrained ALT acoustic model in their core (Gupta et al., 2020). On the other hand, the language model within ALT modules can be used for lyrics generation (Hopkins and Kiela, 2017). ALT models also find utility in using lyrics for song mood/emotion detection (Rachman et al., 2018) or improving vocal source separation (Schulze-Forster et al., 2020; Meseguer-Brocal and Peeters, 2020) and cover song identification (Correya et al., 2018; Vaglio et al., 2021). In addition, these systems can be utilised in musical therapy methods that are developed to treat language deficits (Schlaug et al., 2010). Some other potential applications of ALT include music composition, playlist generation, music recommendation, and royalty forecasting. Considering the above mentioned applications that will be enabled and enhanced via ALT, this research is motivated to improve the applicability of this technology.

1.2 Research Aim

As will be discussed later in Section 2.1, the task of ALT is similar to automatic speech recognition (ASR), or specifically Large Vocabulary Continuous Speech Recognition (LVCSR), as the final desired output of such systems is the transcription of uttered (or sung) words and characters. LVCSR has already been acknowledged as one of the most challenging tasks within the overall machine learning domain, yet ALT is no less challenging, considering the specific characteristics of singing performances and song lyrics mentioned above. Despite the unique characteristics of song lyrics compared to natural speech, not as much research has been done on the ALT domain that could potentially enable its application in the industry. This dissertation is aimed at contributing towards reducing the research and the literature gaps between ASR and ALT to accelerate the development of the latter research field. Within this respect, it strives to adapt the state of the art deep neural network-based LVCSR techniques to the singing domain. Therefore, it ponders which of those techniques is more suitable given the open-source data resources that could be leveraged to build a lyrics transcriber.

This thesis tackles the task of ALT by means of a specific data domain of the LVCSR problem. Although there are several aspects of lyrics to be handled to achieve more context-aware transcriptions, such as the semantics, rhythmic components, rhyming and the segmentation of lyrical lines, this study focuses on adapting the existing state-of-the-art methods in ASR to singing data. The adaptation considers the a number of contrastive dimensions of lyrics and natural speech, like the lexical diversities, utterance lengths, and the pronunciation variances which will be quantitatively examined later in Sections 3.3 and 4.2.1. Furthermore, this study proposes methods to improve transcription performance when there is music accompaniment to singing.

In recent years, ASR systems in research have reached to a performance level that can be comparable to human transcribers. For instance, the professional transcribers in the experiments conducted by Saon et al. (2017) and Xiong et al. (2017) had 5.9 % and 5.1 % *word error rate*s (*WER*s)[1] respectively on the Switchboard corpus (Godfrey et al., 1992) [2]. In 2016, Microsoft reported *WER*s below 6 % on the Switchboard corpus (Saon et al., 2017). Recently, IBM's Research A.I. Team achieved below 5 % *WER* (Tüske et al., 2021). However, at the time when the author of this thesis started his Ph.D. research, the lowest *WER* score reported on solo singing was around 34 % (Gupta et al., 2018), which was even higher when there is musical accompaniment (around 50-60 % *WER* (Gupta et al., 2020)). This highlights the large gap in the performances of word recognition systems for speech and singing data. This

[1] *WER* is the most commonly used evaluation metric in ASR research, which will be studied later in Section 3.4.

[2] The Switchboard corpus is a research benchmark dataset for ASR and consists of human conversational speech data.

poor performance prevented researchers and engineers from developing the above-mentioned applications that would leverage lyrics transcriptions. Moreover, being an amalgamation point for multiple complex information domains, namely music, speech and language, ALT has been considered to be among the most challenging tasks within Music Information Retrieval (*MIR*) research (Humphrey et al., 2018). Therefore, this research aims to improve lyrics transcription performance through leveraging available data resources and adapting the state-of-the-art in ASR to singing data.

1.3 Thesis Structure

The content of this thesis is organised according to the following structure:

- **Chapter 2 - Related Work** provides the background knowledge, goes through the related previous work, and gives a comparison of the common open-source toolkits used in the relevant research field.
- **Chapter 3 - Singing Data for Word Recognition** examines the singing data available for research in lyrics transcription, and gives an explanation of the metrics used to evaluate ALT models.
- **Chapter 4 - Hybrid DNN-HMM Lyrics Transcription** describes the details of the first approach (as the chapter title implies) for building an ALT system included in this study, and
- **Chapter 5 - End-to-end Lyrics Transcription** describes the second approach, which is end-to-end training, where a number of possibilities to apply speech-to-singing transfer learning are explored.
- **Chapter 6 - Discussion & Conclusion** compares the DNN-HMM and end-to-end approaches, discusses the future challenges and opportunities in this research field, and summarises the novel contributions of this thesis.

1.4 Associated Publications

Certain sections in this thesis work were previously published at international peer-reviewed conferences. The related publications are listed below:

- Emir Demirel, Sven Ahlbäck, and Simon Dixon, *Automatic lyrics transcription using dilated convolutional neural networks with self-attention*: The baseline DNN-HMM model presented in Chapter 4, where the around 5% *WER* improvement reported through self-attention and RNNLM[3], compared to the previously best work by the time publication (Dabike and Barker, 2019).

- Emir Demirel, Sven Ahlbäck, and Simon Dixon, *A recursive search method for lyrics alignment*: The model in the above publication is also used in the MIREX 2020: Automatic Lyrics Transcription challenge where it achieved the best results on the monophonic evaluation set (Demirel et al., 2020b). In addition, a recursive audio-to-lyrics alignment procedure is presented in this work.

- Emir Demirel, Sven Ahlbäck, and Simon Dixon, *Computational pronunciation analysis in sung utterances*: The pronunciation analysis and singing-adapted lexicon presented in Sections 4.2 were published in this paper (Demirel et al., 2021c).

- Emir Demirel, Sven Ahlbäck, and Simon Dixon, *MSTRE-Net: Multistreaming acoustic modeling for automatic lyrics transcription*: Three of the novel methods in building the DNN-HMM model in Chapter 4, namely the compact multistream TDNN architecture, music-informed silence modeling and cross-domain training were proposed (Demirel et al., 2021b). A new polyphonic evaluation set was also presented in this work, which is discussed in Section 3.1.2.

Note that all the codework and the papers are written by the author of this thesis, and supervised by the co-authors, Sven Ahlbäck and Simon Dixon.

In addition, further work is done on audio-to-lyrics alignment and sung note segmentation, which resulted in the following publications:

[3]RNNLM refers to Recurrent Neural Network-based Language Model, which will be discussed later in Section 4.1.2

- Emir Demirel, Sven Ahlbäck, and Simon Dixon. Low resource audio-to-lyrics alignment from polyphonic music recordings. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021 (Demirel et al., 2021a) : This work proposes a novel audio-to-lyrics alignment method that can operate on long music recordings with low memory requirements.

- Yukun Li, Emir Demirel, Polina Proutskova, and Simon Dixon. Phoneme-Informed Note Segmentation of Monophonic Vocal Music. In 2nd Workshop on NLP for Music and Audio (NLP4MusA), 2021 (Li et al., 2021): This work presents a novel method for leveraging the output of the ALT system for the sung note segmentation task.

The details of the last two publications are not included in this document as their topics are not within the main scope of this thesis.

1.5 Reproducibility

For the reproducibility of the results of this research, the author used and developed open-source software. These are summarised below:

- Open source ASR Toolkits: Two open-source toolkits are used for building the lyrics transcribers studied in this thesis, namely Kaldi (Povey et al., 2011) and SpeechBrain (Ravanelli et al., 2021) packages. A more detailed discussion regarding these toolkits is provided in Section 2.4.

  Links:
  Kaldi - *https://github.com/kaldi-asr/kaldi*
  SpeechBrain - *https://github.com/speechbrain/speechbrain*

- The (A)utomatic (L)yrics (T)r(A)nscription - ALTA package: The package contains two Kaldi-based recipes for the DNN-HMM lyrics transcription (studied in Chapter 4) approach and the data retrieval is explained in the instructions. The first recipe is for monophonic lyrics transcription which is used to produce the results by Demirel et al. (2020a). The second recipe is for training the MStreNet model (Demirel et al., 2021b). The pretrained models are not shared due to licensing restrictions.

Link: *https://github.com/emirdemirel/ALTA*

- A repository is shared with the community to retrieve the newly introduced polyphonic evaluation set which is curated from a portion of the DALI dataset (Meseguer-Brocal et al., 2019). The repository contains lyrics annotations, the relevant metadata information and a Jupyter notebook tutorial to automatically retrieve the audio recordings.

  Link: *https://github.com/emirdemirel/DALI-TestSet4ALT*

- The novel DNN-HMM based lyrics alignment software is developed as an open-source toolkit. This software uses the pretrained acoustic model from the ALTA package.

  Link: *https://github.com/emirdemirel/ASA_ICASSP2021*

- In addition to the open-source software, all publications and research outcomes are shared online including video presentations. The list of related talks can be found at:

  https://emirdemirel.github.io/alt.html. Specifically, the summary of this thesis is presented at: *https://www.youtube.com/watch?v=r-4JEqyDGPs*.

1.6 Contributions

The novel contributions of this thesis are listed below:

- **Chapter 3:** A quantitative comparison between speech and singing data is given. Evaluation metrics are explained and a novel metric is proposed in 3.4.2, which measures a model's performance drop across multiple test sets or different domains. A lyrics-based corpus for constructing the language model and a new evaluation set for ALT ($DALI^{test240}$) is presented in this chapter.

- **Chapter 4:** The state-of-the-art DNN-HMM based system is presented. Novel methods proposed in this chapter are: a quantitative pronunciation analysis and a singing adapted pronunciation dictionary, cross-domain training for building a more robust acoustic model on polyphonic recordings, the use of the '*music*' token in the target class set to model non-vocal (instrumental) instances in polyphonic recordings (i.e. music-informed silence modeling), and the multistream time delay neural network (TDNN) architecture. In addition, an extensive cross-dataset evaluation is performed to measure the proposed methods' generalisability across all benchmark test sets used in research.

- **Chapter 5:** The state-of-the-art end-to-end ASR approach is applied on the available singing data. As a novelty, transfer learning from speech to singing is applied. This showed that some information between speech and singing is transferable. In addition, a cascaded transfer learning approach is proposed to adapt the monophonic acoustic model to the polyphonic domain. Finally, the cross-domain training approach is tested for the end-to-end models.

- **Chapter 6:** Comparison of the DNN-HMM and the end-to-end approaches, and an exploratory error analysis are conducted. Both models are compared with the previous literature. According to this, the DNN-HMM model outperforms the previous state-of-the-art in ALT by a large margin. While the end-to-end models have considerably higher error rates than the DNN-HMM method, the best performing end-to-end model still has better results than other end-to-end systems reported in the literature. Furthermore, the current challenges and the future opportunities in ALT research are discussed in this chapter.

Chapter 2

Related Work

This chapter introduces the relevant concepts for understanding the lyrics transcription models proposed in the following chapters. It begins by defining the ALT problem, and proceeds with the essential theoretical details of two of the most competitive frameworks in speech recognition, namely Deep Neural Networks - Hidden Markov Models (DNN-HMM) with Lattice-Free Maximum Mutual Information (LFMMI) training and transformer-based end-to-end (E2E) systems using the hybrid Connectionist Temporal Classification (CTC) / Attention objective. After the theoretical introduction, a comparison of the ASR toolkits available for research is given. Finally, the literature on the recent advances in ALT is studied excluding what is presented in the following chapters of this thesis.

2.1 Problem Definition

The task of Automatic Lyrics Transcription (ALT) can be defined as the procedure of transforming a singing voice performance with lyrics that has a finite length of data into a string of text which is targeted to match that of the original lyrics written by the lyricist. From this perspective, it is possible to consider this task as ASR in the singing voice domain. Following the statistical modeling approach presented by Gales and Young (2008), this task can be translated to estimating the posterior probability that any given stream of sung words or lyrics will be uttered. This is equivalent to finding the most probable word sequences, $\mathbf{w}$, given the acoustic observations $\mathbf{X}$, and can be formalised as:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}(P(\mathbf{w}|\mathbf{X})) \tag{2.1}$$

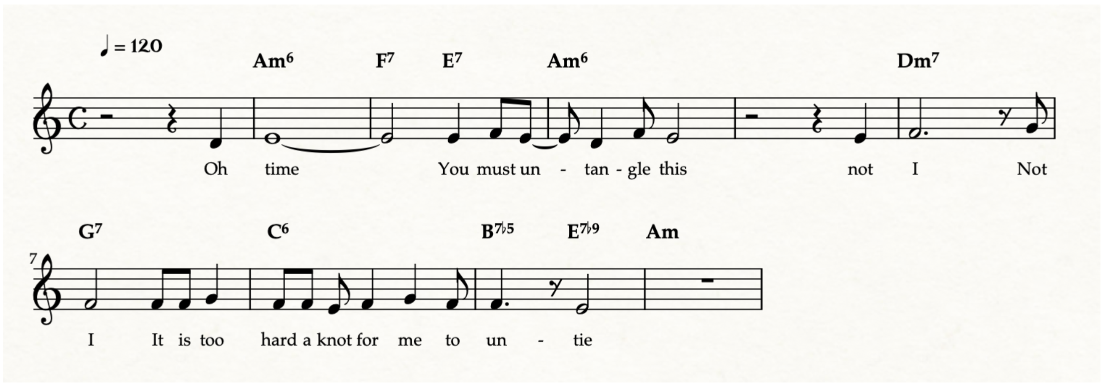
Similar to the art of poetry, prosody is a main concern for song lyrics, which is often influenced by the rhythmic patterns of the underlying musical piece (Palmer and Kelly, 1992; Rasinski, 2006). Likewise the pitch and the melodic structure have an influence when uttering words, which affects how words are pronounced and thus their intelligibility (Fine and Ginsborg, 2014). For instance, consider the following phrase from Act 2 - Scene 2 from Shakespeare's Twelfth Night:

O time, thou must untangle this, not I.
It is too hard a knot for me to untie!

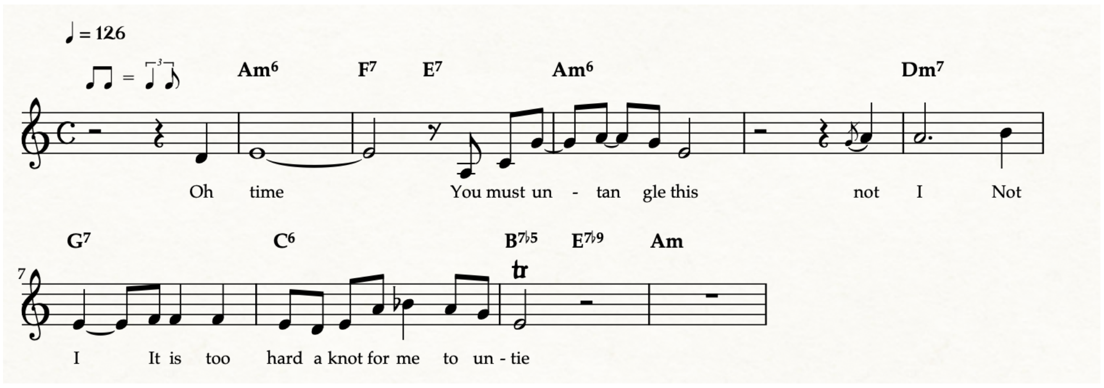
The prosodic elements would not be identical if the above lines were uttered in prose form or natural speech, considering words at the beginnings and the ends of lines, the choice and the articulation of words, and how the sequence of words is ordered. However, the composition and the singing of lyrics are not influenced only by prosody or stress, but also the musical information conveyed through melodic lines. Consider the same phrase by Shakespeare sung with the melodic lines shown in Figure 2.1[1].

Notice the duration, pitch and transition variances between syllables and different interpretations of the piece with the same underlying musical harmonic structure, considering their corresponding musical notes, and the rhythmic patterns. In addition, there are certain words repeated multiple times by the singers that are not in the original lyrics. Although the examples are not necessarily representative of specific music styles, they are provided as a reflection on how the manner of utterance or pronunciation can be varied depending on the content creator's musical creativity. In this respect, generalising rules regarding the acoustics, the linguistics and the semantics of lyrics would not necessarily be a comprehensive perspective when attempting to solve the ALT problem. Broadly, this research considers ALT as a sequence prediction problem, and mainly employs relevant concepts of the state-of-the-art Large Vocabulary Continuous Speech Recognition (LVCSR) frameworks. In this regard, the main approach is developing

[1]The sheets are generated using the ScoreCloud app which can be downloaded from *https://scorecloud.com*.


♩= 120
Am6 F7 E7 Am6 Dm7
Oh time You must un - tan - gle this not I Not
G7 C6 B7♭5 E7♭9 Am
I It is too hard a knot for me to un - tie

(a)


♩= 126
Am6 F7 E7 Am6 Dm7
Oh time You must un - tan gle this not I Not
G7 C6 B7♭5 E7♭9 Am
tr
I It is too hard a knot for me to un - tie

(b)

Figure 2.1: Two examples of lyrics in Western music notation.

methods to adapt the training pipelines and the computational blocks of ASR systems for singing voice. Although there are endless possibilities for injecting lyrics priors in the transcription system with the goal of obtaining more semantically and musically coherent output, this thesis is mainly motivated by providing a baseline for the next generation of lyrics transcribers.

As an additional consideration, common ASR applications often do not include non-linguistic sounds in the target word space as they contribute minimally to the overall context. On the contrary, the accurate transcription of such sounds is a major concern for ALT since they often appear in sung melodies or scat singing. For this reason, the non-linguistic "words" are not excluded from the target search space or the *vocabulary* in ALT.

Decoding

In conventional speech recognition, $\widehat{\mathbf{w}}$ in Equation 2.1 is generally estimated using *beam search*, which can store conditional probabilities of multiple tokens, and provides best *N* outputs based on the accumulated scores. The search chooses predictions with higher probabilities when generating hypotheses. Consider Ω_l is a set of partial hypotheses with length l. Beam search uses the start and end of sentence tokens (<sos> and <eos> respectively) that indicate the beginning and the end of the predictions. The search begins from $l = 0$, and Ω_0 has only one hypothesis which is the *start-of-sentence*, '<sos>' token. For $1 \leq l \leq L_{max}$, the hypothesis set can be expressed as $\Omega_l = \{g_{l-1} \cdot w_l : g_{l-1} \in \Omega_{l-1}, w_l \in \widehat{w}_l\}$, where the operation '$\cdot$' denotes concatenation, $\widehat{w}_l$ is the set of all possible predictions at step l and L_{max} is the maximum allowed length of the hypotheses in question. Each new hypothesis at the l^{th} time step of the beam search (i.e. $h_l \in \Omega_l$) is generated through concatenating a label $w_l \in \widehat{w}_l$ to the partial hypothesis $g_{l-1} \in \Omega_{l-1}$ at the previous time step (i.e. $h_l = g_{l-1} \cdot w_l$). The beam search score, α for every new hypothesis, h, is obtained through summing the scores of the partial hypothesis of the previous time step g_{l-1} and the conditional probability of a token to predict, w_l, in the log domain:

$$\alpha(h_l, \mathbf{X}) = \alpha(g_{l-1}, \mathbf{X}) + \log p(w_l | g_{l-1}, \mathbf{X}). \tag{2.2}$$

The search terminates if w_l is predicted to be <eos>, and h_l is added to the final set of complete hypotheses $\widehat{\Omega}_l$. The final prediction is selected to be the hypothesis with the highest scores:

$$\widehat{\mathbf{w}} = \underset{h \in \widehat{\Omega}}{\operatorname{argmax}} \, \alpha(h, \mathbf{X}). \tag{2.3}$$

In practice for efficiency, only hypotheses with the *N* highest probabilities are kept during the search process (i.e. less likely labels are pruned).

Alignment

Although this dissertation only targets retrieving sung words and phrases in the correct sequence, the ALT task overall is not only concerned with retrieving uttered words in a certain order, but also with their timings, and how they are aligned with the overall musical structure, as the target of transcriptions is to guide performers and listeners about the subject musical piece. From the information retrieval point of view, the task of retrieving word timings can be associated with the alignment problem. The most common approach to this is the procedure referred to as *forced alignment*. In principle, this procedure is based on aligning the target sequence (either on word, phoneme or syllable level) with the sequence of posterior probabilities. However, forced alignment considers $\widehat{\mathbf{w}} = \mathbf{w}$, thus does not need to find the most probable sequence (i.e. no need for 'argmax'). In practice, this is equivalent to finding the least cost path of the given lyrics of a singing performance and its acoustic representation, which is generally achieved by dynamic programming. More details on forced alignment can be found in the works of Rabiner and Juang (1993) and Gales and Young (2008).

Although this thesis acknowledges that retrieving word timings of lyrics is important for the applications of ALT, improving word alignments is not among the prior concerns of the proposed methods in this research. Instead, this work mostly focuses on the retrieval of the intended words in the correct order, and does not attempt to transcribe lyrics in a musically structured manner, or tackle the audio-to-lyrics alignment problem. However, readers are encouraged to read the distinct approaches to the audio-to-lyrics problem proposed by Stoller et al. (2019); Gupta et al. (2020); Demirel et al. (2021a) and Huang et al. (2022), and how such systems are evaluated in the thesis by Dzhambazov et al. (2017).

2.2 DNN-HMM Based Approach

2.2.1 Hidden Markov Models

HMMs are a popular choice for statistically modeling a sequence of symbols or vectors (Gales and Young, 2008). Typically an HMM is defined with a finite number of states $j = 1, 2, ..., N$, transition probabilities ($a_{i,j}$) between state pairs, and output distributions of *emitting* states (b_j) over a set of predefined output symbols. Going back to the speech recognition problem, HMM-based ASR applies Bayes Rule to Equation 2.1 (Gales and Young, 2008), which yields to:

$$\underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}|\mathbf{X}) = \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{X}|\mathbf{w})P(\mathbf{w})). \tag{2.4}$$

It is impractical to compute $P(\mathbf{X}|\mathbf{w})$ directly in large vocabulary applications. Moreover, word predictions should consider pronunciation variations of uttered words. Considering these, most HMM-based ASR systems use *phonemes*[2] to decompose words into smaller units for representing speech sounds. According to the connectionist theory of ASR (Graves et al., 2006), a sequence of phonemes is considered as a Markovian chain where each phoneme is represented as a continuous density HMM, Q with states q, transitions $a_{i,j}$ and output observation distributions $b_{i,j}$ that generates acoustic feature vectors $\mathbf{X}$. This is injected into Equation 2.4 as follows:

$$\begin{aligned}
\underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}|\mathbf{X}) &= \operatorname{argmax} \frac{P(\mathbf{X},\mathbf{w})}{P(\mathbf{X})} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{X},\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_{Q} P(\mathbf{w},\mathbf{Q},\mathbf{X}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_{Q} P(\mathbf{X}|\mathbf{Q},\mathbf{w})P(\mathbf{Q},\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}) \sum_{Q} P(\mathbf{X}|\mathbf{Q},\mathbf{w})P(\mathbf{Q}|\mathbf{w}) \qquad (2.5)
\end{aligned}$$

where $\mathbf{Q}$ is the sequence of phoneme states. Note that the states and the observations have to yield the standard conditional independence assumptions for an HMM (Gales and Young, 2008), so that $P(\mathbf{X}|\mathbf{Q},\mathbf{w}) = P(\mathbf{X}|\mathbf{Q})$. Then, Equation 2.5 can be summarised as:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}|\mathbf{X}) = \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}) \sum_{Q} P(\mathbf{X}|\mathbf{Q})P(\mathbf{Q}|\mathbf{w}). \tag{2.6}$$

According to Equation 2.6, there are three factors used to approximate $P(\mathbf{w}|\mathbf{X})$. The

[2]A phoneme is the basic sonic unit of speech.

acoustic model (AM), $P(\mathbf{X}|\mathbf{Q})$ aims to match an acoustic observation $\mathbf{X}$ to a sequence of phonemes. The term $P(\mathbf{Q}|\mathbf{w})$ is generally obtained from a predefined dictionary that gives a mapping between words and their phonemic pronunciation, and it is associated with the *pronunciation model*. Finally, the grammar information or the *language model* (LM) is obtained through $P(\mathbf{w})$ which is the probability of observing a word in a sequence of words, $w_1 w_2 ... w_N$.

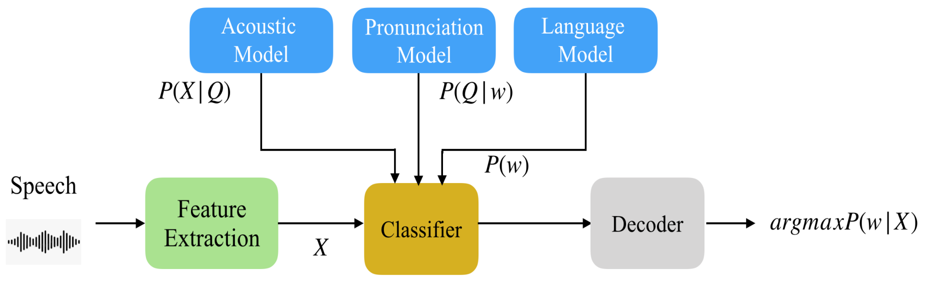


Figure 2.2: Overall bloack diagram for HMM-based ASR systems.

Acoustic Model

Consider a composite HMM $\mathbf{Q}$, formed by the concatenation of the phones $q^{w_1}, ..., q^{w_L}$. Then the acoustic likelihood can be modeled by

$$p(\mathbf{X}|\mathbf{Q}) = \sum_{\theta} p(\theta, \mathbf{X}|\mathbf{Q}), \tag{2.7}$$

where θ is the state sequence, and

$$p(\theta, \mathbf{X}|\mathbf{Q}) = a_{\theta_0,\theta_1} \prod_{t=1}^{T} b_{\theta_t}(\mathbf{X}_t) a_{\theta_t,\theta_{t+1}} \tag{2.8}$$

where θ_0 and θ_{T+1} are non-emitting start and exit states. For simplicity, these non-emitting states are ignored in the model parameter estimation.

In the pre-deep learning era, the traditional approach for building the acoustic models had been modeling output distributions as multivariate Gaussian distributions (or Gaussian Mixture Models - GMM) where the model parameters (like the mean and covariance matrix of these distributions) can be estimated via the forward-backward algorithm (Baum et al., 1970). A

GMM is a probability density function that is obtained by calculating the weighted sum of the Gaussian component densities. In audio and speech processing, it is assumed that a GMM represents the acoustic characteristics of a spectral envelope (Gales and Young, 2008). There are two learnable main sets of parameters of GMMs that can be estimated using machine learning methods for building an acoustic model, which are the mean and the covariance matrices. The GMM-HMM model is initialised with global mean and covariance values and the same number of frames are assigned to each HMM (phoneme) state, q. Then, alignment (via Viterbi decoding) and the parameter estimation are iteratively applied using the Expectation-Maximization (EM) algorithm (Moon, 1996).

Pronunciation Model

Consider a pronunciation of a word as a particular sequence of phonemes $\mathbf{q}^{w}_{M_w} = q_1, q_2, ..., q_{M_w}$. Then, $P(\mathbf{Q}|\mathbf{w})$ can be obtained via the following formula:

$$P(\mathbf{Q}|\mathbf{w}) = \prod_{l=1}^{L} P(\mathbf{q}^{w_l}|w_l), \tag{2.9}$$

where $\mathbf{Q}$ is a particular sequence of pronunciations, L is the number of words in the predictions and the product is done over all possible pronunciation sequences. In practice, there exist only few alternative pronunciations for a particular word, which makes the summation in Equation 2.6 tractable. The pronunciation probabilities, $P(\mathbf{q}^{w_l}|w_l)$ can either be set to be equal for each alternative pronunciation, or can be estimated via a pretrained model as proposed by Chen et al. (2015).

Language Model

The language model is generally trained using statistical learning methods trained on text corpora. In conventional ASR systems, $P(\mathbf{w})$ is computed using the *n-gram* language modeling approach,

$$P(\mathbf{w}) = \prod_{k=1}^{K} P(w_k|w_{k-1}, w_{k-2}, ..., w_{k-N+1}), \tag{2.10}$$

where $\mathbf{w} = w_1, ..., w_K$ is a word sequence with length K. There are often cases where a word in a test sample appears in an *n-gram* that was never seen in the training data. In this case, the LM would assign a zero probability for such words. To prevent the LM assigning zero probability to these unseen events, probabilities of the most frequent events are shaved off and distributed over the unseen *n-gram*. This procedure is called *smoothing*. The DNN-HMM framework employed in this dissertation uses the Kneser-Ney algorithm (Kneser and Ney, 1995) for this purpose.

Context and Position Dependency

The acoustic properties of phonemes may vary in general due to *co-articulation* or omitting (*elision*) during pronunciation. In modern ASR systems, a common way to solve this problem is to use *context-dependent* HMMs, where an HMM state (phone) is labeled with the combination of the base phone and the phones immediately to the left and right. Such state labeling is referred to as *triphone* modeling. However, such labeling would result in a very large set of phones, i.e. target classes. Hence, to make the final decoding graph computationally tractable, less frequently occuring triphones in the training data are pruned, most commonly via *tree-based clustering* (Young et al., 1994).

The pronunciation of phonemes may also vary depending on where they are located within a word. To account for this, phonemes are additionally labeled with respect to their position in a word. There are four categories for doing this: phonemes in the beginnings and endings of words, intra-word phonemes and singleton phonemes which form alone for a single word. This final set of context and position dependent phonemes are generally referred to as *senones*.

Weighted Finite State Transducers

The probabilities estimated via the AM, LM and the pronunciation dictionary are then composed into a graph with hidden states and decoding is applied through the *Viterbi algorithm* on the emission probabilities of the HMM states in order to retrieve the likeliest sequence of words. This procedure is represented by the 'Decoder' block in Figure 2.2. In the DNN-HMM framework we employ, Kaldi (Povey et al., 2011), the decoding graph is constructed using Weighted Finite State Transducers (WFST) (Mohri et al., 2002), where the priors coming from the AM, context dependency, the lexicon and the LM are constructed via separate FSTs, represented by H, C, L and G graphs respectively, and composed into a single decoding graph.

The WFST structure can store multiple pronunciation variants and word predictions through a *lattice* form, a weighted labeled acyclic (directed) graph generated after a first-pass decoding (an example can be seen in Figure 2.3). Readers are encouraged to read more about WFSTs and their application in ASR in the work of Mohri et al. (2002).

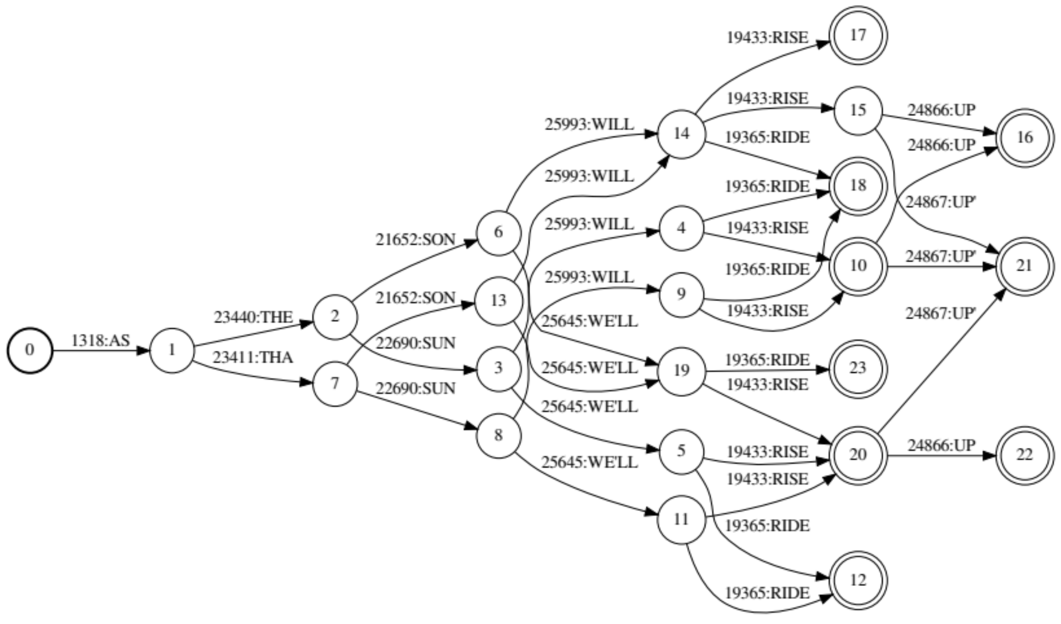


Figure 2.3: An example of the lattice structure constructed at the end of the *HCLG* graph that is generated after the first forward pass. The decoding procedure aims to find the most likely path on this lattice from left-to-right. Numbers on the left-side of the words represent word IDs. Ground Truth : '*AS THE SUN WILL RISE*'.

Language Model Scaling & Word Insertion Penalty

In HMM-based speech recognition, the probabilities estimated by the acoustic and the language models are often on different scales, as the ones estimated by the former are not normalised (Gales and Young, 2008). This is due to the emission probabilities of the HMM states not being independent of each other, which contradicts the conditional independence assumption of HMMs. Due to this, the acoustic model may dominate, causing the influence of the language model to be too low during inference. Therefore, it is necessary to apply scaling on these probabilities. In our pipeline, we keep the acoustic model scaling as 1, and apply scaling on the language model as follows:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}(p(\mathbf{X}|\mathbf{w})P(\mathbf{w})^{\mu_{\mathbf{w}}}), \tag{2.11}$$

where $\mu_{\mathbf{w}}$ is called the language model scaling factor. To improve numerical stability, the log-likelihood is used:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}(\log(p(\mathbf{X}|\mathbf{w})) + \mu_{\mathbf{w}} \log P(\mathbf{w})). \tag{2.12}$$

A typical range for $\mu_{\mathbf{w}}$ is between 5 and 20, though increasing $\mu_{\mathbf{w}}$ too much would increase the risk of making deletion mistakes (Gales and Young, 2008). To balance the deletion mistakes with insertions and substitutions, a *word insertion penalty*, $\sigma_{\mathbf{w}}$, is added as:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}(\log P(\mathbf{X}|\mathbf{w}) + \mu_{\mathbf{w}} \log P(\mathbf{w}) + \log \sigma_{\mathbf{w}}). \tag{2.13}$$

Tuning the values for $\mu_{\mathbf{w}}$ and $\sigma_{\mathbf{w}}$ has an important effect on the output transcriptions which are typically tuned on an external development set.

2.2.2 Training Acoustic Models with Deep Neural Networks

Deep neural networks (DNNs) are artificial neural networks (ANN) with stacked structures of more than one hidden layer between their inputs and outputs (Bengio, 2009) . Due to their capability of modeling nonlinearities in data, DNNs are widely used to train classifiers that classify audio frames into phoneme types (Graves et al., 2013). DNNs in speech recognition are multicategory classifiers which output the posterior probabilities of phoneme states q for an acoustic observation $\mathbf{X}$ at a given time t which can be obtained via the $\mathrm{softmax}$ function:

$$y(q) = p\left(q|\mathbf{X}_{ut}\right) = \frac{\exp\{a\left(q\right)\}}{\sum_{q'} \exp\{a\left(q'\right)\}} \tag{2.14}$$

where $a(q)$ is the activation function at the output layer corresponding to state q (Bourlard and Morgan, 2012). By definition, DNN phoneme classifiers use a pseudo-log-likelihood of state q given the observation $\mathbf{X}$ (Graves et al., 2013),

$$\log p(\mathbf{X}_{ut}|q) = \log p(q|\mathbf{X}_{ut}) - \log P(q),$$
$$= \log y_{ut}(q) - \log P(q), \tag{2.15}$$

where $P(q)$ is the prior probability of state q estimated from the training data (Bourlard and Morgan, 2012; Vesel̀y et al., 2013).

One common approach for obtaining phoneme posteriors using DNNs employs the categorical frame-wise cross-entropy loss as the objective function for training:

$$\mathcal{F}_{CE} = -\sum_{u=1}^{U}\sum_{t=1}^{T_u} \log y_{ut}(q_{ut}), \tag{2.16}$$

where q_{ut} is the phoneme state in an utterance u at time t.

The parameters of DNNs are updated via the standard error backpropagation procedure (Rumelhart et al., 1985). The gradient for this is then calculated as:

$$\frac{\Delta \mathcal{F}_{CE}}{\delta a_{ut}(q)} = -\frac{\delta \log y_{ut}(q_{ut})}{\delta a_{ut}(q)} = y_{ut}(q) - \delta q; q_{ut}, \tag{2.17}$$

where $\delta q; q_{ut}$ is the Kronecker delta function (Vesel̀y et al., 2013). The network parameters are often updated using either Stochastic Gradient Descent (SGD) (Saad, 1998) or Adam (Kingma and Ba, 2014) optimisers.

2.2.3 Sequence Discriminative Training

Sequence discriminative training is essentially based on the maximum mutual information (MMI) criterion where the corresponding objective function, $\mathcal{F}_{MMI}$ is defined as:

$$\mathcal{F}_{MMI} = \sum_{u} \log \frac{p(\mathbf{X}_u|Q_u)^{\mathcal{K}} P(W_u)}{\sum_W p(\mathbf{X}_u|Q)^{\mathcal{K}} P(W)} \tag{2.18}$$

where $\mathbf{X}_u = \{\mathbf{X}_{u1}, \mathbf{X}_{u2}, ..., \mathbf{X}_{uT_u}\}$ is the sequence of acoustic observations for the utterance

u and $Q_u = \{q_{u1}, q_{u2}, ..., q_{uT}\}$ is the sequence of phoneme states corresponding to the word sequence W_u. When training the acoustic model, the neural network parameters are updated w.r.t. phoneme posteriors, $P(\mathbf{X}|Q)$ (Bahl et al., 1986). Hence, the gradient used for optimisation is obtained by differentiating Equation 2.18 w.r.t. these phoneme posteriors:

$$\frac{\delta \mathcal{F}_{MMI}}{\delta \log p(\mathbf{X}_{ut}|r)} = \mathcal{K}\delta_{r;q_{ut}} - \frac{\mathcal{K}\sum_{W:q_t=r} p(\mathbf{X}_u|Q)^{\mathcal{K}} + \mathcal{K}P(W)}{\sum_W p(\mathbf{X}_u|Q)^{\mathcal{K}} P(W)} \tag{2.19}$$

Injecting the output of the $\mathrm{softmax}$ function in Equation 2.14 into the gradient above, we obtain:

$$\frac{\delta \mathcal{F}_{MMI}}{\delta a_{ut}(q)} = \mathcal{K}(\delta_{q;q_{ut}} - y_{ut}(q)), \tag{2.20}$$

where $y_{ut}(q)$ is the posterior probability of observing state q at time instance t. According to Equation 2.19, there are two gradients needed to be computed in sequence discriminative training. The graph in the numerator encodes utterance specific phoneme posteriors whereas the denominator graph is constructed from all possible word sequences available in the labels and is constant across the entire training set (Povey et al., 2016). Considering that the goal of training is to maximise the shared information between the reference and observed word sequences, the gradients are updated to maximise the utterance-specific posteriors in the numerator graph and minimise the corpus-level posteriors in the denominator graph. From this perspective, the DNNs are optimised to discriminate phoneme sequences according to their corresponding acoustic observations in MMI training.

2.2.4 Lattice-free Maximum Mutual Information Objective

The sum in the denominator in Equation 2.20 is calculated over a very large number of word sequences, which is computationally expensive and slows down the training. As a more computationally feasible approach, the lattice structure is generally employed on the word and phoneme levels (Mohri et al., 2002). However, this requires initialization of lattice paths prior to training, using a pretrained model that is typically trained on the cross-entropy loss. Due to this initialization step, sequence discriminative based DNN-HMMs stood out as a less attractive method for many researchers considering to the ease of training in more modern ASR methods, such as E2E models. To discard the necessity of generating lattices prior to training, Povey et al. (2016) introduced a lattice-free version of the sequence discriminative training, which is referred to as LFMMI training. The procedure suggests three major improvements over the standard sequence discriminative training in order to reduce the computational complexity:

- First, the language models in Equation 2.10, $P(\mathbf{w})$ are constructed on the phone-level. Since the number of target phones is much smaller than the number of words in the vocabulary, the total number of possible sequences to compute the language model is also much lower.
- The second improvement proposed is to apply frame-subsampling with a factor of 3 in the input feature sequence which means the DNN outputs are reduced to 1/3 of their size.
- Finally, a 3-state HMM topology is employed as opposed to the 5-state version of GMM-HMM based models, which is simpler and accelerates training speed.

Povey et al. (2016) and other Kaldi based models use FSTs for constructing the graph to store phoneme posteriorgrams. Recalling that the graphs H and C correspond to the phone HMMs (acoustic model) and the context-dependency tree, and considering G^{phone} is now the language model built on the phone-level, the decoding graph decomposition becomes:

$$HCG = H \circ C \circ G^{phone}. \tag{2.21}$$

According to this, the L graph is not included during training. Note that the phone LM, G^{phone} is typically a 4-gram language model where no backoff is allowed lower than 3-grams so to avoid triphones that are not present in the training data.

After the denominator and numerator graphs are created, the forward-backward algorithm is applied to update network parameters. The numerator graph is still much less complex than the denominator graphs, hence for efficient training, numerator and denominator computations are performed on the CPU and GPU respectively.

There has been a few attempts to improve the LFMMI objective, such as *boosted* LFMMI (Povey et al., 2008) and *lattice free state-level minimum Bayes risk* (LFSMBR) (Kanda et al., 2018), however substantial improvements were not consistently observed (Weng and Yu, 2019).

Therefore, the original LFMMI still remains as the standard sequence discriminative training objective for DNN-HMM based ASR at the date of this thesis.

2.3 End-to-end Training

Despite still being used in many industrial products, hybrid DNN-HMM based models have a number of drawbacks which gave rise to the need for the development of purely neural network based *end-to-end* models. The model construction procedure for the hybrid DNN-HMM models is complex and requires several preprocessing systems such as generating alignments for the DNN training. The overall recogniser is composed of different independent modules where an optimal performance of each module does not necessarily lead to global optimality. Finally, constructing a pronunciation dictionary requires language-specific expert knowledge. Due to these reasons and the recent advances in deep learning research, the popularity of end-to-end ASR systems have been growing considerably. In broad terms, end-to-end speech recognisers construct a single module that directly maps the input audio frame sequences to target sequences of words or graphemes (Graves et al., 2013; Hannun et al., 2014), which essentially reduces the overall complexity of training and construction of ASR models.

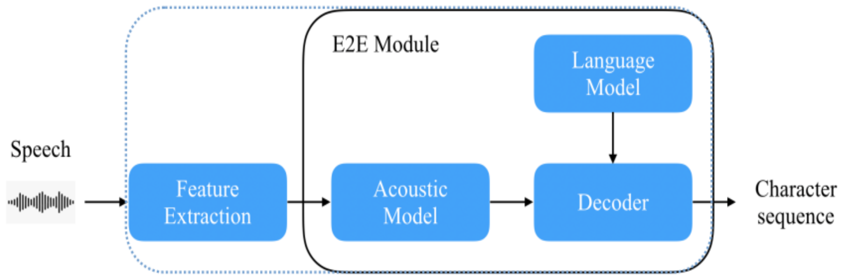


Figure 2.4: E2E speech recognition structure

The E2E module encodes acoustic speech signals into a latent space where the latent features are mapped to target word or grapheme sequences via the decoder (Figure 2.4). Note that the input to the encoder can either be the standard fixed speech features like MFCCs, mel-spectrograms or filterbanks, or directly take the raw speech signal (Ravanelli and Bengio, 2018). The decoding operation is typically based on the beam search algorithm to find the likeliest sequence of output labels (Steinbiss et al., 1994), a procedure similar to the one explained in Section 2.1, however the end goal is often to predict directly the likeliest character or other subword unit sequence, $\mathbf{Y} = y_1, ..., y_T$, and the word-to-phoneme conversion step is not required:

$$\widehat{\mathbf{Y}} = \underset{\mathbf{Y}}{\operatorname{argmax}} \log p(\mathbf{Y}|\mathbf{X}) \tag{2.22}$$

.

The decoder module often includes additional RNNs and/or language model fusion (Sriram et al., 2017) for obtaining more sensible results at the output. Note that an audio-to-text alignment step might still be required before the decoding stage depending on the objective function, such as frame-wise cross-entropy.

2.3.1 Connectionist Temporal Classification

As mentioned above, speech-to-text alignment is required in HMM based systems to perform training. To circumvent the need for alignment, the Connectionist Temporal Classification (CTC) objective was proposed for ASR by Graves et al. (2006). The introduction of CTC was considered a breakthrough in the development of end-to-end LVCSR models. The main reason for this is that the CTC approach does not require alignment between input and output sequences prior to training because the loss is computed through summing over all possible alignment paths between the input and output label sequences.

In the CTC-based end-to-end learning approach, the input acoustic feature sequence $\mathbf{X}$ are encoded into a sequence of latent representation vectors, $\mathbf{H} = h_1, h_2, ..., h_T$;

$$\mathbf{H} = \textit{encoder}(\mathbf{X}). \tag{2.23}$$

The hidden vectors are converted to probability distributions of target classes, $\mathbf{Y} = y_1, y_2, ..., y_T$, via the $\operatorname{softmax}$ operation:

$$p(\pi_t|\mathbf{X}) = \operatorname{softmax}(\textit{Linear}(h_t)), \tag{2.24}$$

where π_t is the output label at time t in the frame-wise letter sequence ϕ and *Linear*(.) denotes a linear layer. The final conditional probability of the frame-wise letter sequence, ϕ is computed using the probabilistic chain rule:

$$p(\phi|\mathbf{X}) = \prod_{t=1}^{T} p(\pi_t|\pi_1, ...\pi_{t-1}, \mathbf{X}). \tag{2.25}$$

Similar to the HMM-based approach, CTC assumes that labels at different time steps are conditionally independent (Graves et al., 2006):

$$p(\phi|\mathbf{X}) = \prod_{t=1}^{T} p(\pi_t|\mathbf{X}). \tag{2.26}$$

Then, the posterior probability of observing the output label sequence is the sum of all compatible ϕ's:

$$\begin{aligned} p_{\mathbf{ctc}}(\mathbf{Y}|\mathbf{X}) &= \sum_{\phi\in\Phi(y)} p(\phi|\mathbf{X}), \\ &= \sum_{\phi\in\Phi(y)} \prod_{t=1}^{T} p(\pi_t|\mathbf{X}), \end{aligned} \tag{2.27}$$

where $\Phi(y)$ denotes the set of all framewise paths that can be mapped to Y. The final CTC loss defined as the negative log likelihood:

$$L_{CTC} = -ln(p_{\mathbf{ctc}}(\mathbf{Y}|\mathbf{X})). \tag{2.28}$$

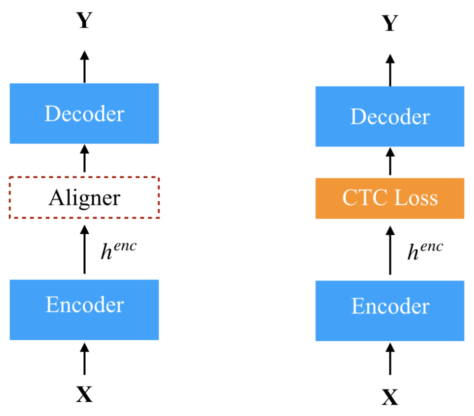
CTC introduces a blank symbol, <b> which marks the label boundaries to separate consecutive identical labels, and the augmented target sequence becomes:

$$\mathbf{Y}' = y_1, \text{<b>}, y_2, \text{<b>}, ..., \text{<b>}, y_T$$

To eliminate the need for a handcrafted pronunciation dictionary, CTC directly uses letters as the output labels $y \in V$ where V is the set of letters in the subject language and $V' = V \bigcup$ <b>. At inference, path aggregation is applied through merging the repeating characters and removing the blank symbol, <b>, to obtain the final label sequence.

2.3.2 Sequence-to-Sequence Training

Initially introduced for the neural machine translation task (Sutskever et al., 2014), sequence-to-sequence (*seq2seq*) architectures have been used extensively for building end-to-end sequence classification models. Particularly for the ASR task, the seq2seq approach has been a popular choice due to its ease of training and folding the pronunciation, language and acoustic models into a single purely neural network based model (Hannun et al., 2014; Amodei et al., 2016; Lu et al., 2016; Graves and Jaitly, 2014).



(a) Basic E2E (b) CTC-based E2E

Figure 2.5: Comparison of basic and CTC-based E2E ASR structures.

Similar to the structure depicted in Figure 2.5, seq2seq models have encoder and decoder networks to perform the mapping between input and output sequences. The encoder transforms the input speech features $\mathbf{X}$ into a hidden state vector, $\mathbf{h}^{enc} = \{h_0, h_1, ..., h_L\}$, with a fixed length L, which is often referred to as the *context* vector:

$$\mathbf{h}^{enc} = encoder(\mathbf{X}). \tag{2.29}$$

Most seq2seq based ASR models use the CTC objective, which already outputs soft alignments, so an aligner is not required at this step. They input the context vector directly to the decoder, which then predicts output labels $\mathbf{Y}$ (Figure 2.5)

$$\mathbf{Y} = \mathit{decoder}(\mathbf{h}^{enc}). \tag{2.30}$$

The encoder network typically consists of a few layers of recurrent neural networks (RNN) (Rumelhart et al., 1986) due to their sequence modeling capabilities. Audio sequences are generally much longer than text data. Due to this, standard RNNs are weak in learning long term context dependencies, essentially due to the vanishing gradient problem (Hochreiter et al., 2001). To overcome this, Long-Short Term Memory (LSTM) cells are introduced (Hochreiter and Schmidhuber, 1997). Research has shown that including information from future is helpful in training. To achieve this the bidirectional version of LSTMs, or BiLSTMs (Schuster and Paliwal, 1997) is employed to achieve better performance. Notably, Gated Recurrent Units (GRU) (Cho et al., 2014) are similarly popular recurrent cells replacing standard RNNs, which essentially operate similarly to LSTMs but with a lower complexity, hence less parameters and a faster training time. The decoder network is a language model, also commonly built with RNNs or their above mentioned variants, which is referred to as RNNLM (Tomas et al., 2011).

Neural network architectures mainly based on RNNs, and specifically (Bi)LSTMs tend to be hard to train due to their unparallelizable nature. An alternative to RNN based encoders is Convolutional Neural Networks (CNN) where the input acoustic features are first processed by a stack of 2-dimensional (2D) CNNs at the front-end of the network, then followed by time-domain 1D CNNs (Time-delay Neural Networks - TDNN) (Waibel et al., 1989) to model the temporal context dependencies (Peddinti et al., 2015). It was shown in (Zeghidour et al., 2018) that a competitive performance can also be achieved via fully CNN based networks. The idea of using the 2D CNN front-end can be also applied in conjunction with RNNs, which was shown to be effective in terms of performance gain (Hannun et al., 2014) and reduction in the number of trainable parameters (Amodei et al., 2016). The effectiveness of the fully CNN-based encoder is also shown in a more recent architecture, called *ContextNet* (Han et al., 2020) where the decoder is essentially an RNN Transducer (Graves, 2012), and in contrast the output of the encoder is fed into a so-called Squeeze-and-Excitation (SE) layer (Hu et al., 2018). The SE layer encodes the sequence of local feature representations into a global vector, and broadcasts it back to the local features. By granting access to the CNN encoder outputs into this global vector, the encoder learns context-aware features and has an increased receptive field without substantial additional computational cost. ContextNet is considered to be one of the state-of-the-art approaches within E2E ASR.

2.3.3 Attention Mechanism in Deep Neural Networks

As mentioned in the previous section, seq2seq models rely on a fixed length context vector. This can hurt training in the presence of training audio with varying durations. Also, the inference performance drops when the input and output sequences are long, especially compared to the training data. Moreover, raw seq2seq models process each input frame with equal weight which is not optimal as it leads to a lack of context and discrimination of temporal information. To overcome this, the *attention* mechanism was introduced by Bahdanau et al. (2014).

Not long after its introduction, attention was successfully implemented for end-to-end ASR, and specifically for seq2seq models in the paper 'Listen, Attend and Spell' (Chan et al., 2016) (LAS), which the model is usually referred to as in the literature. The LAS architecture consists of three main blocks: the *listener* (encoder) module serves as the acoustic model which encodes input features to hidden representations, $\mathbf{h}^{enc}$. Attention mechanism takes part in the *attender* block which weights the encoded representations based on how relevant or important each vector on time domain h_t^{enc} is for predicting the next output symbol, y_i. In practice, the attender acts in a similar way to the aligner in Figure 2.5a. The output of the attender is then fed to the *speller*, which takes the attender output, c_i and the embeddings from the previous prediction, y_{i-1} when generating a posterior probability, $P(y_i|y_{i-1}, ..., y_0, \mathbf{X})$.

The basic idea behind the attention mechanism is weighting input frames with respect to their relevancy to generating the current output. To achieve this, the attention layer learns a context vector, c_i:

$$c_i = \sum_t \alpha_{i,t} h_t^{enc} \tag{2.31}$$

where α is referred to as the attention weight. According to Equation 2.31, this operation

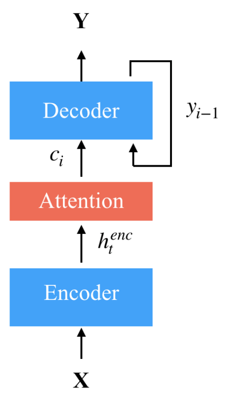


Figure 2.6: Listen, Attend and Spell structure

is essentially a weighted multiplication, where the weights at time step t are obtained via normalizing the *attention energies*, e_i into a probability distribution as:

$$\alpha_i = \underset{t}{\text{softmax}}(e_i). \tag{2.32}$$

The attention energies are essentially similarity scores:

$$e_{i,t} := \text{similarity}(h_{i-1}^{dec}, h_t^{enc}), \tag{2.33}$$

where h_{i-1}^{dec} is the hidden state of the decoder output. There are a number of ways to obtain the similarity score depending on the attention type to be applied such as dot-product, multiplicative or additive attention. Specifically in the LAS architecture, additive attention is employed where the *tanh* function is used as;

$$e_{i,t} = v^T \mathit{tanh}(W[h_i^{dec}, h_t^{enc}, \beta_{i,t}]), \tag{2.34}$$

where W is a trainable matrix, v is trainable vector and $\beta_{i,t}$ is the attention weight feedback:

$$\beta_{i,t} = \text{sigmoid}(v_\beta^T h_t). \sum_{k=1}^{i-1} \alpha_{k,t}. \tag{2.35}$$

Unlike HMMs and the CTC loss, the attention mechanism does not have the conditional independence assumption. The training objective is computed using the probabilistic chain rule similar to Equation 2.26:

$$p_{\textbf{att}}(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{T} p(y_t|y_1^*, ..., y_{t-1}^*, \mathbf{X}), \tag{2.36}$$

where y_t^* are the ground truth labels.

Self-Attention

The basic attention mechanism explained above learns a context vector between the encoder and decoder states, however there is also context between the states of the encoder that needs to be modeled. To achieve this, the *self-attention* layer was proposed (Vaswani et al., 2017), which calculates a similarity score between the encoder states to learn how relevant an encoder state from the past or the future, $h_{t\pm u}$ (where $u = 1, 2, ..., T$) is, for generating the encoder state at the current time step h_t.

Self-attention is based on the principles of query-retrieval modeling given a database with keys and values (Garcia-Molina et al., 1999) and a query to search for. From the perspective of seq2seq structures, the query, Q is projection of the decoder hidden state h^{dec} through a trainable matrix. The key, K is the same as the query but from the hidden states of the encoder, and the value, V, directly corresponds to h_t^{enc} in Equation 2.31. The similarity scores are obtained similar to Equation 2.34, but using the *dot-product* attention, QK^T. This score is then scaled by the square root of the key and query dimensions, d_K. The attention weights are obtained after the softmax operation to ensure the rows in the similarity score add up to 1:

$$\alpha_{i,t} = \text{softmax}(\frac{QK^T}{\sqrt{d_K}}). \tag{2.37}$$

Then, injecting the above equation into Equation 2.31 results in the following formula to summarise self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_K}})V. \tag{2.38}$$

Multiple self-attention blocks can be placed in the network architecture where the weights are learned in parallel. In this context, each attention block is referred to as head, where $\textit{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, and the multi-head attention (MHA) is computed through concatenating heads:

$$\text{MHA}(Q, K, V) = \text{concat}(\textit{head}_1, ..., \textit{head}_N), \tag{2.39}$$

where N is the number of heads that is predefined prior to training.

Speech sequences are generally much longer than word sequences, hence for every time step, calculating the attention matrix over the entire audio can be computationally expensive. Povey et al. (2018b) propose restricting the time context for which the attention scores are calculated. Notably, the aforementioned study is also one of the first implementations of attention within the hybrid DNN-HMM framework.

2.3.4 Transformers

One of the major challenges in training DNNs for sequence prediction or transduction tasks is they are computationally expensive due to the high number of trainable parameters. To reduce model complexity while maintaining the context modeling power, Vaswani et al. (2017) introduced the ‘Transformer’ architecture. Transformers are essentially seq2seq models with typical encoder-decoder structures. However, instead of containing RNNs or CNNs in both the encoder and decoder blocks and having an attention layer in between, the transformers solely consist of combinations of self-attention and fully-connected layers (see Figure 2.7).

Unlike LSTMs, the transformer layers do not involve any recurrence, which implies that the data sequence order is not taken into account when computing the gradients. In order to impose

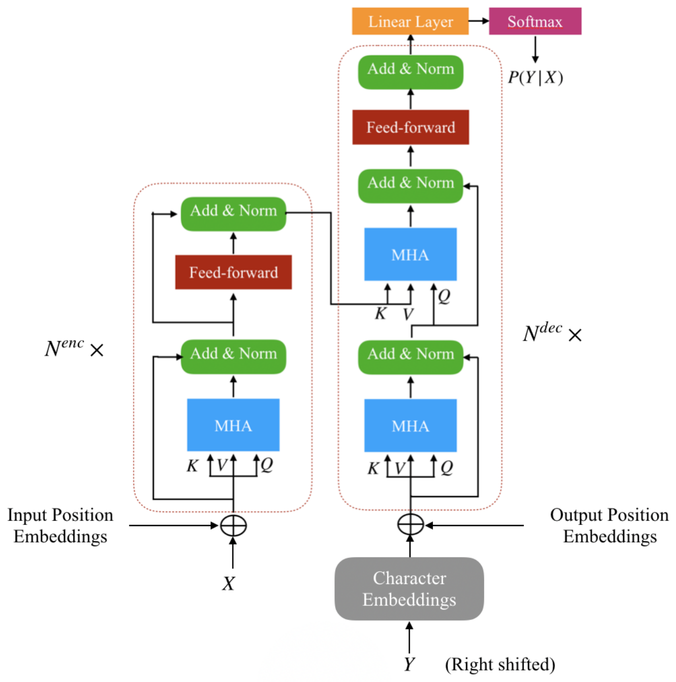


Figure 2.7: The Transformer architecture.

positions of the data points during training positional embeddings are added to the training data at the front-end of the network. While initially sinusoidal embeddings are proposed for this, it was shown to be possible to learn positional embeddings through CNNs. The architectures that use convolutional positional embeddings are generally referred to as *conformers* (Gulati et al., 2020).

After adding the positional embeddings, the data is fed to multi-head attention (MHA) vectors and *layer normalization* is applied (Ioffe and Szegedy, 2015). The normalised hidden vectors are projected to 2D via a linear layer, obtaining the final attention values. The overall operation starting with the multi-attention layer until retrieving linear attention vectors can be stacked to construct deeper networks and increase modeling power. This stack of Transformers between the input features and output of the final Transformer block constitutes the encoder block.

The decoder operates similarly to the encoder in a sense that the output embeddings of the previous time-steps and their positional embeddings are fed to an MHA module. This first MHA is masked to not include embeddings from future time steps because future words would be needed to compute the output embeddings of the current time step, which are not yet generated.

The attention scores are then fed to a *cross-attention* layer. This layer is operationally similar to self-attention as the attention scores are computed via the dot-scale similarity as in Equation 2.33. In this case, the attention scores coming from the output embeddings are considered as queries, and key and values come from the output of the encoder. The cross-attention is then followed by another linear layer before going to the final classification layers.

Transformers have drawn the attention of the deep learning research community since their its introduction due to considerable reduction in model complexity, speeding up training while achieving competitive or better results compared to the state-of-the-art. Specifically within the context of ASR, transformers have been applied successfully in both hybrid DNN-HMM (Wang et al., 2020) and E2E frameworks (Gulati et al., 2020).

2.3.5 Language Model Fusion

In essence, end-to-end training of ASR systems requires paired audio and text data, i.e. $\mathbf{Y}_i = f(\mathbf{X}_i), i \in N$ where $\{\mathbf{X_i}, \mathbf{Y_i}\}$ represent the input audio and the output text pairs for the i^{th} training sample, f is the end-to-end ASR system, and N is the number of samples in the train set. On the other hand, conventional DNN-HMM ASR systems can leverage separate language models trained on external (unpaired) text data, during inference. From this perspective, the paired data restriction stands out as a limiting factor for end-to-end system's modeling power. Addressing this, there have been a number of ways proposed previously to incorporate an external language model trained on unpaired text data during decoding, namely the *shallow*, *deep* (Gulcehre et al., 2015) and *cold* fusion (Sriram et al., 2017) approaches. These techniques differ from each other with respect to at which point they are integrated with the ASR model's output probability distribution computation. Note that the language models are assumed to be neural LM in this scheme (the details of a neural LM training will be explained in Section 4.1.2).

Shallow fusion is the simplest language model integration approach where the word priors obtained from a pretrained external language model are simply added to the posteriors coming from a pretrained acoustic model. Recall the formulation of end-to-end ASR in Equation 2.22, the updated probabilities are computed as:

$$\widehat{\mathbf{Y}} = \underset{\mathbf{Y}}{\text{argmax}}(\log p(\mathbf{Y}|\mathbf{X}) + \gamma \log p_{LM}(\mathbf{Y})) \tag{2.40}$$

where γ is the language model weight which can be tuned after training.

Instead of using the word priors from a pretrained LM, deep fusion (*DF*) fuses the hidden states of pretrained language and acoustic models through a parametric gating procedure (Gulcehre et al., 2015):

$$g_t = \text{sigmoid}(\mathbf{v}_g^\top \mathbf{h}_t^{LM} + b_g) \tag{2.41}$$

$$\mathbf{h}_t^{DF} = [\mathbf{c}_t; h_t^{dec}; g_t h_t^{LM}] \tag{2.42}$$

$$P(y_t|\mathbf{h}^{enc}) = \text{softmax}(\mathbf{W}^{DF}\mathbf{h}_t^{DF} + \mathbf{b}^{DF}) \tag{2.43}$$

where $\mathbf{c}$, $\mathbf{h}^{enc}$ and $\mathbf{h}^{\text{dec}}$ are the context, encoder and decoder hidden state vectors of the pretrained acoustic model respectively (recall notation in Section 2.3.2). The parameters $\mathbf{v}_g, \mathbf{b}^{DF}$ and the matrix $\mathbf{W}^{DF}$ are learned through training the model on a smaller proportion of the training data where the rest of the model parameters are kept fixed. Through fixing most of the model parameters, fine-tuning can be achieved efficiently.

Both shallow and deep fusion approaches are late training integration procedures and require pretrained acoustic and language models. In contrast, cold fusion is an *early training integration* approach where the acoustic model is trained from scratch with a pretrained language model. Note that all of model parameters in cold fusion are learnable. Due to early training integration, cold fusion is costlier than shallow and deep fusion.

2.3.6 Sentence Piece Tokenisation

Most of the traditional large vocabulary continuous speech recognition (LVCSR) systems employ phonemes as the subword units that constitute the target class set to predict, and the predicted phonemes are converted back to words using a lexicon. As mentioned in Section 2.2.1, this procedure requires a predefined mapping between words and phonemes, usually referred to as the pronunciation dictionary, which needs to be curated by experts. Such resources may not be available for every language or speech recognition task. For this reason, researchers have been exploring ways of using different methods for tokenizing words into smaller subword units, such as graphemes (Killer et al., 2003; Le et al., 2019), or syllables (Wu et al., 1997; Ganapathiraju et al., 2001).

Inspired by the NMT research, one popular approach for generating a set of subword units is a text data compression method called *Byte Pair Encoding* (BPE) (Shibata et al., 1999a), which was successfully employed in recent E2E ASR systems (Shin et al., 2019). The main operation for constructing the BPE units is substituting pairs of consecutive characters which frequently appear in the text with single tokens (or characters) Shibata et al. (1999b). This operation will be iterated until either no pair of consecutive characters appears frequently or all characters are used up. Given a fixed size vocabulary (typically between 1000 and 5000 in E2E ASR), BPE ensures that most frequent words are represented by a single token, while less frequent words are divided into more frequent subword BPE tokens.

BPE is essentially based on the frequency of appearance of patterns and characters in a text corpus and it is unsupervised and deterministic. Once the vocabulary of BPE units is established, words can be reconstructed using only the existing BPE's, but this may cause word reconstruction errors, implying a loss of modeling power in general. Improvements over BPE have been observed using the *WordPiece* algorithm (Schuster and Nakajima, 2012), which chooses the pairs that maximise the likelihood of patterns instead of choosing the most frequent ones, which is the subword tokenization employed in the famous *BERT* natural language processing machine (Devlin et al., 2018). To model the subword tokenisation in a probabilistic manner, (Kudo, 2018) introduced a method referred to as *subword regularisation* that ranks all possible subword tokenisations given a word based on likelihoods, and removes the least likely tokens. The likelihoods of the subwords are then obtained via a unigram language model trained on a text corpus. This subword tokenisation method is generally referred to as *unigram subword regularisation* and consistent improvements have been reported compared to the standard BPE approach (Kudo, 2018). Table 2.1 gives an example for various subword tokenisation methods including phonemes, graphemes, BPE and unigram models:

2.4 Research ASR Frameworks

As mentioned above, this thesis tackles the task of automatic lyrics transcription from two major streams in ASR research: DNN-HMM and E2E models. The models presented in Chapters 4

Subword Tokenization	Tokens
Phonemes	D IY P <silence> L ER N IH NG
Graphemes	D E E P <blank> L E A R N I N G
BPE	_DEEP _LEAR NING
Unigram	_DEEP _LEARN ING

Table 2.1: Comparison of different tokenization methods for the phrase '*DEEP LEARNING*'. Underscore '_' is for word beginnings. For the Unigram and BPE approaches, a tokeniser is trained on a the lyrics corpus in (Demirel et al., 2021b) with the vocabulary size of 1000.

and 5 are built using two open-source toolkits: Kaldi (Povey et al., 2011) for the DNN-HMM based approach and recently introduced SpeechBrain (Ravanelli et al., 2021) for the end-to-end model. This section covers the properties, advantages and disadvantages of these frameworks and finally briefly covers other existing frameworks which researchers in ASR field use to report their results. For convenience for the readers, a table is provided at the end of this section that provides an overall comparison between general ASR frameworks available for research.

2.4.1 HMM-based Toolkits

HMM-based toolkits have been available for research for few decades already and are used extensively within the development of ASR research. Among the ones that were more popular during the pre-deep learning era, the Hidden Markov Model Toolkit (HTK) (Young, 1993) and CMU Sphinx (Lamere et al., 2003) have been used the most extensively. Originally developed for the Japanese language, Julius (Lee et al., 2001) has also experienced great attention within the research community due to its multiple language support (similar to CMU Sphinx) and capability for operating in real-time. Last but not least, the RWTH Toolkit (Rybach et al., 2011) is another popular HMM-based toolkit that provides a comprehensive list of speech processing pipelines, similar to Kaldi, such as sequence discriminative training, lattice processing and speaker adaptive training.

Kaldi

Arguably the most commonly used and comprehensive framework within the DNN-HMM based approach is Kaldi (Povey et al., 2011). There are several factors for Kaldi's popularity: first is that there is a large active research community including researchers from varying institutions that are involved in the constant development of the toolkit and implementing the state-of-the-art concepts in research. Secondly, Kaldi has been proven to work well on different tasks such as speech enhancement, speaker identification, key-word spotting as well as lyrics alignment (Gupta et al., 2020; Demirel et al., 2021a), and industrial applications.

On the other hand, most of Kaldi's code is written in C++, which allows it to be deployable on varying operating systems. Due to nature of its implementation Kaldi requires compilation before execution which may be considered to be not very flexible against any custom modifications. Moreover, Kaldi still lags behind the most recent deep learning research as it does not fully support seq2seq models, transformers or contrastive learning methods such as wav2vec. To integrate WFST's with the aforementioned modern deep learning techniques, the new version of Kaldi, *k2*[3] was introduced, however this is still in the development stage and not ready for large-scale applications.

2.4.2 End-to-end ASR Toolkits

As developing E2E models is becoming increasingly accessible due to open-source deep learning libraries, such as Theano (Team et al., 2016), TensorFlow (Abadi et al., 2016), Keras (Ketkar, 2017b) and Pytorch (Ketkar, 2017a), multiple open-source ASR toolkits have been emerging in recent years. One of the first successful large scale applications of such models, Mozilla's DeepSpeech (Amodei et al., 2016) is considered as a breakthrough in the transition from hybrid DNN-HMM systems to purely E2E models. This model was later implemented in many other more recent E2E ASR toolkits, like TensorFlowASR[4]. Although this TensorFlow based repository provides a good baseline for developing E2E systems, it does not support other relevant tasks to ASR, such as speech enhancement and separation, multi-channel ASR, and speaker detection. There are a number of more comprehensive toolkits that provide training pipelines for the aforementioned tasks. Among these, ESPNet (Watanabe et al., 2018) is one of the most popular as its structure resembles Kaldi which is convenient for ASR researchers who want to transition from HMM to E2E models. Note that Gupta et al. (2020) also used ESPNet for building E2E models within the context of ALT. Notably, Facebook's *fairseq* (Ott et al., 2019)

[3]Alpha version can be accessed from *https://github.com/k2-fsa/k2*.
[4]Can be accessed from *https://github.com/TensorSpeech/TensorFlowASR*.

provides a repository with pretrained models and training pipelines. This toolkit also provides the recently introduced *wav2vec* model (Schneider et al., 2019) which is a pretrained speech representation that could perform similarly to the state-of-the-art while requiring much less data (Baevski et al., 2020). On the other hand, fairseq mainly adopts Facebook Team's research and is less flexible in implementing various E2E structures from scratch. Finally, *OpenSpeech* (Kim et al., 2021) is a recently introduced E2E toolkit which covers the state-of-the-art approaches similar to ESPNet and SpeechBrain (see below), however it does not provide pretrained models. Other notable toolkits include Espresso (Wang et al., 2019b), RWTH-RETURNN (Zeyer et al., 2018), Lingvo (Shen et al., 2019) and NeMo (Kuchaiev et al., 2019) which are mostly designed for specific purposes or datasets.

SpeechBrain

Although the large amount of available open source software provides a great opportunity for researchers to develop their own speech processing modules, the toolkits listed above have varying workflows, programming languages and coding styles. This emerges as a challenging factor when various codebases are intended to be integrated together for developing an overall all-in-one module. To overcome this challenge and provide a unified framework, Ravanelli et al. (2021) recently introduced Speechbrain which is purely implemented in Python as the programming language and Pytorch as the deep learning library. Speechbrain is designed to be flexible and convenient for developers proficient in Python language while supporting training pipelines for many of the major ASR tasks.

2.4.3 Commercial APIs Available for Research

Today's digital technology industry enjoys the wide range of commercial speech-to-text (STT) services. Voice assistants like Apple's Siri and Amazon's Alexa are already well-known and widely used by the public. Surely, STT software is not limited to such commercial products designed for end-user applications. Three leading and well-recognised software companies have made their STT APIs available on the web, namely "Google Cloud Speech API", "IBM's Watson", "Amazon Transcribe" and "Microsoft - Azure Speech Services". Although these API's allow limited usage for free and require a service fee to support large-scale applications and training custom models on custom datasets, they have been considered to be of great value by many researchers as they provide an opportunity for a quantitative comparison of research ASR software with commercial STT products. In addition to these, we also added Nvidia's Jasper (Li et al., 2019a) which is another E2E model based on 1D CNNs, similar to the architecture proposed by Zeghidour et al. (2018).

2.4.4 Summary

Below is provided Table 2.2 that shows an overall comparison of the commonly used toolkits available for research. To get a hint on which framework is to be more suitable for the current task, we compare them in several technical and practical aspects: ASR type, support of subword tokens (target classes), model training difficulty, computational complexity, memory requirements, flexibility for modification, level of open-source, programming language, real-time support, and the overall word-recognition rates based on previous comparative studies (Këpuska and Bohouta, 2017; Filippidou and Moussiades, 2020; Georgescu et al., 2021). The training difficulty is related to the data preprocessing steps required prior to training and tuning of hyperparameters for the system to converge. Computational complexity is related to the number of trainable parameters. The flexibility property is proportional to the support for different models, datasets and tasks. The ASR performance index is a relative measure and values are included according what were reported in the aforementioned comparative studies.

Among the HMM based toolkits, it is clear that Kaldi stands out to be the most preferable due to its word recognition performance, support for a wide range of ASR tasks, benchmark datasets and training frameworks, and a fairly easy pipeline for training custom models.

The memory requirements for the speech-to-text APIs listed in Section 2.4.3 are the lowest in Table 2.2 as the computation is done on cloud servers instead of the local machine. On the other hand, these APIs are publicly available for inference only, and do not support training custom models. Moreover, the amount of inference is limited and further usage requires subscription, which makes them less appealing for large-scale research applications. In contrast, Nvidia's Jasper provides an open-source repository where training is possible. However, it supports a single E2E-ASR type, which is the fully CNN based architecture (Li et al., 2019a). As reported by Georgescu et al. (2021), this model has a very large number of trainable parameters making it difficult to train compared to the rest of the models in Table 2.2. Facebook's fairseq repository contains smaller networks than some other state-of-the-art approaches in ASR (Baevski et al., 2020), however the list of models is mostly limited to Facebook's publications. ESPNet and SpeechBrain share essentially similar properties such as ASR performance, flexibility and computational complexity, but the latter is purely *pythonic*, which provides a unified coding framework and easier to create models based on custom models and datasets. Motivated by the concerns listed above, we decided to use SpeechBrain as the E2E ASR framework in our study.

ASR System	System Type	Subword Tokens	Training Difficulty	Computation Complexity	Memory Req.	Flexibility	Programming Language	Real-time	ASR Performance
HTK	GMM-HMM	Phoneme	● ● ● ● ●	● ● ○ ○ ○	● ○ ○ ○ ○	● ● ○ ○ ○	C++	Yes	● ● ◐ ○ ○
CMUSphinx	GMM-HMM	Phoneme	● ● ○ ○ ○	● ● ● ○ ○	● ○ ○ ○ ○	● ● ● ○ ○	C++	Yes	● ● ● ○ ○
Kaldi	GMM-HMM, DNN-HMM, LFMMI	Phoneme, Grapheme, BPE	● ● ● ○ ○	● ● ● ● ○	● ● ● ○ ○	● ● ● ◐ ○	C++, Perl Python	Yes	● ● ● ● ◐
TensorFlowASR	E2E	Char	● ● ● ○ ○	● ● ● ● ○	● ● ● ● ○	● ● ● ◐ ○	Python	Yes	● ● ● ● ○
ESPNet	E2E	Char, BPE	● ● ● ◐ ○	● ● ● ● ◐	● ● ● ◐ ○	● ● ● ● ○	Python, Bash	Yes	● ● ● ● ◐
Facebook - fairseq	E2E	Char, BPE	N/A	● ● ● ● ○	● ● ● ◐ ○	● ● ● ○ ○	Python	Yes	● ● ● ● ◐
SpeechBrain	E2E	Char, BPE unigram	● ● ○ ○ ○	● ● ● ● ◐	● ● ● ◐ ○	● ● ● ● ○	Python	No	● ● ● ● ◐
Google Cloud - Speech API	DNN-HMM	Phoneme	N/A	N/A	● ○ ○ ○ ○	● ○ ○ ○ ○	Python	Yes	● ● ● ● ○
Microsoft Bing - Azure Speech	DNN-HMM	Phoneme	N/A	N/A	● ○ ○ ○ ○	● ○ ○ ○ ○	C++, Python JavaScript	Yes	● ● ● ● ○
IBM - Watson	DNN-HMM	Phoneme	N/A	N/A	● ○ ○ ○ ○	● ● ● ○ ○	JavaScript	Yes	● ● ● ● ○
Nvidia - Jasper	E2E	Char	● ● ● ● ◐	● ● ● ● ●	● ● ● ● ●	● ● ○ ○ ○	Python	Yes	● ● ● ● ◐

Table 2.2: ASR Frameworks commonly used in research.

2.5 Advances in Automatic Lyrics Transcription Research

ASR technologies have experienced considerable improvement in the course of the last few decades, letting them be integrated into large-scale commercial products, which have brought convenience to people's lives and already has an important impact on the global society. On the other hand, ALT has not been utilised in large-scale industrial applications yet, as word recognition performance from singing had not reached similar levels as for ASR prior to this thesis. In this section, an outline of the recent advances in ALT research is provided. The section begins with the initial attempts that tackled the ALT problem with limited data and conventional machine learning methods. Then, the commonly used datasets in research are briefly introduced. Third, data augmentation methods for singing data are described, which has been a popular research direction for improving ALT. Then, various methods previously proposed to leverage the musical information embedded in the singing data are discussed. The chapter finishes by providing a numerical comparison between state-of-the-art ALT approaches. In this, we include results for both DNN-HMM and E2E ASR approaches.

The early attempts for ALT prior to the advent of deep learning and the availability of large training datasets focused on leveraging already existing speech recognisers. Fujihara et al. (2006) compose a lyrics focused language model with a pretrained acoustic model for speech. Mesaros and Virtanen (2010a,b) employ the traditional GMM-HMM models trained on speaker-adaptative features and the results improve when a lyrics-focused language model is incorporated. Hansen (2012) trains Multilayer Perceptrons (MLP) using MFCC and Temporal Pattern (TRAP) features. Even though these results are hard to compare due to reporting results on different datasets, it is worth mentioning them as these results constitute the initial baselines for lyrics transcription research.

The word recognition rates of lyrics transcribers published in the literature have improved drastically since the recent introduction of two open-source datasets, namely *DAMP* [5] and *DALI* (Meseguer-Brocal et al., 2019). *DAMP* was initially used for training an ALT module by Dabike and Barker (2019), marking a breakthrough in the word recognition rates reported previously. Gupta et al. (2018) used *DALI* for the first time, to train an acoustic model for polyphonic recordings. Since these works, both datasets have been used to benchmark different

[5]URL: https://ccrma.stanford.edu/damp/, accessed in Jan, 2022.

ALT models.

A number of open-source evaluation sets have been used to report word recognition rate results. Among these, Dabike and Barker (2019) presented a subset of the *DAMP* dataset (notated as *DAMP*^{test} in this thesis) with hand-checked annotations as a new benchmark test set of a cappella recordings. In addition, the NUS Sung and Spoken Lyrics Corpus (Duan et al., 2013) was utilised in our previous work (Demirel et al., 2021c) to compare word error rates (WER) on speech and singing data. For evaluating ALT from polyphonic recordings, the *Jamendo* dataset (Stoller et al., 2019) is mostly used for benchmarking, which is an open source dataset containing 20 polyphonic recordings from the Jamendo catalogue, with corresponding lyrics and word-level time annotations. Despite not having publicly available recordings due to copyright, the Hansen (Hansen, 2012) and the Mauch (Mauch et al., 2011) datasets have also been commonly used in the literature, including the most recently introduced Music Information Retrieval Evaluation Exchange - Automatic Lyrics Transcription (MIREX - ALT)[6] public evaluation. More detailed information and analysis of the commonly used training and evaluation datasets are provided in Chapter 3.

On one of the earlier releases within Smule's *DAMP* repository, the *DAMP - multiple songs* dataset, where no line-level lyrics annotations were provided, Kruspe (2016) trained a DNN-HMM system on weak labels obtained from a speech-trained forced aligner, and reported 77% Phoneme Error Rate (PER) on a subset of the aforementioned dataset. To generate labels and line-level weak annotations to be used for training, Gupta et al. (2018) automatically segmented recordings with respect to words detected by a pretrained speech recogniser. Since the line-level lyrics are already provided in the later released *DAMP* corpus, namely *Dsing! - 300*$\times$*30*$\times$*2*, the interest in semi-supervised learning for ALT decreased, and researchers focused on generating 'synthetic' singing data.

In her work, Kruspe (2015) transformed speech data into singing through digital signal processing (DSP) techniques such as adding vibrato, time stretching and pitch shifting (transposition) to capture the acoustic variations in singing and yet observed only a moderate performance improvement using the 'songified' training data. In a later attempt, Basak et al. (2021) applied *style-transfer* using opera data to convert speech into singing. In particular, the authors resyn-

[6]To access; https://www.music-ir.org/mirex/wiki/2020:Lyrics_Transcription

thesise speech by adapting fundamental frequency contours and spectral envelope parameters estimated from opera data. However, the melody and the lyrics are not necessarily aligned, potentially causing unrealistic prosody and pronunciation in the synthesised vocals. To cope with this, Zhang et al. (2021) developed a speech-to-singing conversion method, namely *PD-Augment* where pitch and duration are readjusted in the original speech utterances guided by melodic patterns extracted from a MIDI dataset. The authors report a considerable improvement compared to training on non-augmented data.

Task-specific language models have been shown to be beneficial for the development of ASR systems. Similarly in ALT, building a language model on a lyrics-specific corpus has been the preferred method since the early work of Fujihara et al. (2006). Recently, Gupta et al. (2020) showed that a lyrics-specific language model is superior to using either speech-only or speech-mixed-with-lyrics corpus. Notably, most of the *WER* improvement in their experiments was due to the lyrics-specific language model. In our recent work, we observed around 25% relative *WER* improvement when a neural network based language model, namely RNNLM, is used (Demirel et al., 2020a), which is later shown in Chapter 4. Finally, Zhang et al. (2021), scraped over 45 million lyrics lines from the web and trained a Transformer-based LM, which verified a lyrics-LM being beneficial against a speech LM in the context of E2E ALT.

The prosodic elements of sung utterances are often motivated by melody construction and expressivity which may affect how words are pronounced. Singers tend to utter longer vowels according to statistical observations (Duan et al., 2013; Dabike and Barker, 2021; Demirel et al., 2021c). To model longer vowels, Gupta et al. (2018) proposed representing word pronunciations with repeated vowels in the pronunciation model, which resulted in a considerable improvement. Kawai et al. (2017) showed that including pitch in the feature space was beneficial on a private Japanese language dataset. Later, Dabike and Barker (2021) concluded that the improvements observed due to including pitch features diminishes as the training data gets larger.

One of the major challenges in ALT is transcribing words when there is music accompaniment, i.e. polyphonic recordings. There have been few methods proposed to leverage music related information present in the acoustic scene of polyphonic recordings, in order to improve overall performance in ALT. Gupta et al. (2020) leveraged the genre annotations in the *DALI* dataset and labeled utterances with genre-tags, essentially constructing a genre-aware pronunciation model. While improvements were especially noticeable for the task of audio-to-lyrics alignment[7], modeling genre-specific phonemes results in a very large set of target classes and a huge acoustic model, however this was mentioned to be extremely memory consuming (Demirel et al., 2021a).

The state-of-the-art

In Table 2.3, previously reported results for both the DNN-HMM and E2E approaches are compared for four commonly used evaluation sets: *DAMP^{test}*, *Hansen*, *Mauch* and *Jamendo*. The *WER* scores in Table 2.3 do not include my previous publications, although these report the state-of-the-art results. The improvements achieved in my work are compared with the previous literature in Chapter 6, as outlined in Section 1.4 where the relevant sections of this thesis can be found.

The E2E approach for ALT was first implemented by Stoller et al. (2019), who used a basic CTC structure with a U-Net as the encoder. In their work, Gupta et al. (2020) tested an attention-based seq2seq model but obtained worse results, possible due to training on a much smaller dataset. In the same work, the authors compared their E2E model with the genre-aware DNN-HMM LFMMI model mentioned above, which had considerably better results than Stoller et al. (2019).

DNN-HMM models were also considered to be the better option for monophonic recordings until very recently. Zhang et al. (2021) trained a Transformer with convolutional positional embeddings on the *DAMP* dataset, and reported 6% absolute WER improvement over the previous best results in Demirel et al. (2020a). Although their results for models trained on non-'PD-augmented' data are still worse (27.6% WER) than the best performing DNN-HMM model, the study shows the possibility for bridging the gap between E2E and DNN-HMM based systems within the context of ALT. Note that this paper has not yet been peer-reviewed, and it is published only on *arxiv*.

[7]This work is still the state-of-the-art for audio-to-lyrics alignment to date.

Dataset	DNN-HMM		End-to-end	
DAMP^{test}	LFMMI (Dabike and Barker, 2019)	19.60	Transformer w PD-augment (Zhang et al., 2021)	**9.60**
Hansen^{poly}	LFMMI +	**47.01**	Seq2seq w/ Attention + Lyrics LM (Gupta et al., 2020)	80.1
Mauch	+ Genre-labeled phonemes	**44.02**	U-Net + CTC (Stoller et al., 2019)	70.9
Jamendo	+ Vowel extended lexicon (Gupta et al., 2020)	**59.57**		77.8

Table 2.3: State of the Art in ALT (in terms of % *WER*).

Chapter 3

Singing Data for Word Recognition

In this chapter, singing data used in building and evaluating the lyrics transcription system is studied in order to establish a better understanding of how it is processed and leveraged within this task. First, the training and evaluation datasets are examined which are commonly used in ALT research and also included in our experiments in Chapters 4, 5 and 6. Then, a new evaluation set is introduced, namely *DALI*^{test240}, which is curated for benchmarking lyrics transcription results on polyphonic recordings. Finally, a statistical and quantitative comparison between speech and singing data is provided, where their distinct properties are identified that are taken into consideration for adapting conventional ASR systems to singing data.

3.1 Datasets

As mentioned in Chapter 2, the recent availability of training datasets - *DAMP*-Sing! 300$\times$30$\times$2 and *DALI* - had a great impact on the development of ALT research, and most of the reproducible studies have used either of these datasets in building their transcribers. On the other hand, evaluation in ALT has been on a variety of publicly available datasets, each of them having unique properties. Thus, curating a single framework for testing ALT models through including all the publicly available evaluation sets would potentially contribute to establishing a more comprehensive outlook of varying models at operation. In this section, we introduce the datasets employed in our study for the training and testing stages, and highlight their specific properties. The overall statistics of the subject datasets are provided at the end of this section in Table 3.1.

3.1.1 Training Data

Each data sample for training the transcriber is typically a line of lyrics. This requires time stamps pointing to the beginning and end of each line of lyrics. The only publicly available datasets with such time annotations and sufficient size are the *DAMP* and *DALI* datasets (first mentioned in Section 2.5). These datasets have distinct characteristics making them suitable for a specific scenario within the ALT problem, which are discussed below:

DAMP - Sing!300x30x2

The *DAMP* repository consists of karaoke recordings collected via a commercial mobile application, Smule. The performers are the users of the mobile app, and generally amateur singers. Most recordings are monophonic and there is no dominant background music accompanying singing. Note that the acoustic properties of the recording environment vary due to the data creation procedure. This is typically reflected as variance in reverberation or echo, distorted singing voice signals, users' languages and recording locations, making the dataset fairly representative of real-world solo singing data.

There are a few separate datasets within the *DAMP* repository.[1]. Specifically, this study uses the Sing! 300x30x2 dataset within the *DAMP* repository, where line-level timing annotations of the corresponding lyrics are provided. In addition, the dataset was initially released as a clean version of its predecessors while having singers' gender balance, and containing samples from 30 different countries. Because of this, the dataset provides a large variety of accents, prosody and pitch ranges, making it even more suitable for the ALT task.[2].

DALI

The *DALI* dataset is a large corpus of commercial polyphonic recordings, also with weak labels obtained via a semi-supervised learning strategy (Meseguer-Brocal et al., 2019) where the time annotations are also provided on the lyrics line level. The recordings can be obtained via the Youtube links and the code repository provided at *https://github.com/gabolsgabs/DALI*. *DALI* is frequently used in research to train polyphonic ALT models (Gupta et al., 2020; Basak et al.,

[1]The data can be retrieved from *https://ccrma.stanford.edu/damp/* upon request to Smule.

[2]The data identifiers and annotations are shared at *https://github.com/emirdemirel/ALTA/s5/data/train_damp*.

2021) The version reported in this study (*DALI*-v2.0) (Meseguer-Brocal et al., 2020) has around 200 hours of data, however a subset of the original version (156 hours) is used due to what was available on the links provided at the time of retrieval[3]. Note that, there were overlapping songs in our initial version of *DALI* set with some of the samples in the evaluation sets. Hence, we removed the overlapped songs from the training set.

The Lyrics Corpus

As mentioned in Section 2.5, constructing a lyrics-specific language model is found to be beneficial for ALT (Gupta et al., 2020). For this, the lyrics corpus introduced by Dabike and Barker (2019) is taken as the starting point, which consists of two resources: the songs of artists from the Billboard charts between the years 2015 and 2018[4], and the sentence annotations in the training subset of the *DAMP* dataset. This corpus is extended with the lyrics of the train split of the *DALI* data to form the final version of our *Lyrics Corpus*. For scientific evaluation, the lyrics of the songs are excluded from the test sets for training the language model.

3.1.2 Evaluation Data

NUS Sung and Spoken Lyrics Corpus

The NUS Corpus (Duan et al., 2013) contains sung and spoken performances of 20 pieces in English language by 12 singers with varying native and non-native accents. The dataset includes manual annotations of phonemes, which enabled a data-driven comparison of sung and spoken utterances. The data and annotations are exploited in our computational pronunciation analysis (Demirel et al., 2021c), which will be discussed in Chapter 4.

DAMP^{test}

Dabike and Barker (2019) who first reported results on DAMP - Sing!300x30x2 dataset, introduced a test split where the annotations are manually verified or fixed. This set, generally referred to as *DAMP^{test}* is also used to compare results by Demirel et al. (2020a); Zhang et al.

[3]Similar to the *DAMP* dataset, the metadata information for our version of *DALI* set used in training are shared publicly at *https://github.com/emirdemirel/ALTA/s5/data/train_dev*.

[4]This was first curated in Dabike and Barker (2019), then utilised by Demirel et al. (2020a).

(2021); Demirel et al. (2021b). The dataset contains a higher number of singers compared to other test sets and similar to the *DAMP* data overall.

Hansen

Introduced by Hansen (2012), the Hansen dataset consists of 10 commercial pop songs in English released in early 2010s, which is used as one of the evaluation sets in the MIREX - ALT challenge. Despite its limited musical variability, this set is interesting for ALT research as it contains the original vocal stems making a direct comparison between monophonic and polyphonic recordings possible (referred to as $\textit{Hansen}^{\textit{mono}}$ and $\textit{Hansen}^{\textit{poly}}$ respectively).

Mauch

The *Mauch* dataset (Mauch et al., 2011) is another set used in the MIREX - ALT challenge which has 20 commercial songs in English and a higher variability in terms of the release years and music styles, though it still mostly consists of pop music. Word-level time annotations are also provided within this dataset, and hence it is also used as an evaluation set for the *audio-to-lyrics alignment* task.

Jamendo-lyrics

The above-mentioned datasets are prone to have *evaluation bias* as both *Hansen* and *Mauch* consist of songs in similar styles. In their work, Stoller et al. (2019) introduced the *Jamendo* dataset which consists of 20 songs from a variety of music genres including *hiphop*, *metal*, *reggae*, *R&B* and *country* (Stoller et al., 2019) which provides a higher musical (and lyrical) variability. In addition, the recordings have an open source license (Creative Commons) and can be publicly retrieved at *https://github.com/f90/jamendolyrics*. Similar to the *Hansen* and *Mauch* dataset, the *Jamendo*-lyrics dataset also has manually annotated word-level time stamps and is used to evaluate lyrics alignment models.

$\textit{DALI}^{\textit{test240}}$

Data within *DALI* is also used to evaluate ALT from polyphonic recordings (Gupta et al., 2020; Basak et al., 2021; Zhang et al., 2021). However, usually sufficient information is not provided on the curation of these splits and data identifiers are not publicly shared, which makes evaluation across different papers difficult. For these reasons, we have curated a new split of the *DALI* dataset to be used as a benchmark evaluation set for polyphonic recordings (Demirel et al., 2021b). We began the curation of the set from the test split used by Vaglio et al. (2020), which consisted of 513 recordings. However, several recordings were not retrievable from the links provided at the time of data retrieval. To cope with this, we obtained the YouTube links through *automatic search* using relevant key words. We eliminated live performances, recordings with low audio quality or that contained extra utterances compared to the corresponding original lyrics. In addition, we excluded songs where the dominant language was not English. For each artist, we included at most 5 songs. Among the remaining ones, a subset is chosen manually for maintaining a balanced distribution of singers' gender, official release dates over decades (see Figure 3.1) and variability of singing styles, vocal effects and music genre. We manually verified the lyrics according to the original *DALI* annotations and other online resources.

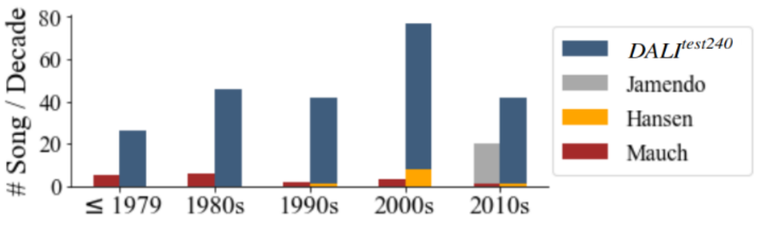


Figure 3.1: The distribution of release years of recordings per test set studied in this study.

The final version consists of 240 recordings, setting the largest evaluation set for ALT in polyphonic recordings with manually verified annotations. In the rest of this thesis, this dataset is referred to as *DALI*^{test240}. For reproducibility, the data identifiers and cleaned lyrics annotations are shared publicly. In addition, a tutorial to retrieve the correct versions of the audio files via their corresponding YouTube links can be found at "*https://github.com/emirdemirel/DALI-TestSet4ALT*".

3.1.3 Summary

Regarding the presence of musical accompaniment, the above-mentioned datasets used in this study can be categorised under two distinct data domains:

- *Monophonic* : *DAMP*, *NUS*, *Hansen^{mono}*
- *Polyphonic* : *DALI*, *Hansen^{poly}*, *Mauch*, *Jamendo*

Below in Table 3.1, a statistical summary of these datasets is provided:

Set	Words	Unique Words	Num. Samples	Mean Sent. Length	Num. Rec.	Num. Singers	Avg. Utt. Dur.	Total Dur.
Librispeech	9,613,824	90,153	292,367	40	292,367	3052	14.38sec	960h
DAMP^{train}	685,956	5304	79,959	9	4155	3052	6.74sec	112h
DALI^{train}	1,077,133	25,477	227,021	5	4132	1507	2.48sec	156h
DAMP^{test}	4630	840	479	10	70	40	6sec	48min
DALI^{test240}	62,800	4200	240	262	240	160	233sec	15.5h
NUS	8667	684	48	181	48	12	143sec	2h
Hansen	2874	585	10	287	10	9	214sec	35min
Mauch	5181	820	20	259	20	19	245sec	82min
Jamendo	5688	995	503	11	20	20	216sec	72min

Table 3.1: Statistics of datasets used in experiments. Titles (from right-to-left): dataset name, total number of words, total number of unique words, number of training samples, average sentence length, number of recordings, number of singers, average duration of training samples (in seconds), and the total duration of the audio data.

3.2 Preprocessing Lyrics

Lyrics automatically retrieved from web resources[5] often contain textual noise. Thus, this raw lyrics data requires a few denoising and normalization steps prior to training, which are listed below:

- All non-ASCII characters are removed except apostrophe ("’") as this occasionally contributes to the context and how words are pronounced. The non-ASCII characters include punctuation marks such as exclamation marks ("!"), question marks ("?"), dots ("."), etc.
- All numeric characters are converted to their alphabetic correspondence.
- All letters are converted to upper case.
- Lyrics text automatically retrieved from public online resources have certain specific noise, such as explicit hyphenation or syllabification of single words (e.g. HY-PHEN-ATION). These explicit notations occur possibly to guide singers to utter words in separate syllables during melody construction by providing cues regarding rhythms or durations.

[5]This includes the lyrics in the *DALI* dataset, as they are also scraped from the web.

- Such symbols or repeated letters are removed and words are converted back to their canonical form using the standard open-source Natural Language Toolkit (*NLTK*) toolkit.[6]. Note that these steps can be language specific.
- The output of the procedure explained above are then corrected and verified manually.

3.3 Speech vs. Singing

In this section, singing and speech data are compared in quantitative terms to provide evidence for the design choices in constructing our baseline lyrics transcribers in Chapters 4 and 5. The analysis begins with the lyrics data in text form to identify its domain-specific characteristics in contrast to typical speech utterances used to train speech recognition models. Later, the major differences between sung and spoken utterances as audio data are highlighted.

3.3.1 Text

At this stage, speech and singing text data are compared based on a number of properties related to linguistic complexity, which are shown in Table 3.2. As the speech data, the transcriptions of the Librispeech - train960h dataset (Panayotov et al., 2015) are used, and the lyrics corpus mentioned in Section 3.1.1 is used for singing. It can be seen that the lyrics corpus in this analysis is larger than Librispeech in terms of total number of training samples and words.

	Librispeech	Lyrics Corpus
# samples	292,367	2,020,776
# words	9,613,824	12,773,430
# types	90,153	101,386
# OOV types	46,264	66,873
# types (% 95)	11,377	4,965
# OOV types (% 95)	368	457
MTLD (mean,std.)	(63.41, 35.25)	(8.66,8.65)

Table 3.2: Comparison between speech and lyrics corpora used in this study. The rows represent the number of training samples, words, types (unique words), out-of-vocabulary types and their 95% percentile, and the measure of textual lexical diversity (MTLD) scores (top-to-bottom).

The number of unique words (*types*) or the *vocabulary* size is also larger in the lyrics corpus. However, if we consider the most common 95% of all words, the vocabulary size drops sharply

[6]Source code is available at *https://github.com/nltk/nltk*

for both the corpora, indicating a high number of types occurring only for few instances. In order to provide more numerical evidence on this matter, or the *lexical diversity*, we make use of the measure of textual lexical diversity (MTLD) metric (McCarthy, 2005) as it provides a sentence-length invariant index (Torruella and Capsada, 2013). Figure 3.2 and the last line in Table 3.2 show that the MTLD index is much higher for the Librispeech dataset compared to the lyrics corpus.

Additionally, the out-of-vocabulary (OOV) words per corpus are provided. Here, a word is considered to be OOV if it does not exist in the standard CMU English Dictionary used in most ASR baseline systems. It can be seen that more than half of the words are OOV in the lyrics corpus, which stands out as a factor to consider for phoneme-based word recognition models. In this regard, a grapheme-to-phoneme conversion procedure is required which is also a standard data preprocessing procedure in LVCSR pipelines.

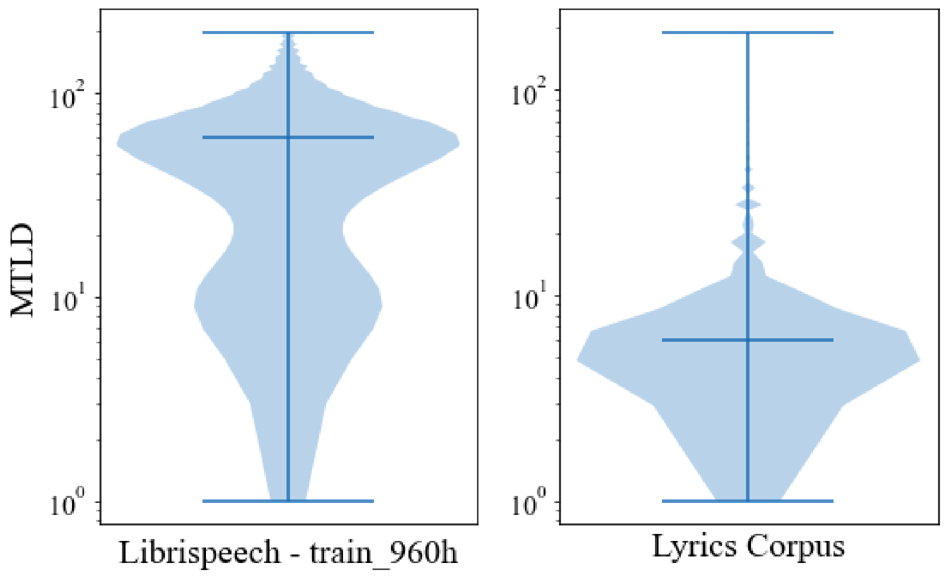


Figure 3.2: Distribution of MTLD scores computed on the transcriptions for each training sample. The x-axis shows the concentration of samples given an MTLD value (i.e. the wider the shaded region, the more samples there are for the corresponding value on the y-axis). The y-axis is log-scaled for visualisation purposes.

On the other hand, the samples in the lyrics corpus generally have fewer words compared to the speech corpus. This is illustrated via the violin plots in Figure 3.3. According to this, the number of words per training sample in the Librispeech corpus is concentrated around 40 whereas this slightly lower than 10 for the lyrics corpus. This shows that lyrics at line level have lower contextual depth compared to speech transcriptions in the Librispeech set. Note that these transcriptions are often formed of multiple sentences which contributes to the lexical density, whereas the lyrics samples are restricted to be a single line of lyrics. Moreover, it can be inferred that most of the lyrics samples (i.e. lines of lyrics) have less than 20 words whereas the number of words per sample varies a lot more for Librispeech.

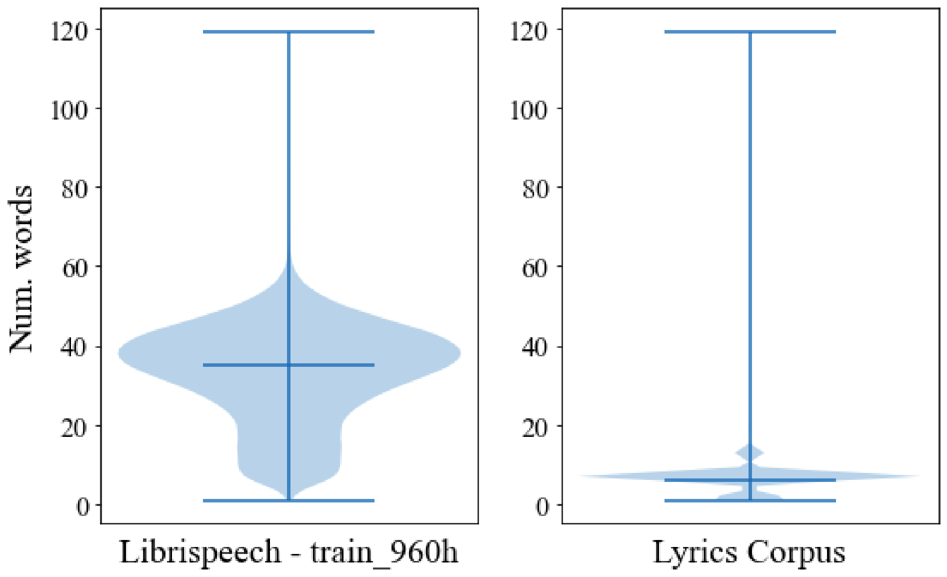


Figure 3.3: Distribution of number of tokens (words) per utterance in the Librispeech dataset and the Lyrics Corpus used in this study.

According to the above comparison, the corpus size seems sufficient for training a language model on lyrics data. Thus, we decided not to extend the lyrics corpus further. However, the analysis above also shows that the lyrics corpus has much lower lexical diversity compared to speech, which might be potentially due to the repetitive patterns in lyrics. In Section 4.4.2, the effect of increasing the corpus' lexical diversity is tested through including Librispeech data in language model training.

3.3.2 Audio

Singing and speech have a number of commonalities such as having a hierarchical structure and complexity (Fitch, 2006), and containing information about the speaker's / singer's gender, identity and emotion (Weninger et al., 2011), however the sound production mechanism may be used differently. For instance, singers may use their vocal organs in an unusual way during opera singing (Sundberg, 1977, 2018), may perform different breathing techniques (Leanderson et al., 1987) or change the location of the vowel formants in the spectral domain (Sundberg and Romedahl, 2009). Moreover in singing, performers tend to articulate syllabic patterns in an expressive and an artistic manner. Compared to speech, these articulations in singing result in vowels having considerably higher variance in duration and acoustic characteristics like pitch, loudness and timbre, and thus in overall prosody of utterances Fujisaki (1981); Lindblom and Sundberg (2014); Duan et al. (2013); Dabike and Barker (2021); Sharma et al. (2021). For a basic prosodic comparison, we compare the articulation rates and vowel duration distributions of the sung and spoken utterances in the NUS Corpus. The articulation rate (AR) can be simply computed as the number of pronounced linguistic units or syllables per minute. Note that the computation of AR does not take silent regions into account.

In general, an utterance or a sequence of speech (or singing) sounds is made of syllable nuclei which are optionally edged by initial and final margins. Vowels often are found within these syllabic boundaries where the energy concentration or the sound intensity is the highest. Thus, the articulation rate can be estimated by tracking the number of vowels in an utterance. Following the steps of De Jong and Wempe (2009), syllable detection is performed based on thresholding the sound intensity level and observing whether a preceding dip exists for each peak above the threshold. Then, the fundamental frequencies are extracted to exclude unvoiced segments.

	Speech	Singing
Articulation Rate (per min)	266.25	172.50
Duration (min)	0.03	0.18
Duration (max)	0.77	3.86
Duration (mean)	0.10	0.34

Table 3.3: Mean articulations rates (syllables per minute) and syllable duration stats (in seconds).

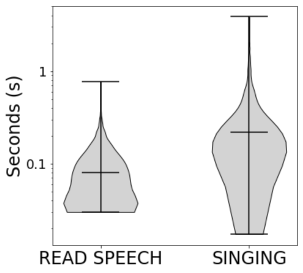


Figure 3.4: Vowel duration distributions.

According to Table 3.3, having a lower AR value in sung utterances implies fewer syllables are uttered per minute and the presence of longer vowels. Longer uttered vowels can also be verified via the vowel duration distributions in Figure 3.4. In addition to the above analysis, readers are encouraged to read data-driven spectral analyses by Duan et al. (2013) and Dabike and Barker (2021) on the articulation differences and pitch variances between sung and spoken phonemes and syllables.

Although this study acknowledges the aforementioned differences between speech and singing and the relevant literature on the subject, it focuses on the data-driven aspects in both domains that would be of importance in the design principles of our deep learning based lyrics transcribers. From this perspective, we are mainly concerned with the total size of the training data. According to the statistics in Table 3.1, the available speech audio data (Librispeech 960h) is much larger than the combination of both the ALT training sets (960 vs. ≈ 270 hours). Unlike the lyrics corpus used for the language model, the number of words and training samples are less for singing. Moreover, training utterance lengths appear to be shorter for singing data compared to Librispeech (second column from right in Table 3.1) which is beneficial for memory consumption during training.

3.4 Evaluation Metrics for Automatic Lyrics Transcription

3.4.1 Word / Character Error Rate

The applications of ALT expect accurate word transcriptions. Similar to ASR, the standard metric for evaluating ALT systems is the *word error rate* (*WER*), which is derived from the *Levenshtein (edit) distance*. Given two strings, the Levenshtein distance computes the minimum number of modifications required to transform one string to the other through dynamic programming (Levenshtein, 1965). Within the context of ASR, Levenshtein distance is used to measure the difference between an automatically recognised word sequence and a reference word sequence (McCowan et al., 2004; Morris et al., 2004). This edit distance is normalised by the number of words in the reference word sequence, N_{ref}. This normalised value is called the word error rate.

There are three types of errors in computing the Levenshtein distance: number of word substitutions (S), number of words in the reference that are deleted in the transcription (D), and number of words in the transcription that do not appear in the reference, i.e. insertion errors, (I). Considering $N_{\text{ref}} = C + S + D$, where C is the number of correctly predicted words, then, *WER* can be formulated as:

$$WER = \frac{S+D+I}{N_{\text{ref}}} = \frac{S+D+I}{C+S+D} \tag{3.1}$$

There may be cases where the predicted word is incorrect but orthographically very similar to the reference. For instance, suppose reference lyrics to transcribe is “*I AM TITANIUM*", and the prediction is “*I’M TITANIUM*”. In this case, *WER* penalises the prediction two times, one for substituting the first words, and a deletion error for the missing second word. As an alternative metric to penalise such instances less, the error rate can be calculated on the character-level, which is often referred to as *Character Error Rate* (*CER*).

3.4.2 Cross-dataset Performance Drop

In this section, we introduce the *Cross-dataset Performance Drop* (*CPD*) metric that measures a specific model’s performance drop across different test sets. *CPD* is proposed to summarise a specific model’s performance across all the datasets used in evaluation with the goal of making the model selection procedure more convenient at each experimental step.

Given an objective function, deep neural networks (DNN) can learn a hyper-dimensional function that can map input features to target labels. This provides the ability to model high level nonlinear information. However, they may overfit to their training data and their performances may get degraded depending on the statistical properties and distributions of the datasets used in training and evaluation. In deep learning evaluation frameworks, this phenomenon has been referred to as the *dataset* or *aggregation bias* (Torralba and Efros, 2011). This inherent bias from the training data should be taken into account when evaluating different deep learning models.

Dataset bias can be observed via scrutinising the performance generalisation of a machine learning model across different datasets. In this study, we formulate the typical cross-dataset evaluation scheme as follows: Given a root dataset $\mathcal{D}$, each subject model is trained on a $\mathcal{D}^{train}$ split and a smaller subset $\mathcal{D}^{valid}$ is used for validation. The hyperparameters of models are fine-tuned and tested on different unseen portions of $\mathcal{D}$, namely the development, $\mathcal{D}^{dev}$, and the test set, $\mathcal{D}^{test}_{self}$, is used to report the final evaluation scores. This implies that a unique pair of $\mathcal{D}^{dev}$ and $\mathcal{D}^{test}_{self}$ is defined for each training set. These splits are illustrated in Figure 3.5.

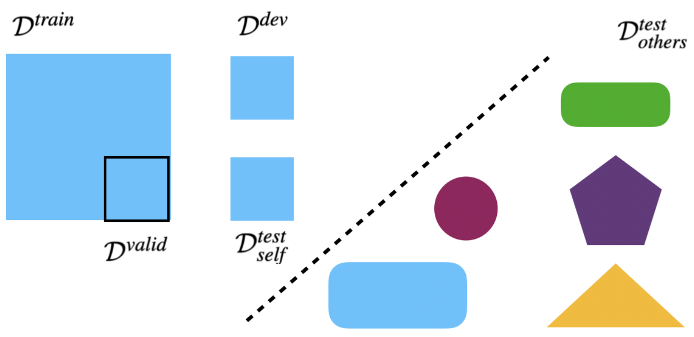


Figure 3.5: The dataset splits considered in the cross-dataset evaluation framework.

In cross-dataset evaluation, subject models are evaluated on other corpora unrelated to the root dataset $\mathcal{D}$, (noted as $\mathcal{D}^{test}_{others} = \{\mathcal{D}^{test}_0, \mathcal{D}^{test}_1, ..., \mathcal{D}^{test}_N\}$ in Figure 3.5). The datasets in $\mathcal{D}^{test}_{others}$ are selected to have distinct data distributions to simulate a more generalisable evaluation. According to the cross-dataset evaluation studies by Torralba and Efros (2011), it is considered that the smaller the gap between a model's performance on $\mathcal{D}^{test}_{others}$ and its performance on $\mathcal{D}^{test}_{self}$, the higher the model's generalisability (or lower the dataset bias) is. Conversely, a higher performance drop on unseen data may indicate that the system is biased either due to overfitting or dataset bias.

In the context of ALT, the performance drop across $\mathcal{D}^{test}_{others}$ according to the above explained generalisation assumption, can be formalised in terms of word recognition rate (*WRR*), which is the opposite of *WER* (McCowan et al., 2004):

$$\begin{aligned} WRR(\%) &= 100 - WER(\%) \\ &= (1 - \frac{S+I+D}{C+S+D}) \times 100 \\ &= (\frac{C-I}{C+S+D}) \times 100 \\ &= (\frac{C-I}{N_{\text{ref}}}) \times 100. \end{aligned} \tag{3.2}$$

Then the *WRR* drop between $\mathcal{D}^{test}_{self}$ and $\mathcal{D}^{test}_{others}$ can be expressed as:

$$\Delta WRR(\%) = WRR^{test}_{self}(\%) - \sum_{m}^{M} \frac{WRR^{test}_{m}(\%)}{M} \tag{3.3}$$

where M is the number of datasets in $\mathcal{D}^{test}_{others}$. Equation 3.3 gives a relative measure w.r.t. WRR^{test}_{self} and calculating the performance drop in terms of percentages does not take the size of $\mathcal{D}^{test}_{self}$ into full consideration. For this reason, Tommasi et al. (2017) proposed measuring the performance drop directly based on the absolute differences of the numbers of correctly predicted instances. However, using direct differences may result in biased measurements in case of large differences in the sizes of test sets to compare. For instance, consider an extreme case where the number of words in $\mathcal{D}^{test}_{self}$ is 10, and the words in hypothetical $\mathcal{D}^{test}_{others,a}$ and $\mathcal{D}^{test}_{others,b}$ are 2 and 1 million respectively. Suppose half of the words both in $\mathcal{D}^{test}_{self}$ and $\mathcal{D}^{test}_{others,b}$ are predicted correctly (i.e. $WRR = 50\%$ with no insertion errors) and both words in $\mathcal{D}^{test}_{others,a}$ are predicted correctly ($WRR = 100\%$). According to Equation 3.3, the resulting cross-dataset performance drop would be -25% meaning that the subject model performs much better on unseen evaluation sets, while the overall performance is almost the same according to the ratio between the total number of correct predictions and the number of words in the reference. Therefore, in order to reduce the bias introduced by sizes of the subject test datasets, we propose using this ratio as the cross-dataset performance drop, *CPD*, which can be expressed as:

$$CPD(\%) = \left(\frac{C^{test}_{self} - I^{test}_{self}}{N^{test}_{self}} - \frac{\sum_m^M (C_m - I_m)}{\sum_m^M N_m} \right) \times 100 \tag{3.4}$$

The above expression can be interpreted as merging all sets in $\mathcal{D}^{test}_{others}$ into a single test set. According to Equation 3.4, $CPD = 0$ implies that the subject model's average word recognition performance is the same as for $\mathcal{D}^{test}_{self}$. In other words, the results obtained on $\mathcal{D}^{test}_{self}$ are generalisable to other datasets. In this context, $CPD > 0$ means performance drop is observed and signals the presence of dataset bias. As *CPD* gets larger, the risk of overfitting increases and the influence of aggregation bias increases. $CPD < 0$ might occur where the root dataset $\mathcal{D}$ has sufficient diversity compared to $\mathcal{D}^{test}_{others}$ although there is a room for improvement and hints the risk of *underfitting*. This can be interpreted as the lower *CPD* gets for $CPD < 0$, the risk of underfitting increases and the performance of the model M should be improved further, possibly. The above described interpretation of the performance drop measure can be summarised as follows:

$$\mathit{CPD} > 0 \rightarrow \text{Dataset bias, risk of overfitting}$$

$$\mathit{CPD} < 0 \rightarrow \text{More fine-tuning needed, risk of underfitting}$$

Research in automatic lyrics transcription has focused on two distinct data domains: solo singing performances (monophonic) and recordings with musical accompaniment (polyphonic). Specifically for the polyphonic domain, the background acoustic scene is highly variable depending on the music accompaniment, i.e., music styles, instruments, sound effects, musical harmony, etc. Moreover, the data acquisition process (e.g. recording environment, device, audio encoding) is likely to vary across different sources of data. These domain differences have motivated researchers to focus on building *domain-specific* models[7] and thus reporting domain-specific evaluation results (Kruspe, 2016; Stoller et al., 2019; Gupta et al., 2018, 2020; Dabike and Barker, 2019; Demirel et al., 2020a; Basak et al., 2021). In contrast, one of the main focuses of the methods proposed in this thesis is to improve lyrics transcription models' performance generalisability and scalability across varying domains. Therefore, we measure subject models' performance drop on the combination of monophonic and polyphonic recordings which is referred to as $\mathcal{D}^{test}_{others}$ in cross-dataset evaluation (Section 4.4.6). For this, we use the *CPD* metric for comparing different models.

3.4.3 Other Performance Metrics

In addition to evaluating ALT systems based on word recognition rates, we assess different models in terms of effective model complexity (Hu et al., 2021) and inference runtime. The former, which is also referred to as *practical complexity* in the literature (Novak et al., 2018), reflects the complexity of the functions of a model with a fixed set of parameters represented by DNNs (Hanin and Rolnick, 2019). This measure has been utilised to improve model selection strategies (Myung, 2000) and explore model compression techniques (Cheng et al., 2018). A common metric for measuring the effective model complexity is the number of trainable DNN parameters given the same model framework is employed (Kurkova, 2018), which is the one

[7] In this context, the term domain-specific is used for models trained on either monophonic or polyphonic recordings.

utilised in this study.

The second operational performance metric mentioned above is concerned with how fast the transcription system responds given the input speech / singing signal. A popular metric used in both research and industry for this purpose is called the *real-time factor* (*RTF*), which is simply the ratio between the system's response time and the duration of the input utterance. To specifically measure the word inference block of the overall transcription pipeline, we compute *RTF* based on the graph decoding times, and exclude the initial data preprocessing and feature extraction steps, which are identical across all models tested in this study. In the experiments, we repeat decoding for 5 times and report the average of these iteration as the final *RTF*s. All tests are performed on the same machine.

3.5 Summary

To sum up, we went through the datasets available for research and mentioned their specific properties relevant to the ALT task. According to these properties, we categorised datasets with respect to having background music accompanying the singing voice. A lyrics corpus is curated for constructing the language model and a new evaluation set, $DALI^{test240}$ is presented. Furthermore, we studied the singing data within the context of word recognition both in text and audio forms, and analysed its mutual and contrasting aspects with speech. In summary, while the lyrics corpus is sufficiently large, its lexical diversity is much lower than that of the benchmark speech dataset. Moreover, the audio dataset available for training the acoustic model is also much smaller than Librispeech. On the other hand, training samples in singing are shorter both in terms of audio and text lengths. These data properties are taken into consideration for adapting state-of-the-art speech recognition engines to singing data and building the lyrics transcribers in Chapters 4 and 5. Finally, we studied the evaluation metrics that will be used for model selection during experiments, and introduced a novel evaluation metric, *CPD* which measures a lyrics transcriber's performance generalisability.

Chapter 4

Hybrid DNN-HMM Based Lyrics Transcription

For decades, HMMs had been the standard approach for modeling speech signals due to their capability of modeling sequential data. According to the HMM based speech processing framework, the most common approach for modeling utterances is through decomposing spoken utterances into streams of phonemes where each phoneme is represented by an HMM (Gales and Young, 2008). Conceptually, the overall function of HMMs in this context is modeling the transition probabilities and observation distributions of phoneme sequences statistically based on acoustic observations. The transition and output phoneme posterior probability distributions (or emission probabilities) of HMM sequences are estimated from training data. Phoneme probabilities are converted to word probabilities through a phoneme-to-word symbol transduction procedure based on the mapping predefined by a pronunciation dictionary. The raw posterior probabilities are then smoothed via a word-level language model.

Before the deep learning era, Gaussian Mixture Models (GMM) were employed to estimate the HMM emission probabilities from acoustic feature observations, i.e. the acoustic model. With the advances in deep learning research, Deep Neural Networks (DNN) have become the standard classifiers for estimating the HMM emission probabilities. The framework examined in this chapter is referred to as the hybrid DNN-HMM approach for speech recognition, and the first lyrics transcription system presented in this thesis follows the principles of this framework.

Modern HMM based ASR frameworks (including Kaldi) use the concept of Weighted Finite

State Transducers (WFST) for modeling the phoneme states[1]. As their name implies, WFSTs have the ability to perform transduction / translation between symbolic sequences from two different domains. This is achieved via storing a weight between symbols to translate, which can later be used to implement a probability distribution over a sequence of events (Mohri et al., 2002). According to this framework, a cascading transduction operation is applied for converting phoneme posterior probabilities to word probabilities. The main transducers in this cascaded pipeline are described below:

Transducer	Function	Input sequence	Output sequence
H	acoustic model	HMM topology	context-dependent phonemes
C	context-dependency	context-dependent phonemes	context-independent phonemes
L	pronunciation dictionary	context-independent phonemes	words
G	language model	words	words

Table 4.1: Finite state transducers in the decoding graph

The acoustic model transducer, H, takes acoustic features as input and stores context-dependent phoneme state probabilities. The context-dependency is removed by the C transducer. These context-independent phoneme labels are then converted to words via the lexicon transducer, L. The output word probability distributions are then composed with word posteriors estimated by the language model or the grammar transducer, G. The cascaded composition operation can be summarised as:

$$HCLG = H \circ C \circ L \circ G, \tag{4.1}$$

where $HCLG$ is referred to as the final decoding graph. In the above equation, $\circ$ stands for the *graph composition* operation which is used to define the binary relationship between the input and output sequences in different domains. For instance, consider the symbol q is converted to s via the T_1 transducer ($q \rightarrow s$) and T_2 translates s to w. Then the cascaded transducer $T_1 \circ T_2$ can be used to define the transduction $q \rightarrow w$. Following this principle, it can be inferred that the composition in Equation 4.1 is applied unidirectionally from left to right. According to this,

[1]Kaldi uses the Open-FST toolkit's WFST implementation (Allauzen et al., 2007).

the acoustic model and the grammar transduction are the initial and the final operations during the construction of the overall transcriber, i.e. decoding graph composition.

This chapter outlines the details of the proposed methods to improve the performance of the DNN-HMM lyrics transcription system following a *top-down* approach. It follows the reverse order of the graph composition procedure and begins with building the language model. Then, a comparative study is held to adapt the lexicon (pronunciation) transducer to singing. The next two sections focus on the acoustic model. First, the study proposes cross-domain training to achieve domain invariant performance. Secondly, a novel compact multistream neural network architecture is presented which is designed with the goal of improving robustness in noisy environments and polyphonic recordings. The chapter concludes with experiments to find the optimal setting for each of the major computational blocks of the DNN-HMM based lyrics transcriber. In this final section, a cross-dataset evaluation framework is provided to measure models' performance generalisability across varying evaluation datasets used in research.

4.1 Language Model

Language models in DNN-HMM frameworks are typically built on *n-gram approximation*. Once word lattices are generated after a first-pass scoring with an n-gram model, the word posteriors can be rescored with a larger n-gram (Ljolje et al., 1999). In addition, rescoring can also be performed via RNNLMs (Sundermeyer et al., 2014; Liu et al., 2016). In this section, an explanation of the theory behind the construction of these language models is given.

4.1.1 n-Gram Approximation

The word probabilities, $P(\mathbf{w})$, can be defined according to the n-gram approximation approach as:

$$P(\mathbf{w}) = \prod_{k=1}^{K} P(w_k | w_{k-1}, w_{k-2}, ..., w_{k-N+1}), \tag{4.2}$$

where $\mathbf{w} = w_1, ..., w_K$ is a word sequence with a finite length K. Similar to the previous studies by Dabike and Barker (2019) andGupta et al. (2020), the DNN-HMM based lyrics transcriber in this study adopts the *maximum entropy* (ME) modeling approach (Rosenfeld, 1994; Alumäe and Kurimo, 2010) within the SRILM toolkit (Stolcke, 2002), a standard open-source natural language processing software. The n-gram probabilities are estimated in the ME optimization approach as follows:

$$P(w_k|w_{k-1}, w_{k-2}, ..., w_{k-N+1}) = \frac{\exp(\sum_k \lambda_k f_k(w_k, w_{k-1}, w_{k-2}, ..., w_{k-N+1}))}{\sum_{w_k \in V_k} \exp(\sum_j \lambda_j f(w_{k-1}, w_{k-2}, ..., w_{k-N+1}))}, \tag{4.3}$$

where f corresponds to the word feature functions used in training, and V_k is the set of all possible word predictions at time step k. The numerator in Equation 4.3 represents the likelihood of the word w_k given the word history, and the denominator is the likelihood of all words in V on the same observations which serves as the normalisation factor. The goal of ME training is to learn the optimal weights, λ_k corresponding to features $f_k(w_k, w_{k-1}, w_{k-2}, ..., w_{k-N+1})$, to maximise the likelihood of all observed word sequences in the training data $L(\mathbf{w}; \lambda)$. The weights are usually learned via the iterative scaling algorithm (Rosenfeld, 1994) or gradient descent. Our system in particular performs parameter optimization using the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) method for this (Andrew and Gao, 2007).

There are often cases where a word in a test sample appears in an *n-gram* that was never seen in the training data. In this case, the LM would assign zero probability to such words. To prevent this, the probabilities of the most frequent events are shaved off and distributed over the unseen *n-gram*. This procedure is called *smoothing*. Our framework uses the Kneser-Ney smoothing algorithm (Kneser and Ney, 1995).

Once first-pass decoding is done, a lattice structure is obtained as in Figure 2.3 which stores the word posterior probabilities, labels and graph costs. Previous studies showed that further performance improvement can be achieved after a second-pass scoring of word lattices using another language model trained on a larger corpus or with a larger n-gram (Liu et al., 2016; Ljolje et al., 1999). WFST based ASR (as in Kaldi) provides the flexibility to rescore the word posteriors with an external or a larger language model. This is achieved by subtracting the costs of the initial language model from the overall graph cost and replacing them with the costs of the new language model $G^{'}$ (Liu et al., 2016). For instance, Dabike and Barker (2019) initially obtain word probabilities with a *3-gram* LM, then rescore the word lattices with a *4-gram*. Their experiments showed that the rescoring with the *4-gram* LM was consistently superior over the *3-gram*. Note that the final decoding graph can as well be constructed through composing the *HCL* graph directly with a *4-gram* language model.

4.1.2 RNNLM

The external language model used for lattice rescoring can also be built with *neural language models* (Bengio et al., 2003). Recurrent Neural Networks (RNN) are the standard choice for this purpose due to their capability of capturing longer context dependencies (in text data) compared to n-grams. Although this approach of language modeling is generally referred to as RNNLM, the network topology may include other variants of RNNs, such as LSTMs or GRUs.

The basic training principles of the neural language model can be summarised as follows: suppose the training samples are sequences of words, $w_1, w_2, ..., w_T$ with $w_t \in V$, where V is the set of words in the training set, or the *vocabulary*. Then, the objective function is to learn word likelihoods given a certain context with length n:

$$f(w_t, ..., w_{t-n+1)} = P(w_t | w_1^{t-1}) \tag{4.4}$$

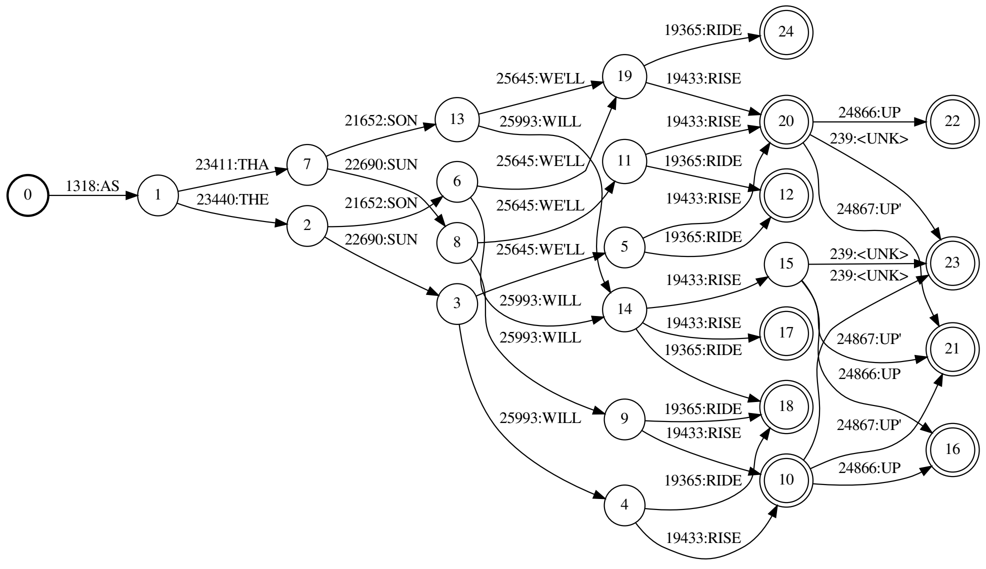
The words are represented with embeddings, or *distributed feature vectors*, C^2. The network, g takes C as input and learns a context representation, z_i. This representation correspond to unnormalised word log-probability distributions which are normalised with $\mathrm{softmax}$ function to guarantee they sum up to 1:

$$P(w_t | w_{t-1}, ..., w_{t-n+1}) = \frac{e^{z_{w_t}}}{\sum_i e_i^z}. \tag{4.5}$$

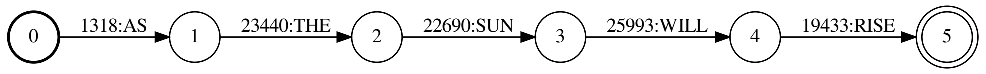
In their work, Chen et al. (2017) successfully integrated RNNLMs for word lattice rescoring within the DNN-HMM framework. This initial approach used word classes as the target units. However, considering that there can be thousands of words in a dataset, the computation of the word probabilities at the end of the $\mathrm{softmax}$ layers becomes expensive. To reduce computational complexity and speed up rescoring, Sundermeyer et al. (2014) introduced *lattice pruning* which

[2] Here we use C for representing the context vector symbol only within this context as it is the most common symbol used for this in the literature. It should not be confused with the symbol to represent the number of correct word predictions.

removes the predictions with lower probabilities from the search space. This pruned search procedure was later implemented within Kaldi's DNN-HMM based framework (Xu et al., 2018b,a). In our previous work, we showed that the pruned RNNLM rescoring leads to a much simplified lattice structure at inference (see Figure 4.1) which helps to reduce the decoding runtime (Demirel et al., 2020a). In the aforementioned paper, RNNLM rescoring was also shown to improve the lyrics transcription performance. Although there are more modern neural language modeling approaches, this study uses Kaldi's RNNLM rescoring implementation due to it being already well-integrated with the other computational blocks of the DNN-HMM based lyrics transcription system.



(a)



(b)

Figure 4.1: Lattices obtained with a 4-gram LM (a) and RNNLM (b). The figures are obtained from (Demirel et al., 2020a).

Kaldi's RNNLM implementation uses the word representation proposed by Huang et al. (2013). According to this approach, words are decomposed into character-level n-grams which may also be referred as *word hashing* procedure. For instance:

$$\#\textit{flow}\# \rightarrow \{\#\text{f, fl, flo, low, ow, w}\#\},$$

where $\#$ indicates word boundaries. Word hashing is a data compression method and can be perceived as feature dimensionality reduction. Consider representing words using *one-hot encoding* for a vocabulary with the size of 40K words where each bit in the feature space corresponds to a word. This would result in feature vectors having the dimensions of 40K bits. In word hashing, the above-explained subword n-grams are used to construct the feature vectors. Therefore each element in the feature space corresponds to a subword unit. According to the statistics reported by Huang et al. (2013), using word hashing with unigrams reduced the size of feature vectors from 40,000 to 1107 bits.

One issue with word hashing is *collision* where feature representations of different words have the same set of character-level n-grams. Huang et al. (2013) showed that the number of collisions can be reduced through increasing the order of n-grams. However, this would increase the size of the word representation vector to 10,306. To distinguish different words with the same set of unigrams, Xu et al. (2018b) append the number of per-word n-grams and word-level unigram log-probabilities on top of the subword-level unigram vectors. These feature vectors are fed to the neural network through a learnable embedding matrix which is trained in conjuction with the RNNLM.

The architecture of the RNNLM is a simple two-layer LSTM network with a hidden dimension size of 256 and the word features are projected to the LSTM network via an embedding matrix of 1024. A pair of layer normalisation and ReLU activation is applied after each LSTM layer (Figure 4.2). The data order is randomised and one out of 50 samples are chosen for the validation set. The network is trained for 30 epochs. At inference, the pretrained RNNLM takes word likelihoods obtained after the n-gram approximation scoring as input, and outputs new likelihoods. These are composed with the scores in the final decoding graph $HCLG$ through removing the costs from the n-gram LM and adding the new RNNLM cost (Xu et al., 2018a). Then the standard beam search is applied on the updated costs to obtain the final word transcriptions.

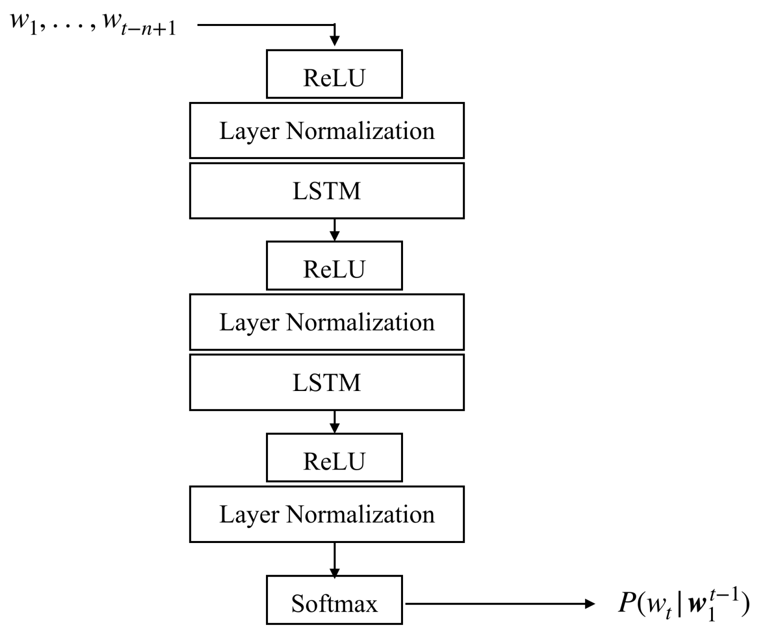


Figure 4.2: The topology of the RNNLM architecture

4.2 Pronunciation Model

The acoustic model in the DNN-HMM setup learns a function to translate acoustic features into a probability distribution over a phoneme set. As the expected output of a lyrics transcriber is word sequences, the transduction from a stream of phonemes to word sequences is required. In the DNN-HMM speech recognition framework this transduction, or word-to-phoneme mapping is achieved using a predefined dictionary (or lexicon), which is referred to as the pronunciation model.

For most cases, LVCSR research for the English language employs the standard CMU dictionary (Weide, 1998) as the lexicon transducer *L.fst*. The CMU dictionary is curated by phonetics experts and models word pronunciations based on the North American English accent. This restriction regarding accents becomes less apparent when DNNs are used to build the acoustic model as they can model higher levels of nonlinearity between target classes and the variances in the input acoustic features. On the other hand, previous research showed an explicit modeling of domain-specific common pronunciation variances can improve recognition rates (Adda-Decker and Lamel, 2000; Morales and Cox, 2007; Yilmaz and Pelemans, 2014). Specifically for sung utterances, Gupta et al. (2018) showed that adapting a standard speech lexicon to singing data through extending vowel occurrences in pronunciations is beneficial for transcribing lyrics.

Motivated by the aforementioned research, potential directions for adapting a standard speech pronunciation model to singing data is investigated in this section. The first approach is generating word pronunciations alternative to their canonical form, based on the common pronunciation variances in singing compared to natural speech. For this a data-driven confusion analysis is initially applied on sung phonemes, a method that is similar to one presented by Morales and Cox (2007); Yilmaz and Pelemans (2014), and the alternative pronunciations are generated based on the observations driven from the confusion analysis. Later, graphemes as an alternative subword token to phonemes is studied.

Phoneme Set

In constructing the target class set, the DNN-HMM based transcriber the 39-phoneme set for English which is used in the CMU Pronunciation Dictionary[3] and does not take the variance in lexical stress into account. The phonemes in this set are referred to as the *base* or *context-independent* phones. The set of phonemes can be seen in the pronunciation analysis tables (Table 4.2) in the following section.

4.2.1 Pronunciation Analysis

The goal of the pronunciation analysis in this section is to reveal statistically observable pronunciation variances in sung utterances compared to the canonical word pronunciations in speech. The analysis is applied on the *NUS Corpus* (Duan et al., 2013), due to the availability of phoneme-level annotations, and considers only the native English speakers within dataset. Initially, the orthographic word transcriptions are extracted from singing recordings in the analysis set using the pretrained ALT model in (Demirel et al., 2020a), which uses the standard CMU pronunciation dictionary. The word predictions are converted back to their phonemic representations $\widehat{\mathbf{Q}}$ by decomposing the grammar transducer G from the decoding graph *HCLG*. To get the phoneme confidences, $\widehat{\mathbf{Q}}$ is aligned with their corresponding manually annotated phoneme sequences. The alignments are generated following the procedure explained below:

1. The alignment score matrix $\mathbf{D}$ is computed through the Levenshtein alignment, *lev*,

[3]The phoneme set can be found here *http://www.speech.cs.cmu.edu/cgi-bin/cmudict?in=desert&phones*.

between the phoneme tokens q of the predictions $\widehat{\mathbf{Q}}_M$ and the ground truth $\mathbf{Q}_N$:

$$\mathbf{D}_{M\times N} = \mathit{lev}(\widehat{\mathbf{Q}}_M, \mathbf{Q}_N), \tag{4.6}$$

and find the best alignment path, $\mathbf{A}_{2\times K}$ through reverse tracing to find the path with the lowest pairwise gap cost:

$$\mathbf{A}_{2\times K} = \begin{pmatrix} \cdots & q_{k-1} & q_k & q_{k+1} & \cdots \\ \cdots & \widehat{q}_{k-1} & \widehat{q}_k & \widehat{q}_{k+1} & \cdots \end{pmatrix}. \tag{4.7}$$

$\mathbf{A}$ can be interpreted as a sequence of phoneme pairs.

2. There are three operations defined on these phoneme pairs to match $\widehat{\mathbf{Q}}_M$ to $\mathbf{Q}_N$: insertions (I), substitutions (S) and deletions (D). These operations are represented in $\mathbf{A}$ with the symbol ϵ. An alignment instance $a_k = \begin{pmatrix} \epsilon \\ \widehat{q}_k^* \end{pmatrix}$ is a deletion and the opposite case would be an insertion, and $a_k = \begin{pmatrix} \epsilon \\ q_k^* \end{pmatrix}$ is a substitution instance if $\widehat{q}_k^* \neq q_k$.

3. Let the number of correctly matching pairs in $\mathbf{A}$ be C, then the confidence score per phoneme type, c_q, can be retrieved as:

$$c_q = \frac{\sum_i^T C_{q,i} - (S_{q,i} + I_{q,i} + D_{q,i})}{\sum_i^T C_{q,i} + S_{q,i} + I_{q,i} + D_{q,i}}, \qquad q \in \Omega_E, \tag{4.8}$$

where T is the number of utterances in the analysis set, q is the phoneme type[4] and Ω_E is the English phoneme set used in our analysis. The denominator is necessary to normalise with respect to the total number of pairs in $\mathbf{A}$, since the phonemes in Ω_E are not necessarily represented equally in the analysis data set.

Table 4.2 shows the confidence scores obtained through the procedure explained above. The first two columns from the left are the list of English phoneme categories and types. In the third

[4]Note that in the case of non-correct predictions, $q_k \neq \widehat{q}_k$.

column from the left, the confidence scores and their confidence rankings R are given. Note that c_q are normalised values according to Equation 4.8, $-1 \leq c_q \leq 1$. Finally, three most frequently confused phonemes per target phoneme type, Φ', are included in Table 4.2.

Vowels	q	$c_q(R)$	Φ'_N
Short Vowels	AE	-0.42 (38)	AH, EH, AA
	AH	0.17 (33)	AA,EH,OW
	EH	0.3 (32)	AH,AE,IH
	IH	0.48 (26)	IY,AH,EY
	UH	0 (36)	AO,UW,AH
Long Vowels	AA	0.5 (24)	AO,AW,AE
	AO	0.06 (35)	AA,AH,OW
	ER	0.36 (31)	AH,OW,EH
	IY	0.87 (6)	EY,IH,EH
	UW	0.88 (4)	OW,AH,UH
Diphthongs	AY	0.86 (8)	AA,AH,EH
	AW	0.71 (18)	AA,AH
	EY	0.87 (7)	IY,AY,EH
	OW	0.76 (17)	AO,AA,AH
	OY	0.4 (28)	OW,AO,AY

Consonants	q	$c_q(R)$	Φ'_N
Plosives	B	0.77 (16)	D,P,W
	D	0.16 (34)	T,N,JH
	G	0.77 (15)	NG,K
	K	0.85 (15)	G,HH
	P	0.78 (14)	B,M,F
	T	0.37 (29)	D,S,CH
Affricates	CH	0.79 (13)	JH,SH,T
	JH	0.88 (5)	CH,S,Y
Nasals	M	0.93 (2)	N,NG
	N	0.85 (12)	M,NG,D
	NG	0.85 (9)	N,M,T
Fricatives	DH	0.36 (30)	TH,D,N
	F	0.91 (3)	V,P,TH
	HH	0.70 (19)	DH,W,Y
	S	0.95 (1)	Z,TH,T
	SH	0.85 (10)	CH,S,Z
	TH	0.57 (21)	S,T,DH
	V	0.56 (22)	F,R,DH
	Z	-0.05 (37)	S,T
	ZH	N/A	N/A
Approximants	L	0.44 (27)	AA,OW,AH
	R	0.48 (25)	AA,AH,IH
	W	0.66 (20)	AA,OW,V
	Y	0.55 (23)	IH, AH, IY

Table 4.2: Results of the phonetic analysis.

Note that, $c_q < 0.25$ means that there are less true positives than the sum of false negatives and positives in per phoneme type predictions, i.e. in most cases, q is predicted incorrectly. These phoneme types with low confidence scores are highlighted in gray colour in Table 4.2. It can be seen that it is mostly the vowel types within short and long vowel categories fall in this low confidence zone. On the other hand, diphthongs are more accurately predicted compared to the other vowel types. In particular, the phoneme '*AE*' has the lowest c_q and is generally associated with the *schwa* sound in phonetics (Silverman, 2011). This very low c_q is not surprising as it is often pronounced weakly and is one of the most frequently occurring vowel sounds in the English language (Roach, 2004).

Consonants have higher c_q values in general with certain exceptions. For instance, plosives ‘*D*’ and ‘*T*’ are severely confused indicating a systematic error, likewise the fricative sound ‘*Z*’. On the other hand, plosives ‘*B,G,K,P*’ have rather high confidences which may be interpreted as singers in our analysis not omitting ‘*B,G,K,P*’ sounds in word pronunciations. It was also mentioned in the literature that singers may utilise such phonemes to utter strong note offsets during melody construction (Bauer, 2002).

In addition, it can be seen that approximants are mostly confused with vowels. This might imply either *deletion* errors during the text alignment step or again omitted approximants similar to plosives as explained above. The former can occur when multiple different vowels are annotated consecutively in the reference to represent the actual uttered phonemes during melody construction, whereas the predictions based on a canonical pronunciation dictionary cannot predict such extra vowels. This is represented as a missing phoneme prediction, and hence a deletion error. Hence, it can be inferred that there are systematic confusions of *approximants* with vowels.

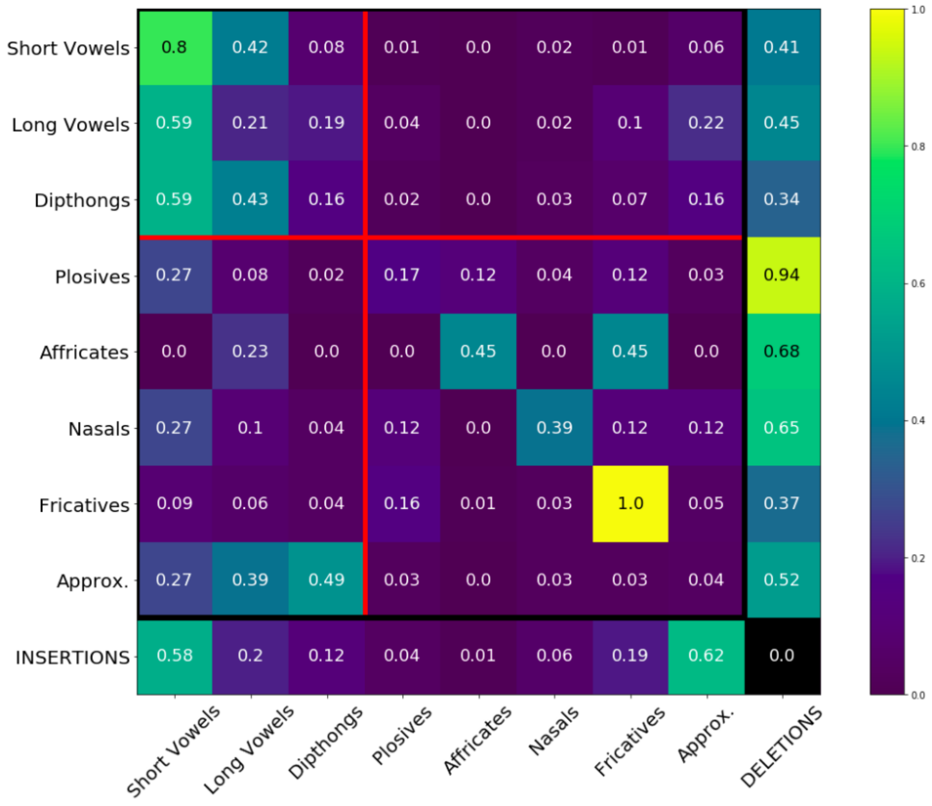


Figure 4.3: Confusion matrix. The x and y axes are represent the ground-truth and predictions respectively.

Figure 4.3 shows the phoneme confusion matrix summarised with respect to phonetic categories where the numbers in each cell represent the phoneme category-wise confusion scores. In computation of these, C_qs are discarded and the sum considers only S, I and D values for each phonetic category. Therefore the diagonal axis does not represent self-confidences. Instead it represents the domestic confusions within each phonetic category. Phonetic-category-wise normalization is applied based on unit sum. These normalization steps are crucial to get confusion values independent of the number of occurrences. Moreover, per-phoneme-category insertions and deletions are also included in the confusion matrix representing extra pronounced or omitted phoneme instances.

The concentration of high confusion rates can be observed for vowels (top left in Figure 4.3). Short vowels are mostly confused with each other. The annotated longer vowels are not necessarily represented in the standard speech lexicon, thus causing the system to assign a higher likelihood for the short vowels when making word predictions. Note the high number of deleted *plosives* signaling them being omitted from pronunciations during singing. Overall, a high frequency of deletions is observed for all categories. In addition to alignment errors, one possible cause for this could be the word liaisons being annotated as single phonemes in human annotations whereas the ALT system would predict such instances as separate phonemes. For example, in ‘*DREAM MAKER*’, ‘*M*’ is annotated once in the corresponding $\mathbf{Q}$, but detected twice in $\widehat{\mathbf{Q}}$.

4.2.2 Singing Adapted Lexicon

In this section, a number of methods for adding alternative pronunciations to the standard speech lexicon is proposed which are based on the observations of the previous section. Through this, we aim to observe the effect of explicit pronunciation dictionary adaption on the sung word recognition performance within the DNN-HMM framework.

The initial method is extending the vowel representations in word pronunciations. This was first shown to be effective by Gupta et al. (2018). In this study, a similar strategy is applied. For instance, consider the word *OCEANS* with its phonemic representation *OW SH AH N Z* in the lexicon. Gupta et al. (2018) extends vowels for 4 times which might cause a large increase in the search space, thus making it computationally expensive. The number of vowel repetitions as a hyperparameter is tested in Section 4.4.3.

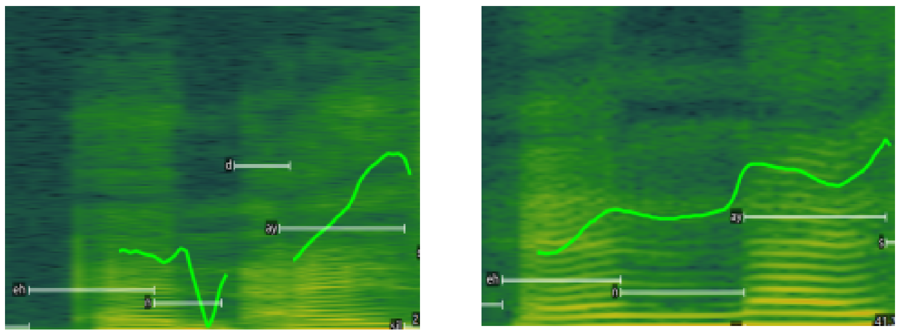


Figure 4.4: An example of an omitted plosive in singing. W = 'AND I ' ; Q^{read} = 'AE N **D** AY' (left) ; Q^{sing} = 'EH N AY'. The gray horizontal lines show the temporal phoneme regions and the bright green curves are the pitch tracks extracted using pYIN (Mauch and Dixon, 2014).

Next, we consider the omitted consonants observed in the previous section. It is not seldom that in singing, performers may omit some consonants at the endings of words. This phenomenon can be explained as a stylistic convention that singers exhibit in their performances in order to maintain the sonority of their singing (Sundberg and Rossing, 1990), or it could as well be a microphone technique to avoid unpleasant pops. The analysis in Section 4.2.1 suggests that this occurs most likely for *plosives* as the phoneme category with highest number of deletions. An example of an omitted *plosive* is illustrated in Figure 4.4[5]. The spectrogram segments in Figure 4.4 show the same words uttered as speech (left) and singing (right) by the same performer. According to the human annotators, the phoneme 'D', is not present during singing. This can also be seen from the discrepancies in the spectrogram and the undisturbed pitch curve in the singing segment[6]. For explicitly modeling such instances, we propose adding alternative pronunciations to words ending with consonants 'D' and 'T' by removing their last phoneme in the corresponding word pronunciation, q^{w_l}. These consonants are selected due to their low confidence scores in Table 4.2. We refer to the model built with omitted plosives as *Pron-1*. According to Figure 4.3, the phoneme category with the lowest confidence seems to be the *fricatives*. By combining this observation with the statistics in Table 4.2, it can be inferred that it is only the Z sound within fricatives that is severely misdetected and mostly confused with the sound S. To test this, we substitute the Z sounds with S when generating alternative

[5]The analysis is performed using Sonic Visualiser software (Cannam et al., 2010).

[6]According to the empirical study by Goldsworthy (2015), pitch and phoneme perception are found to be cognitively correlated processes. Hence, we have chosen explicitly to show the pitch tracks.

pronunciations. This method is referred to as *Pron-2*. Finally, we apply the vowel extension in *Pron-3*. An example of the resulting alternative pronunciations generated with each of these methods is shown below:

Word	FLOAT	OCEANS
Canonical	F L OW T	OW SH AH N Z
Pron-1	F L OW	OW SH AH N Z
Pron-2	F L OW T	OW SH AH N S
Pron-3	F L OW OW T	OW SH AH AH N Z, OW OW SH AH N Z

Table 4.3: Comparison of different alternative word pronunciation generation methods.

4.2.3 Computing Pronunciation Probabilities

The above explained alternative pronunciation generation steps would increase the size of the search space, potentially making the decoding procedure computationally more expensive. In addition, these steps are based on data-driven heuristics, hence not necessarily generalisable to all the words in our vocabulary. To reduce the size of the pronunciation dictionary, pruning can be applied through discarding the redundant pronunciation variants and keeping only the likeliest ones.

Although different pronunciations of a given word can have distinct probabilities, they are considered to be equally likely in practice taken as $P(q|w) = 1$ (i.e. *max-norm* applied) for numerical stability. On the other hand, these probabilities can be calculated explicitly following the steps in Chen et al. (2015). According to this, phoneme alignments are first obtained via a pretrained acoustic model using the raw lexicon with equally likely pronunciations. At this point, it is assumed that the acoustic model chooses the likeliest pronunciation. The list of word-pronunciation pairs are retrieved during the training data alignments and their counts, $\mathit{Count}(w, \mathbf{q}_i^w)$, are computed. Then, the probability $p(\mathbf{q}_i^w|w)$ of the i^{th} pronunciation $\mathbf{q}$ of a word w is computed as:

$$p((\mathbf{q}_i^w|w) = \frac{\mathit{Count}(w, \mathbf{q}_i^w) + \lambda}{\sum_{i=1}^{N_w}(\mathit{Count}(w, \mathbf{q}_i^w) + \lambda)} \tag{4.9}$$

where N_w is the number of pronunciation variants of a word defined in the dictionary, and λ is smoothing constant which is typically set to 1. Then, pronunciation variances with low probabilities are pruned to reduce the overall search space.

4.2.4 Graphemes as Alternatives to Phonemes

One of the major caveats of the state-of-the-art DNN-HMM based ASR systems is their dependence on a handcrafted pronunciation modeling which typically requires human expertise. Several attempts have been proposed to discard this requirement through using *graphemes* (Killer et al., 2003; Rao and Sak, 2017). Le et al. (2019) showed that competitive performance can be achieved via tied context and position dependent graphemes, i.e. *chenones*, in the presence of large training sets. In our study, we leverage the Kaldi framework for building the grapheme context trees and tied states for injecting the context dependency. We add position encoding via adding a *word boundary* tag (_WB) to phonemes (as shown in Table 4.4). Finally, the context dependency is also added similar to the phoneme-based models. The final set of graphemes consists of all the English alphabetic characters and the apostrophe sign (" ’ "). An example of a grapheme-based subword tokenization is shown below:

Words	FLOATIN’
Raw Tokens	F L O A T I N ’
+ Position Dependency	F_WB L O A T I N ’_WB
+ Context Dependency	<null>_F_L_WB F_L_O L_O_A O_A_T A_T_I I_N_’ N_’_<null>_WB

Table 4.4: From raw graphemes to chenones.

4.3 Acoustic Model

The acoustic model is trained on the LFMMI scheme[7] explained in Section 2.2.4. The novel contributions of this study in building the lyrics transcriber acoustic model are two-fold: first, we explore ways to construct a single acoustic model where the recognition performance is undisturbed across both monophonic and polyphonic domains. Secondly, we propose and fine-tune a novel neural network architecture for improving the model’s robustness against noisy environments and polyphonic recordings.

[7]The main pipeline is similar to that of Kaldi’s chain recipe, which can be found at: *https://github.com/kaldi-asr/kaldi/tree/master/egs/librispeech/s5/local/chain/run_tdnn.sh*.

4.3.1 Cross-Domain Training

Conventional ASR systems were initially designed for standard data domains like conversational telephone speech or broadcast news. As the performance of such systems improves, they have been adopted for speech-based human-computer interaction applications used widely in everyday life. Consequently, this has resulted in newly emerging data domains such as web / Youtube data, podcasts, radio talks, movies, as well as utterances with different accents or in low-resource languages. Handling such data domains using a model trained on standard or out-of-domain data is among the main challenges in current ASR research.

Domain mismatch also stands out as a major bottleneck in ALT. It has been previously mentioned that the music background accompanying the singing voice affects the intelligibility of words negatively for human listeners (Fine and Ginsborg, 2014) as it tends to mask lyrics content due to overlapping frequencies with the singing voice. A similar trend applies for ALT machines according to lyrics recognition rates reported previously in the literature (Demirel et al., 2021a) and the MIREX 2020: Automatic Lyrics Transcription challenge[8]. According to these results, the performance of lyrics transcribers trained on monophonic singing recordings is degraded considerably when there is musical accompaniment. The presence of the accompaniment might affect the acoustic scene drastically, potentially masking formants and phonemes. This is illustrated in the images on the left and in the middle of Figure 4.5. The one on the left is the spectral image of a singing excerpt. The second image has the polyphonic music accompaniment.

Within this context, the most common approach for suppressing the music accompaniment has been extracting the vocal track through source separation (Stoller et al., 2019; Gupta et al., 2020; Demirel et al., 2021a). Although better recognition rates could be achieved (Stoller et al., 2019; Demirel et al., 2021a), there are two major caveats in this approach. The first one is that the current state of source separation models available for research usually introduces artifacts such as removed consonants or added background noise. For instance, consider the rightmost image in Figure 4.5. This is the spectrogram of the source-separated version of the same excerpt as shown on the left. While the phonemic structures in this image have lower resolution than the original vocal track, additional high frequency content can also be observed, which are potentially reflected as noise (or artifacts) during lyrics transcription. Secondly, in real-world

[8]*https://www.music-ir.org/mirex/wiki/2020:Lyrics_Transcription_Results.*

applications, source separation would have to be applied prior to the lyrics transcription module, which would increase the computational cost and complexity of the overall pipeline. This is especially undesirable in such real-world applications where the users want to retrieve lyrics as quickly as possible.

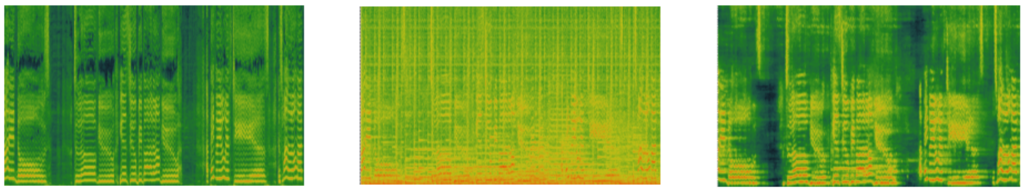


Figure 4.5: Excerpts from the dataset presented by Hansen (2012): (left) Original vocal stem, (middle) polyphonic mix, (right) vocal separated using Spleeter (Hennequin et al., 2020).

At first sight, the domain mismatch can be thought to be resolved through training a model on in-domain data, however such resources in sufficient quantities to train robust models may not be available all the time. Notable methods to resolve the domain mismatch problem for low-resource data domains include domain adaptation (Samarakoon et al., 2018; Yi et al., 2018), speech enhancement(Liao et al., 2018), data generation/augmentation through semi-supervised learning (Park et al., 2020) or using domain-invariant features for training the model (Hsu and Glass, 2018).

Fortunately in ALT, labeled data is available through the *DAMP* and *DALI* datasets respectively, which researchers have been able to leverage to construct domain-specific models for monophonic and polyphonic recordings separately (Dabike and Barker, 2019; Demirel et al., 2020a; Gupta et al., 2020). To circumvent the source separation step, a polyphonic acoustic model can be trained solely on polyphonic recordings. However, these models do not perform well on monophonic datasets compared to monophonic acoustic models. In order to construct a single acoustic model where the transcription performance is maintained across both monophonic and polyphonic recordings, the cross-domain acoustic model training includes both monophonic and polyphonic recordings as proposed in our previous work (Demirel et al., 2021b), which was shown to be effective and led to improved results specifically for polyphonic recordings.

Music Informed Silence Modeling

Phoneme based ASR systems employ extra phonemes in addition to the set of phonemes given by the natural language to be processed. These extra phonemes, generally referred to as '*silence*' and '*spoken noise*' are included to represent the non-voice instances or sounds that are not relevant to the transcription (such as laughter, sneeze sounds, etc.). In this cross-domain training framework, a new extra phoneme is introduced, which is referred to as the '*music*' phoneme that is inserted for modeling the acoustic representations of instrumental music sounds or the non-vocal instances in the polyphonic recordings.

The music and silence information are explicitly embedded by tagging the monophonic and polyphonic recordings. Tagging is performed by inserting the respective phoneme type at the beginnings and ends of each training sample. As it is known that the *DAMP* and *DALI* datasets consist of only monophonic or polyphonic recordings respectively, this prior knowledge is exploited during tagging of the data.

	w
Raw	$w_1\ w_2\ ...\ w_N$
DAMP	*<silence>* $w_1\ w_2\ ...\ w_N$ *<silence>*
DALI	*<music>* $w_1\ w_2\ ...\ w_N$ *<music>*

Table 4.5: Music / silence phoneme tagging during cross-domain training.

Recall that training sample is a line of lyrics. Therefore, adding the *<silence>* and *<music>* tags in the beginning and end of each sample allows the training to be context aware on the lyrics line-level. This is especially useful for the *audio-to-lyrics* alignment task (Demirel et al., 2021a) or when the line-level segmentation is required.

4.3.2 Compact Multistreaming Time Delay Neural Networks

This section explains the design principles of the compact multistreaming neural network architecture developed in our research. First, time delay neural networks (TDNN) (Waibel et al., 1989) are studied which are the main building blocks of the our neural network architecture. Then, a convolutional front-end structure is studied to reduce model complexity while maintaining operational power. Then, the novel compact variant of the multistream TDNN architecture is presented.

Time-Delay Neural Networks

TDNNs are essentially one dimensional CNNs where the convolution is applied on the time domain with *dilated* frames. Consider a stream of input acoustic features like filterbanks or MFCCs, $\mathbf{x}_t \in \mathbb{R}^m$ are concatenated to form the input matrix to a TDNN, $X \in \mathbb{R}^{m \times t}$ where m is the height of input features and each column represents a time frame t. Consider a learnable matrix $\mathbf{W} \in \mathbb{R}^{m \times l}$ where l is the width of the matrix, and the height is the same as the height of input features, m. Then, $m \times l$ indicates the kernel size for the convolution operation. The kernel width of a TDNN layer determines the span of temporal context covered by the layer which is often referred to as the receptive field.

Similar to standard CNNs, the learnable matrix $\mathbf{W}$ slides through the input, X, with a time stride s. This means that the TDNN layer applies convolution for s time steps. To make the input feature matrix compatible with the TDNN convolutions, padding (with a size of p) might be added at each end of the input vectors. Given these, the output width of a TDNN layer, o can be calculated as:

$$o = \lfloor \frac{t + 2p - l}{s} \rfloor + 1, \tag{4.10}$$

where $\lfloor . \rfloor$ is the floor function. In a typical TDNN setting, the kernel slides over the input one step at a time, meaning $s = 1$. In this framework, zero (*null*) padding is applied[9].

At each time step t of a TDNN layer, a convolution operation is applied which is essentially an element-wise multiplication of the kernel weights, W and the input features X, then these products are summed at the output. In neural networks, a bias matrix is typically added before passing through the sigmoid function, ϕ. According to this, the output of a TDNN layer at output step $q \in \{1, 2, ..., o\}$, $z(q)$ is computed as:

$$z(q) = \phi(\mathbf{W} * \mathbf{X}_q + \mathbf{b}), \tag{4.11}$$

where $\mathbf{X}_q$ represents the portion of the input features within the receptive field and $*$ stands for the convolution operation. The above equation can be unfolded as:

[9]In this context, zero padding means padding zero vectors.

$$z(q) = \phi(\sum_{m=0}^{M}\sum_{l=0}^{L} w_{m,l}.x_{m,l} + b). \tag{4.12}$$

The description above is the same as for a one dimensional convolution operation where the height of the kernel and the input feature vectors are the same. According to Equation 4.12, the convolution operation is applied at every time step. For TDNNs in particular, certain frames are dropped out during the convolution and the summation is done over frames dilated in time. This operation is often referred to as *dilated convolution*. Specifically, consider a set of context indices, J, where the element $j = 0$ is the current time step to be processed, for example, $J = \{-3, 0, +5\}$. Then the output size of the TDNN layer becomes:

$$o = \lfloor \frac{T - (\max(J) - \min(J)) + 2p}{s} \rfloor + 1 \tag{4.13}$$

This time, the TDNN output, $z(q)$ is computed within the range of J:

$$z(q) = \phi(\sum_{m=0}^{M}\sum_{j\in J} w_{m,j}.x_{m,j} + b) \tag{4.14}$$

Notice that the length of J is smaller than L due to subsampling. Hence, fewer frames are required to achieve the same receptive field as the standard one dimensional convolution. In other words, dilated CNNs can cover a larger context compared to a similarly complex one dimensional CNN with non-dilated convolutions.

TDNNs can be stacked together to form a deep neural network structure where the context index set for each TDNN layer J_n can be layer specific, where $n \in \mathbb{N}$. For instance, consider the 4 layer TDNN structure summarised in Table 4.6. The first layer ($n = 1$) has a dilation rate $\tau = 1$, meaning that the convolution is applied only with the neighbouring frames. Hence, the receptive field (*RF*) is 3 frames[10]. The *RF* of the following layer relative to the preceding TDNN layer is $5 - (-3) + 1 = 9$. Considering the overall receptive field aggregates as the

[10]The current frame is included in the receptive field computation.

layers go deeper in the network, the absolute receptive field of the second layer with respect to the beginning of the network becomes 32 frames.

Layer	Input context	*RF* (relative)	*RF* (absolute)
1	{-1,0,1}	3 frames	3 frames
2	{-5,0,7}	13 frames	15 frames
3	{-3,0,5}	9 frames	23 frames
4	{-3,0,3}	7 frames	29 frames
5	{-1,0,3}	5 frames	33 frames

Table 4.6: A dummy TDNN example.

According to the above computation, the receptive field of the overall network can be formulated as:

$$RF = \left(\sum_{n=1}^{N} (\max(J_n) - \min(J_n)) \right) + 1, \tag{4.15}$$

where N is the number of TDNN layers in the network. Although having varying context spaces for each layer seems to be a way of diversifying how the temporal information is processed, previous research showed that having the same symmetrical past and future context led to optimal results within the context of DNN-HMM based ASR (Povey et al., 2018a). In this case, amount of dilation per layer (i.e. *dilation rate)* can be referred to as τ_n, where n is the index of a TDNN layer. For instance, suppose the frame at time step t is convolved with frames at $t = \mp 3$, which means the dilation rate is $\tau = 3$, and the *RF* of this layer is $2 \times 3 + 1 = 7$ frames. This computation can be generalised as:

$$RF = (2 \times \tau \times N) + 1, \tag{4.16}$$

where $\tau_n = |\max(J_n)| = |\min(J_n)|$. Then, the time span covered by the receptive field at the end of the TDNN stack, rf_N can be simply computed by multiplying *RF* with the input frame length H:

$$\begin{aligned} rf_N &= (2 \times \tau \times N + 1) \times H \\ &= RF \times H \end{aligned} \tag{4.17}$$

As the computation of network weights is based on the convolution operation, training TDNNs is parallelisable providing a more efficient training compared to LSTMs. Although TDNNs are introduced more than three decades ago (Waibel et al., 1989), they are still used in the state-of-the-art architectures in both DNN-HMM based speech recognition (Pan et al., 2020) and lyrics transcription and alignment (Gupta et al., 2020; Demirel et al., 2020a; Dabike and Barker, 2019).

Factorising TDNN Layers

As Equation 4.17 implies, the more TDNN layers the architecture has, the larger time span it covers, i.e. it is capable of modeling longer context dependencies. However, increasing the number of TDNN layers is computationally expensive. To reduce the number of trainable parameters and establish a more efficient training, Povey et al. (2018a) proposed a factorised version where the trainable matrices $\mathbf{M}^{TDNN}$ are decomposed (or factorised) into multiplication of two matrices:

$$\mathbf{M}^{TDNN} = \mathbf{AB}, \tag{4.18}$$

where one of the multiplicant matrices is constrained to be *semi-orthogonal*. In linear algebra, $\mathbf{M}$ is called to be semi-orthogonal if,

$$\mathbf{MM}^T = \mathbf{M}^T\mathbf{M} = \mathbf{I}, \tag{4.19}$$

where $\mathbf{I}$ is the identity matrix.

In practice, TDNN layers are factorised as follows: consider a standard TDNN topology with a hidden dimension of h and a convolution kernel of 3×1. Then the parameter matrix,

$\mathbf{M}^{TDNN}$, would have the dimensions of $h^{TDNN} \times 3h^{TDNN}$. Suppose we choose $\mathbf{A}$ to be a much smaller matrix with a size $h^{TDNN} \times h^{\mathbf{A}}$ where $h^{\mathbf{A}} < h^{TDNN}$. Consequently, $\mathbf{B}$ has the size of $h^{\mathbf{A}} \times h^{TDNN}$ and is constrained to be semi-orthogonal. Here, the matrix $\mathbf{A}$ is referred to as linear bottleneck and its size the bottleneck dimension. As the main goal of this procedure is to reduce computational complexity, a small value is chosen for the bottleneck dimension. In the same paper, the factorisation step was shown to reduce the trainable parameters to a quarter compared to the original TDNN and had a faster training while *WER* improvement is also observed (Povey et al., 2018a). Finally, in the implementation of the factorised TDNN (or TDNN-f) blocks, the input parameter matrix is passed through ReLU activation, batch normalisation and dropout layers placed in cascade before the TDNN factoriwation procedure.

Similar to LSTMs, the design principle of TDNNs is to learn temporal information. However, the TDNN is mainly based on the convolution operation, and hence is parallelisable. In addition, factorising TDNNs (as mentioned above) leads to a reduction in the number of trainable parameters. Considering these points, providing a more efficient training stands out as one of the major advantages of using TDNNs compared to LSTMs.

Baseline Network Topology

The input features can be fed directly into the TDNN blocks as in Figure 4.6, which is the architecture in Kaldi's standard Librispeech recipe, and later used for lyrics transcription (Dabike and Barker, 2019; Gupta et al., 2020). According to this architecture, the output of the TDNN blocks is projected to the final classification (softmax) layer via a couple of fully connected layers (FC).

Convolutional Front-end

Two dimensional CNNs are widely used to capture patterns and temporal information from audio spectral images (Huang et al., 2015). Applied in conjuction with *pooling* (or subsampling) steps, CNNs can also be used for dimensionality reduction for the deep layers in the network (Cai et al., 2014; Passricha and Aggarwal, 2018; Pardede et al., 2018). Inspired by this, we propose adding a CNN-based front-end network to summarise the input acoustic features prior to being processed by the TDNN-f blocks (Figure 4.7).

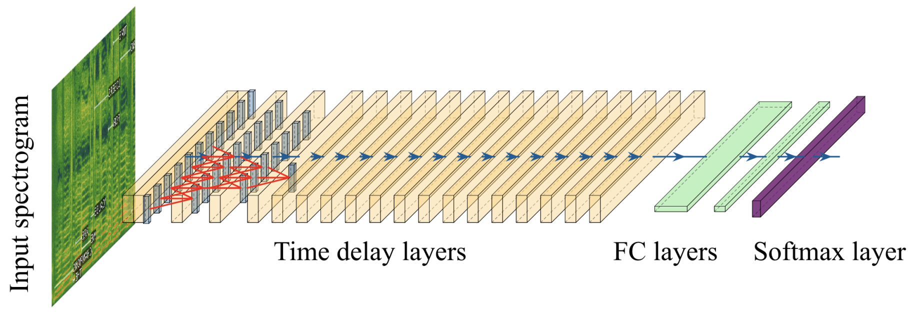


Figure 4.6: Single stream TDNN architecture. The stack of yellow blocks are the TDNN part of the network. Each small blue block represent within yellow blocks represent a TDNN-f node, and the red lines are for the convolutional operations between these nodes.

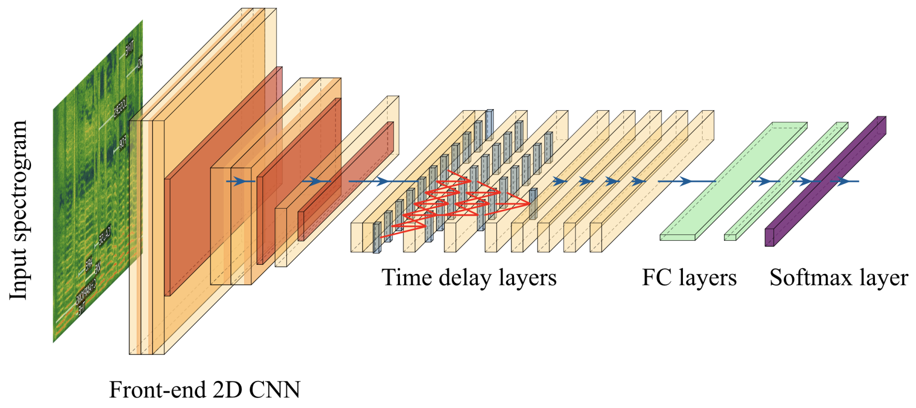


Figure 4.7: Single stream TDNN architecture with 2D CNN front-end. Notice the reduced number of TDNN-f layers.

The front-end structure consists of a stack of 2D-convolutional layers with 3×3 kernels, where feature subsampling with a factor of 2, is applied. With this, we aim to obtain compact embeddings for the following part of the network. For instance, suppose using 40-band filterbank features in the input. Instead of directly feeding a feature vector of 40 to TDNN-f blocks, the latent space vector has a height of 5 (after three subsampling stages). We tune the front-end topology and test the effectiveness of this method in terms of recognition performances and the reduction in the number of trainable parameters in Section 4.4.5.

Multistream TDNN - MstreNet

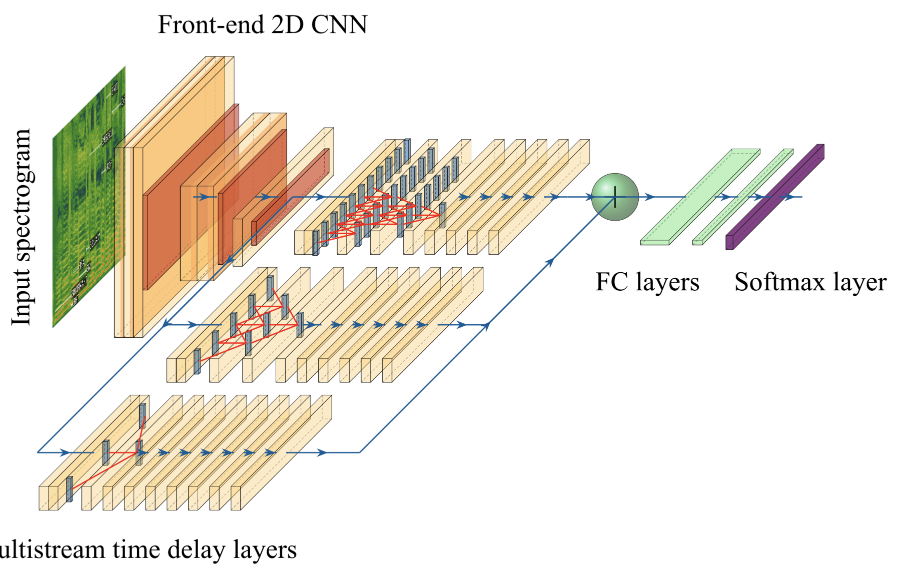
In consideration of the observations mentioned above in Section 4.3.1, this study treats polyphonic recordings as singing performances in noisy environments within the context of automatic lyrics transcription, similar to the famous cocktail party problem (Cherry, 1953). One method for ASR in noisy environments is *multistream acoustic modeling*, which is initially proposed based on empirical analysis of the speech decoding and perception principles in the human auditory system (Allen, 1995; Hermansky, 2019). According to this, acoustic signals enter into the cochlea and are decomposed into multiple frequency bands through hair cells, where all the decomposed streams are processed in parallel.

The multistream ASR approach was initially implemented in the DNN-HMM framework through splitting the frequency bands in the spectrogram into multiple subbands with different recognisers. Then the output phoneme posteriorgrams are recombined through either a weighted summation or a fully connected neural network layer (Hermansky, 2013; Zhao and Morgan, 2008). This approach was found to have comparable recognition rates to a standard single stream approach on clean speech, however performed much better on noise-corrupted data. Later multistream attempts applied spectrum modulation and processing acoustic features at multiple resolutions (Mesgarani et al., 2010). Mallidi and Hermansky (2016a) and Mallidi and Hermansky (2016b) simplified the multi-recogniser framework in (Hermansky, 2013) into a single neural network where input feature streams are randomly dropped out. At inference, the likeliest word sequence is obtained via a tree search algorithm. Instead of random activation of feature streams, Wang et al. (2019a) and Li et al. (2019b) proposed using the attention mechanism which was shown to be effective specifically for the multiple microphone setting (Barker et al., 2017; Ravanelli et al., 2017).

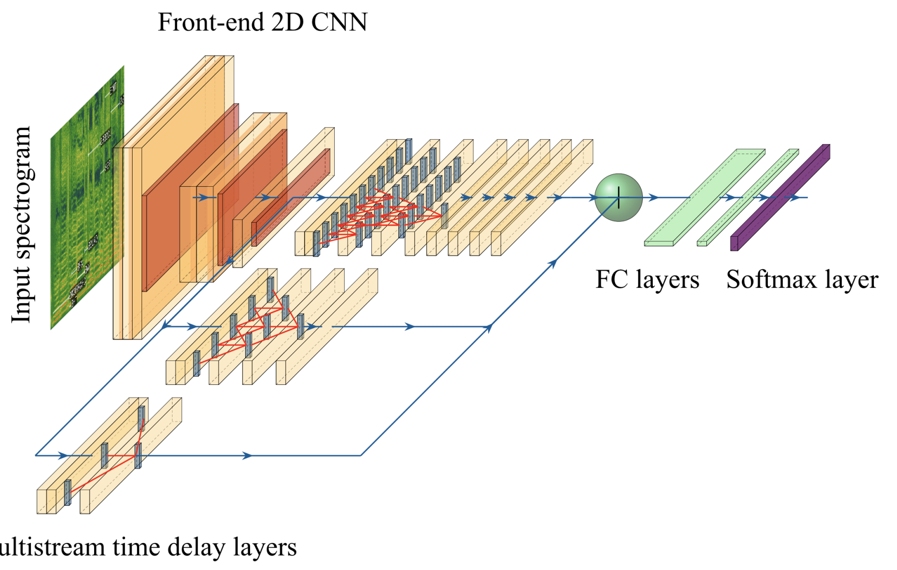
Recently in (Han et al., 2021; Pan et al., 2020), multistream ASR was applied through a single network in which the acoustic features are directly processed through multiple streams of TDNN-f blocks, where each stream has a unique dilation rate τ. An illustration of the overall structure of this multistream TDNN-f architecture can be seen in Figure 4.8a. According to this, input features, $\mathbf{x}_i$ are first processed by a single stream of a 2D-CNN front-end (described above),

$$\mathbf{h}_i = \textit{Stacked-2D-CNN}(\mathbf{x}_i) \tag{4.20}$$

The compressed hidden representations, $\mathbf{h}_i$ are then fed into the multiple streams of TDNN-f's. Each stream j of TDNN-f's has a unique time dilation rate, d_j, encoding temporal information with different resolutions into separately trained representations,



(a) Multistream architecture with even TDNN streams



(b) Multistream architecture with diverse TDNN streams

Figure 4.8: The TDNN of the architecture in 4.8a have identical hyperparameters (such as number of TDNN-f layers and hidden dimensions) except their time dilation rates. 4.8b shows diverse TDNN-f streams where the number of layers decrease with increasing dilation rate. The lesser the frequency of blue points, the higher the dilation rate is.

$$\mathbf{z}_i^j = \textit{Stacked-TDNN-}f_j(\mathbf{h}_i|d_j). \tag{4.21}$$

The output of each stream, $\mathbf{z}_i^j$, are then concatenated and succeeded by a pair of fully connected (FC) layers before projecting to the output layer,

$$a_i(s) = \mathrm{softmax}(\mathit{FC}(\mathit{FC}(\mathit{Concat}(\mathbf{z}_i^1, \mathbf{z}_i^2, ..., \mathbf{z}_i^J)))). \tag{4.22}$$

where $a_i(s)$ is the activations of the output layer obtained via the $\mathrm{softmax}$ function.

The architectures proposed by Han et al. (2021) and Pan et al. (2020) have identical streams in terms of the number of hidden layers within TDNN-f blocks and their hidden dimensions, despite having varying dilation rates and thus receptive fields (*RF*). This setting may not be optimal as the *rf*s of deeper layers in the streams with greater dilation rates may exceed the length of the audio chunk to be processed, making corresponding network connections redundant. For instance, consider training the DNNs on 1.5-second audio chunks (as in (Povey et al., 2016)), having a frame rate of 30 ms for the feature vectors, and 8 layers across all streams. According to Equation 4.17, the streams with dilation rates $d_i = \{3, 6, 9\}$ have receptive fields, $\mathit{rf}_3 = 1440$, $\mathit{rf}_6 = 2880$, $\mathit{rf}_9 = 4320$ milliseconds.

In our variant of the multistream TDNN architecture (see Figure 4.8a), the model complexity is reduced by adjusting the number of TDNN layers, N_j, with respect to their dilation rates, ideally in order to exclude network nodes that are temporarily redundant. In addition, we test the effect of diversifying TDNN streams in terms of their hidden dimensions k_j. According to this, the hidden representations at the end of the multistream TDNN blocks can be expressed as:

$$\mathbf{z}_i^j = \textit{Stacked-TDNN-}f_j(\mathbf{h}_i | d_j, N_j, k_j). \tag{4.23}$$

In our recent paper (Demirel et al., 2021b), this architecture is referred to as MStreNet, its performance is shown to be superior to the single-stream TDNN architecture, especially on the polyphonic recordings. In Section 4.4.5, the experiments in the paper extended to find an optimal network setting.

4.4 Experiments and Results

This section includes the experiments held to test the proposed methods and design choices for the DNN-HMM framework. Similar to the direction presented at the beginning of this chapter, this section follows a top-to-bottom approach in the progression of the experiments and performs ablative tests to evaluate the proposed methods. It begins with introducing the baseline model and continues with finding the most suitable corpus for building the language model. After that, the alternative pronunciation generation methods proposed in Section 4.2.2 are tested. Then, the effectiveness of cross-domain training and music informed silence modeling approaches are tested. Next step of experiments is concerned with finding the optimal setting for the compact multistream time delay neural network architecture. To provide more scalable results, we measure model performance generalisability across a number of benchmark evaluation sets used in research (i.e. all the test sets studied in Chapter 3).

4.4.1 Constructing the Baseline Model

The DNN-HMM setup in our study follows the pipeline of the *chain* recipe within the Kaldi framework (Povey et al., 2011), which is based on the LFMMI / sequence discriminative training principles explained in Sections 2.2.3 and 2.2.4. In order to initiate the training of neural networks, phoneme level alignments and lattices are necessary as a preprocessing step. For this reason, the overall training pipeline starts with training a Gaussian Mixture Model (GMM) - HMM baseline model that will be used to generate alignments. The overall pipeline follows the standard Kaldi - LVCSR recipe for this, and the GMM-HMM training procedure is explained below. Note that all experiments until the cross-domain training stage (Section 4.4.4) are done on the monophonic *DAMP* dataset and no data augmentation is applied.

The GMM-HMM Model

The training pipeline initiates with training a monophone GMM-HMM acoustic model[11]. The monophone model is trained on the 13-dimensional MFCC features. Monophone acoustic modeling treats phoneme types as singular units, meaning that a phoneme type would have a certain acoustic model independent from the context. This can be interpreted as meaning a certain phoneme type is pronounced the same way in all contexts. Context and word position information is embedded in phonemes through the procedure explained in Section 2.2.1, which results in a triphone model. To include the context in the acoustic domain, the Δ and Δ-Δ

[11]A monophone based acoustic model uses the base phoneme classes as explained in Section 4.2 where no context or position dependency is applied. Note that the monophone model should not be confused the term *monophonic* which is used to refer models trained on solo singing recordings.

features (i.e. *dynamic coefficients*) are concatenated to the MFCC features at each time step t (Gales and Young, 2008). The training of this model is initiated with the alignments obtained with the initial monophone model. The triphone model trained on dynamic features is used to regenerate alignments that are used to train another triphone model. In this next model, the input feature space is reduced using Linear Discriminant Analysis (LDA) (Haeb-Umbach and Ney, 1992) and the reduced features are mapped into a speaker-invariant space using feature-map maximum linear likelihood regression (fMLLR) transformation (Anastasakos et al., 1996). Finally, the pronunciation probabilities are computed through the procedure explained in Section 4.2.3. With these updated probabilities the lexicon transducer, *L.fst*, is reconstructed, which is used to train another singer adaptive triphone GMM-HMM model. The final model described here is referred to as the GMM-HMM baseline model in our experiments.

Neural Network Training

This section outlines the training details of DNNs [12]:

- The networks takes 40-band filter bank features as the input. To extract these, a hop size of 10 ms and window size of 30 ms are used. According to this pipeline, frame subsampling with a factor of 3 on the input feature vectors is applied. This means that one in every 3 frames is taken into account as the input features and the next 2 frames are skipped. In other words, there is one feature vector for every 30 ms. The Kaldi chain recipe also makes use of the skipped frames in training. Once the training examples are generated, for each sample, the feature vector indices are shifted by 1 and 2. Shifting by 1 means that the feature vectors correspond to the first stream of skipped frames after frame subsampling, and shifting by 2 would correspond to the last stream of skipped frames. By applying frame-shifting and regenerating training data from the skipped frames (feature vector streams), we create different versions of the same training data, hence this procedure serves as data augmentation. In research, this is referred to as *frame-shift* augmentation.

- Because of the reduced frame rate, a modified HMM topology is employed in the LFMMI training (Povey et al., 2016). As opposed to the 3-state topology in the GMM setting,

[12]The DNN training employs the standard settings in the Kaldi chain recipe.

the Kaldi chain recipe uses a 1-state HMM topology which can be traversed only in one transition. The transition probabilities between consecutive HMM states is constant and set to be 0.5, so the input and output transition probabilities would sum up to 1. According to Povey et al. (2016), learning transition probabilities did not improve the modeling power whereas it slowed down training.

- Training of the acoustic model is performed on fixed size audio chunks of 1.5 seconds instead of using the entire singing performance. In order to inform training on the utterance-level when doing the forward-backward computation, a number of steps are taken. Recall from Equation 2.18 that the MMI objective contains two separate gradients, one for the numerator which is utterance-specific and one for the denominator that is computed on the entire training data. For the former, the chunk-wise numerator graphs are constructed as follows: Prior to training the neural networks, phoneme lattices containing alternative pronunciations are obtained using a pretrained GMM-HMM aligner. Each state transition in the resulting lattices is labeled with a unique label, which is then used to split the utterance-wise numerator graphs into appropriate chunks. For the denominator graph, the phone-level FST G is composed with the context-dependency C from the left, and then composed again with the updated 3-state HMM H_{LFMMI} from the left. However, the probabilities in the decoding graph reflect the overall singing performance, hence they are incompatible with the numerator graph considering training on chunks of 1.5 seconds. Povey et al. (2016) circumvent this by running the denominator HMM for 100 iterations and take the average of state distributions accordingly. Then, this average is used as the initial state probabilities for the chunk-wise denominator HMM graphs. The final state probabilities are set to 1.

- Neural network training is performed on minibatches of 128 where each training sample is a 1.5-seconds audio chunk as explained above. The numerator FSTs are initially obtained on the chunk-level, then the FSTs of chunks belonging to the same utterances are appended one top of each other linearly, so that the backward-forward pass is performed only once per utterance instead of applying it separately for each audio chunk. A decaying learning rate is applied with the beginning and final learning rates of 10^{-4} and 10^{-5}

respectively. Stochastic Gradient Descent (SGD) (Kiefer and Wolfowitz, 1952) is used as the neural network optimiser. The network is trained for 6 epochs. Early stopping is not applied though, but increasing the number of training epochs did not result in improved results.

- To achieve *singer-adaptive training*, i-Vectors with the dimension of 100 are extracted, which provide global speaker-id embeddings (Kenny et al., 2005). The i-Vector training is performed on the *DAMP* recordings. For the cross-domain models, the i-Vector model is retrained on the combination of *DAMP* and *DALI*.
- To be able to report results in line with our most recent publication on the topic (Demirel et al., 2021b), the same single stream CTDNN structure with 8 TDNN layers is used as the baseline neural network architecture. For the same reason, the experiments involve the same dataset as in our previous work (Demirel et al., 2021b) for building the baseline language models, which is the Lyrics Corpus presented in Chapter 3. The global optimal DNN-HMM setup is then determined according to the final cross-dataset evaluation where each proposed method is tested individually.

4.4.2 Language Model

Previous research in ALT noted that a lyrics-focused language model is beneficial in terms of word recognition compared to using transcriptions of spoken utterances (Gupta et al., 2020; Zhang et al., 2021). At this initial step, a similar comparison is provided in Table 4.7 including both the baseline GMM-HMM and LFMMI models. All models apply a single-pass decoding via a 4-gram LM.

LM Corpus	GMM-HMM		LFMMI	
	Dev	*Test*	*Dev*	*Test*
Lyrics	38.55	45.69	14.73	17.73
Librispeech	65.34	67.26	39.46	42.65
Both	38.85	45.94	12.99	17.08

Table 4.7: *WER* scores produced by 4-gram LMs trained on different corpora.

The above comparison indicates that an in-domain language model for ALT is beneficial for word recognition in the DNN-HMM setting. In addition, although increasing the lexical diversity of the lyrics corpus via merging it with the Librispeech transcriptions was not clearly beneficial for the GMM-HMM model, it led to a slight *WER* improvement in the LFMMI / DNN-HMM model.

RNNLM Rescoring

The choice of the training set is an important factor also for the RNNLM rescoring procedure. Specifically, the modeling power of RNNLMs is prone to get degraded when trained on noisy data. Table 4.8 gives a comparison of RNNLM models trained on different corpora. The first RNNLM is trained on the transcriptions of the *DAMP*^{train} split. Next, the training set is extended with the lyrics of the *DALI*^{train} split. The model on the fourth column is trained on the lyrics corpus used to train the lyrics-specific n-gram language model above. Next, we test the effectiveness of combining Librispeech and the lyrics corpora.

	4-gram LM		*DAMP*		*DAMP* $\bigcup$ *DALI*		Lyrics Corpus		Lyrics + Speech	
	Dev	*Test*	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*
WER	13.76	17.73	**10.20**	**14.07**	13.02	14.27	13.12	16.86	13.79	16.45
C	87.08	83.48	**90.57**	**87.07**	87.76	86.88	87.61	84.26	87.06	84.57
S	7.52	11.71	**5.48**	9.21	7.32	**9.19**	7.37	11.18	7.42	10.68
I	0.85	1.10	0.77	1.13	0.77	0.97	**0.72**	1.10	0.85	**0.96**
D	5.40	4.91	**3.96**	**3.72**	4.93	4.13	5.03	4.62	5.53	4.81

Table 4.8: RNNLM scores trained on different corpora

In addition to overall *WER*s, Table 4.8 provides scores explicitly in terms of the percentage of correct word predictions, (C), plus substitution (S), insertion (I) and deletion errors (D). It can be seen that extending the training corpus led to increased deletion errors. Table 4.8 shows that the best results are achieved when the RNNLM is trained only on the transcriptions of the *DAMP*^{train} set. This is interesting because RNNLM training on larger corpora did not seem to improve word recognition results compared to the smaller *DAMP*^{train} set, which is counter-intuitive from the deep learning perspective.

4.4.3 Pronunciation Model

At this stage, the effects of alternative pronunciation generation methods proposed in Section 4.2.2 are tested in terms of word error rates. For this, we begin by constructing the pronunciation model-specific GMM-HMM models to generate phoneme alignments. This is crucial for this stage of experiments as the acoustic model optimization is based on phoneme alignments and the DNN training is performed on phoneme-based LMs.

At first, the vowel extension factor is tuned when generating the vowel extended alternative pronunciations. In their work, Gupta et al. (2018) use the combination of vowel extensions ($VE = \{1, 2, 3, 4\}$)[13]. However this potentially increases the size of the lexicon transducer (*L.fst*). To find the optimal setting in this context, lexicons extended with only one *VE* are compared first (results for 1-4 in Table 4.9). Then in the two rightmost columns, results for the lexicons where vowel extended pronunciations with smaller *VE*s are also added. For example, for the lexicon for the column denoted with ≤ 3, the resulting list of alternative generations of the word *OCEANS* also includes *OW OW SH AH N Z*, *OW SH AH AH N Z*, *OW SH AH AH AH N Z*, etc. In addition to *WER*s, we compare models with respect to size of the resulting decoding graph (*HCLG.fst*) and real-time factors.

VE	1	2	3	4	≤ 3	≤ 4
WER	17.73	**17.27**	17.47	17.33	17.85	17.61
Size	2.5GB	2.5GB	2.8GB	3.1GB	3.0GB	3.4GB
RTF	0.6899	0.4275	0.4169	0.4115	0.4455	0.4459

Table 4.9: Tuning the vowel extension factor

Table 4.9 above shows that using alternative pronunciations with extended vowels generally leads to better performance, however none of the extension factors improves the recognition rates considerably more than others. Moreover, extending vowels also resulted in a faster inference performance (lower RTF), possible due to an increased recognition rate. According to these results, $VE = 2$ is sufficient to achieve performance improvement while having a smaller decoding graph and a comparable RTF with other *VE*s. For this reason, $VE = 2$ is chosen to build the vowel extended lexicon used in the next experimental step.

In Table 4.10, the comparison of different alternative pronunciation generation methods is given. The leftmost column with the label L_{CMU} represents the model based on the standard

[13]Note that $VE = 1$ means using the non-vowel-extended lexicon.

CMU pronunciation dictionary. In L_1, the plosive consonants mentioned in Section 4.2.2 are removed, '*D,T*', from word endings in the alternative pronunciations. L_2 corresponds to the lexicon extended with *Pron-2* where the *Z* sounds are removed from word endings. L_3 is the vowel extended lexicon which was tuned above. In order to observe whether the pronunciation generations are beneficial specifically for singing, the read / singing splits of the NUS Corpus (Duan et al., 2013) are included in the evaluation and native English speakers are excluded in evaluation as those samples are the ones used in the pronunciation analysis in Section 4.2.1. Finally, L_4 includes the pronunciation variants in both L_1 and L_3. Here, the method for L_2 is not included in L_4, as it did not lead to improvements.

	L_{CMU}	L_1	L_2	L_3	L_4
$DAMP^{test}$	17.73	17.41	18.56	17.33	**17.17**
NUS^{read}	7.47	**7.15**	8.32	7.24	7.31
NUS^{sing}	8.04	7.85	9.07	7.71	**7.68**

Table 4.10: WERs obtained using different lexicon variants

According to the table above, both L_1 and L_3 lead to slightly improved word recognition rates, and the best results are achieved by combining them in L_4. Replacing the Z sound with S did not help with the performance, hence we did not include the *Pron-2* method when generating L_4. Note that improvements due to the extended lexicon are more apparent for singing according to the results for the reading and singing splits of the NUS Corpus. This indicates that the heuristics used to generate the alternative pronunciations are more suitable for the singing data.

Table 4.11 shows a more detailed comparison between the speech and singing adapted lexicons (L_{CMU} and L_4 respectively), specifically in terms of the rate of substitutions, deletions and insertions as well as *CER*s. This shows that most improvements are observed on deletions while the improvement in insertions is marginal. Word recognition systems might produce no word output when the prediction is ambiguous. The observation related to deletion errors might be interpreted as follows: the recognition procedure suffers less from the pronunciation ambiguities due to prolonged vowels or omitted plosives when using L_4 as the pronunciation dictionary.

Furthermore, we extract case-specific results to observe whether the proposed lexicon extension methods lead to improved results relevant to newly generated alternative pronunciations. Specifically to test the method *Pron-1*, the results for the words ending with the low confidence

	L_{CMU}				L_4			
	ER	*S*	*I*	*D*	*ER*	*S*	*I*	*D*
word								
DAMP^{test}	17.73	11.70	1.10	4.91	17.17	11.02	1.36	3.89
NUS^{read}	7.47	5.34	0.82	1.31	7.31	5.31	0.88	1.12
NUS^{sing}	8.04	5.67	0.68	1.69	7.56	5.61	0.59	1.36
character								
DAMP^{test}	11.56	5.13	1.47	4.96	11.26	5.25	1.74	4.27
NUS^{read}	4.24	2.07	1.05	1.12	4.32	2.20	1.01	1.12
NUS^{sing}	4.37	2.22	0.55	1.60	4.27	2.08	0.64	1.55

Table 4.11: Word and character error rates using the standard (L_{CMU}) and the singing-adapted (L_4) pronunciation dictionaries.

plosives are compared. For *Pron-3*, results only on vowel characters are given in Table 4.12. Note that while the results for the former are *WER*s, *CER* is used to evaluate vowels, i.e *vowel error rate (VER)*. The results in Table 4.12 show that the lexicon extensions led to improvements in the aforementioned specific cases. This hints that improvements due to adding alternative pronunciations not only provide more flexibility in the search space during decoding but also can be beneficial for specific utterances.

	L_{CMU}			L_4		
	ER	*S*	*D*	*ER*	*S*	*D*
words ending with plosives (*WER*)						
DAMP^{test}	22.84	13.06	7.78	**17.67**	10.15	7.21
NUS^{read}	9.74	8.82	0.91	**9.01**	7.90	1.10
NUS^{sing}	14.01	7.76	5.73	**7.94**	5.73	2.21
vowels (*VER*)						
DAMP^{test}	13.20	6.47	6.72	**9.80**	5.59	4.21
NUS^{read}	4.02	2.44	1.58	**3.99**	2.55	1.44
NUS^{sing}	7.23	2.98	4.26	**6.71**	3.03	3.68

Table 4.12: Error analysis for plosives and vowels

Discussion

The pronunciation dictionary can possibly be extended further to adapt to the common pronunciation variances in singing. Additional alternative pronunciations may be generated through a statistical analysis of the interchange of phonemes between sung and spoken utterances. This can be performed in a context-dependent manner via observing the neighbouring phonemes. The analysis above shows that the proposed singing adapted lexicon methods lead to consistent but modest improvements in the overall word error rates and the combination of two methods (i.e.

$L_1 + L_3 := L_4$) resulted in the best performance. According to the case-specific error analysis, both methods were beneficial in their targeted direction. In addition, the vowel extension method is also observed to speed up inference.

Note that this section of Chapter 4 follows the same experimental steps as our previous study in (Demirel et al., 2021c). In contrast, a smaller language model is used in the aforementioned study. In addition, data augmentation by means of speed perturbation is also applied and the neural network architecture has an extra self-attention layer (Demirel et al., 2021c). In order to provide a coherent and consistent results with the rest of the experiments in this thesis work, we have redone the experiments using the baseline setup explained in Section 4.4.1. Overall, these results are perfectly in line with our previous study (Demirel et al., 2021c).

Senones vs. Chenones

Here, the potential of using graphemes (or specifically chenones) within the DNN-HMM lyrics transcription setup is tested. For a clear comparison, the results for senone-based models using both the standard and the singing adapted lexicons are included.

Token	*WER*	*CER*
Senones (CMU)	17.73	11.56
Senones (singing)	17.17	11.26
Chenones	19.98	14.75

Table 4.13: *WER*s of different lexicon variants. Test set is *DAMP*^{test}.

Although the singing adapted lexicon is clearly superior to using chenones according to Table 4.13, the error rate differences between using senones and chenones are smaller in the case of using the standard pronunciation dictionary. Although having around 2% higher *WER*s, using chenones can be justified where a pronunciation dictionary for the language in question is not available. This analysis shows the potential of chenones in the contrast to senones. These results are interesting in particular for applications where a senone based system would require a manually curated pronunciation model, such as for multilingual lyrics transcription. For such applications, chenones are potentially a good candidate as subword units to train the acoustic model as they do not require language specific phonological expert knowledge to define a mapping between words and subword units.

4.4.4 Cross-Domain Training

In this section, we test our hypothesis on the possibility of constructing a single lyrics transcriber where the word recognition performance is not degraded across varying domains through training the acoustic model on multiple domains. Specifically, we suggest using the combination of *DAMP* and *DALI* as the training set.

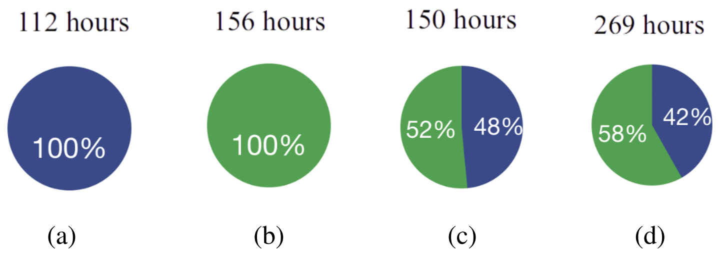


Figure 4.9: Summary of the training sets. a) *DAMP*, b) *DALI*, c) $(DAMP \bigcup DALI)_{small}$, d) $(DAMP \bigcup DALI)_{large}$: M_b (Gupta et al., 2020) uses a 200 hour version of this dataset.

The models in Table 4.14 are based on the baseline model used in the previous experimental stages, that uses the neural network architecture of 8-layer TDNN-f's with a CNN front-end, the lyrics corpus for the language model, and the standard pronunciation dictionary. The tested models differ only by their training sets. The top two lines are the domain-specific models trained on either *DAMP* or *DALI* datasets respectively. The second-to-last line is the first model that is trained jointly on monophonic and polyphonic recordings. For ablation tests, we initially combine subsets of both datasets keeping their relative proportion the same with respect to the number of singers and in a similar range in terms of total utterance duration. The total utterance duration of this initial combined set (denoted as $(DAMP{+}DALI)^{train}_{small}$) is kept at 150 hours to enable a fair comparison with domain-specific models (a,b in Figure 4.9). In training the final cross-domain model, the full versions of both training sets are combined. These dataset combinations are illustrated in Figure 4.9. Note that for evaluating M_{poly}, a subset of $DALI^{train}$ with 20 recordings $(DALI)^{dev}$ is chosen for tuning the language model scaling factor. Then, the cross-domain models use the combination $DAMP^{dev}$ and $DALI^{dev}$ as the development set.

According to Table 4.14, the model trained only on polyphonic recordings (M_{poly}) considerably outperforms the baseline monophonic model (M_{mono}) on polyphonic data although its performance is considerably poorer on monophonic recordings. On the other hand, results for the models in the last two lines show that including data from both domains can be observed to

Model	Train Set	Dev	*DAMP*^{test}	*DALI*^{test240}
M_{mono}	*DAMP*	13.69	17.73	78.42
M_{poly}	*DALI*	37.90	61.95	59.19
$M_{\text{CD-small}}$	*DAMP+DALI*$_{small}$	30.27	17.83	56.75
$M_{\text{CD-large}}$	*DAMP+DALI*$_{large}$	28.05	**17.14**	**53.86**

Table 4.14: Cross-domain evaluation results. The cells in *gray* show the results on the development sets. Results on $\mathcal{D}_{self}^{test}$ are highlighted according to their corresponding training set ($\mathcal{D}^{train}$).

help preserving transcription performance across domains. In particular, consider the $M_{\text{CD-small}}$ which is trained on equal portions of monophonic and polyphonic recordings with a similar overall size compared domain-specific models. Despite the data from both domains being reduced to half of their size, the model performance is kept to a similar level across different domains. This validates the advantage of training an acoustic model both on polyphonic and monophonic recordings compared to building a domain-specific one. Moreover, including full versions of both *DAMP* and *DALI* datasets improved the performance on the polyphonic domain further. However, this did not help much on monophonic recordings.

Music Informed Silence Modeling

To enhance the modeling capability of cross-domain training, the music informed silence modeling proposed in Section 4.3.1 is applied. In Table 4.15, the effectiveness of this approach is shown over the initial cross-domain model.

Model	*dev*	*DAMP*^{test}	*DALI*^{test}
Cross-domain	28.05	**17.14**	53.86
+ music/sil tag	25.40	17.29	**47.00**

Table 4.15: Cross-domain training and music/silence tagging results

The results in Table 4.15 show that music informed silence tagging was effective on the polyphonic domain. In particular, close to 7% absolute *WER* improvement is observed on *DALI*^{test240}. However, this method did not lead to improvements on the monophonic case where the overall performance is similar to not using the explicit silent token tagging. Due to the improvements observed specifically on the polyphonic domain, music-informed silence tagging is always applied when referring to the cross-domain model within the cross-dataset evaluation in Section 4.4.6.

4.4.5 Neural Network Architecture

At this stage, the neural network architecture used to construct the acoustic model is tuned. The goal is to find an optimal setting for achieving an improved word recognition performance while keeping the architecture as compact as possible. For this, the experiments begin with the 2-D convolutional front-end network studied in Section 4.3.2 for ALT. Then, the number of layers in the TDNN-f blocks is tuned with respect to the receptive field of the last TDNN-f layer.

2-D CNN Front-end

In the presence of larger training datasets, purely TDNN-f based networks can perform well and do not necessarily need the convolutional front-end. However, the training sets used to train a lyrics transcriber in this study are much smaller than benchmark speech datasets. Because of this, a more compact feature representation is needed to exploit the potential of TDNN-f's to a larger extent. A 2 dimensional CNN at the front-end of the network is placed to summarise features to a smaller height while preserving the vertical (spectral) information in the learned representations. For feature dimension reduction, subsampling with a factor of 2 is applied on the vertical axis only. Subsampling is not applied on the temporal axis to preserve the time information to be later processed directly by the TDNN-f blocks.

	# layers	# subsamp.	h_{input} dim.	Kernel dim.	# params	*DAMP*^{dev}	*DAMP*^{test}
1)	0	0	40	N/A	7,939,520	16.72	19.93
2)	3	1	20	3×3	8,284,608	15.15	17.81
3)	4	2	10	3×3	8,166,912	14.46	18.35
4)	5	2	10	3×3	8,187,696	14.01	17.44
5)	6	3	5	3×3	5,893,312	**13.49**	**17.14**

Table 4.16: Parameter combinations tested for tuning the CNN front-end.

In Table 4.16, several front-end settings are tested in terms of number of CNN layers, subsampling steps, kernel dimensions and the total number of trainable network parameters[14]. The model on the top consists of only TDNN-f's and has 8 hidden layers. In the second model, a 3-layer CNN network is included where a subsampling is applied in between the second and third CNN layers, thus reducing the dimension to be fed to the TDNN-f's from 40 to 20. In the next model, another pair of subsampling and a CNN layer on the top of the previous front-end

[14]The experiments in Tables 4.16, 4.17 and 4.18 regarding NN architectures use data augmentation prior to training, hence these baseline results are slightly better than the baseline in the previous sections.

setting are added. The next model has an extra CNN layer between the two subsampling operations. Finally in the fifth row, the final front-end CNN structure is constructed by adding one more subsampling + CNN pair, reducing the feature height to 5 before the TDNN-f block. This final setting achieved the best performance and the most compact representation compared to previous front-end settings. Moreover, compared to the raw TDNN-f architecture, this final model achieved around 2.5% absolute *WER* improvement with a less complex network. According to these observations, we can conclude that the addition of the 2D CNN front-end is beneficial for our task of lyrics transcription given the limited size of the training data.

Time-delay Layers

The main goals in determining the optimal number of TDNN-f layers are: (1) to process all input feature frames at each time step t during the forward-backward pass, and (2) to avoid too large receptive fields (*rf*) that would require too much zero padding at the borders of training chunks, hence a sub-optimal model. Since, the training chunks have a fixed length of 1.5 seconds, we tune the number of hidden layers within the TDNN-f block with respect to the resulting *rf* at the end of the final layer. The architectures used by Dabike and Barker (2019); Gupta et al. (2020) have 16 TDNN-f layers with heterogeneous τ's across layers. The first 3 layers have $\tau = 1$, the fourth layer has $\tau = 0$, which is essentially a 1-D CNN with no dilation. The rest of the layers $[5 - 16]$ have $\tau = 3$. According to Equation 4.17, this architecture has *rf* $= 2310$ ms at the output of its top layer which exceeds the length of the audio chunks considerably. Motivated by this and in order to provide evidence that better performance can be achieved via a smaller TDNN-f network, the number of hidden layers is also tested in Table 4.17.

	# layers	*rf*	Num. params	$DAMP^{dev}$	$DAMP^{test}$
1)	7	1290 ms	5,614,208	14.41	17.21
2)	8	1470 ms	5,876,864	13.49	17.14
3)	9	1650 ms	5,893,312	13.54	**17.08**
4)	10	1830 ms	6,139,520	14.82	17.21
5)	16	2310 ms	7,978,112	13.42	17.38
6)	16	2880 ms	19,171,248	**13.21**	17.84

Table 4.17: Tuning the TDNN layers.

According to Table 4.17, models with *rf* < 2 seconds perform better than the models with *rf* > 2 seconds. The model on the third row which has 9 TDNN-f layers has comparable results with models with fewer hidden layers and also with similar model complexities. On the other hand, this model has slightly larger *rf* than 1.5 seconds, hence the network is able to process all the input features at every time step and does not require as much padding as models with larger *rf*'s while having less model complexity. We choose this model as the baseline TDNN stream during the design of the multistream architecture in the following section.

Finally, we add the architecture that is previously used by Dabike and Barker (2019) and Gupta et al. (2020) which does not have the CNN front-end. Compared with the network having the identical TDNN network but also the 2-D CNN front-end (line 5 in Table 4.17), the purely TDNN based architecture does not have better *WER* scores on $DAMP^{test}$ while having a much more complex network. This is an additional evidence for the benefit of using the CNN front-end studied in the previous stage of experiments.

Multistream CTDNN

Now that the front-end and the baseline TDNN stream is tuned, we can proceed with the design of the multiple TDNN streaming architecture. As previously mentioned in Section 4.3.2, the motivation behind multistream acoustic modeling is to enhance the robustness of transcribers specifically on polyphonic recordings. For this reason, the $DALI^{test240}$ is included in evaluating the multistream architectures. For hyperparameter tuning, we merge $DAMP^{dev}$ and $DALI^{dev}$ and use this as the development set.

The design principles follow the receptive field tuning approach similar to the preceding step. According to the empirical results on the design of multistream TDNN architectures used in DNN-HMM speech recognition previously reported in the literature (Han et al., 2021), the best setting in terms of dilation rates was achieved when layers have a greatest common divisor (g.c.d.) greater than 1. Some examples for this could be, $\tau = \{2, 4, 6, 8\}$ with g.c.d. 2 or $\tau = \{6, 9, 12\}$ with g.c.d. 3. Specifically the second example was shown to be the best setup (Pan et al., 2020). In Table 4.18, we test similar models where the g.c.d. of all dilation rates is 3.

The effect of using different combinations of τ's can be seen through the results in lines 2,3 and 4 where the model complexity is kept the same. Although including larger τ seems to improve results on $DAMP^{test}$, the opposite applies on the polyphonic $DALI^{test240}$. As the motivation for the multistream approach is to observe improvements in the polyphonic case,

	τ	# Layers	Hidden Dim.	Num. params	*dev*	*DAMP*^{test}	*DALI*^{test240}
1)	3	9	512	5,893,312	27.68	17.08	52.25
2)	$3 \times 6 \times 9$	$9 \times 9 \times 9$	$512 \times 512 \times 512$	11,720,320	26.69	16.75	**51.38**
3)	$6 \times 9 \times 12$	$9 \times 9 \times 9$	$512 \times 512 \times 512$	11,720,320	28.13	16.64	54.31
4)	$3 \times 9 \times 15$	$9 \times 9 \times 9$	$512 \times 512 \times 512$	11,720,320	27.92	16.41	54.73
5)	$3 \times 6 \times 9$	$9 \times 4 \times 3$	$512 \times 512 \times 512$	8,831,104	26.65	16.45	**49.32**
6)	$3 \times 6 \times 9$	$9 \times 9 \times 9$	$512 \times 256 \times 172$	7,293,968	27.13	**16.08**	52.54
7)	$3 \times 6 \times 9$	$9 \times 4 \times 3$	$512 \times 256 \times 172$	6,825,796	27.38	16.62	51.92

Table 4.18: Parameter combinations tested for tuning the Multistream TDNN architecture.

we choose the setting in line 2 when further tuning the number of layers and hidden dimension sizes in TDNN streams. This architecture corresponds to the one in Figure 4.7.

Next, the variants of the setting in line 2 are compared. The architecture in line have identical TDNN structures (except for τ). Its variants in lines 6,5 have either reduced number of hidden layers (N) or hidden dimensions respectively depending on the τ. Both directions of model reduction are applied in line 7. N is reduced for the streams with larger dilation rates ($\tau = \{6, 9\}$ to keep *rf* similar across all streams. Lines 5,7 have 4 and 3 layers at the streams with $\tau = 6$ and $\tau = 9$ having *rf* values of 1470 and 1650 ms respectively. Note that, adding one more TDNN layer on these streams would result in increasing the *rf* too much, which is shown above to be suboptimal in the single-stream TDNN experiments.

It is clear from the number of trainable parameters in Table 4.18 that diversifying TDNN streams helped reducing the model complexity. When model reduction is applied in terms of both aforementioned directions, as in line 7, the model complexity merges to a comparable level to its single stream counterpart while leading to improved performance both in polyphonic and monophonic models. However, this is not the best performing models for neither of the domains. The model with the same number of TDNN layers across streams (line 6) achieves the best results on the monophonic *DAMP*^{test} set. However, this model does not lead to any improvement on polyphonic data. On the other hand, the model with adjusted per-stream number of TDNN layers improved the *WER* around 3% on the polyphonic *DALI*^{test240} set while some improvement is also observed on monophonic recordings. Since our initial motivation for using the multistream TDNN architecture is to improve recognition rates from noisy setting and specifically polyphonic recordings at this stage of experiments, we choose the model in line 5 as the best multistream TDNN setting that will later be used to compare results in Section 4.4.6.

Note that this is the architecture illustrated in Figure 4.6.

In addition to *WER*s, we evaluate the efficiency of the multistream architecture during inference in Figure 4.10. We compare the RTFs of single-stream models {1,2,3,4} in Table 4.17 (noted as M_i^{single}) with the multistream models {2,5,6,7} in Table 4.18, which are noted as M_i^{multi} in Figure 4.10[15].

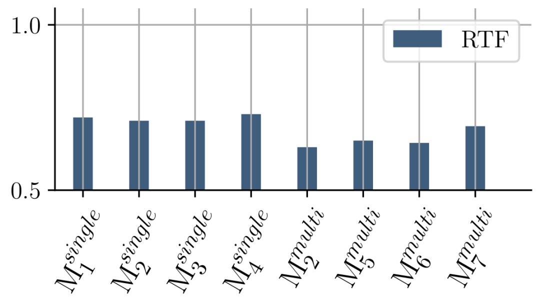


Figure 4.10: Comparison of RTFs between single and multistream models.

The analysis shows that the multistream architectures have a faster inference response compared to their single stream counterparts. We suspect that this might be due to two reasons: improved recognition performance and processing distant frames in early stages.

Robustness Against Reverberation

Reverberation is a well-known source of mismatch and a cause of reduction in word intelligibility (Poissant et al., 2006) in ASR. Similarly in ALT, reverberation may cause performance reduction in real-world applications due to varying acoustic environments in recording settings. Moreover, artificial reverberation is often applied on vocal tracks as a vocal effect. In this section, the model robustness is tested in different settings via curating simulated data from the a cappella evaluation sets.

Reverberation is generally modeled by means of a convolution of the input signal with a room impulse response (RIR). Recording real RIRs is quite standard in acoustics but requires effort, hence these are generally simulated by sampling the room size variables and the receiver position in the room. Following the RIR simulation steps in (Ko et al., 2017), the room size and receiver position parameters are uniformly sampled. Based on the set of sampled parameters, a number of RIRs are randomly generated using the method presented by Allen and Berkley

[15]We have included the MTDNN variants having the same set of τ's across TDNN streams.

(1979)[16]. The simulated RIRs are categorised into three based on the length and width of the sampled room.

- *Small room*: Ranges from 1 m to 10 m.
- *Medium room*: Ranges from 10 m to 30 m.
- *Large room*: Ranges from 30 m to 50 m.

In Figure 4.11, the *WER*s of the baseline single stream CTDNN and the compact MTDNN architectures (Models 1 and 5 in Table 4.18) across different reverberation conditions are given. According to the *WER* scores, the multistream model clearly outperforms its single stream counterpart. Interestingly, *WER*s tend to be lower for the large-reverb condition, which is counter-intuitive from the ASR perspective.

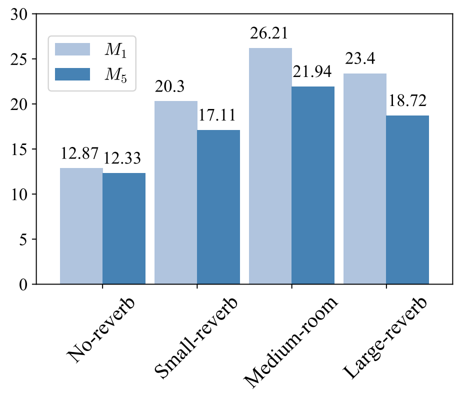


Figure 4.11: *WER* comparison across different reverberation settings.

In this part of the analysis, the neural network architecture of the acoustic model was constructed and tuned. The first step of tuning was performed on the 2-D convolutional network with subsampling at the front-end to summarise the features in the spectral domain. Then, the multistream TDNN architecture was tuned to get optimal settings. The multistream approach was compared to the traditional single stream structure in varying reverberation conditions and also compared for inference performance. We showed the superiority of the proposed compact multistream TDNN architecture over the standard single stream network.

[16]The experiments in this section use an open-source implementation of this method, available from `http://home.tiscali.nl/ehabets/rirgenerator.html`.

4.4.6 Cross-Dataset Evaluation

In the previous experimental stages, we tested the proposed methods for an improved DNN-HMM based lyrics transcriber. We tuned each of the transcriber's computational blocks (e.g. acoustic, pronunciation and language models) individually. For this, the test splits of *DAMP* and *DALI* sets are used. In this section, an evaluation of the proposed methods on the other benchmark test sets used in ALT research is provided.

The main goal for the cross-dataset evaluation is to observe improvements emerging from each of the proposed methods individually. At first, one method at a time is applied on the baseline model. Similar to the above experimental setups, the baseline model has a single-stream architecture, is trained on monophonic recordings, and uses the standard CMU pronunciation dictionary and 4-gram MaxEnt language model built on lyrics only (see Table 4.19). Model 2 is the same as the baseline model but is trained on the combination of *DAMP* and *DALI*. The model in line 3 has the compact multistream architecture tuned above. Model 4 uses the singing adapted lexicon. In line 5, the language model is trained on the combination of speech and lyrics corpora which was shown to be effective above in Section 4.4.2. In the last line, 2nd-pass RNNLM rescoring is applied on the baseline model that uses the 4-gram LM. The RNNLM is trained only on the *DAMP*^{train} transcriptions as it was shown above to be the best setting.

Model	Domain	Network	Pron.Model	Lang.
1) (*Baseline*)	Monophonic	Single-stream	CMU	Lyrics 4-gram
2)	**Cross-domain**	Single-stream	CMU	Lyrics 4-gram
3)	Monophonic	**Multistream**	CMU	Lyrics 4-gram
4)	Monophonic	Single-stream	**Singing-adapted**	Lyrics 4-gram
5)	Monophonic	Single-stream	CMU	**Mixed LM** 4-gram
1+)	Monophonic	Single-stream	CMU	Lyrics **RNNLM**

Table 4.19: Summary of models compared in the cross-dataset evaluation.

The model comparison begins with respect to the training and validation loss trends. This is shown in Figure 4.12 where each data point corresponds to the training or validation losses computed after each batch processing[17]. Notice that the losses of Models 1, 4 and 5 are almost indistinguishable. This is expected as all of them use the same baseline (single-stream) neural network architecture for training the acoustic model. It can be seen that the multistream architecture has slightly lower loss compared to the single-stream models. On the other hand,

[17]The reason why Model 2 has a larger number iterations is due to having a larger training set, i.e *DALI* $\bigcup$ *DAMP*.

the cross-domain model (Model 2) has much larger loss which is not surprising as the training uses larger training and validation sets. According to the overall loss trends, none of the models seems either to overfit, as the gap between training and validation losses does not seem to increase or to underfit, as the training loss is still lower than the validation loss.

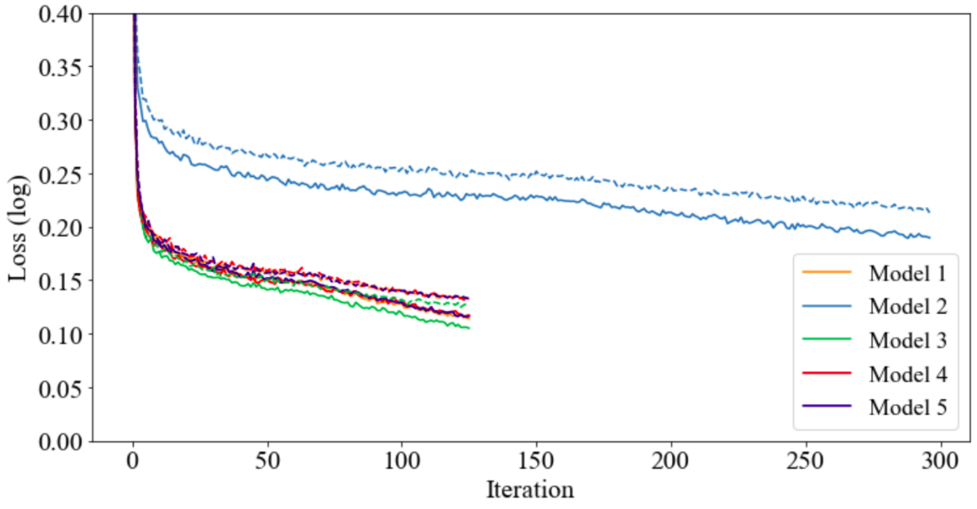


Figure 4.12: Train (full-lines) vs. validation losses (dashed lines).

To observe how generalisable the resulting *WER* scores are, we use the cross-dataset performance drop metric introduced in Section 3.4.2. The performance drop metric provides a measure of a model's performance generalisability across different evaluation datasets. According to Equation 3.4, it compares the scores obtained on the test split of the dataset used to build a model ($\mathcal{D}_{self}$) and on completely unseen data ($\mathcal{D}_{others}$). The evaluation scheme in this section uses all of the evaluation sets introduced in Chapter 3 and follows the framework in Section 3.4.2. For $\mathcal{D}_{self}^{test}$, we use either the $DAMP^{test}$ or its combination with $DALI^{test240}$ depending on the training set. In particular, consider the second line in Table 4.20 which is the cross-domain model. As this model is trained both on *DAMP* and *DALI* sets, we need to include polyphonic recordings in $\mathcal{D}_{self}^{test}$ for a fair comparison of the performance drop across different domains of singing data. The rest of the evaluation sets, namely the singing portion of NUS Corpus (i.e. NUS^{sing}), *Hansen*, *Mauch* and *Jamendo* form $\mathcal{D}_{others}^{test}$. In addition to overall *CPD* computed across all these test sets, we observe the performance drop specifically in monophonic or polyphonic domains. Domain specific *CPD*s are obtained on NUS^{sing} and the monophonic version of *Hansen* datasets for the monophonic, and on *Jamendo*, *Mauch* and $Hansen^{poly}$ sets for the polyphonic cases.

Model	Dev	$DAMP^{test}$	NUS	$Hansen^{mono}$	$CPD_{mono}(\%)$	$Hansen^{poly}$	*Mauch*	*Jamendo*	$DALI^{test240}$	$CPD_{poly}(\%)$	$CPD(\%)$
1)	13.76	17.73	8.04	**18.96**	-6.47	85.66	81.97	73.47	82.17	60.59	51.75
2)	32.59	17.48	7.92	19.14	-6.46	**48.19**	**47.23**	**41.74**	**52.93**	30.66	25.77
3)	13.29	**16.82**	8.44	19.90	-5.44	81.94	79.62	67.84	79.31	59.07	50.34
4)	13.66	17.17	7.68	19.00	-6.86	82.25	81.53	69.48	80.67	60.92	51.98
5)	12.99	17.08	**7.47**	19.45	-6.73	84.52	78.69	69.53	80.44	59.52	50.79
1+)	**10.20**	**14.09**	7.99	**18.62**	-3.44	84.52	80.00	73.12	80.05	64.28	55.36

Table 4.20: Cross-domain evaluation results. The best *WER*s for each set is highlighted with bold font. If best ones are achieved via RNNLM rescoring for a test set, the best result with the 4-gram LM is also highlighted.

There are several arguments deduced from the results in Table 4.20. At first sight, each of the proposed methods improved *WER*s on most of the test sets in $\mathcal{D}^{test}_{others}$. However, the best results for each test data occur with varying models. This is not surprising as each test set has distinct properties. In order to provide more insight on the comparison of the proposed methods, we summarise the implications drawn from Table 4.20 below:

- Model 1: The baseline model is trained on monophonic recordings. Although it performs well on monophonic $\mathcal{D}^{test}_{others}$, there seems to be a room for improvement as $CPD < 0$. Clearly this monophonic model suffers from dataset bias when tested on polyphonic recordings.

- Model 2: This is the baseline cross-domain model. *CPD* shows that cross-domain training helped increasing the model generalisability mostly on polyphonic recordings. The performance drop across $\mathcal{D}^{test}_{others}$ is by far the least with this model indicating the highest generalisability. The *WER* improvement on monophonic recordings is marginal, and CPD_{mono} is almost the same as the baseline model, which indicates cross-domain training does not affect the performance on monophonic recordings considerably. The performance drop improvement can be seen more explicitly in Figure 4.13.

- Model 3: The multistream architecture did not seem to improve *WER*s on monophonic $\mathcal{D}^{test}_{others}$. However, it led to the smallest $|CPD_{mono}|$ (with an n-gram LM) indicating its performance is the most generalisable on monophonic recordings. The multistream modeling approach seems to consistently improve *WER*s also on polyphonic sets, which supports the initial motivation for the design of the network architecture. Moreover, this method is the most effective method compared to other monophonic models (1,4,5,1+)

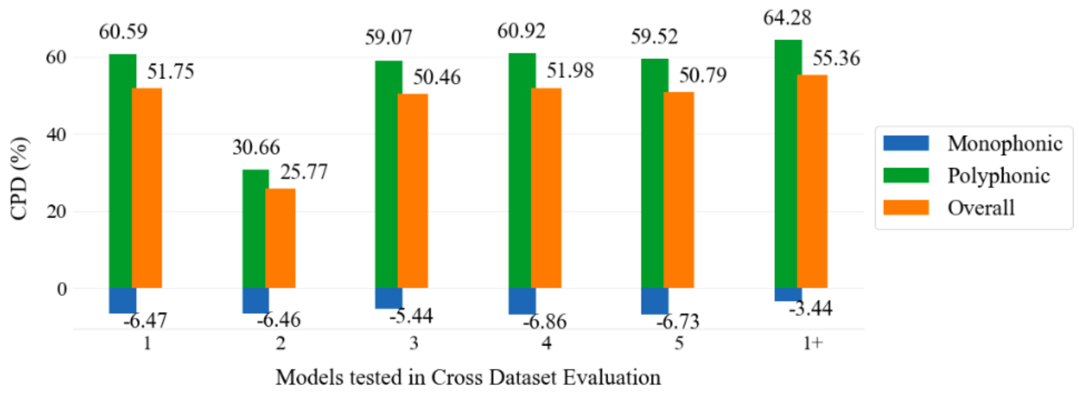


Figure 4.13: Performance drop across $\mathcal{D}_{others}^{test}$.

in terms of overall model generalisability. The best results on $\mathcal{D}_{self}^{test}$ and the smallest $|CPD_{mono}|$ are achieved with this model compared to other models with the 4-gram LM.

- Model 4: According to the *CPD* values, the singing adapted lexicon does not seem to lead to generalisable improvements as much as other methods, despite the slight but consistent decrease in per dataset *WER*s. The improvements have a parallel trend with Model 3 with slightly lower magnitude of improvement.

- Model 5: Combining lyrics and speech data in training the 4-gram language model improved *WER* scores on every set in $\mathcal{D}_{others}^{test}$ where the improvement on polyphonic recordings is more apparent. Compared to other monophonic models, the second least overall performance drop is achieved with this model on monophonic recordings.

- Model 1+: RNNLM rescoring improved *WER* on $\mathcal{D}_{self}^{test}$ considerably but was not as beneficial for monophonic $\mathcal{D}_{others}^{test}$. Therefore, $|CPD|$ decreased, meaning that the performance gap between $\mathcal{D}_{self}^{test}$ and monophonic $\mathcal{D}_{others}^{test}$ decreased. However, it did not lead to as much improvement on polyphonic sets, hence *CPD* increased due to the increased performance gap between D_{self}^{test} and $\mathcal{D}_{others}^{test}$. Thus, the second-pass RNNLM scoring did not increase the overall generalisability.

To sum up the findings of the above studied cross-dataset evaluation, all proposed methods improved the model's generalisability, where the improvement was by far the largest on the cross-domain model (Model 2). Each of Models 3,4,5 improved *WER* scores in most cases and their contribution to generalisability were modest compared to Model 2. Although the best scores on $\mathcal{D}_{self}^{test}$ are achieved via RNNLM rescoring, this did not improve recognition rates as much on $\mathcal{D}_{others}^{test}$ and did not contribute to the model's generalization power.

Performance on Separated Vocals

Here, *WER*s on the source-separated vocal tracks are compared. Since only the *Hansen* set has both the original vocal track and polyphonic mix available, the comparison is performed on this dataset in Table 4.21 and Figure 4.14. We measure the performance drop on separated vocals and polyphonic versions with respect to model performance on *Hansen*^{mono}. This is equivalent to considering *Hansen*^{mono} as $\mathcal{D}_{self}^{test}$ and the rest as $\mathcal{D}_{others}^{test}$. The goal for this comparison is to observe the performance degradation and how much gain can be achieved via source separation. Note that we use Deezer's *spleeter* toolkit (Hennequin et al., 2020) for source separation as it is representative of the state of the art in music source separation.

Model	Monophonic	Separated	Polyphonic
1	18.96	62.46	85.66
2	19.14	**56.72**	**44.05**
3	20.25	60.40	81.94
4	19.00	60.89	82.25
5	19.45	61.20	84.52
1+	18.62	60.93	84.52

Table 4.21: *WER* comparison on separated vocals and original polyphonic mix. Test set: *Hansen*.

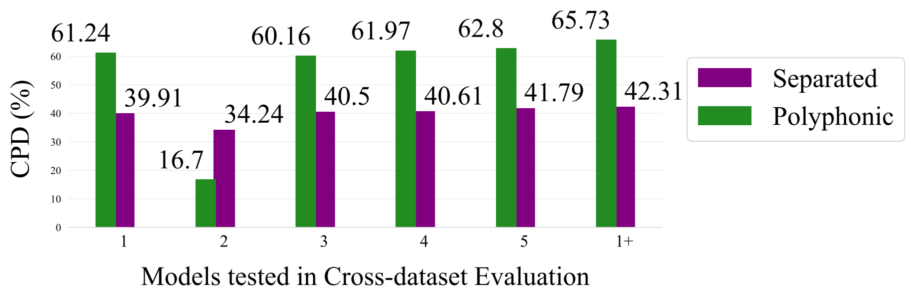


Figure 4.14: Comparison of performance drop on separated vocals and polyphonic mix.

According to the results shown above, the *WER* scores of monophonic models (1,3,4,5,+1) are much better on separated vocals, which is in line with previous research (Demirel et al., 2021a; Stoller et al., 2019). On the other hand, the cross-domain model (2) performs worse on separated vocals compared to the original polyphonic mix, yet still better than the monophonic models. This shows that separating vocals does not necessarily lead to improved recognition rates if a cross-domain model is employed. In addition, the performance drop (*CPD*) on separated vocals with respect to monophonic recordings is the lowest for the cross-domain model, which can be considered as evidence for the better domain-invariant performance of cross-domain models compared to domain-specific ones. Moreover, the multistream architecture (Model 3) has the lowest *CPD* compared to other monophonic models.

Top-down Integration

Now that each proposed method is compared above individually, we can proceed with integrating them into a single model. In this section we apply one method at a time progressively when building each of the analysis models. Table 4.22 gives the summary of the models tested at this stage. Similar to the top-down direction of the *HCLG* graph composition, we begin by integrating the improved language model. The first model (line 2) is Model 5 in the above analysis which has the same overall structure as the baseline model (line 1), but the language model is trained on the combination of speech and lyrics corpora. On top of Model 5, the alignments prior to DNN training are generated with the singing adapted lexicon in Model 6, and this is also used as the lexicon transducer. In Model 7, the acoustic model has the proposed multistream TDNN architecture. Model 7 is retrained on the combination of *DAMP* and *DALI* training sets to build the cross-domain model resulting in Model 8. To scale up performance, we apply data augmentation through speed perturbation in Model 8+ which has been shown to improve performance consistently Ko et al. (2015). Speed perturbation is applied with the factors of 0.9 and 1. Finally, we report results in Model 8++ obtained via RNNLM rescoring. In addition, the state-of-the-art DNN-HMM model published in our previous work (Demirel et al., 2021b) is included in Table 4.22, labeled as 9+. Model 9+ differs from Model 8+ only in terms of the language model where the former uses only the lyrics corpus for building the LM. Rescoring is applied using the same RNNLM for Model 9++.

According to Table 4.23, the improvements during the top-down integration of the proposed methods in this chapter are not extremely consistent across $\mathcal{D}^{test}_{others}$ and some of the improvements are dataset specific. Because of this, the implications of these results are multidimensional, and hence require to be scrutinised in detail. Below are itemised the key implications that can be drawn upon with respect to the *WER* and *CPD* scores in Table 4.23:

Model	Domain	Network	Pron.Model	Lang.
1) (*Baseline*)	Monophonic	Single-stream	CMU	Lyrics 4-gram
5)	Monophonic	Single-stream	CMU	**Mixed LM** 4-gram
6)	Monophonic	Single-stream	**Singing-adapted**	**Mixed LM** 4-gram
7)	Monophonic	**Multistream**	**Singing-adapted**	**Mixed LM** 4-gram
8)	**Cross-domain**	**Multistream**	**Singing-adapted**	**Mixed LM** 4-gram
8+)	**Cross-domain**	**Multistream**	**Singing-adapted**	**Mixed LM + RNNLM**
8++)	**Cross-domain**	**Multistream**	**Singing-adapted**	**Mixed LM + RNNLM**
9+)	**Cross-domain**	**Multistream**	**Singing-adapted**	**Lyrics LM** 4-gram
9++)	**Cross-domain**	**Multistream**	**Singing-adapted**	**Lyrics LM + RNNLM**

Table 4.22: Summary of top-down integration models.

Model	Dev(*self*)	*DAMP*^{test}	NUS	*Hansen*^{mono}	*CPD*$_{mono}$	*Hansen*^{poly}	*Mauch*	*Jamendo*	*DALI*^{test240}	*CPD*$_{poly}$	*CPD*
1)	13.76	17.73	8.04	18.96	-6.47	85.66	81.97	73.47	82.17	60.59	51.75
5)	12.99	17.33	7.47	19.45	-6.73	84.52	78.69	69.53	80.44	59.52	51.28
6)	12.17	17.64	7.48	19.35	-7.34	84.55	79.32	67.51	79.01	59.54	52.05
7)	12.39	17.03	**7.14**	18.55	-6.77	81.32	77.04	66.94	78.35	57.97	49.99
8)	27.68	17.29	7.57	**18.16**	-6.93	**35.49**	45.32	39.22	48.48	27.76	24.06
8+)	24.03	16.13	7.82	23.97	**-4.86**	37.61	41.47	**32.07**	43.36	27.61	**23.46**
8++)	24.93	13.40	8.13	23.45	-1.46	37.13	39.76	**31.21**	43.49	29.21	25.16
9+)	24.27	**15.89**	7.49	23.00	-4.95	36.78	**39.00**	34.94	**42.98**	**26.68**	23.74
9++)	25.54	**12.56**	7.66	22.93	-1.38	37.82	**38.81**	35.05	43.07	29.61	26.58

Table 4.23: Top-down integration results

- Overall, the cross domain training (models >=8) is the most effective method in terms of *CPD* which indicates a better performance generalisability.
- The multistream TDNN architecture is also consistently effective for performance generalisability, however the improvements are more modest compared to cross-domain training.
- One-by-one integration of the speech + lyrics language model and pronunciation models did not lead to consistent improvements. In addition, although the speech + lyrics language model seems to improve the results compared to the baseline model (1), this was not the case when models were trained on augmented data (Models 8+ and 9+). This indicates that improving one of the computational blocks in DNN-HMM based models does not necessarily lead to global optimal performance. This phenomenon has been mentioned in the literature as a cause of different computational blocks (i.e. language, pronunciation and acoustic models) using different objectives (Xiao et al., 2018; Watanabe et al., 2017; Li, 2021).
- Similar to the previous stage of cross-dataset evaluation (Table 4.20), the RNNLM led

to considerable improvements on $DAMP^{test}$, however it was not as beneficial on $\mathcal{D}^{test}_{others}$. This indicates a bias towards *DAMP* data and hence the overall *CPD* is higher.

- Speed perturbation generally improves the results. Interestingly, this caused a sharp drop only in *Hansen* datasets which is counter-intuitive. However, *Hansen* is a small dataset, thus the performance drop on this dataset did not affect the overall *CPD* considerably.

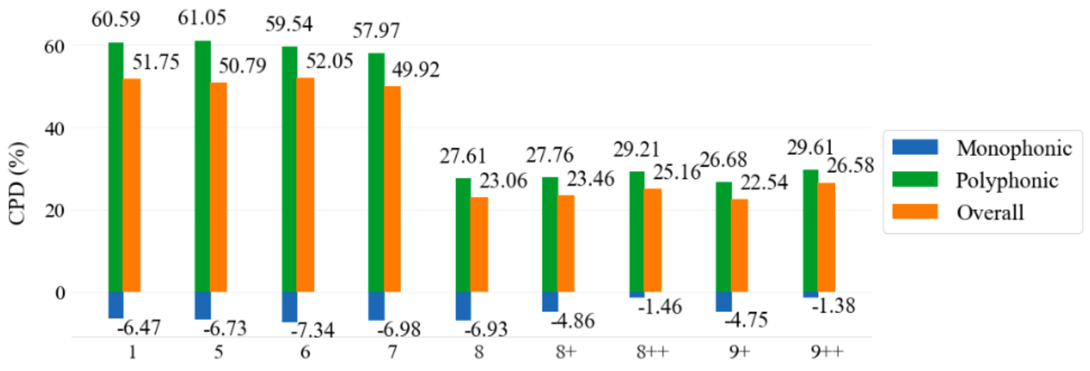


Figure 4.15: Top-down integration cross-dataset performance drop results.

The results in Figure 4.15 and Table 4.23 consider $DAMP^{test}$ as the $\mathcal{D}^{test}_{self}$ when computing the cross-dataset performance drop metric, *CPD*. This was done so to make a fair comparison across different proposed methods. However for the cross-domain models in particular, $DALI^{test240}$ is needed to be included in $\mathcal{D}^{test}_{self}$ and excluded from $\mathcal{D}^{test}_{others}$, in order to observe models' operational performance more accurately. Model 7 in Figure 4.16 is the MStreNet model trained on only monophonic recordings, hence it considers $\mathcal{D}^{test}_{self} = DAMP^{test}$. The columns in the middle section are for the cross-domain multistream TDNN model (8) where *CPD*'s are computed similarly with the monophonic model. On the other hand, the rightmost set of columns also represent scores for the cross-domain model but the overall *CPD* computation includes $DALI^{test240}$ in $\mathcal{D}^{test}_{self}$. For domain-specific *CPD*'s, $DAMP^{test}$ is still used to compute CPD_{mono} and $DALI^{test240}$ for CPD_{poly} .

- If *CPD* for the cross-domain Model (8) is calculated considering $\mathcal{D}^{test}_{self} = DAMP^{test}$ (the bars in the middle in Figure 4.16), its overall value is positive, implying that the model has dataset bias towards its training or in-domain data. However, Model 8 is trained on both monophonic and polyphonic recordings, hence its in-domain data should not be

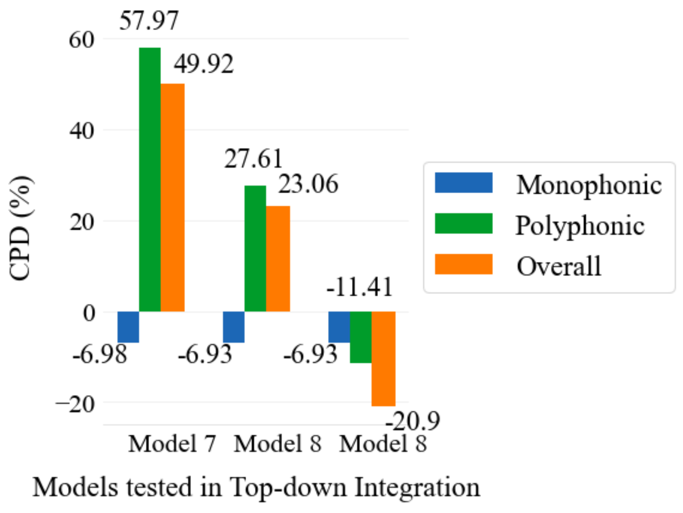


Figure 4.16: Cross-dataset performance drop. The first two bars on the left side use $\mathcal{D}_{self}^{test}$ = $DAMP^{test}$. The *CPD* scores on the right consider $DALI^{test240} \in \mathcal{D}_{self}^{test}$.

restricted to monophonic recordings. This may cause the risk of misinterpretation of *CPD* observations.

- When the *CPD* calculation for Model 8 considers the combination of polyphonic and monophonic recordings as $\mathcal{D}_{self}^{test}$, in other words in-domain data, its overall value is negative. This indicates that cross-domain training is helpful to mitigate the dataset bias problem, however, there is still a room for improvement to achieve overall domain invariant performance, and hints the risk of underfitting.

- CPD_{poly} when $\mathcal{D}_{self}^{test}$ = $DAMP^{test}$ is positive indicating the risk of overfitting. However, the performance drop on polyphonic recordings is negative when $\mathcal{D}_{self}^{test} = DALI^{test240}$, hence the model does not seem to overfit to its training data.

- Domain-specific *CPD*s are both negative. In this case, $|CPD_{mono}| < |CPD_{poly}|$, which means lyrics transcription performance on monophonic recordings is better than on polyphonic recordings.

4.5 Summary

For the task of automatic lyrics transcription, the DNN-HMM based speech recognition approach is a robust but complex framework. This study proposes adapting this LVCSR framework to singing data in a number of ways. Following a top-down approach, the first approach is concerning the language model. Experiments show that using a large lyrics corpus is considerably beneficial compared to training the n-grams on speech transcriptions. We further explore the effect of including the available spoken utterance transcriptions in the language model training corpus on lyrics transcription performance, where a minor gain is observed despite the increased data size. The reason for this might be associated with the prosodic and structural differences of spontaneous speech transcriptions and song lyrics. The pronunciation dictionary used to decompose words into phonemes is adapted to singing data based on the knowledge obtained from a data-driven phoneme confusion analysis. This analysis highlighted prolonged vowels and omitted plosives during singing. The acoustic model employs a novel compact multistream TDNN architecture which is shown to be effective in terms of *WER*, robustness against reverberation and faster inference. To develop a scalable lyrics transcription solution, cross-domain training is applied where a single lyrics transcriber can operate well undisturbed by the domain differences such as the presence of musical accompaniment. Experiments regarding this can also be considered as an evidence for including monophonic recordings in training the acoustic model is useful for ALT from polyphonic recordings.

The above mentioned proposed methods are tested on a cross-dataset evaluation framework which is based on several benchmark test sets used in research. In order to observe performance generalisability across datasets with different properties, we make use of the cross-dataset performance drop (*CPD*) metric introduced in Chapter 3. Among all methods, the cross-dataset training (including music informed silence modeling) and the multistream acoustic model architecture are the most effective ones. Overall, all proposed methods lead to improvements when applied independently. However, combining these methods to build the final transcriber did not always improve results overall, and the word error rates across $\mathcal{D}^{test}_{others}$ do not follow a consistent trend depending on the method tested. One major implication that can be drawn upon this is that improving an independent computational block of DNN-HMM based systems does not necessarily lead to a global performance improvement. This makes DNN-HMM models hard to tune for a downstream word recognition task. In addition to the tuning difficulty, the overall system includes several data preprocessing steps which can be cumbersome during experimental iterations.

Although having robust and scalable performance, DNN-HMM based systems have been becoming less attractive for the research community compared to end-to-end models. Some of the major factors for this trend are the tuning difficulty, training complexity and dependency on human expertise for building the models (and specifically the pronunciation dictionary). The cross-dataset evaluation study in this chapter verifies these previously mentioned drawbacks of DNN-HMM based systems within the context of automatic lyrics transcription. To mitigate these, we develop an end-to-end lyrics transcriber in Chapter 5 and provide a comparison with the DNN-HMM based approach in Chapter 6. Furthermore, a comparison with other published methods is given in Chapter 6.

Chapter 5

End-to-end Lyrics Transcription

This chapter takes an alternative approach to the DNN-HMM method for building a lyrics transcriber, which is an end-to-end training method. According to the previous research on ALT, end-to-end models have not been able to surpass the DNN-HMM approach until very recently (Gupta et al., 2020; Stoller et al., 2019). Zhang et al. (2021) reported better results on the monophonic $DAMP^{test}$ set using a Transformer architecture which highlighted the potential of end-to-end lyrics transcription given the data resources available for research. However, similar success has not been achieved on polyphonic recordings yet. In this chapter, we follow a similar approach with Zhang et al. (2021) to explore the potential of the end-to-end Transformer architecture in multiple domains (i.e. monophonic, polyphonic and source separated) . As a novelty, we apply cross-domain training and speech-to-singing transfer learning. The models' performances are tested following the cross-dataset evaluation scheme studied in the previous chapter. Furthermore, this chapter provides the first attempt at building a lyrics transcription system using the SpeechBrain toolkit.

The end-to-end learning scheme offers a simplified paradigm that can directly map given input representations to output labels using a single classification or regression model. With the emergence of promising models in computer vision (Krizhevsky et al., 2012) and natural language processing (Bengio, 2009; Devlin et al., 2018), the popularity of end-to-end models has grown significantly in the last decade. This framework has also been applied successfully to ASR, where a purely neural network based system translates input audio signals directly into orthographic transcriptions (Graves and Jaitly, 2014). Although the hybrid DNN-HMM models had been the most widely used framework for large-scale speech processing applications both in academia and industry (Hinton et al., 2012), most recently, end-to-end models have begun to surpass the performance of the traditional HMM-based models in a number of speech processing tasks (Watanabe et al., 2017, 2018; Sainath et al., 2020; Graves and Jaitly, 2014; Li et al., 2020; Han et al., 2020; Baevski et al., 2020), which marks a breakthrough in research.

Research in end-to-end ASR has mainly evolved to overcome the major drawbacks of the traditional DNN-HMM models that limit their modeling capability and/or performance scalability. Watanabe et al. (2017) outlines these problems as follows:

- *Linguistic information:* The mapping between the acoustic and language models is achieved with a pronunciation model, which is typically based on a handcrafted dictionary. The construction of the dictionary requires linguistic expertise and thus is subject to human errors. In addition, the word-phoneme mapping has language-dependent structures.

- *Complex inference:* The final decoding path uses a composition of probabilities computed from separate computational blocks. This composition is often a quite complex procedure. Although modern systems use WFST's for this purpose, their optimization for a downstream task is still complicated (Mohri et al., 2002; Allauzen et al., 2007; Stolcke, 2002).

- *Complex preprocessing steps:* A DNN-HMM based speech recogniser cannot be built from scratch, and the initialization of DNN training requires several complex preprocessing steps. These include training a GMM-HMM acoustic model, building context-dependency decision trees and generating alignments and lattices. Errors in any of these procedures may cause a failure of the overall pipeline or end up building a suboptimal model.

- *Incoherence in optimization:* The above listed issues introduce several variables separate from the core neural network training, that may affect the overall performance of the recogniser. Moreover, each computational block is constructed independently with different objectives. Therefore, having an optimal model for each of these blocks does not necessarily lead to a global optimum. This observation was verified in Section 4.4.6

where combining improved pronunciation and language models did not consistently lead to a better performance.

- *Conditional independence assumption:* HMMs and n-gram LM approximation models are based on Markov chains and hence rely on the conditional independence assumption (Gales and Young, 2008). However, this assumption does not hold for real-world speech data.

The end-to-end lyrics transcription system studied in this chapter is constructed using the SpeechBrain framework (Ravanelli et al., 2021), which is introduced to address and provide solutions to the above listed drawbacks of DNN-HMM models. In particular, the system architecture is based on the transformer encoder-decoder structure trained on the hybrid CTC / Attention objective[1]. The advantages of this framework over the Kaldi-based DNN-HMM system studied in Chapter 4 are as follows: first of all, the classifier uses the *SentencePiece* (unigram) tokens (Kudo, 2018) as the target subword classes which were introduced in Section 2.3.6. These tokens can be extracted automatically from a text corpus without the need of expert linguistic information. The inference pipeline directly applies beam search on the neural network outputs, which provides a single block between inputs and output probabilities. The attention mechanism can learn alignments between input and output sequences during training, therefore generating alignments prior to training is not necessary. The attention mechanism also does not rely on the conditional independence assumption. In addition to these advantages, SpeechBrain can process and expand datasets dynamically. Because of this capability, the feature extraction and data augmentation steps are applied on the fly during training (Ravanelli et al., 2021) which does not require storing the augmented data. Unlike SpeechBrain, Kaldi requires to store raw and augmented features separately which can be costly in terms of time, storage and memory resources.

The following sections explain the details of the end-to-end lyrics transcription system built in this study. First, the implementation details of the hybrid CTC / Attention training principles are given. Then, a few methods to leverage low resource singing data are presented. These methods are tested in Section 5.4 where the training and inference details are also provided.

[1]The network and training configuration of the system can be found here: `https://github.com/speechbrain/speechbrain/blob/develop/recipes/LibriSpeech/ASR/transformer/hparams/transformer.yaml`.

In this section, a similar cross-dataset evaluation is applied to compare the generalisability of end-to-end models and their performance on source-separated vocals.

5.1 Hybrid CTC/Attention Training

The Hybrid CTC/Attention learning objective was first introduced in the work of (Watanabe et al., 2017). In this section, the formulation and implementation details of this training approach is provided.

5.1.1 Multiobjective Learning

The neural network training is based on the *multiobjective learning* approach (Kim et al., 2017) using the combination of CTC and attention objectives explained in Section 2.3. The performance of the attention based models drops considerably in noisy environments (Kim et al., 2017) as the network may assign high attention weights to noisy audio frames that do not contribute to the overall context of the utterance in question. In order to suppress the undesired attention to noisy instances, the network can be conditioned with the timing information of labels, which can be achieved through alignment. Because there is no conditional independence assumption in the attention mechanism, the objective in Equation 2.36 does not enforce predictions to be aligned with the input sequence and it can attend anywhere in the input sequence when predicting the next label. Due to this, the attention mechanism is inherently prone to make deletion and insertion mistakes (Watanabe et al., 2017). As CTC based training already enforces a monotonic alignment between the input audio and output labels, Kim et al. (2017) proposed jointly training the networks using both CTC and attention objectives. According to this architecture, the CTC layer takes the output of the encoder $\mathbf{h}^{enc}$ when computing the objective $\mathcal{L}_{\text{CTC}}$. The final objective to maximise is computed through the logarithmic linear combination of the log-scaled CTC and attention objectives:

$$\mathcal{L}_{\textbf{multiobj}} = \lambda \log p_{\textbf{att}}(\mathbf{Y}|\mathbf{X}) + (1-\lambda) \log p_{\textbf{ctc}}(\mathbf{Y}|\mathbf{X}) \tag{5.1}$$

where λ is a tunable hyperparameter with $0 \leq \lambda \leq 1$. Note that $\mathbf{Y}$ corresponds to either character or subword tokens to predict.

5.1.2 Decoding

Inference in the hybrid CTC / Attention models uses beam search for decoding as explained in Section 2.1, similar to the traditional DNN-HMM and other conventional end-to-end models. The multiobjective decoding objective at inference can be defined as:

$$\widehat{\mathbf{Y}} = \underset{\mathbf{Y}}{\operatorname{argmax}}(\lambda \log p_{\mathbf{att}}(\mathbf{Y}|\mathbf{X}) + (1-\lambda) \log p_{\mathbf{ctc}}(\mathbf{Y}|\mathbf{X})) \tag{5.2}$$

According to Equation 2.2, the beam search procedure computes a score for each partial hypothesis. From this perspective, combining attention and CTC scores is not straightforward, as the former performs decoding synchronously with the output labels while the latter's scores are frame-synchronous. To cope with this, Watanabe et al. (2017) proposed utilizing the *CTC prefix probability* which is defined in Kawakami (2008) as the cumulative probability of all sequences which have the partial hypothesis h as their prefix (recall the notation in Section 2.1:

$$p_{\mathbf{ctc}}(h, ...|\mathbf{X}) = \sum_{v \in (\mathcal{U} \bigcup \{<eos>\})} p_{\mathbf{ctc}}(h \cdot v|\mathbf{X}), \tag{5.3}$$

where v is the set of all possible sequences excluding the blank symbol <b>. Then the multiobjective beam search uses the log CTC-prefix objective:

$$\alpha_{\mathbf{ctc}}(h, \mathbf{X}) \triangleq \log p_{\mathbf{ctc}}(h, ...|\mathbf{X}). \tag{5.4}$$

The attention scores also use the logarithmic objective:

$$\alpha_{\mathbf{att}}(h, \mathbf{X}) \triangleq \log p_{\mathbf{att}}(h|\mathbf{X}). \tag{5.5}$$

Then the multiobjective scores during beam search can be expressed as:

$$\widehat{\mathbf{Y}} = \underset{h \in \widehat{\Omega}}{\operatorname{argmax}}(\lambda \alpha_{\mathbf{att}}(h, \mathbf{X}) + (1-\lambda)\alpha_{\mathbf{ctc}}(h, \mathbf{X})) \tag{5.6}$$

For a more detailed explanation of the algorithmic implementation of the multiobjective beam search decoding, readers are encouraged to study the work of Watanabe et al. (2017).

5.1.3 Language Model Integration

Previous research showed that shallow fusion outperforms both deep and cold fusion in addition to being the most inexpensive language model integration approach (Toshniwal et al., 2018). For this reason, the shallow fusion language model integration is applied in this end-to-end lyrics transcription model. Recall the inference equation for end-to-end ASR updated with the shallow language model fusion in Equation 2.40. This is integrated with the hybrid CTC / attention model as follows:

$$\widehat{\mathbf{Y}} = \underset{h \in \widehat{\Omega}}{\operatorname{argmax}}(\lambda p_{\mathbf{att}}(h, \mathbf{X}) + (1-\lambda)p_{\mathbf{ctc}}(h, \mathbf{X}) + \gamma \log p_{LM}(\mathbf{Y})) \tag{5.7}$$

5.1.4 Overall System Structure

The overall structure of the end-to-end lyrics transcription system employed in this thesis is illustrated in Figure 5.1. To summarise its components: the encoder consists of a stack of transformer layers. The decoder is also a transformer network which takes the encoded hidden representation or the context vector at its input and generates the attention loss $p_{\mathtt{att}}$. To enforce the timing information, the context vector is also used to generate the CTC loss, $p_{\mathtt{ctc}}$. The output predictions are generated through the joint decoding procedure explained in Section 5.1.2. At inference, the token priors coming from a pretrained TransformerLM are added with shallow fusion when computing the final output likelihoods.

5.2 Speech-to-Singing Transfer

Up until very recently, the performance of traditional DNN-HMM based ASR approaches surpassed the end-to-end training systems. Some of the commercial applications of end-to-end models, such as Google's LAS (Chan et al., 2016) or Mozilla's *Deep Speech* (Hannun

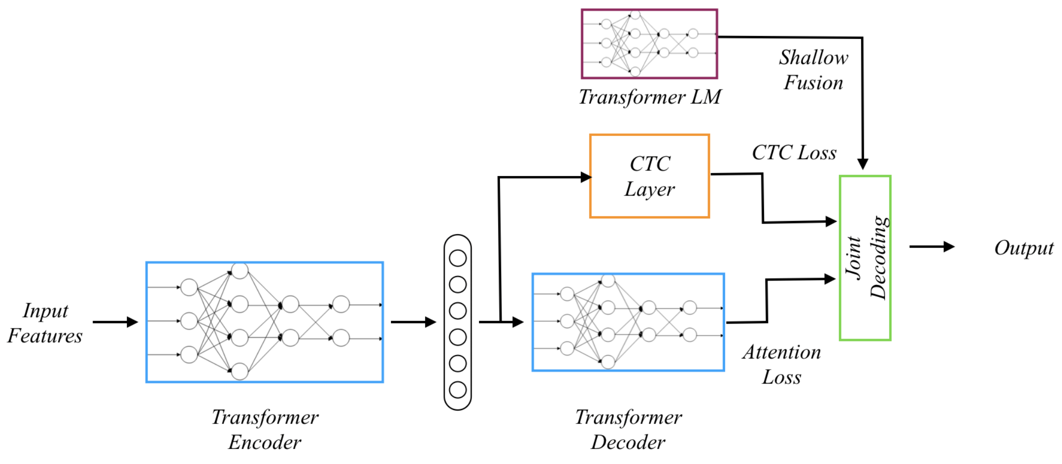


Figure 5.1: The overall hybrid CTC/Attention training procedure with shallow language model fusion at the joint decoding stage.

et al., 2014) showed that the performance gap between DNN-HMM and end-to-end models dwindle as the training data gets larger. This large training data requirement has been commonly acknowledged due to the end-to-end learning being expected to learn high level information abstractions purely from data (Bengio, 2009). However, large training sets are not always available to be able to train well-performing end-to-end models.

Research has shown that the parameters of a pretrained end-to-end model can be transferable to different data domains (Taylor and Stone, 2009) or tasks (Bengio, 2012). Methods that aim to construct end-to-end models through leveraging information gained from a relevant task are broadly referred to as *transfer learning* (Pan and Yang, 2009). In speech recognition, transfer learning is generally applied in low resource scenarios such as multilingual applications (Swietojanski et al., 2012; Ghoshal et al., 2013; Schuster et al., 2018; Cho et al., 2018), and ASR from children's (Shivakumar and Georgiou, 2020) or dysarthric speech (Xiong et al., 2020). There are numerous ways proposed to apply transfer learning for speech processing tasks such as model adaptation (Pan et al., 2010; Paredes et al., 2012), multilingual and cross-lingual adaptation (Swietojanski et al., 2012; Ghoshal et al., 2013), and deep representation learning (Bengio, 2012). This study exploits the available pretrained open-source ASR models and applies transfer learning specifically through *model transfer* from speech to singing data.

Model transfer can be applied when the output labels of the source and target domains are the same. As shown in Figure 5.2, a model is initially trained on data from the source domain

(1). Then, a model for the target domain is retrained **(3)** where the parameters of the network are initialised with the parameters of the pretrained source model **(2)**.

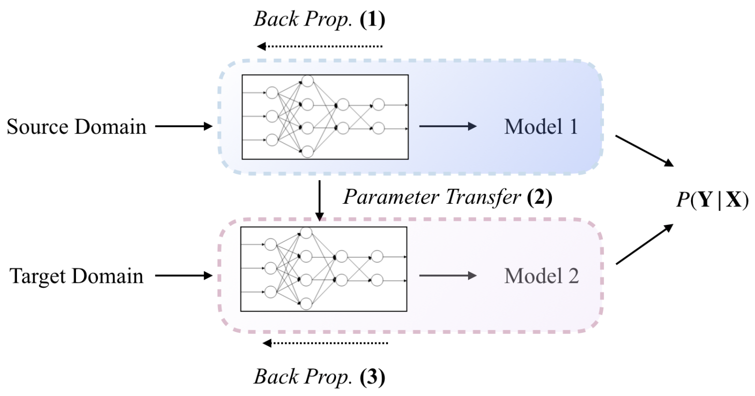


Figure 5.2: Model fine-tuning on a target domain.

The language model integration method employed in this end-to-end ASR framework, i.e. shallow fusion, is a *late training integration* approach where the posteriors to compute the final objective in Equation 5.7 are obtained from separately trained language and acoustic models. Thus, a complete adaptation of the end-to-end ASR model should include adapting the language model to lyrics as well. In this framework, this is achieved via model transfer from Librispeech to $DAMP^{train}$ transcriptions. It was previously shown in Section 4.4.2 that using larger corpora for training the neural lyrics language models did not necessarily improve results and using only the $DAMP^{test}$ data was the most beneficial setup. Similarly, this was observed to apply also for the end-to-end training framework employed in this study. A larger corpus for transferring the language model is not used due to this reason.

One common application of transfer learning is transferring knowledge between different domains, i.e. *cross-domain transfer learning* (Wang and Zheng, 2015). In the context of ASR, cross-domain transfer learning has two distinct directions; the first one is concerned with transferring features from different modalities of the same data, such as leveraging video or image features that are associated with an audio recordings used in training, in addition to the standard audio features. The other major direction in transfer learning is achieved through transferring knowledge in-between different applications of the same task. For example the tasks of lyrics transcription from monophonic and polyphonic recordings can be considered as different applications of the same task considering their distinct acoustic properties. This type of transfer learning is referred to as *domain transfer*. This study focuses on the latter transfer learning approach due to different data modalities not being current available in the training data. Moreover, domain transfer is more relevant to the applications of ALT.

The methods proposed in this section applies domain transfer in three ways. The basic approach is illustrated in Figure 5.3b which is simply fine-tuning a pretrained speech recogniser either on monophonic or polyphonic recordings. Figure 5.3c shows a recursive transfer learning approach, where fine-tuning is applied on different domains in cascade. The domain transfer in Figure 5.3d is similar to the one in Figure 5.3b, but the model transfer is applied directly in the combination of monophonic and polyphonic recordings, a method that is also similar to the cross-domain training approach explained in Section 4.3.1. These different ways to apply domain transfer in ALT are evaluated in Section 5.4.1.

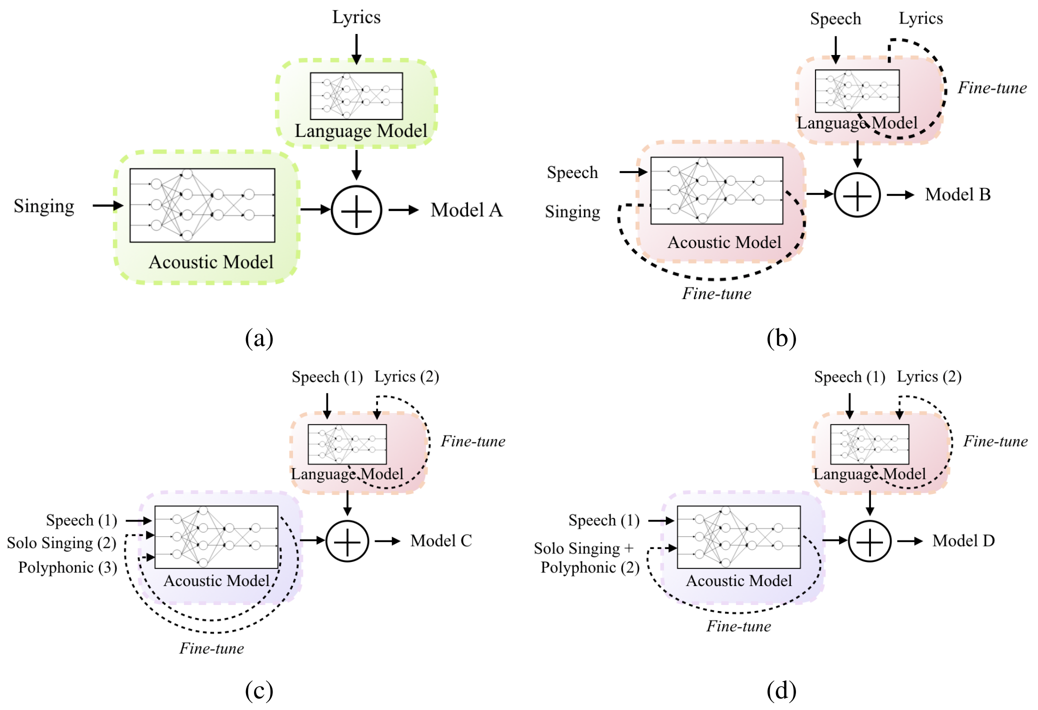


Figure 5.3: Speech-to-singing transfer learning. (a) Training on singing data from scratch. (b) Fine-tuning a pretrained speech model on singing data. (c) Cascaded transfer learning to train a polyphonic acoustic model. (d) Cross-domain fine-tuning for end-to-end ASR.

5.3 Experimental Details

5.3.1 Training

The overall system architecture follows the standard Transformer recipe within the SpeechBrain toolkit, and is summarised in Figure 5.4. There are two main training blocks of the overall system: the acoustic and language models. The network topology of the acoustic model follows the encoder-decoder framework (i.e. sequence-to-sequence models explained in Section 2.3.2), where both encoder and decoder consist of transformer layers. The acoustic model takes 80-band mel-spectrogram features at the input which are extracted with a window size of 25 ms, a hop size of 10 ms. The input audio recordings are sampled at the rate of 16kHz. The mel-spectrograms are fed to a 3-layer 2D CNN block with 3×3 filters at the front-end of the network. The convolutional front-end is used to extract input positional embeddings, inspired by ContextNet (Han et al., 2020). The encoder and decoder have 12 and 6 transformer layers respectively where each transformer layer has a multi-head self-attention layer with the hidden dimension of 768 and 8 heads, followed by a feed-forward layer with the dimension of 3072. No dropout is applied between transformer layers, which use the Gaussian Error Linear-activation Unit (GELU) (Hendrycks and Gimpel, 2016) as the activation function. The training objective uses the CTC weight $\lambda = 0.3$, which is determined empirically on $DAMP^{dev}$. At inference λ is also empirically set to 0.4 as the final likelihood computation uses probabilities coming from the language model (recall Equation 5.7).

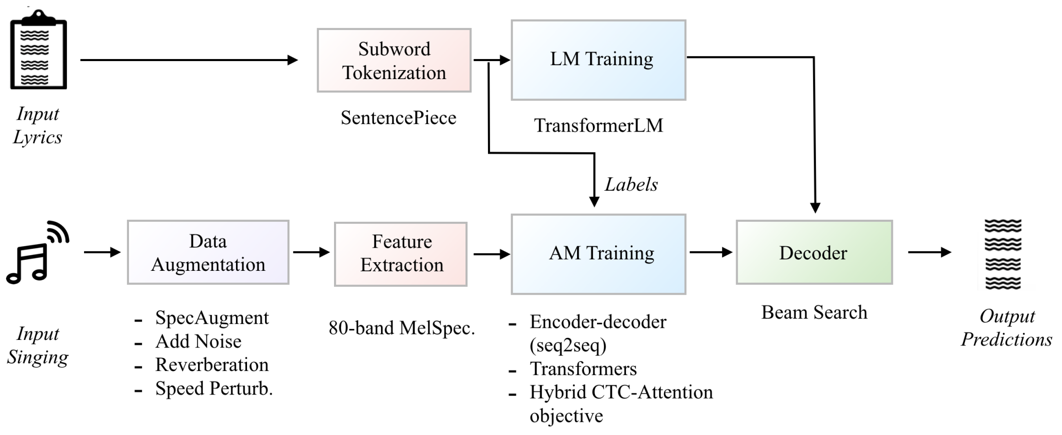


Figure 5.4: The overall training pipeline.

The language model is a TransformerLM (Vaswani et al., 2017), which has a 12-layer encoder with 12 self-attention heads and hidden dimensions of 768. The transformer feed-forward layer has the size of 3072 similar to the acoustic model. Words are tokenised using the SentencePiece algorithm (Kudo, 2018) with the unigram vocabulary size of 5000. Both the acoustic and the language models are trained on the same set of unigram tokens as the target class set, which is the pretrained Librispeech tokeniser obtained from the HuggingFace repository.[2] Therefore, both neural networks have the output size of 5000. The language model weight used to compute the overall objective, γ is empirically set to 0.1.

Training samples that are shorter than 1.5 seconds and longer than 15 seconds are discarded. Moreover, malformed audio files are also excluded. [3] These procedures are similar that of Kaldi's, hence similar portions of the training data are actually used in training both the E2E model in this chapter and the DNN-HMM model in Chapter 4. After this initial filtering, the total utterance durations of the training data for *DAMP* and *DALI* datasets are around 112 and 150 hours respectively.[4] All recordings in the train set are sampled at 16 kHz. The dataset is augmented with the *SpecAugment* method (Park et al., 2019) with a time warp window size of 5, frequency mask width of 30 frames and a time mask of 40 frames. Speed perturbation is also applied with a time-warping factors of 0.9 and 1.1 similar to the DNN-HMM setup in Chapter 4.

The training is performed on Nvidia ® - Quadro RTX 6000 ™ machines with 22 GB memory capacity. End-to-end models, and transformers in particular, are large neural network models and have very large numbers of trainable parameters, and thus large memory requirements. One way to avoid exceeding memory capacity is to use smaller minibatches. However, large models require larger minibatches for them to be able to converge. To circumvent these issues, large-batch training is simulated via gradient accumulation (Ravanelli et al., 2021). According to this approach, the gradients are accumulated for a predefined number of batches (N), and they are updated after the N^{th} minibatch. For each minibatch, zero-padding is applied to match the sizes of the samples within the batch. The amount of padding for each sample in a minibatch is another issue during batch processing. In order to minimise the effect of the features computed from zero-padded samples, and to save computational resources due to these, the training dataset is sorted by duration and fed into the network in an ascending order with minibatches of 2. The

[2]The model can be retrieved from *https://huggingface.co/speechbrain/asr-transformer-transformerlm-librispeech*.

[3]Most malformed files were in the *DALI* dataset, possibly due to them being automatically retrieved from online resources.

[4]This means around 6 hours of *DALI* data is discarded after this procedure.

gradients are accumulated for 64 minibatches to simulate training on 128 minibatches.

For all iterations, the acoustic models are trained for 10 epochs using the Adam optimiser with an initial learning rate of 1, then trained for an additional 5 more epochs using the SGD optimiser with a learning rate of 25×10^{-6}. This hybrid optimiser training has been shown to be useful within the end-to-end ASR training framework (Ravanelli et al., 2021).

5.3.2 Inference

Unlike the DNN-HMM setup, end-to-end transfomer models require fixed length audio samples at the inference step. If an utterance in question is shorter than the tensors used in training, padding is applied. However, longer utterances are not compatible, hence they need to be split into smaller chunks. The audio recordings are split using the timestamp annotations if they are available. Otherwise, automatic segmentation is performed based on heuristics [5]. One straight-forward segmentation approach is through splitting audio linearly with overlapping chunks (Stoller et al., 2019). However, this method is prone to cut the context within utterances in the middle, leading to an inherent reduction in performance during inference. In order to perform inference on cleaner audio chunks, the long utterances are segmented based on a neural *voice-activity detection* (VAD) model. The inference pipeline in this study uses the pretrained CRDNN model retrieved from the HuggingFace repository[6], which is essentially a binary speech / non-speech classifier. The classifier outputs the posterior probability of a frame being a speech instance or not, and a threshold is set to make a decision. The pretrained model is trained on the LibriParty dataset (Ravanelli et al., 2021) which does not generally have music instruments within the examples. For this reason, the VAD segmentation is performed on the source separated vocal tracks. Once the raw VAD frames are retrieved, they are smoothed to obtain a minimum of 5 and a maximum of 15 seconds. Finally, these VAD segments are manually corrected and verified to make sure they are sensible audio chunks for lyrics transcription evaluation. The overall inference pipeline is illustrated in Figure 5.5.

[5] *Jamendo*, *Hansen* and *Mauch* sets do not have such annotations, therefore the automatic segmentation is applied on these. The rest test sets use the available timestamps to segment the full-length audio.

[6] The model can be retrieved via the online tutorial at *https://huggingface.co/speechbrain/asr-transformer-transformerlm-librispeech*.

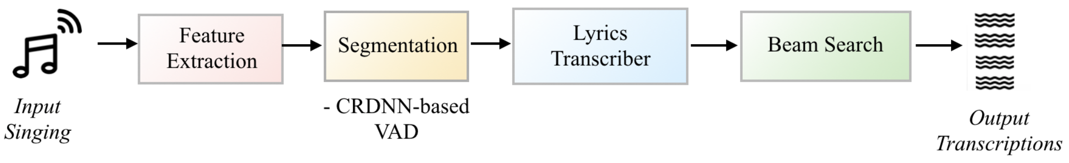


Figure 5.5: The inference pipeline.

5.4 Results

The experiments use the same training pipeline and network topology for all the tested models, and focus on constructing a baseline model with available data resources. Similar to Chapter 4, the experiments begin with monophonic data and continue with cross-dataset evaluation.

5.4.1 Speech-to-singing Transfer

Table 5.1 shows the results of the transfer learning experiments. The initial model uses the acoustic and language models both pretrained on Librispeech dataset (the version with around 960 hours of speech). Models 2 and 3 are trained on *DAMP* from scratch. The acoustic model for the rest of the models (4,5,6) uses the pretrained ASR in Model 1 which is fine-tuned (i.e. parameter transfer) on the *DAMP* dataset. As Models 2 and 3 do not benefit from any pretraining, they may require more epochs to converge. Therefore, these models are trained for an additional 5 epochs (20 epochs in total) compared to the rest of the transferred models (4,5,6). Models 1,2,4 use the language model trained on the Librispeech transcripts, and the language models in 3,5 are trained from scratch on the lyrics corpus introduced in Chapter 3. In model 6, the Librispeech language model is also transferred to singing data.

	Acoustic Model	Language Model	Dev	Test
1)	Librispeech	Librispeech	54.18	53.18
2)	*DAMP*	Librispeech	94.07	85.92
3)	*DAMP*	Lyrics Corpus	106.00	131.2
4)	Librispeech $\rightarrow$ *DAMP*	Librispeech	21.44	18.05
5)	Librispeech $\rightarrow$ *DAMP*	Lyrics Corpus	31.99	36.48
6)	Librispeech $\rightarrow$ *DAMP*	Librispeech $\rightarrow$ Lyrics Corpus	**15.00**	**16.69**

Table 5.1: Summary of speech-to-singing transfer learning results (*WER*) on the *DAMP* dataset. The symbol $\rightarrow$ stands for the model transfer operation.

As mentioned above, transformers require large training sets in order to have a converging training loss. This can be seen from the losses of the Models 2 and 3 in Figure 5.6, which have acoustic models trained from scratch on $DAMP^{train}$. Moreover, notice the large gaps between the train and validation losses for these models. While this usually signals the risk of overfitting, the *WER* scores for Models 2 and 3 (Table 5.1) are extremely high. Hence the models do not seem to converge to near the global minima. On the other hand, models initialised with a pretrained ASR model (4,5,6) have loss curves converging to a much lower loss. Even tuning only the acoustic model (4) led to a *WER* score of 18% on $DAMP^{test}$, which is comparable with the baseline DNN-HMM model (Model 1 in Table 4.19). The gap between DNN-HMM and end-to-end gets even lower when transfer learning is applied on the language model (6). Notice that training the language model solely on lyrics (Models 3,5) did not lead to optimal results.

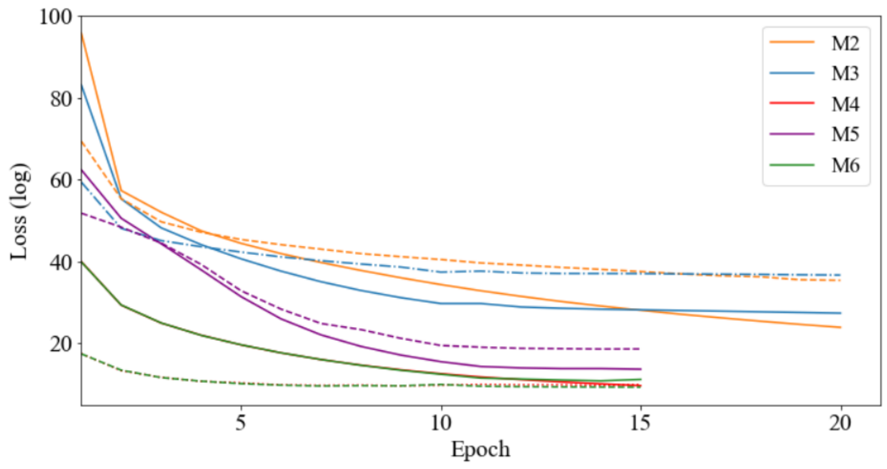


Figure 5.6: Train (full-lines) and validation (dashed) loss curves for the end-to-end models.

The above analysis provides evidence that end-to-end models require large training sets, and transfer learning is a useful method to achieve considerable performance boost in low resource scenarios.

5.4.2 Cross-Dataset Evaluation

Similar to the motivation in Section 4.3.1 on achieving a domain-invariant lyrics transcription performance, this stage of experiments is concerned with exploring ways for building an end-to-end cross-domain model. In order to provide quantitatively comparable results to these in Chapter 4, the same evaluation framework is utilised in Table 5.2, i.e. $DAMP^{test} \bigcup DALI^{test}$ are used as $\mathcal{D}_{self}^{test}$ for the cross-domain models.

	Acoustic Model	Train Set
1)	Monophonic	Librispeech $\rightarrow$ *DAMP*
2)	Polyphonic	Librispeech $\rightarrow$ *DALI*
3)	Cross-domain	Librispeech $\rightarrow$ *DAMP* $\rightarrow$ *DALI*
4)	Cross-domain	Librispeech $\rightarrow$ (*DAMP* $\bigcup$ *DALI*)

Table 5.2: Summary of the end-to-end models in the cross-dataset evaluation.

The baseline model (1) is the best performing one in the previous stage of experiments, which is fine-tuned on the monophonic $DAMP^{train}$. Model 2 is trained solely on the polyphonic $DALI^{train}$. Model 3 is constructed through the cascaded transfer learning illustrated in Figure 5.3c, where the acoustic model is fine-tuned on $DAMP^{train}$, then $DALI^{train}$ consecutively. Finally, model 4 has a similar approach to the cross-domain DNN-HMM based model, where the acoustic model of the pretrained ASR model is fine-tuned directly on $(DAMP \bigcup DALI)^{train}$. Following the observations in Section 4.4.6, $\mathcal{D}^{test}_{self}$ is set depending on the training set. Specifically, while the monophonic (Model 1) and polyphonic (Model 2) models use $DAMP^{test}$ and $DALI^{test}$ respectively, the cross-domain models (3,4) use the combination of both when computing the cross-dataset performance drop *CPD*.

Model	$DAMP^{test}$	$DALI^{test}$	NUS	$Hansen^{mono}$	CPD_{mono}	$Hansen^{poly}$	*Mauch*	*Jamendo*	CPD_{poly}	*CPD*
1)	16.69	81.16	**19.20**	**53.23**	9.48	80.23	77.50	91.75	64.82	57.65
2)	72.55	76.29	92.11	79.17	8.56	75.15	60.58	67.95	-9.41	0.12
3)	64.72	76.22	89.23	77.71	11.26	76.08	**59.68**	**67.40**	-9.25	0.14
4)	**16.04**	**67.21**	19.93	53.92	-36.05	**72.43**	65.16	79.12	5.73	-13.40

Table 5.3: Results of the cross-domain experiments on end-to-end framework. $\mathcal{D}^{test}_{self}$ used in evaluating each model is highlighted in blue.

The implications of the above cross-dataset evaluation are studied below:

- Model 1 performs well on $\mathcal{D}^{test}_{self}$ ($DAMP^{test}$), however a performance drop is observed on other monophonic sets ($CPD_{mono} > 0$), indicating the possibility that the model suffers from overfitting to $DAMP^{train}$. *CPD* gets much larger on polyphonic data, hence the monophonic end-to-end model does not have performance generalisation across varying domains.

- Although finetuning the pretrained speech recogniser on the polyphonic *DALI* dataset improves recognition rates on polyphonic data, the corresponding model's (2) performance on $\mathcal{D}^{test}_{self}$ is considerably lower than that of the monophonic model 1. For this model,

$CPD_{poly} < 0$, meaning that this model performs worse on $DALI^{test240}$ than the polyphonic $\mathcal{D}^{test}_{others}$, which might be due to the former having a larger size, and can be an indication for being a more challenging dataset compared to the latter. On the other hand, this is also understandable as $DALI^{test240}$ is much larger in size with a higher musical variability than $DAMP^{test}$. The overall performance drop CPD is near 0 (zero), implying a generalisable performance. However, this is mostly due to model 2 already having a very high WER on $\mathcal{D}^{test}_{self}$.

- Applying cascaded transfer learning (model 3) helps improve the performance on monophonic recordings compared to model 2, while the improvement on polyphonic data is not as apparent. Note that this model uses both $DAMP^{test}$ and $DALI^{test240}$ sets as $\mathcal{D}^{test}_{self}$ and the computation of CPD is done accordingly.

- Cross-domain training in Model 4 improved the performance on part of the polyphonic sets while maintaining the performance on monophonic recordings. However, a sharp drop is observed on the *Jamendo* and *Mauch* datasets. Note that the performance on polyphonic recordings is still much poorer compared to monophonic.

- The CPD scores of Models 2 and 3 are near the perfect performance balance (zero). However, this alone cannot be interpreted to signify a powerful model.

- Moreover, the interpretation of CPD scores also needs to consider the data which is included in D^{test}_{self} and $\mathcal{D}^{test}_{others}$ sets. In particular, Model 2 uses a polyphonic $\mathcal{D}^{test}_{self}$, but Model 3 uses $(DALI \bigcup DAMP)^{test}$. The overall CPD for Model 2 gives a performance generalisability measure with respect to the *in-domain* polyphonic data, whereas the CPD scores for Models 3 and 4 consider both monophonic and polyphonic recordings as in-domain data.

From the above perspective, the following observations can be inferred regarding the domain and dataset biases:

- Model 1 is biased towards the monophonic domain, hence its performance cannot be generalised to the polyphonic domain. There is also a dataset bias as $CPD_{mono} > 0$.

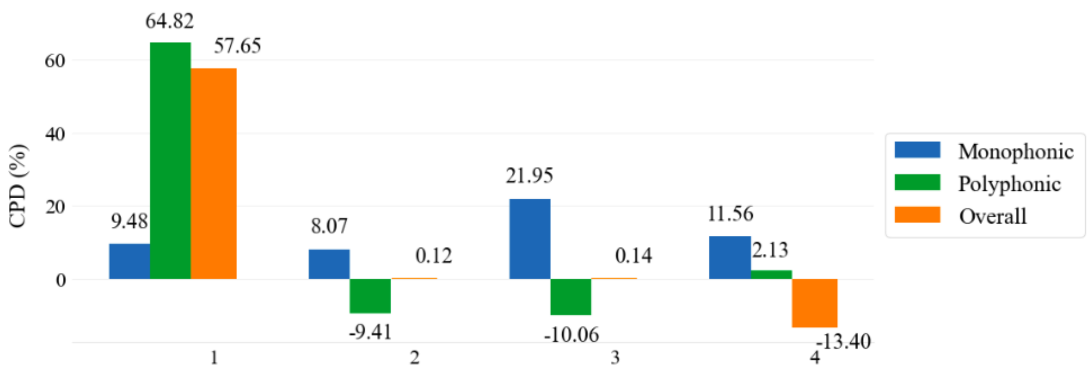


Figure 5.7: Cross-dataset performance drop. The first two bars on the left side use $\mathcal{D}_{self}^{test} = DAMP^{test}$. The *CPD* scores on the right consider $DALI^{test240} \in \mathcal{D}_{self}^{test}$.

- Model 2 is biased towards the polyphonic domain as $CPD_{mono} > 0$, $CPD_{poly} < 0$, and considering $\mathcal{D}_{self}^{test}$ is polyphonic.

- Further improvement is observed on monophonic recordings through cascaded transfer learning. However, this model (3) is a cross-domain model, and $DAMP^{test} \in \mathcal{D}_{self}^{test}$. Therefore, the monophonic $\mathcal{D}_{others}^{test}$ can also be considered as in-domain data, and the reason for $CPD_{mono} > 0$ is due to dataset bias.

- The overall *CPD* is near zero for Models 2 and 3, their performance on $DAMP^{test}$ and $DALI^{test240}$ sets can be generalised to other datasets. However, consider that both of these models have much higher *WER*s on monophonic recordings. This means that their poorer performance in both domains can be generalisable.

- Model 4 is also a cross-domain model. The risk of overfitting to monophonic recordings still applies as $CPD_{mono} > 0$. The performance on $DALI^{test240}$ is similar to other polyphonic sets, so CPD_{poly} is near zero. In contrast to Model 1, the cross-domain model has $CPD_{overall} < 0$, hence it carries the risk of *underfitting* and further tuning is required.

5.4.3 Performance on Separated Vocals

Experiments in Section 4.4.6 showed that a cross-domain DNN-HMM lyrics transcriber that performs well on polyphonic recordings does not necessarily benefit from source separation (recall the results in Table 4.21). On the other hand, previous research leveraged source separation for performance improvement on end-to-end models even when trained on polyphonic recordings (Stoller et al., 2019; Gupta et al., 2020). The results in Table 5.4 show the performance of the end-to-end models tested in the above cross-dataset evaluation stage on source separated vocals. The most commonly used polyphonic test datasets are used at this stage. Vocals are separated using the *spleeter* toolkit (Hennequin et al., 2020), as in the relevant experiments in Chapter 4. In order to observe the relative gain due to source separation compared to inference on the original polyphonic mixes, the *CPD* metric is utilised. The polyphonic versions of the *Hansen*, *Mauch* and *Jamendo* sets are included in $\mathcal{D}^{test}_{self}$ and their separated versions are in $\mathcal{D}^{test}_{others}$. This means the *CPD* measures the performance drop on seperated vocals with respec to the original polyphonic recordings. Within this perspective, the *CPD* values in Table 5.4 are not directly comparable with the ones in Table 5.3.

Model	*Hansen*	*Mauch*	*Jamendo*	CPD_{sep}
1)	78.60	66.38	83.23	-7.13
2)	87.17	86.38	79.64	16.76
3)	86.32	81.01	80.53	15.65
4)	**75.07**	**57.16**	**77.22**	-15.18

Table 5.4: *WER* scores end-to-end models on source-separated vocals. The columns named with test sets are the *WER* scores and the rightmost column is *CPD* w.r.t performance on original polyphonic versions.

Following a similar outline with Section 5.4.2, the findings of the results in Table 5.4 are listed below:

- Source separation helped improve *WER*s of the monophonic model (1) which is reflected as $CPD_{sep} < 0$ (negative performance drop, i.e. performance on separated vocals is better than on polyphonic recordings).
- However, it did not improve word recognition rates for the acoustic models fine-tuned on *DALI* (2,3), and hence $CPD > 0$. This is inline with the relevant observation for the DNN-HMM approach in Section 4.4.6 regarding models that have polyphonic recordings in their training data performing better on polyphonic recordings than on separated vocals.
- Similar to the monophonic model, the cross-domain model that is trained on $DAMP \bigcup DALI$ (4) also benefits from source separation (i.e. $CPD < 0$), which is the opposite case for the DNN-HMM models.

- It should be noted that a further performance improvement on separated vocals might be achieved if the training used spleeter-separated data.

5.5 Summary

In this chapter, a method is presented for building an end-to-end lyrics transcription system, which can perform similarly to the DNN-HMM based model on the monophonic test set $DAMP^{test}$. The main method can be summarised as fine-tuning a Transformer + Hybrid CTC / Attention ASR model on singing data. The best performance is achieved by fine-tuning both the acoustic and language models. Furthermore, potential for cross-domain transfer to polyphonic data is investigated based on the observations drawn upon cross-dataset evaluation. This analysis shows that the monophonic end-to-end model, similar to its DNN-HMM counterpart, has a domain bias towards in-domain data and a large performance drop is observed in the polyphonic domain. On the other hand, the analysis also highlights a number of contrasting observations with DNN-HMM models regarding cross-domain training. First of all, cross-domain training did not consistently improve polyphonic results. In contrast, it was slightly useful for monophonic recordings. For all models, *WER*s on polyphonic data are very high for both $\mathcal{D}^{test}_{self}$ and D^{test}_{others}, which is reflected in *CPD* being closer to zero than for their DNN-HMM counterparts. This implies that the low recognition rates of end-to-end models reported in this chapter can be generalisable to unseen polyphonic data, hence an overall improvement is needed. In addition, source separation helped improve word recognition rates for the cross-domain model, but not for the polyphonic models fine-tuned only on *DALI* (2) and the cascaded transfer learning method in Model 3.

To sum up, an end-to-end model is constructed in this chapter using the same data settings as for the DNN-HMM based lyrics transcriber so that the results obtained from these two distinct approaches are comparable. As a novel contribution, a cascaded transfer learning approach is proposed, and cross-domain end-to-end training is applied to improve the performance of the acoustic model on polyphonic recordings. From this perspective, this study also provides evidence for the possibility of the information between speech and singing being transferable. While both approaches lead to better results, the improvements are not consistent among different evaluation sets. Overall, the cross-domain model had the best results on both sets in $\mathcal{D}^{test}_{self}$. Within the scope of the above observations, the performance gap between end-to-end and DNN-HMM based lyrics transcription models evaluated in this study is evident. This performance gap is further discussed in Chapter 6.

Chapter 6

Discussion & Conclusion

The previous two chapters explained the system principles of the two major paradigms in LVCSR and proposed novel methods to adapt them to singing data, and thus for the ALT task. This section provides the final discussions regarding the performance robustness, generalisability, and scalability aspects of these paradigms. At the end of Section 6.1, a comparison of the best-performing proposed models with what was reported in the previous literature is given. After that, the novel findings of this thesis are summarised. Based on the knowledge derived from the data and the experiments, the potential future directions in ALT research are highlighted in the final section. This chapter finishes with a summary of the overall contributions of this dissertation, specifically regarding the data, knowledge and user-driven aspects.

6.1 DNN-HMM vs. End-to-End

Cross-Dataset Evaluation

For fairness in evaluation, the best performing DNN-HMM and end-to-end models are selected for comparison based on their results on the $DAMP^{test}$ and $DALI^{test240}$ sets (i.e. $\mathcal{D}_{self}^{test}$) (refer to Tables 4.23 and 5.3). The results of the DNN-HMM approach considered in this section are for the Model 9+ in Table 4.22 which is a cross-domain model with the MStreNet network architecture and uses a 4-gram language model that is trained only on lyrics (Demirel et al., 2021b). The end-to-end model is also a cross-domain model where the pretrained speech acoustic model is fine-tuned directly on $DAMP \bigcup DALI$ (Model 4 in Table 5.3).

Model	*DAMP*^{test}	*DALI*^{test240}	NUS	*Hansen*^{mono}	CPD_{mono}	*Hansen*^{poly}	*Mauch*	*Jamendo*	CPD_{poly}	*CPD*
DNN-HMM	**15.89**	**42.98**	**7.49**	**23.00**	-4.75	**36.78**	**39.00**	**34.94**	-6.6	-16.62
E2E	16.04	67.21	19.93	53.92	11.56	72.43	65.16	79.12	2.13	-13.40

Table 6.1: *WER* (%) comparison of the cross-domain DNN-HMM and end-to-end models. The *CPD*s are illustrated in 6.1. The highlighted cells (in blue colour) indicate $\mathcal{D}^{test}_{self}$ used to compute *CPD*s.

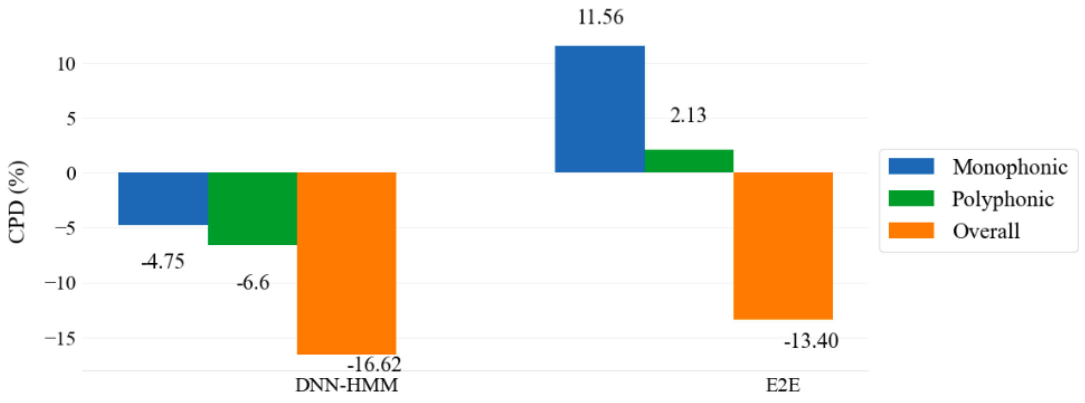


Figure 6.1: DNN-HMM vs E2E: Cross-dataset performance drop.

It is clear from the *WER* comparison in Table 6.1 that the DNN-HMM approach has lower error rates on all of the evaluation sets. The end-to-end model has a comparable performance on *DAMP*^{test}. However, it performs generally much worse on polyphonic recordings. From this perspective, it can be inferred that the size of the monophonic training dataset is sufficient to build a monophonic end-to-end lyrics transcriber (through transfer learning) that has a comparable performance to the DNN-HMM model on solo-singing data. However, this does not apply for the polyphonic training dataset (which has a similar size to the monophonic one) as *WER*s are generally much higher, and hence increasing its size is required. This might be associated with the higher spectral complexity and varying music styles in polyphonic recordings. On the other hand, there is a considerable performance drop on other monophonic test sets compared to the DNN-HMM model. This results in a positive *CPD* indicating the risk of overfitting to *DAMP* data. This also implies that increasing the variance of the monophonic training set is necessary to achieve a more generalisable performance. The overall *CPD*s for both models are in a comparable range, indicating that their performances on $\mathcal{D}^{test}_{self}$ are similarly generalisable on other sets in this cross-dataset evaluation. However, considering that the DNN-HMM model has much lower *WER*s overall, its performance is generally better than the end-to-end model. Furthermore, the DNN-HMM model has negative *CPD*s, therefore, overfitting does not seem to be the primary issue. To sum up, this comparison verifies that the DNN-HMM model outperforms end-to-end models under the low data resource circumstances for the ALT problem.

Error Analysis

Further analysis is undertaken in Figure 6.2 where errors (substitution, insertion and deletion) and overall *WER*s are computed singer-wise, and are illustrated via box-whisker plots. The interquartile range (IQR) of the boxes consider the 25-75 % of the data and the horizontal line within indicates the median value. The whiskers extend to 5-95%. The statistics are computed on the polyphonic $DALI^{test240}$. It can be seen that the medians of the singer-wise error distributions for the end-to-end model are generally higher. On the other hand, there appears to be more outlier errors for the DNN-HMM model. This observation is in line with the fact that $|CPD_{poly}^{\text{DNN-HMM}}| > |CPD_{poly}^{\text{E2E}}|$, and hence slightly less generalisable results on polyphonic recordings despite having superior performance. From this perspective, it can be inferred that increasing the data variability (for instance in terms of music styles, singers and recording conditions) is needed to improve the DNN-HMM model.

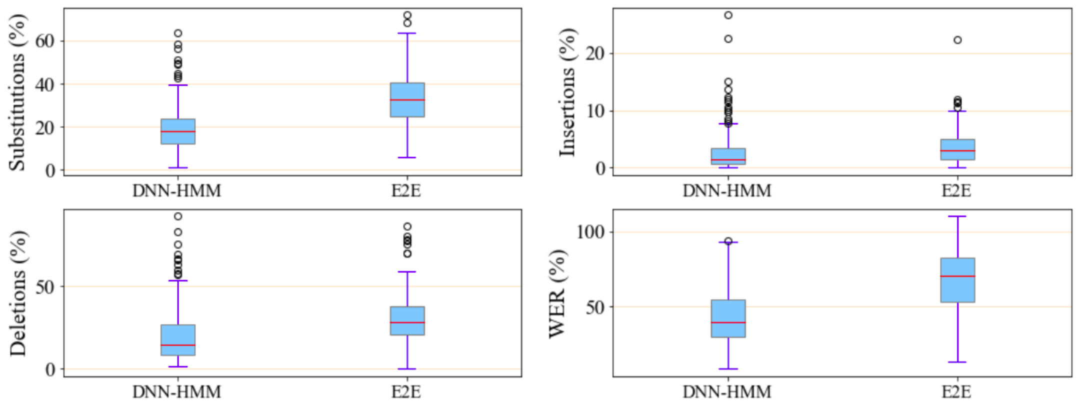


Figure 6.2: DNN-HMM vs E2E: Error analysis. The data samples are computed singer-wise.

Model Complexity

Figure 6.3 shows the number of trainable parameters per model. The end-to-end model is considerably larger with more than 160 million parameters compared to the DNN-HMM model with less than 9 million parameters. This is one of the main reasons why the end-to-end models require much larger training datasets. Note that very large models also require more powerful computational resources for training.

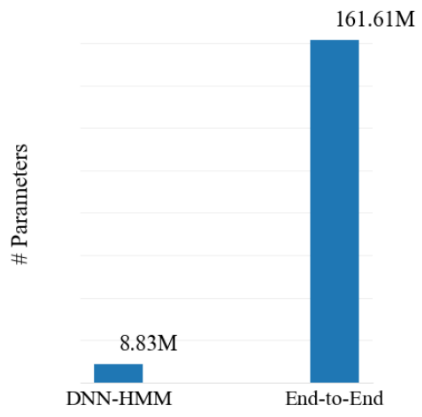


Figure 6.3: Number of trainable parameters

The Effect of Data Preprocessing

Besides the word recognition rates and models' performance generalisability across completely unseen data and domains, one needs to consider certain preprocessing steps included in the overall inference pipeline. For instance, the end-to-end model requires automatic segmentation (see Figure 5.5) and according to the steps explained in Section 5.3.2, manual correction is applied to the test samples. Therefore, the end-to-end results in this study are not obtained by means of pure inference, and a performance drop is expected when using suboptimal segments obtained via automatic segmentation. On the other hand, the Kaldi-based DNN-HMM solution can process longer audio recordings at once[1]. Moreover, due to the requirement of fixed-length audio at the input, the utterance duration (hence the amount of padding) is another determining factor for the inference performance of the end-to-end models. These external dependencies can be considered as disadvantages towards the scalability of the end-to-end approach.

Comparison with the literature

Tables 6.2 and 6.3 provide a comparison of the DNN-HMM and end-to-end models with what was reported in the ALT literature excluding the publications which are the outcomes of this research project (Demirel et al., 2020a, 2021a,c,b). The best performing model for each

[1]In practice, the recordings are split into 1.5 second chunks and the forward pass is applied on these. The probabilities are accumulated for all chunks until the last one and decoding (beam search) is applied on the final accumulated probabilities, hence the context is preserved.

evaluation set is included in this comparison. For this reason, results obtained via the 2nd pass RNNLM rescoring are reported for *DAMP^{test}* and Mauch sets on top of the DNN-HMM model in Table 6.1. The best end-to-end results for *DAMP^{test}* and *Jamendo* and *Mauch* sets are obtained using the Cross-Domain / Transfer Learning approach while the cascaded transfer learning approach (model 3 in Table 5.3) had the lowest *WER* scores.

	DNN-HMM			
Dataset	**Previous Work**		**This Study**	
DAMP^{test}	LFMMI (Dabike and Barker, 2019)	19.60	LFMMI + MStreNet	**12.56***
Hansen^{poly}	LFMMI	47.01	+ Cross-domain Training	**36.78**
Mauch	+ Genre-labeled phonemes	44.02	+ Music informed Sil. (Demirel et al., 2021b)	**38.81***
Jamendo	+ Vowel extended lexicon (Gupta et al., 2020)	59.57	+ Singing adapted lexicon (Demirel et al., 2021c)	**34.94**

Table 6.2: Comparison of the DNN-HMM *WER* results with the previous literature. Scores marked with '*' are obtained with the RNNLM rescoring (Demirel et al., 2020a) due to better results compared to 4-gram.

	End-to-end			
Dataset	**Previous Work**		**This Study**	
DAMP^{test}	Transformers + PD Augment (Zhang et al., 2021)	**9.80**	Transformers + Transfer Learning	16.04
Hansen^{poly}	Seq2seq / Attention (Gupta et al., 2020)	80.10	+ Cross-domain Training	**72.43**
Mauch	UNet + CTC (Stoller et al., 2019)	70.90	Transformers	**59.68**
Jamendo		77.80	+ Cascaded Transfer Learning	**67.40**

Table 6.3: Comparison of the end-to-end *WER* results with the previous literature.

The MstreNet based cross-domain DNN-HMM model performs better than the other DNN-HMM results reported in the literature. On the polyphonic *Hansen*, *Mauch* and *Jamendo* sets, this model achieves the *state-of-the-art* by a considerably large margin (around 15-30 % relative *WER* improvement depending on the test set). On these datasets, the end-to-end models presented in this study achieve around 10% absolute *WER* improvement compared to previously reported scores in the literature, however there is still a large performance gap between the DNN-HMM and the end-to-end models.

Most recently, Zhang et al. (2021) achieved *WER* scores below 10 % on the monophonic *DAMP^{test}*. This model is similar to the end-to-end model presented in this study in several aspects. First of all, both models use Transformer architectures with similar hyperparameters. For training, the same *DAMP* and *DALI* sets are used. Both models leverage speech data, i.e. the Librispeech corpus, where Zhang et al. (2021) apply training from scratch and include the Librispeech data in the training set while the model in this study applies finetuning on the Librispeech pretrained model on *DAMP* and *DALI*. In addition, Zhang et al. (2021) apply pitch and duration-wise data augmentation, i.e. *PDAugment*, which was shown to improve *WER* results considerably. With this approach, the aforementioned model achieves around 6 % lower *WER* than the end-to-end model presented in this study. Without PDAugment, their model had around 20 % *WER* and the hybrid CTC / attention + Transfer Learning based end-to-end model performs 4 % better than this. With this setup, both models have very similar results as both leverage the data from *DAMP*, *DALI* and Librispeech. In this regard, they are comparable and better *WER* shows the benefit of using hybrid CTC / attention loss compared to using only the attention loss. On the other hand, the benefit of the PDAugment is also evident and future research has to be conducted to scale-up results.

6.2 Summary of Findings

This thesis tackles the task of automatic lyrics transcription in numerous aspects, raises several research questions, proposes methods for improved word recognition performance and provides an extensive evaluation framework. In order to summarise and highlight the research outcomes of this study, a list of important findings is given below:

1. **DNN-HMM vs. End-to-end:** According to the result tables in Section 6.1, the LFMMI based DNN-HMM approach to lyrics transcription is superior to end-to-end models. However, this is true given the size of the data resources available for training was around a few hundred hours. It is known to the research community that the gap between end-to-end and DNN-HMM models gets smaller as the training data gets larger. Therefore, this comparison also needs to be conducted in the presence of large training sets ($>$ 1000 hours). According to the cross-dataset evaluation study in this chapter, the poor performance of the end-to-end model is generalisable to unseen data, which is especially true for the polyphonic domain. While its performance on the monophonic *DAMP*^{test} is comparable to the DNN-HMM results, this cannot be generalised much to completely unseen monophonic data, which signals the risk of overfitting. The DNN-HMM model does not seem to suffer from overfitting according to the negative *CPD* values in Figure

6.1.

2. **Performance generalisability and evaluation in Automatic Lyrics Transcription:** This thesis considers two data domains within the ALT task and its applications: monophonic (solo-singing) and polyphonic (with music background) recordings. The cross-dataset evaluation scheme offers a way to benchmark lyrics transcription scores using and categorising the training and evaluation data based on the domain properties. It offers the use of *CPD* to summarise the behaviour of a model in question across a number of evaluation sets. This evaluation showed that both DNN-HMM and end-to-end models perform much better on monophonic recordings than polyphonic recordings, and adding solo-singing samples in the training sets helps to improve performance in the polyphonic domain. Finally, the singer-wise error analysis in Section 6.1 highlighted that the lyrics transcription rate may vary depending on the singer.

3. **Cross-domain training:** Training the acoustic model both on monophonic and polyphonic recordings is shown to be beneficial for achieving a more domain-invariant performance. Including solo singing recordings especially benefits the transcription performance on the polyphonic domain. Moreover, modeling non-vocal sounds distinctively for silence and instrumental instances (by using separate labels - <silence> vs. <music>, as in the music-informed silence modeling approach proposed in Section 4.3.1) improves performance on polyphonic data.

4. **MStreNet architecture:** The multistream TDNN architecture performs better than its single stream counterpart and is more robust on polyphonic recordings and against varying reverberation conditions. Through diversifying the parallel TDNN streams, a more compact model can be achieved, which also benefits the word recognition rate and inference runtime.

5. **Speech-to-singing transfer:** The experiments on end-to-end models show that some information between speech and singing can be transferable within the context of automatic lyrics transcription as the speech-to-singing transfer model performance is superior to that of the model trained on either speech or singing.

6. **Inference on separated vocals:** Although source separation help improve word recognition rates for the monophonic models, the cross-domain DNN-HMM models perform better on the original polyphonic mixes. In contrast, source separation slightly improves word recognition rates for the cross-domain end-to-end model. However, this could as well be due to the overall poor performance on polyphonic data. Note that this study utilises one source separation model (spleeter). Because of this, the effect of the artifacts introduced by the source separation procedure on lyrics transcription needs to be investigated further. From this perspective, whether or not source separation is beneficial for ALT from polyphonic recordings still remains an open question.

7. **Model optimisation:** There are a number of independently tuned computational blocks within the DNN-HMM approach and this study proposes improvements for each block. Combining improved blocks did not always lead to an optimal performance. This is especially true for improvements proposed in the language and pronunciation models. Due to this, tuning the DNN-HMM model is difficult, whereas tuning the end-to-end model is more straightforward as both the acoustic and language models are optimised on the same objective.

8. **Computational cost:** As mentioned before, the DNN-HMM model involves several preprocessing steps required prior to training the acoustic model. Some of these steps need to be re-executed for each experimental iteration. Due to this, model optimization is time costly. On the other hand, the end-to-end training can be initiated directly on audio data paired with text, hence the overall training pipeline is much simpler. Because of this, iterating over hyperparameters and thus model optimisation is less time costly compared to the DNN-HMM framework. However, the end-to-end model has a much larger model size and thus complexity (Figure 6.3), which requires powerful machines, and again, much larger training sets.

6.3 Future Prospects

One of the goals of this thesis is to identify the opportunities and challenges for the future development of ALT research. Considerable improvements compared to the previous literature have been reported, however this alone does not mark a robust performance and further improvement in lyrics recognition rates is required. Although this study provides one of the most extensive evaluation frameworks presented in research, it still has several limitations, such as using data from a single language and limited span of music and singing styles. Within this respect, there still exists a long list of directions that the research on ALT can focus on. In this section, the current challenges and the future opportunities for the development of ALT research are mentioned.

6.3.1 Challenges

- **More training data:** Most recent benchmark ASR models use at least 1000 hours of audio data for training the acoustic model. *DAMP* and *DALI* have around 300 hours in total. Therefore, the size of the open-source training data needs to be at least tripled. Although creating time annotations can be costly, these can be generated on the sentence-level through alignments using a pretrained model, an approach similar to the one used to generate annotations for the Librispeech dataset (Panayotov et al., 2015).

- **Multilingual lyrics transcription:** This thesis focuses on lyrics transcription for the English language only, and research on other languages has to be undertaken considering ALT's potential applications. Other than the data requirement for the language in question, one main challenge for multilingual lyrics transcription is the dependency on linguistic expert knowledge to build the word-to-phoneme pronunciation modeling. In this study, possibilities for eliminating such dependency are investigated via the grapheme-based DNN-HMM and end-to-end models which could be leveraged for future research. Note that the gap between the grapheme and phoneme based DNN-HMM models might dwindle with more training data as well, which would decrease the dependency on linguistic expertise for constructing such models.

- **Style-independent acoustic modeling:** The acoustic properties of singing performances may vary depending on the singers, music and singing styles. Some of the directions for a more style-independent performance are learning domain/style-invariant features or leveraging genre embeddings, similar to the singer identity embeddings using i-vectors

(Saon et al., 2013; Liu et al., 2018).

- **Ambiguity in evaluation:** Lyrics in text form may include altered word forms to cue melody construction, such as repeated vowels (eg. 'HIGHS $\rightarrow$ HI-I-I-I-IGHS'[2]), which may cause alterations in word pronunciations as well.

- **Performance ceiling:** As mentioned before, the surveys by Johnson et al. (2013) and Fine and Ginsborg (2014) demonstrated that word intelligibility according to human listeners reduces in sung utterances. However, the average human lyrics transcription performance has not yet been quantitatively measured. In order to benchmark lyrics transcription results in this respect, listening tests similar to the one by Amodei et al. (2016) that measure the word recognition rates by human listeners need to be conducted.

6.3.2 Opportunities

- **Training on big data:** The results reported in the recent work by Zhang et al. (2021) showed that including the available speech data, like the Librispeech set, in training the end-to-end lyrics transcriber is useful for improved transcription performance. More recently in MIREX 2021: Automatic Lyrics Transcription challenge, this was shown to be also true for the LFMMI-based DNN-HMM system (Yang et al., 2021). In addition, the team added an available private dataset of polyphonic recordings for training the acoustic model and a much larger lyrics corpus for building the language model. With this, Yang et al. (2021) showed that big improvements can be achieved in the presence of very large training sets. These studies substantiate the possibility of robust acoustic and language modeling when sufficient resources are available.

- **Data augmentation:** Zhang et al. (2021) also proposed a novel data augmentation approach through pitch and duration shifting which was shown to improve the transcription performance for the end-to-end approach. This augmentation method would possibly help improve the performance of the DNN-HMM approach.

- **Leveraging music priors:** Although they can be expressed by the same mathematical expression, the tasks of ASR and ALT have their own domain specific characteristics.

[2]This is an example taken from $DAMP^{test}$.

This is especially true when attempting to transcribe lyrics from polyphonic recordings where there is music accompaniment and/or multiple singers. Previous research showed that music style information can be leveraged for improved performance (Gupta et al., 2020). This can be expanded to including genre / music style embeddings in the feature space, a method similar to speaker-adaptive training using i-Vectors (Saon et al., 2013). Genre-adaptive training can also be achieved via learning music-style invariant features using representation learning paradigms, such as contrastive learning (Schneider et al., 2019; Spijkervet and Burgoyne, 2021). Another direction in this regard is using additional musical features. For instance, Dabike and Barker (2021) showed that adding pitch in the feature space can be helpful, yet the improvement due to this gets negligible as the training data gets larger. Instead of using pitch in the feature space, one could apply multitask learning through optimising the model parameters with respect to the combination of pitch and lyrics (phonemes, graphemes, etc.) losses.

- **Lyrics and musical structures:** Many songs can be segmented into distinct sections, such as verses, choruses, intros, outros, interludes, etc., and each section may have distinct semantics and characteristics. One possible direction in this regard would be to apply 'section-aware' training. This could be achieved via testing different granularities of lyrics when curating the training samples. For instance, the language model can use the whole section of verses and choruses as training samples, instead of using line-level lyrics. This would essentially bring more context to the lyrics transcriber during inference. In addition to this, leveraging the rhythmic, prosodic and melodic structures as a prior to lyrics transcription has a potential to achieve more context-aware models. The pitch and beat tracks and the dynamic properties of sung melodies can be taken into consideration in this direction.

- **Multi-task learning:** As mentioned in Section 1.1, singing involves both semantic and musical context. In this regard, the training of a lyrics transcriber can benefit from learning information about other relevant tasks, such as vocal automatic music transcription, source separation, language and cover song identification. Conversely, the aforementioned tasks can also leverage model optimisation considering the lyrics transcription objective. For instance, the concept of multi-task learning can be utilised to reconstruct more intelligible lyrics at the output of vocal source separation. A similar perspective was presented by Basak et al. (2021) where the ALT model is jointly optimised with a singing voice synthesiser.

- **Audio transformation:** The *WER* scores reported in this thesis show that the lyrics transcription performance is much better on solo singing performances than polyphonic recordings. According to the experiments on the effect of source separation, the performance does not necessarily improve on separated vocals when cross-domain models are used. This is potentially due to the artifacts introduced during separation. In this regard, source separation is still an interesting research direction to take. One potential direction specific for ALT is building a vocal separation algorithm that has the objective of maintaining or enhancing the *word intelligibility*. Another possibility for audio transformation to improve ALT is *songifying* the available speech data by means of data augmentation (Kruspe, 2016). This idea was later improved by Zhang et al. (2021) and considerable improvements are observed. The opposite transformation is also possible, in which the singing data would be *speechified*, i.e. converted to speech utterances.

6.4 Summary of Contributions

6.4.1 Data-Driven Aspects

Chapter 3 showed that sung utterances, as data for LVCSR, have specific properties different to speech such as input audio length, the variability of the phonetic and prosodic contents, and the presence of music accompaniment. This chapter also provided the details of the basic data preprocessing steps included the ALT pipelines. In addition, considering the open-source data resources available for research being limited in size compared to the benchmark training datasets used in LVCSR, two distinct approaches to ASR are employed: the DNN-HMM-based sequence discriminative training, and the transfer learning of end-to-end models. In particular, the experiments for the latter approach showed that some information between speech and singing can be transferable, thus pretrained speech models can be utilised for ALT in the case of low data resources.

Based on the potential applications of ALT and available data resources, this research considers singing data under two major data domains: solo singing (monophonic) performances and singing with music accompaniment (polyphonic recordings). Two methods for avoiding domain mismatch are proposed. First, cross-domain training including data from both monophonic and polyphonic recordings in the training set, and second, labelling and representing non-vocal music and silent instances with explicit tokens (i.e. music informed silence modeling).

One of the most important steps in developing a machine learning model is model evaluation. In this study, an extensive evaluation framework is presented including a new test set, and a novel evaluation metric is introduced for quantitatively observing model generalisability and performance drop across different datasets, domains and environmental conditions. These results provide a benchmark over all the test sets used in ALT research.

6.4.2 Knowledge-Driven Aspects

ALT is a complex problem that requires domain knowledge from several areas during its design. Pertinent fields include speech recognition, natural language processing, music information retrieval, neural networks, audio signal processing and, last but not least, linguistics and phonology. This study considers the domain knowledge from these fields when designing the ALT system. The first aspect concerns explicit language modelling: influenced by the prosodic elements of both natural language and music, song lyrics exhibit unique structures. Often rhyming is the priority over rules of grammar. Motivated by these aspects a lyrics-specific language model is built. Secondly, since lyrics are structured in unique ways depending on the musical structure and semantics, the training of both the language and acoustic models are performed on the line-level. Third, tokens are embedded explicitly in training that indicate the beginnings and endings of the lyrical lines. This is performed with distinct tokens depending on whether a recording is monophonic or polyphonic, knowing that the non-vocal silence and music instances exhibit distinct acoustic properties (i.e. music informed modeling in Section 4.3.1), which is shown to be a simple but effective method for improving ALT performance especially on polyphonic recordings. Moreover, a computational analysis is held to get a better understanding of the systematic pronunciation variations between sung and spoken utterances. Using the knowledge that the study reveals, a novel pronunciation dictionary for singing voice is developed.

The design of a neural network architecture requires a deep understanding of the training objective, the properties of data to be processed and, how neural networks process information. Based on the knowledge of these aspects, a novel compact variant of a state-of-the-art neural network architecture is presented for noise-robust LVCSR that is shown to have a lower inference time and model complexity while achieving an improved performance.

Furthermore, based on the knowledge derived from the comparison of the proposed ALT models, the challenges and future opportunities of this research field are outlined to set initial guidance for future researchers. Specifically, this study provides examples for the uses of a couple of open-source toolkits in building a lyrics transcriber. Within this regard, this dissertation contributes towards reducing the research and literature gaps in ALT research.

6.4.3 User-Driven Aspects

The present project had the opportunity to test the lyrics transcription model on a real-world application as a result of the collaboration with Doremir. In this, a singing-to-text module is developed to be integrated and utilised within an automatic music transcription software. Imposed by practical concerns of end-users, a successful singing-to-text module requires a fast runtime, low memory footprint, robustness and performance generalisability. For the last, singers from 30 different countries are included in training the transcriber to make it familiar with varying accents. For robustness against environmental variability and to reduce the effect of domain-mismatch the cross-domain training approach is proposed. For the first two aforementioned requirements, a multistream neural network architecture is developed which is also shown to perform faster at inference than its single stream counterpart. In addition, the architecture is tuned for achieving a more compact model to reduce model complexity, thus memory requirements. Finally, this study holds an extensive evaluation and comparison of the two major paradigms in ASR, with the goal of finding the most scalable and better performing approach considering ALT's potential real-world use cases, and given the available resources.

6.5 Final Remarks

Automatic lyrics transcription has been considered as one of the most complex tasks within MIR as it requires expertise and domain knowledge from multiple data domains: music, speech and language. One of the major bottlenecks prior to the initiation of this research was the lack of open-source datasets to train models. Because of this, ALT was not among the most popular tasks within the MIR community, and the existing literature was limited. Soon after I started my research on ALT, two datasets were released that could be leveraged for the purpose, the *DAMP* and *DALI* sets. At this point, these datasets and the reproducible work by Dabike and Barker (2019) granted the opportunity to achieve considerable performance improvement. In the beginning of my Ph.D. journey, the *WER*s reported in the literature were around 34 % and above 70 % on monophonic and polyphonic recordings respectively, whereas this thesis approximately halved the error rates compared to what was reported previously depending on the dataset. With such improvement, the technology of ALT began to seem to be applicable for real-world use cases. Through our collaboration with Doremir Music Research AB, we developed the first industrial application of ALT and integrated this with their music transcription system, with the goal of creating a 360-degree songwriting software. More can be found on this at *https://scorecloud.com/*.

I hope that the work presented in this thesis paves the ground for future researchers to improve the current state of lyrics transcription even further. For open-science and reproducibility, the codebase of this thesis is shared publicly with the research community at *https://github.com/emirdemirel/ALTA*.

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

Martine Adda-Decker and Lori Lamel. The use of lexica in automatic speech recognition. In *Lexicon Development for Speech and Language Processing*, pages 235–266. Springer, 2000.

S Omar Ali and Zehra F Peynircioğlu. Songs and emotions: Are lyrics and melodies equal partners? *Psychology of Music*, 2006.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFST: A general and efficient weighted finite-state transducer library. In *International Conference on Implementation and Application of Automata*, pages 11–23. Springer, 2007.

Jont B Allen. How do humans process and recognize speech? In *Modern Methods of Speech Processing*. Springer, 1995.

Jont B Allen and David A Berkley. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 1979.

Tanel Alumäe and Mikko Kurimo. Efficient estimation of maximum entropy language models with n-gram features: An SRILM extension. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep Speech

2: End-to-end speech recognition in English and Mandarin. In *International Conference on Machine Learning (ICML)*, pages 173–182. PMLR, 2016.

Tasos Anastasakos, John McDonough, Richard Schwartz, and John Makhoul. A compact model for speaker-adaptive training. In *Proceedings of the Fourth International Conference on Spoken Language Processing*. IEEE, 1996.

Galen Andrew and Jianfeng Gao. Scalable training of l-1 regularized log-linear models. In *Proceedings of the 24th International Conference on Machine learning*, pages 33–40, 2007.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*, 2020.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Lalit Bahl, Peter Brown, Peter De Souza, and Robert Mercer. Maximum mutual information estimation of Hidden Markov Model parameters for speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1986.

Mary E Ballard, Alan R Dodson, and Doris G Bazzini. Genre of music and lyrical content: Expectation effects. *The Journal of Genetic Psychology*, 160(4):476–487, 1999.

Jon Barker, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe. The third 'CHiME' speech separation and recognition challenge: Analysis and outcomes. *Computer, Speech & Language*, 46:605–626, 2017.

Gonçalo T Barradas and Laura S Sakka. When words matter: A cross-cultural perspective on lyrics and their relationship to musical emotions. *Psychology of Music*, 2021.

Sakya Basak, Shrutina Agarwal, Sriram Ganapathy, and Naoya Takahashi. End-to-end lyrics recognition with voice to singing style transfer. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 266–270. IEEE, 2021.

William R Bauer. Scat singing: A timbral and phonemic analysis. *Current Musicology*, 2002.

Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

Yoshua Bengio. *Learning Deep Architectures for AI*. Now Publishers Inc., 2009.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36. JMLR Workshop and Conference Proceedings, 2012.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 2012.

Meng Cai, Yongzhe Shi, Jian Kang, Jia Liu, and Tengrong Su. Convolutional maxout neural networks for low-resource speech recognition. In *The 9th International Symposium on Chinese Spoken Language Processing*, pages 133–137. IEEE, 2014.

Chris Cannam, Christian Landone, and Mark Sandler. Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the 18th ACM International Conference on Multimedia*, pages 1467–1468, 2010.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

Guoguo Chen, Hainan Xu, Minhua Wu, Daniel Povey, and Sanjeev Khudanpur. Pronunciation and silence probability modeling for ASR. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Xie Chen, Xunying Liu, Anton Ragni, Yu Wang, and Mark JF Gales. Future word contexts in neural network language models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 97–103. IEEE, 2017.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

E Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the acoustical society of America*, 25(5):975–979, 1953.

Jaejin Cho, Murali Karthick Baskar, Ruizhi Li, Matthew Wiesner, Sri Harish Mallidi, Nelson Yalta, Martin Karafiat, Shinji Watanabe, and Takaaki Hori. Multilingual sequence-to-sequence speech recognition: Architecture, transfer learning, and language modeling. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 521–527. IEEE, 2018.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Henry Leland Clarke. The basis of musical communication. *The Journal of Aesthetics and Art Criticism*, 10(3):242–246, 1952.

Lauren B Collister and David Huron. Comparison of word intelligibility in spoken and sung phrases. 2008.

Albin Andrew Correya, Romain Hennequin, and Mickaël Arcos. Large-scale cover song detection in digital music libraries using metadata, lyrics and audio features. *arXiv preprint arXiv:1808.10351*, 2018.

Gerardo Roa Dabike and Jon Barker. Automatic lyrics transcription from karaoke vocal tracks: Resources and a baseline system. *Interspeech*, 2019.

Gerardo Roa Dabike and Jon Barker. The use of voice source features for sung speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

Nivja H De Jong and Ton Wempe. Praat script to detect syllable nuclei and measure speech rate automatically. *Behavior Research Methods*, 2009.

Emir Demirel, Sven Ahlbäck, and Simon Dixon. Automatic lyrics transcription using dilated convolutional neural networks with self-attention. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020a.

Emir Demirel, Sven Ahlbäck, and Simon Dixon. A recursive search method for lyrics alignment. In *MIREX 2020 Audio-to-Lyrics Alignment and Lyrics Transcription Challenge*, 2020b.

Emir Demirel, Sven Ahlbäck, and Simon Dixon. Low resource audio-to-lyrics alignment from polyphonic music recordings. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021a.

Emir Demirel, Sven Ahlbäck, and Simon Dixon. Mstre-net: Multistreaming acoustic modeling for automatic lyrics transcription. In *Proceedings of International Society in Music Information Retrieval Conference (ISMIR)*, 2021b.

Emir Demirel, Sven Ahlbäck, and Simon Dixon. Computational pronunciation analysis in sung utterances. In *29th European Signal Processing Conference (EUSIPCO 2021)*. IEEE, 2021c.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Nicole Scotto Di Carlo. Effect of multifactorial constraints on intelligibility of opera singing (i). *Journal of Singing*, 2007.

R Donald, G Kreutz, L Mitchell, and R MacDonald. What is music health and wellbeing and why is it important? In *Music, Health, and Wellbeing*, pages 3–11. Oxford University Press, 2012.

Zhiyan Duan, Haotian Fang, Bo Li, Khe Chai Sim, and Ye Wang. The NUS sung and spoken lyrics corpus: A quantitative comparison of singing and speech. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE, 2013.

Georgi Dzhambazov et al. *Knowledge-based probabilistic modeling for tracking lyrics in music audio signals*. PhD thesis, Universitat Pompeu Fabra, 2017.

Foteini Filippidou and Lefteris Moussiades. A benchmarking of IBM, Google and Wit automatic speech recognition systems. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 73–82. Springer, 2020.

Philip A Fine and Jane Ginsborg. Making myself understood: Perceived factors affecting the intelligibility of sung text. *Frontiers in Psychology*, 2014.

W Tecumseh Fitch. The biology and evolution of music: A comparative perspective. *Cognition*, 100(1):173–215, 2006.

Hiromasa Fujihara, Masataka Goto, Jun Ogata, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Automatic synchronization between lyrics and music CD recordings based on Viterbi alignment of segregated vocal signals. In *Eighth IEEE International Symposium on Multimedia (ISM'06)*, pages 257–264. IEEE, 2006.

HIROYA Fujisaki. Dynamic characteristics of voice fundamental frequency in speech and singing. acoustical analysis and physiological interpretations. *Dept. for Speech, Music and Hearing, Tech. Rep*, 1981.

Mark Gales and Steve Young. The application of hidden Markov models in speech recognition. *Foundations and Trends in Signal Processing*, 2008.

Aravind Ganapathiraju, Jonathan Hamaker, Joseph Picone, Mark Ordowski, and George R Doddington. Syllable-based large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(4):358–366, 2001.

Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Prentice Hall, 1999.

Susan C Gardstrom. Positive peer culture: A working definition for the music therapist. *Music Therapy Perspectives*, 4(1):19–23, 1987.

Alexandru-Lucian Georgescu, Alessandro Pappalardo, Horia Cucu, and Michaela Blott. Performance vs. hardware requirements in state-of-the-art automatic speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2021(1):1–30, 2021.

Arnab Ghoshal, Pawel Swietojanski, and Steve Renals. Multilingual training of deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7319–7323. IEEE, 2013.

Rhonda Gibson, Charles F Aust, and Dolf Zillmann. Loneliness of adolescents and their choice and enjoyment of love-celebrating versus love-lamenting popular music. *Empirical Studies of the Arts*, 18(1):43–48, 2000.

John J Godfrey, Edward C Holliman, and Jane McDaniel. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 1, pages 517–520. IEEE Computer Society, 1992.

Raymond L Goldsworthy. Correlations between pitch and phoneme perception in cochlear implant users and their normal hearing peers. *Journal of the Association for Research in Otolaryngology*, 16(6):797–809, 2015.

Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning (ICML)*, 2014.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine learning (ICML)*, 2006.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

Jean Westerman Gregg and Ronald C Scherer. Vowel intelligibility in classical singing. *Journal of Voice*, 2006.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015.

Chitralekha Gupta, Haizhou Li, and Ye Wang. Automatic pronunciation evaluation of singing. In *Interspeech*, 2018.

Chitralekha Gupta, Emre Yılmaz, and Haizhou Li. Automatic lyrics alignment and transcription in polyphonic music: Does background music help? In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

Reinhold Haeb-Umbach and Hermann Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13–16, 1992.

Kyu J Han, Jing Pan, Venkata Krishna Naveen Tadala, Tao Ma, and Dan Povey. Multistream CNN for robust acoustic modeling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6873–6877. IEEE, 2021.

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. Contextnet: Improving convolutional neural networks for automatic speech recognition with global context. *arXiv preprint arXiv:2005.03191*, 2020.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

Jens Kofod Hansen. Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients. In *9th Sound and Music Computing Conference (SMC)*, 2012.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. Spleeter: A fast and efficient music source separation tool with pretrained models. *Journal of Open Source Software*, 2020.

Hynek Hermansky. Multistream recognition of speech: Dealing with unknown unknowns. *Proceedings of the IEEE*, 101(5):1076–1088, 2013.

Hynek Hermansky. Coding and decoding of messages in human speech communication: Implications for machine recognition of speech. *Speech Communication*, 106:112–117, 2019.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Jack Hopkins and Douwe Kiela. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

Wei-Ning Hsu and James Glass. Extracting domain invariant features by unsupervised learning for robust automatic speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5614–5618. IEEE, 2018.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. Model complexity of deep learning: A survey. *Knowledge and Information Systems*, 63(10):2585–2619, 2021. URL `https://doi.org/10.1007/s10115-021-01605-0`.

Jiawen Huang, Emmanouil Benetos, and Sebastian Ewert. Improving lyrics alignment through joint pitch detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

Jui-Ting Huang, Jinyu Li, and Yifan Gong. An analysis of convolutional neural networks for speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4989–4993. IEEE, 2015.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *ACM International Conference on Information & Knowledge Management*, pages 2333–2338, 2013.

Eric J Humphrey, Sravana Reddy, Prem Seetharaman, Aparna Kumar, Rachel M Bittner, Andrew Demetriou, Sankalp Gulati, Andreas Jansson, Tristan Jehan, Bernhard Lehner, et al. An introduction to signal processing for singing-voice analysis: High notes in the effort to automate the understanding of vocals in music. *IEEE Signal Processing Magazine*, 36(1): 82–94, 2018.

Karim M Ibrahim, David Grunberg, Kat Agres, Chitralekha Gupta, and Ye Wang. Intelligibility of sung lyrics: A pilot study. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

Randolph B Johnson, David Huron, and Lauren Collister. Music and lyrics interactions and their influence on recognition of sung words: An investigation of word frequency, rhyme, metric stress, vocal timbre, melisma, and repetition priming. *Empirical Musicology Review*, 2013.

Naoyuki Kanda, Yusuke Fujita, and Kenji Nagamatsu. Lattice-free state-level minimum bayes risk training of acoustic models. In *Interspeech*, pages 2923–2927, 2018.

Dairoku Kawai, Kazumasa Yamamoto, and Seiichi Nakagawa. Lyric recognition in monophonic singing using pitch-dependent DNN. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017.

Kazuya Kawakami. Supervised sequence labelling with recurrent neural networks. *Ph. D. thesis*, 2008.

Patrick Kenny, Gilles Boulianne, and Pierre Dumouchel. Eigenvoice modeling with sparse training data. *IEEE Transactions on Speech and Audio Processing*, 13(3):345–354, 2005.

Veton Këpuska and Gamal Bohouta. Comparing speech recognition systems (Microsoft API, Google API and CMU Sphinx). *Int. J. Eng. Res. Appl*, 7(03):20–24, 2017.

Nikhil Ketkar. Introduction to pytorch. In *Deep Learning with Python*, pages 195–208. Springer, 2017a.

Nikhil Ketkar. Introduction to Keras. In *Deep Learning with Python*, pages 97–111. Springer, 2017b.

Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466, 1952.

Mirjam Killer, Sebastian Stüker, and Tanja Schultz. Grapheme based speech recognition. In *Interspeech*, 2003.

Soohwan Kim, Sangchun Ha, and Soyoung Cho. Openspeech: Open-source toolkit for end-to-end speech recognition. `https://github.com/openspeech-team/openspeech`, 2021.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4835–4839. IEEE, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Reinhard Kneser and Hermann Ney. Improved backing-off for n-gram language modeling. In *International Conference on Acoustics, Speech, and Signal processing (ICASSP)*. IEEE, 1995.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25: 1097–1105, 2012.

Anna M Kruspe. Training phoneme models for singing with 'songified' speech data. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

Anna M Kruspe. Bootstrapping a system for phoneme recognition and keyword spotting in unaccompanied singing. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, et al. Nemo: A toolkit for building AI applications using neural modules. *arXiv preprint arXiv:1909.09577*, 2019.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

Vera Kurkova. Constructive lower bounds on model complexity of shallow perceptron networks. *Neural Computing and Applications*, 29(7):305–315, 2018.

Paul Lamere, Philip Kwok, Evandro Gouvea, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. The CMU SPHINX-4 speech recognition system. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 2–5, 2003.

Duc Le, Xiaohui Zhang, Weiyi Zheng, Christian Fügen, Geoffrey Zweig, and Michael L Seltzer. From senones to chenones: Tied context-dependent graphemes for hybrid speech recognition.

In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 457–464. IEEE, 2019.

R Leanderson, J Sundberg, and C Von Euler. Breathing muscle activity and subglottal pressure dynamics in singing and speech. *Journal of Voice*, 1(3):258–261, 1987.

Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Julius - an open source real-time large vocabulary recognition engine. In *EUROSPEECH*, pages 1691,1694, 2001.

Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. 163(4):845–848, 1965.

Daniel J Levitin. *This Is Your Brain on Music: The Science of a Human Obsession*. Penguin, 2006.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An end-to-end convolutional neural acoustic model. *arXiv preprint arXiv:1904.03288*, 2019a.

Jinyu Li. Recent advances in end-to-end automatic speech recognition. *arXiv preprint arXiv:2111.01690*, 2021.

Jinyu Li, Rui Zhao, Zhong Meng, Yanqing Liu, Wenning Wei, Sarangarajan Parthasarathy, Vadim Mazalov, Zhenghao Wang, Lei He, Sheng Zhao, et al. Developing RNN-T models surpassing high-performance hybrid models with customization capability. *arXiv preprint arXiv:2007.15188*, 2020.

Ruizhi Li, Xiaofei Wang, Sri Harish Mallidi, Shinji Watanabe, Takaaki Hori, and Hynek Hermansky. Multi-stream end-to-end speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:646–655, 2019b.

Yukun Li, Emir Demirel, Polina Proutskova, and Simon Dixon. Phoneme-informed note segmentation of monophonic vocal music. In *2nd Workshop on NLP for Music and Audio (NLP4MusA 2021)*, 2021.

Chien-Feng Liao, Yu Tsao, Hung-Yi Lee, and Hsin-Min Wang. Noise adaptive speech enhancement using domain adversarial training. *arXiv preprint arXiv:1807.07501*, 2018.

Björn Lindblom and Johan Sundberg. The human voice in speech and singing. In *Springer handbook of acoustics*, pages 703–746. Springer, 2014.

Xunying Liu, Xie Chen, Yongqiang Wang, Mark JF Gales, and Philip C Woodland. Two efficient lattice rescoring methods using recurrent neural network language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(8):1438–1449, 2016.

Yi Liu, Liang He, Jia Liu, and Michael T Johnson. Speaker embedding extraction with phonetic information. *arXiv preprint arXiv:1804.04862*, 2018.

Andrej Ljolje, Fernando Pereira, and Michael Riley. Efficient general lattice generation and rescoring. In *Sixth European Conference on Speech Communication and Technology*, 1999.

Liang Lu, Xingxing Zhang, and Steve Renals. On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5060–5064, 2016.

Raymond AR MacDonald, David J Hargreaves, and Dorothy Miell. *Musical identities*. OUP Oxford, 2002.

Sri Harish Mallidi and Hynek Hermansky. Novel neural network based fusion for multi-stream ASR. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5680–5684. IEEE, 2016a.

Sri Harish Reddy Mallidi and Hynek Hermansky. A framework for practical multistream ASR. In *Interspeech*, pages 3474–3478, 2016b.

Matthias Mauch and Simon Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014.

Matthias Mauch, Hiromasa Fujihara, and Masataka Goto. Integrating additional chord information into HMM-based lyrics-to-audio alignment. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

Philip M McCarthy. *An Assessment of the Range and Usefulness of Lexical Diversity Measures and the Potential of the Measure of Textual, Lexical Diversity (MTLD)*. PhD thesis, The University of Memphis, 2005.

Iain A McCowan, Darren Moore, John Dines, Daniel Gatica-Perez, Mike Flynn, Pierre Wellner, and Hervé Bourlard. On the use of information retrieval measures for speech recognition evaluation. Technical report, Technical Report, IDIAP, 2004.

Annamaria Mesaros and Tuomas Virtanen. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010a.

Annamaria Mesaros and Tuomas Virtanen. Recognition of phonemes and words in singing. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2010b.

Gabriel Meseguer-Brocal and Geoffroy Peeters. Content based singing voice source separation via strong conditioning using aligned phonemes. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

Gabriel Meseguer-Brocal, Alice Cohen-Hadria, and Geoffroy Peeters. DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

Gabriel Meseguer-Brocal, Rachel Bittner, Simon Durand, and Brian Brost. Data cleansing with contrastive learning for vocal note event annotations. *arXiv preprint arXiv:2008.02069*, 2020.

Nima Mesgarani, Samuel Thomas, and Hynek Hermansky. A multistream multiresolution framework for phoneme recognition. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 2002.

Todd K Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.

Omar Caballero Morales and Stephen Cox. Modelling confusion matrices to improve speech recognition accuracy, with an application to dysarthric speech. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

Andrew Cameron Morris, Viktoria Maier, and Phil Green. From WER and RIL to MER and WIL: Improved evaluation measures for connected speech recognition. In *Eighth International Conference on Spoken Language Processing*, 2004.

In Jae Myung. The importance of complexity in model selection. *Journal of Mathematical Psychology*, 44(1):190–204, 2000.

Adrian North and David Hargreaves. *The Social and Applied Psychology of Music*. OUP Oxford, 2008.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: An empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. Fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.

Caroline Palmer and Michael H Kelly. Linguistic prosody and musical meter in song. *Journal of memory and language*, 31(4):525–542, 1992.

Jing Pan, Joshua Shapiro, Jeremy Wohlwend, Kyu J Han, Tao Lei, and Tao Ma. ASAPP-ASR: Multistream CNN and self-attentive SRU for SOTA speech recognition. In *Interspeech*, 2020.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.

Hilman F Pardede, Asri R Yuliani, and Rika Sustika. Convolutional neural network and feature transformation for distant speech recognition. *International Journal of Electrical & Computer Engineering (2088-8708)*, 8(6), 2018.

Bernardino Romera Paredes, Andreas Argyriou, Nadia Berthouze, and Massimiliano Pontil. Exploiting unrelated tasks in multi-task learning. In *Artificial Intelligence and Statistics*, pages 951–959. PMLR, 2012.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

Daniel S Park, Yu Zhang, Ye Jia, Wei Han, Chung-Cheng Chiu, Bo Li, Yonghui Wu, and Quoc V Le. Improved noisy student training for automatic speech recognition. *arXiv preprint arXiv:2005.09629*, 2020.

Vishal Passricha and Rajesh Kumar Aggarwal. *Convolutional Neural Networks for Raw Speech Recognition*. IntechOpen, 2018.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *16th Annual Conference of the International Speech Communication Association*, 2015.

Terry F Pettijohn and Donald F Sacco Jr. The language of lyrics: An analysis of popular Billboard songs across conditions of social and economic threat. *Journal of Language and Social Psychology*, 28(3):297–311, 2009a.

Terry F Pettijohn and Donald F Sacco Jr. Tough times, meaningful music, mature performers: Popular Billboard songs and performer preferences across social and economic conditions in the USA. *Psychology of Music*, 37(2):155–179, 2009b.

Sarah F Poissant, Nathaniel A Whitmal III, and Richard L Freyman. Effects of reverberation and masking on speech intelligibility in cochlear implant simulations. *The Journal of the Acoustical Society of America*, 119(3):1606–1615, 2006.

Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah. Boosted MMI for model and feature-space discriminative training. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4057–4060. IEEE, 2008.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, and Petr Schwarz. The Kaldi speech recognition toolkit. In *IEEE - Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for ASR based on lattice-free MMI. In *Interspeech*, 2016.

Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Interspeech*, 2018a.

Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur. A time-restricted self-attention layer for ASR. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5874–5878. IEEE, 2018b.

Kaila C Putter, Amanda E Krause, and Adrian C North. Popular music lyrics and the covid-19 pandemic. *Psychology of Music*, 2021.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., 1993.

Fika Hastarita Rachman, Riyanarto Sarno, and Chastine Fatichah. Music emotion classification based on lyrics-audio using corpus based emotion. *International Journal of Electrical & Computer Engineering*, 2018.

Kanishka Rao and Haşim Sak. Multi-accent speech recognition with hierarchical grapheme based models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4815–4819. IEEE, 2017.

Timothy Rasinski. Reading fluency instruction: Moving beyond accuracy, automaticity, and prosody. *The Reading Teacher*, 59(7):704–706, 2006.

Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.

Mirco Ravanelli, Piergiorgio Svaizer, and Maurizio Omologo. Realistic multi-microphone data simulation for distant speech recognition. *arXiv preprint arXiv:1711.09470*, 2017.

Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, et al. SpeechBrain: A general-purpose speech toolkit. *arXiv preprint arXiv:2106.04624*, 2021.

Peter Roach. British English: Received pronunciation. *Journal of the International Phonetic Association*, 34(2):239–245, 2004.

Ronald Rosenfeld. *Adaptive Statistical Language Modeling; A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, 1994.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, Technical Report, University of California, San Diego, La Jolla Institution for Cognitive Science, 1985.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

David Rybach, Stefan Hahn, Patrick Lehnen, David Nolden, Martin Sundermeyer, Zoltan Tüske, Simon Wiesler, Ralf Schlüter, and Hermann Ney. RASR-the RWTH Aachen University open source speech recognition toolkit. In *IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

David Saad. Online algorithms and stochastic approximations. *Online Learning*, 5:6–3, 1998.

Tara N Sainath, Yanzhang He, Bo Li, Arun Narayanan, Ruoming Pang, Antoine Bruguier, Shuo-yiin Chang, Wei Li, Raziel Alvarez, Zhifeng Chen, et al. A streaming on-device end-to-end model surpassing server-side conventional model quality and latency. In *IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6059–6063. IEEE, 2020.

Lahiru Samarakoon, Brian Mak, and Albert YS Lam. Domain adaptation of end-to-end speech recognition in low-resource settings. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 382–388. IEEE, 2018.

George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013.

George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. *arXiv preprint arXiv:1703.02136*, 2017.

Shin-ichi Sato and Nicola Prodi. On the subjective evaluation of the perceived balance between a singer and a piano inside different theatres. *Acta Acustica united with Acustica*, 95(3): 519–526, 2009.

Gottfried Schlaug, Andrea Norton, Sarah Marchina, Lauryn Zipse, and Catherine Y Wan. From singing to speaking: Facilitating recovery from nonfluent aphasia. *Future Neurology*, 2010.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

Kilian Schulze-Forster, Clement SJ Doire, Gaël Richard, and Roland Badeau. Joint phoneme alignment and text-informed speech separation on highly corrupted speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

Mike Schuster and Kaisuke Nakajima. Japanese and Korean voice search. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE, 2012.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45:2673–2681, 1997.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. *arXiv preprint arXiv:1810.13327*, 2018.

Bidisha Sharma, Xiaoxue Gao, Karthika Vijayan, Xiaohai Tian, and Haizhou Li. Nhss: A speech and singing parallel database. *Speech Communication*, 133:9–22, 2021.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. Lingvo: A modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*, 2019.

Yusuke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, and Takeshi Shinohara. Byte pair encoding: A text compression scheme that accelerates pattern matching. Technical report, Technical Report DOI-TR-CS-161, Department of Informatics, Kyushu University, 09 1999a.

Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. Byte pair encoding: A text compression scheme that accelerates pattern matching. 1999b.

Joonbo Shin, Yoonhyung Lee, and Kyomin Jung. Effective sentence scoring method using BERT for speech recognition. In *Asian Conference on Machine Learning*, pages 1081–1093. PMLR, 2019.

Prashanth Gurunath Shivakumar and Panayiotis Georgiou. Transfer learning from adult to children for speech recognition: Evaluation, analysis and recommendations. *Computer Speech & Language*, 63:101077, 2020.

Daniel Silverman. Schwa. *The Blackwell Companion to Phonology*, pages 1–15, 2011.

L Robert Slevc. Language and music: Sound, structure, and meaning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2012.

Lloyd A Smith and Brian L Scott. Increasing the intelligibility of sung vowels. *The Journal of the Acoustical Society of America*, 67(5):1795–1797, 1980.

Janne Spijkervet and John Ashley Burgoyne. Contrastive learning of musical representations. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*, 2017.

Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. Improvements in beam search. In *3rd International Conference on Spoken Language Processing*, 1994.

Andreas Stolcke. SRILM - an extensible language modeling toolkit. In *7th International Conference on Spoken Language Processing*, 2002.

Daniel Stoller, Simon Durand, and Sebastian Ewert. End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

Valerie N Stratton and Annette H Zalanowski. Affective impact of music vs. lyrics. *Empirical Studies of the Arts*, 12(2):173–184, 1994.

Johan Sundberg. The acoustics of the singing voice. *Scientific American*, 236(3):82–91, 1977.

Johan Sundberg. Acoustic and psychoacoustic aspects of vocal vibrato. *Vibrato*, pages 35–62, 1995.

Johan Sundberg. The singing voice. *The Oxford Handbook of Voice Perception*, page 117, 2018.

Johan Sundberg and Camilla Romedahl. Text intelligibility and the singer's formant — a relationship? *Journal of Voice*, 2009.

Johan Sundberg and Thomas D Rossing. *The Science of Singing Voice*. Acoustical Society of America, 1990.

Martin Sundermeyer, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. Lattice decoding and rescoring with long-span neural network language models. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

Pawel Swietojanski, Arnab Ghoshal, and Steve Renals. Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 246–251. IEEE, 2012.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

Mikolov Tomas, Deoras Anoop, Kombrink Stefan, Burget Lukas, and H Gernocky Jan. Rnnlm-recurrent neural network language modeling toolkit. In *IEEE Automatic Speech Recognition and Understanding Workshop*, 2011.

Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. A deeper look at dataset bias. In *Domain Adaptation in Computer Vision Applications*, pages 37–55. Springer, 2017.

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 2011.

Joan Torruella and Ramón Capsada. Lexical statistics and tipological structures: A measure of lexical richness. *Procedia-Social and Behavioral Sciences*, 95:447–454, 2013.

Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, Yonghui Wu, Tara N Sainath, and Karen Livescu. A comparison of techniques for language model integration in encoder-decoder speech recognition. In *IEEE Spoken Language Technology Workshop*, pages 369–375. IEEE, 2018.

Zoltán Tüske, George Saon, and Brian Kingsbury. On the limit of english conversational speech recognition. *arXiv preprint arXiv:2105.00982*, 2021.

Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gael Richard, and Florence d'Alché Buc. Multilingual lyrics-to-audio alignment. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

Andrea Vaglio, Romain Hennequin, Manuel Moussallam, and Gael Richard. The words remain the same: Cover detection with lyrics transcription. In *22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

Robert W Van Sickel. A world without citizenship: On (the absence of) politics and ideology in country music lyrics, 1960–2000. *Popular Music and Society*, 28(3):313–331, 2005.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey. Sequence-discriminative training of deep neural networks. In *Interspeech*, 2013.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

Dong Wang and Thomas Fang Zheng. Transfer learning for speech and language processing. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1225–1237. IEEE, 2015.

Xiaofei Wang, Ruizhi Li, Sri Harish Mallidi, Takaaki Hori, Shinji Watanabe, and Hynek Hermansky. Stream attention-based multi-array end-to-end speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7105–7109. IEEE, 2019a.

Yiming Wang, Tongfei Chen, Hainan Xu, Shuoyang Ding, Hang Lv, Yiwen Shao, Nanyun Peng, Lei Xie, Shinji Watanabe, and Sanjeev Khudanpur. Espresso: A fast end-to-end neural speech recognition toolkit. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 136–143. IEEE, 2019b.

Yongqiang Wang, Abdelrahman Mohamed, Due Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, et al. Transformer-based acoustic modeling for hybrid speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6874–6878. IEEE, 2020.

Kento Watanabe and Masataka Goto. Query-by-blending: A music exploration system blending latent vector representations of lyric word, song audio, and artist. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al. Espnet: End-to-end speech processing toolkit. *arXiv preprint arXiv:1804.00015*, 2018.

Robert Weide. The CMU pronunciation dictionary, release 0.6. Technical report, Carnegie Mellon University Pittsburgh, PA, 1998. URL `http://www.speech.cs.cmu.edu`.

Chao Weng and Dong Yu. A comparison of lattice-free discriminative training criteria for purely sequence-trained neural network acoustic models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6430–6434. IEEE, 2019.

Felix Weninger, Martin Wöllmer, and Björn Schuller. Automatic assessment of singer traits in popular music: Gender, age, height and race. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

Su-Lin Wu, Michael L Shire, Steven Greenberg, and Nelson Morgan. Integrating syllable boundary information into speech recognition. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 987–990. IEEE, 1997.

Zhangyu Xiao, Zhijian Ou, Wei Chu, and Hui Lin. Hybrid CTC-attention based end-to-end speech recognition using subword units. In *11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 146–150. IEEE, 2018.

Feifei Xiong, Jon Barker, Zhengjun Yue, and Heidi Christensen. Source domain data selection for improved transfer learning targeting dysarthric speech recognition. In *IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7424–7428. IEEE, 2020.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Michael L Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Toward human parity in conversational speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(12):2410–2423, 2017.

Hainan Xu, Tongfei Chen, Dongji Gao, Yiming Wang, Ke Li, Nagendra Goel, Yishay Carmiel, Daniel Povey, and Sanjeev Khudanpur. A pruned RNNLM lattice-rescoring algorithm for automatic speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018a.

Hainan Xu, Ke Li, Yiming Wang, Jian Wang, Shiyin Kang, Xie Chen, Daniel Povey, and Sanjeev Khudanpur. Neural network language modeling with letter-based features and importance sampling. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018b.

Zhen Yang, Qichen Han, Xiang Li, Dong Liu, and Peng Li. Technical report: Lyrics transcription for MIREX 2021. In *MIREX 2021: Lyrics Transcription Challenge*, 2021.

Jiangyan Yi, Jianhua Tao, Zhengqi Wen, and Ye Bai. Adversarial multilingual training for low-resource speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4899–4903. IEEE, 2018.

Emre Yilmaz and Joris Pelemans. Automatic assessment of children's reading with the flavor decoding using a phone confusion model. *Interspeech*, pages 969–972, 2014.

Steve Young. The HTK hidden markov model toolkit: Design and philosophy. Technical report, Technical Report, University of Cambridge, Department of Engineering Cambridge, 1993.

Steve J Young, Julian J Odell, and Phil C Woodland. Tree-based state tying for high accuracy modelling. In *Human Language Technology*, 1994.

Neil Zeghidour, Qiantong Xu, Vitaliy Liptchinsky, Nicolas Usunier, Gabriel Synnaeve, and

Ronan Collobert. Fully convolutional speech recognition. *arXiv preprint arXiv:1812.06864*, 2018.

Albert Zeyer, Tamer Alkhouli, and Hermann Ney. RETURNN as a generic flexible neural toolkit with application to translation and speech recognition. *arXiv preprint arXiv:1805.05225*, 2018.

Chen Zhang, Jiaxing Yu, LuChin Chang, Xu Tan, Jiawei Chen, Tao Qin, and Kejun Zhang. Pdaugment: Data augmentation by pitch and duration adjustments for automatic lyrics transcription. *arXiv preprint arXiv:2109.07940*, 2021.

Sherry Y Zhao and Nelson Morgan. Multi-stream spectro-temporal features for robust speech recognition. In *9th Annual Conference of the International Speech Communication Association*, 2008.