THE UNIVERSITY
of EDINBURGH

Robust learning of acoustic representations from diverse speech data

Joachim Fainberg

Doctor of Philosophy
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
2020

Abstract

Automatic speech recognition is increasingly applied to new domains. A key challenge is to robustly learn, update and maintain representations to cope with transient acoustic conditions. A typical example is broadcast media, for which speakers and environments may change rapidly, and available supervision may be poor. The concern of this thesis is to build and investigate methods for acoustic modelling that are robust to the characteristics and transient conditions as embodied by such media.

The first contribution of the thesis is a technique to make use of inaccurate transcriptions as supervision for acoustic model training. There is an abundance of audio with approximate labels, but training methods can be sensitive to label errors, and their use is therefore not trivial. State-of-the-art semi-supervised training makes effective use of a lattice of supervision, inherently encoding uncertainty in the labels to avoid overfitting to poor supervision, but does not make use of the transcriptions. Existing approaches that do aim to make use of the transcriptions typically employ an algorithm to filter or combine the transcriptions with the recognition output from a seed model, but the final result does not encode uncertainty. We propose a method to combine the lattice output from a biased recognition pass with the transcripts, crucially preserving uncertainty in the lattice where appropriate. This substantially reduces the word error rate on a broadcast task.

The second contribution is a method to factorise representations for speakers and environments so that they may be combined in novel combinations. In realistic scenarios, the speaker or environment transform at test time might be unknown, or there may be insufficient data to learn a joint transform. We show that in such cases, factorised, or independent, representations are required to avoid deteriorating performance. Using i-vectors, we factorise speaker or environment information using multi-condition training with neural networks. Specifically, we extract bottleneck features from networks trained to classify either speakers or environments. The resulting factorised representations prove beneficial when one factor is missing at test time, or when all factors are seen, but not in the desired combination.

The third contribution is an investigation of model adaptation in a longitudinal setting. In this scenario, we repeatedly adapt a model to new data, with the constraint that previous data becomes unavailable. We first demonstrate the effect of such a constraint, and show that using a cyclical learning rate may help. We then observe that these successive models lend themselves well to ensembling. Finally, we show that the impact of this constraint in an active learning setting may be detrimental to performance, and suggest to combine active learning with semi-supervised training to

avoid biasing the model.

The fourth contribution is a method to adapt low-level features in a parameter-efficient and interpretable manner. We propose to adapt the filters in a neural feature extractor, known as SincNet. In contrast to traditional techniques that warp the filterbank frequencies in standard feature extraction, adapting SincNet parameters is more flexible and more readily optimised, whilst maintaining interpretability. On a task adapting from adult to child speech, we show that this layer is well suited for adaptation and is very effective with respect to the small number of adapted parameters.

Lay Summary

Automatic speech recognition is enjoying increasingly widespread use. Smart assistants have placed this technology in the hands of millions of users. There are still particular situations for which the performance leaves something to be desired. Some examples of challenging data includes child speech and broadcast data. What makes this data challenging is the variation in acoustics over time, speakers speaking over each other, background noise, and much more. Additionally, to build a speech recognition system, we need large amounts of speech data with all the speech transcribed. Inaccurate transcription is another source of noise to the system. This thesis concerns the learning of the inner representations for a model in a robust fashion such that they are robust to the variations and inaccuracies present in such data, to ultimately improve the accuracy of the system.

In the thesis we first look at how we may handle inaccurate transcriptions. We usually can not train a system directly on such transcriptions: this may deteriorate the system because the transcriptions do not always match the audio and the model would learn the wrong words for certain speech. However, some of the words in the transcription are likely to be correct, and since procuring new transcriptions is expensive, we would like to make the most of the data that we have available. If we have an existing system trained on good data, we can use that model in combination with the transcriptions that we know to be inaccurate, to create new training data. This new data includes information of where we should expect the transcriptions to be incorrect, and where we should expect them to be correct. With such information, we can train a new system robustly, because it will pay more attention to the part of the data that we consider correct, and less attention to the rest. Using this technique we show considerable improvements in accuracy on broadcast data, training on subtitles.

Another source of mismatch is when a system is used on a new speaker or environment that the model did not see in its training data. In this thesis we consider especially the interaction of speakers and environments. For example, it may be that we have seen a speaker and an environment previously, but, crucially, never in a particular combination. That is, if we have learned something about a speaker in a different environment than what we observe at test time, we may see reduced performance because there is a mismatch in the environments. We propose a method to separate speakers from their environments, and environments from their speakers, also known as factorisation. In this way speakers and environments can occur together in any combination at test time, without a loss of performance. We show that our method can improve the accuracy of a system in some challenging scenarios.

In some cases data may not be available all at once. In particular, if we consider TV series, we may expect a system to improve for every episode that is broadcast. Traditionally, we would build a new system using all the broadcast episodes thus far. However, if those previous episodes are unavailable, e. g. for copyright reasons, then we are limited to updating the current model to individual episodes one at a time. We call this scenario longitudinal learning. This scenario has implications for how we may update a model and the resulting performance. We study several aspects of this situation, and look both at how we may improve models by combining multiple models over time, and how we may incorporate a small amount of carefully transcribed data for each episode to further improve the performance.

Lastly, we study a new way to update a system to new speakers or new domains. We identify the case of child speech, for which we usually do not have much data, but for which the acoustic properties are much different from adult speech. Traditionally, this mismatch has been corrected for by using physiologically motivated preprocessing of the data that is fed to the learning algorithm. There is, however, an increasing interest in avoiding separate processing steps, and rather have the model as a whole learn how to best make use of the data. We study a model that takes raw audio directly as input in an efficient manner, and we find that we can obtain large improvements in accuracies on child speech data, having started from a model trained on adult speech. This model is further designed in such a way that we can interpret what it learns, and we observe that it seems to control for the key acoustic differences between child and adult speakers, namely vocal tract length.

Acknowledgments

I am tremendously grateful to my supervisors, Steve Renals and Peter Bell, for providing a relaxed, accommodating, and extremely knowledgeable research environment in which I have had the freedom to pursue my research interests whilst learning from their insight and guidance. I would like to thank Sharon Goldwater for her insightful feedback in my yearly reviews. I would also like to thank my examiners, Thomas Hain and Catherine Lai, for an interesting and challenging viva that can only have improved the final result herein. I am grateful to Bloomberg for funding my PhD and for my contact Gary Kazantsev.

The Centre for Speech Technology Research (CSTR) has been a wonderful place to work, and in ways a second home over many years, solely thanks to the wonderful people therein and its visitors. The daily lunches and semi-regular cake events were always a great time, and my random encounters in the corridors with Aciel Eshky, Erfan Loweimi, Sam Ribeiro, Mirjam Wester or Gustav Henter always either left me laughing or pondering upon some great advice. I would like to thank my fellow PhD students in whose great company I've shared this journey, Marco Damonte, Ondřej Klejch, Sameer Bansal, Joanna Równicka, Mihai Sorin Dobre, Abhirup Gosh, Ben Krause, Craig Innes, Joana Ribeiro, and many more. Special thanks go to Ondřej for our daily apple breaks, and to Marco for enduring living with me for so many years. Mike Lincoln and Nick Rankin from Quorate Technology have provided wonderful support and encouragement in my every interaction with them. I would also like to extend a thank you to Aleksandra, Margaret and Joe, part of the wonderful staff of the Forum who greeted me practically every day for five years and made me feel so welcome.

Outside of work I have enjoyed the ability to switch contexts with friends from Ascham Court turned Brew Crew: Meagan, Will, Robin, and Rachel. I am grateful to my English Tonmeister friends, Daniel and Tashi, who came to visit me in Scotland several times, as well as my friends back home in Norway who have not left me despite emigrating for more than 10 years, Rune, Harald, Charlotte, and Siv. During my PhD I have also had the pleasure of two internships during which I have learned an awful lot from wonderful mentors in Daniele Giacobello (Sonos) and Ahmad Emami (Bloomberg) - and I have picked up some fine friends along the way in Jennie and Geoff.

Finally, mom, dad, and my sister Julie, who have provided me with so much love and encouragement throughout sometimes difficult years: Tusen takk for uttallige, betryggende FaceTime samtaler og fine besøk, jeg kunne ikke fullført uten dere. And last but not least to Lillepus, whose carefree and happy demeanour (being a cat) always put my problems into perspective.

Contents

Chapter 1

Introduction

Automatic Speech Recognition (ASR) systems have in the last few years obtained impressive accuracies in many real-world tasks. Improvements to acoustic and language modelling, hardware-acceleration of neural networks, and the ability to leverage increasingly larger amounts of data, have enabled companies to provide end-user ASR systems that enjoy widespread use. Examples include personal assistants such as Apple's Siri, Google Assistant, and Amazon Alexa which are becoming ubiquitous within mobile phones and other smart devices. Many of these applications provide constrained environments in which one or more factors are predictable. Smart devices are typically used by few speakers, sometimes also in a fixed environment (consider home products), and there are a few extremely common voice requests that can help language modelling and downstream natural-language understanding tasks[1]. This contributes to their success in using ASR.

There are many remaining challenges in large-vocabulary speech recognition that surface in less constrained conditions and with more diverse data. Mismatch between training and test time conditions can have detrimental effects on recognition performance, such as changing speaker characteristics, environments or background noise (Barker et al., 2013; Vincent et al., 2013). Adding distance or reverberation to the captured audio exacerbates those existing challenges (Barker et al., 2018; Carletta, 2007; Vincent et al., 2017). For certain domains we may have access to only a limited amount of data, yet the inherent diversity of the domain usually requires more data in comparison to more homogeneous domains. Consider for example the acoustic and linguistic properties of child speech, with varying vocal tract lengths and pronunciation mistakes (Lee, Potamianos, and Narayanan, 1999; Shivakumar et al., 2014).

A medium that inherently embodies many of these challenges is broadcast data, which is very diverse and often highly unpredictable. The data may contain overlapping speech from multiple speakers of all ages and accents, across many genres and domains. Each domain may exhibit its own idiosyncrasies, such as news media, where topics may change fast, along with changing background noise between a studio and reporting from the field. In drama or children's shows, the vocal characteristics of the speakers may vary dramatically. The Multi-Genre Broadcast (MGB) challenge (Bell et al., 2015)

[1] On iOS 13.1, Apple's Siri will – in my own limited testing – provide the weather forecast for any three-word utterance, as long as the word "weather" is correctly transcribed.

reflects the difficulty of this diverse data, with the best submitted system obtaining a Word Error Rate (WER) of 21.8% after considerable engineering efforts, and the combination of multiple systems (Woodland et al., 2015).

There are many growing strands of research attempting to tackle the challenges associated with such diverse data and to ultimately provide better ASR performance. Robust techniques are required to improve error rates for these situations, in spite of the difficulties they present. Fast and robust adaptation methods (Karanasou et al., 2014; Saz and Hain, 2017; Sim et al., 2018) counter rapidly changing acoustics. New approaches to data selection (Doulaty, Saz, and Hain, 2015; Lanchantin et al., 2016) can help select appropriate training data for the task at hand. Lightly- and semi-supervised techniques (Drugman, Pylkkönen, and Kneser, 2016; Lamel, Gauvain, and Adda, 2002; Manohar et al., 2018; Zavaliagkos et al., 1998) enable better usage of much real-world data.

The aim of this thesis is to build and study methods that robustly learn acoustic representations from diverse speech data. By robust learning we mean techniques that can yield representations that improve error rates in situations that normally would be detrimental to a speech recognition system. Specifically, how may we robustly obtain representations despite acoustic mismatch, whether that be due to incorrect supervision or a change of speaker and environment acoustics? In other words, we require both methods to improve or incorporate mismatched or inaccurate training data as well as suitable methods to update a model to such diverse and changing contexts. As we will see, training directly on inaccurate transcriptions, or using mismatched representations at test time may considerably increase error rates. Many of the characteristics and transient conditions embodied by broadcast media are applicable to this subject, and we will ground much of the discussion in broadcast media. The challenges of diverse data, however, occur more broadly, and throughout the thesis we may choose corpora that highlight a particular characteristic. The corresponding challenges we have chosen to address are motivated next.

1.1. Motivation and research questions

One of the key applications of ASR in the broadcast domain is to supply subtitles. Subtitling provides an additional information stream and an additional modality. It is used for public information where sound would otherwise be drowned out by noise, it is of crucial importance to the hard-of-hearing, and it enables search and categorisation. It may also help reading proficiency and content understanding (McCall and Craig, 2009; Perego et al., 2010). Studies have shown that, in the UK, a significant amount of viewership use subtitles – a large fraction of whom did not report having a hearing impairment (Ofcom, 2006; RNID, 2008). Advertising agencies report that up to 85% of videos on Facebook are watched without sound (Patel, 2016). In the US, a range of civil rights laws require subtitles in certain domains to ensure equal access to information.

Given the rapidly expanding amount of broadcast media, subtitling is an important application of ASR. It is infeasible to manually write transcriptions for all new content. However, as alluded to above, constructing a suitably high-performing model is not trivial. Broadcast media is highly varied, both in acoustic and linguistic content. One may first ask, how is training data obtained? It seems reasonable to be able to use existing subtitles from previously transcribed media. Yet, subtitles are not verbatim transcriptions of the speech, and label mismatch may negatively affect the resulting model. This leads to the first research question:

1. **Light supervision**. How may inaccurate transcripts be used effectively during training?

By effective we specifically mean that their use does not deteriorate model performance and that their inclusion improves error rates, exceeding semi-supervised methods which do not make use of the transcriptions. In other words, our method should be robust to inaccurate transcriptions, and improve error rates over the direct use of such transcriptions and over semi-supervised techniques. We develop a method to combine the transcriptions from subtitles with hypotheses from a seed model. In effect, the result is confidence weighted supervision given agreeing words from the two sources of supervision.

However, even if we had perfect supervision, we may still need to consider the context in which the data occurs. For example, much data will have a mixture of speakers and their environments. TV series may have a protagonist appearing in many changing environments, environments in which other characters may have been seen previously. Adaptation of an ASR model to a joint scenario, a speaker and an environment, can have dramatic effects on performance. However, the delay from adaptation may prove disruptive, and re-using representations extracted in mismatched conditions may increase error rates. It would be convenient to be able to combine existing knowledge across speakers and environments. This requires that speaker and environment representations are independent, or *factorised*:

2. **Factorised adaptation**. How can feature representations be factorised, such that they can be combined in novel combinations at test time?

There are existing approaches that obtain factorised representations using constrained optimisation. We instead make use of the ability of neural networks to learn latent representations that implicitly factor out irrelevant information to the task at hand. This yields a flexible approach which may factorise (in effect) any feature representation. Carefully constructed test-scenarios show that these new representations are robust to unseen speaker and environment combinations, improving error rates.

Changing contexts in the data is particularly evident when the data is time-limited, or ephemeral, which may be the case with growing privacy and copyright concerns (see e. g. Zimmeck et al., 2016). By this we mean that data from a domain that was available

for training today, may be unavailable tomorrow, replaced by new data. Models are often still expected to improve day by day, particularly in relatively constrained domains such as a running TV series or parliamentary hearings. We call this *longitudinal learning*:

3. **Longitudinal learning**. When data is ephemeral, how well does a model improve with continuous training, and what are the implications for active learning?

We first look at the difference longitudinal training makes, compared to the case of retraining on all data. We then study active learning in the longitudinal context, which is a special case of active learning in which it is not possible to pool newly obtained data with previous data. This has ramifications for how data is chosen for active learning. We demonstrate that a naive application of active learning leads to poor results, and instead present a robust, suitable methodology for enabling active learning in this setting.

When updating acoustic models to account for changing contexts, we may need to seek compact representations for practical and modelling reasons. For example, the acoustic variety in broadcast media is particularly pronounced with adult and child speech, requiring adaptation of the acoustic model. Additionally, child speech has particular privacy concerns: it may be necessary to adapt a background model repeatedly, with small amounts of data. For a large number of speakers, or on-device applications, this would need to be parameter-efficient. For unsupervised adaptation, it would need to be robust to errors in the supervision, specifically with such a large difference in acoustic traits between the speakers. We identify the following problem:

4. **Adaptation**. What is a parameter-efficient, robust way to adapt a model to new speaker acoustics, such as child speech?

We experiment with adapting a particular parameterisation of a neural feature extractor. This results in a highly parameter efficient adaptation method that nearly matches the performance of adapting several times more parameters in the supervised case, and which enables robust adaptation to first-pass targets. A side-effect is improved interpretability as the method acts on well-defined parameters.

1.2. Thesis outline

Chapter 2: Acoustic modelling. The main topics and techniques for acoustic modelling that will be used throughout the thesis are presented, starting with a concise review of hybrid acoustic models. Sequence-discriminative training is discussed. We then look at some common adaptation and normalisation techniques.

Chapter 3: Lattices for decoding and supervision. Lattices, encoded as Weighted Finite State Transducers (WFSTs), are core to modern ASR systems. Chapter 5 will manipulate lattices using techniques reviewed here. We discuss how lattices can be applied for supervision in systems trained with sequence-discriminative

criteria.

Chapter 4: Data. A variety of data is used throughout the thesis. Each corpus is presented, along with typical error rates from the literature. The notion of filtering with Matching Error Rates (MERs) is introduced for corpora where the majority of labels stem from subtitles, or are otherwise known to be inaccurate.

Chapter 5: Lightly-supervised training. We present an algorithm to handle inaccurate transcriptions when training acoustic models using sequence-discriminative criteria. The algorithm combines transcripts and hypothesis lattices to create improved lattice supervision. An extension using an external paraphrase database is further proposed. Experiments compare the algorithm with filtering methods using matching error rates, and all experiments are repeated with biased language models.

Chapter 6: Factorised representations. We propose a method to adapt to individual acoustic factors independently, enabling the re-combination of factors in any configuration. Multi-condition networks are used to create bottleneck features from i-vectors with opposing factors implicitly factored out. Experiments demonstrate the importance of factorisation in the case of mismatch during i-vector extraction.

Chapter 7: Longitudinal training. We explore how to adapt and train in a longitudinal fashion, when previous data has to be discarded. We first experiment with model combination across time, using both posterior ensembling and weight averaging. We then experiment with the effect of confidence-based selection for active learning in the longitudinal framework.

Chapter 8: Adaptation with raw waveform acoustic models. We explore the adaptation of raw-waveform acoustic models, and suggest to adapt a particular parameterisation of a raw waveform filterbank, known as SincNet (Ravanelli and Bengio, 2018). The technique is shown to share similarities with traditional techniques to normalise for vocal tract length, but also with other well-known adaptation methods. Adaptation experiments from adult to child speech demonstrate interpretable adaptation transforms that are parameter efficient.

Chapter 9: Conclusions. We conclude the thesis, and present ideas for future work.

1.3. Published work

The core idea of Chapter 5 on lattice-based lightly supervised training is based on a paper presented at *Interspeech 2019* (Fainberg et al., 2019b). Chapter 6 on factorised

adaptation is largely based on work published at *Interspeech 2017* (Fainberg, Renals, and Bell, 2017). Lattice-based adaptation results in Chapter 3 form a small part of an *arXiv* publication on unsupervised adaptation (Klejch et al., 2019). The main idea and results in Chapter 8 on adaptation with raw waveform acoustic models were presented at *ASRU 2019* (Fainberg et al., 2019a).

Chapter 2

Acoustic modelling

The goal of Automatic Speech Recognition (ASR) is to determine the most likely sequence of words spoken, $\mathbf{W}^*$, given some acoustic observations, $\mathbf{O}$:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{O}). \tag{2.1}$$

This posterior can be estimated directly in models known as end-to-end (Chorowski et al., 2014; Graves and Jaitly, 2014). More traditionally, the problem has been factored into an Acoustic Model (AM) and a Language Model (LM) using Bayes' rule:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{O}) = \arg\max_{\mathbf{W}} \underbrace{P(\mathbf{O} \mid \mathbf{W})}_{\text{AM}} \underbrace{P(\mathbf{W})}_{\text{LM}}. \tag{2.2}$$

The factorisation into two dedicated models is useful from a research perspective, for which unrelated strands of research can easily be incorporated. The modularity is also useful from a practical perspective, since the individual models can impose known structure, such as the pronunciation of words, and the models may be updated and customised independently of each other. It is, for example, typical to have much more text data to train a language model, than corresponding audio for the acoustic model.

On the other hand, end-to-end models that estimate the posterior directly may find useful information that otherwise would not be shared across modules, and they may be easier to understand and to put on devices. It can also be argued that such models can learn any required structure from sufficient amounts of data[1]. While theory and implementation can differ considerably, the two methodologies can both produce excellent systems. On the Librispeech corpus (Panayotov et al., 2015), the current Kaldi recipe[2] obtains 8.76% on the more difficult `test-other` test set with a factored Time-Delay Neural Network (TDNN) model and the LF-MMI criterion (Povey et al., 2018, 2016). A character level end-to-end architecture (Chan et al., 2016) with attention in recurrent networks obtains 6.8% (5.8% using a LM) with significant data augmentation (Park et al., 2019). With a transformer architecture as a hybrid acoustic model and a neural language model, Wang et al. (2020) obtain 4.85%. On the Switchboard corpus (Godfrey,

[1]The author of Kaldi, Dan Povey, warns however that "by taking the structure out of the system, the fairy dust of neural networks will improve the performance, but I think that's a mirage." (Kincaid, 2018).

[2]See `Kaldi-Libri` in Appendix A for a model description.

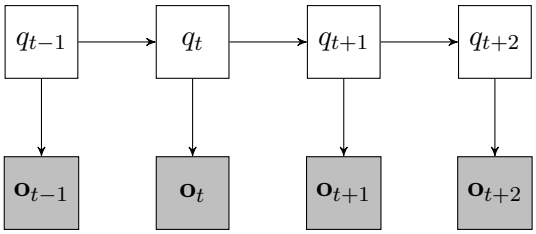


Figure 2.1: Visualisation of HMMs for speech recognition as a graphical model. Grey nodes indicate observed variables. This representation encodes independence between the random variables, in contrast to an alternative representation using automata which shows traversals through individual states. Independence relations can be observed directly using e. g. the Bayes' ball algorithm (Schachter, 1998).

Holliman, and McDaniel, 1992), Kaldi's recipe[3] using a TDNN+LSTM model and 4-gram rescoring obtains 8.8% WER on the Switchboard portion of the `eval2000` test set. In comparison, the character level end-to-end architecture with data augmentation obtains 7.2% (6.8% with an LM).

This thesis concerns how to robustly estimate and adapt acoustic models in the modular framework, specifically hybrid acoustic models which will be reviewed below. But the ideas discussed herein can also be applied to end-to-end systems.

2.1. HMM acoustic models

The most dominant model structure for ASR since the late 1980s has been Hidden Markov Models (HMMs) with either Gaussian Mixture Models (GMMs) or Deep Neural Networks (DNNs) to model the observation distributions. This model structure uses HMMs with associated densities as a generative model of speech, $p(\mathbf{O} \mid \mathbf{Q})$, given some underlying state sequence $\mathbf{Q} = [q_1, \ldots, q_T]$. The properties of HMMs emerge from two conditional probability assumptions that in turn provide tractable and efficient algorithms for training and inference. These relations can be visualised with a graphical model, shown in Figure 2.1. We use rectangular boxes to distinguish it from the automata used elsewhere in the thesis. Conditioning on the present state, the future is independent of the past given the present, and the corresponding output distribution is independent of all other states and observations.

To parameterise the HMM we require transition probabilities and emission densities. The transition probabilities from state i to state j are written $a_{i,j} = P(q_t = j \mid q_{t-1} = i)$. The emission density (assuming continuous observations) for state j at time t is written $b_j(\mathbf{o}_t) = P(\mathbf{o}_t \mid q_t = j)$. See e. g. Poritz (1988) or Bilmes (2006) for more in-depth treatments of HMMs and the algorithms for training and inference.

The standard HMM topology used in ASR is to model a phone, or similar, by a three-state HMM that only allows transitions to the next state, as well as self-loops. This is shown as an automaton in Figure 2.2. The restricted set of transitions implies

[3] See `Kaldi-SWBD` in Appendix A.

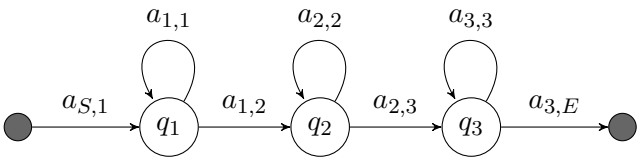


Figure 2.2: Typical three-state HMM left-to-right model used to model phones.

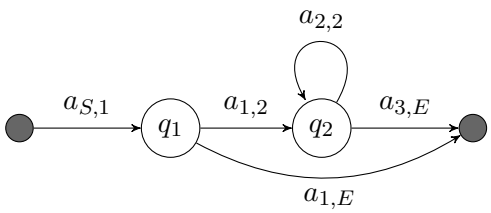


Figure 2.3: Two-state HMM with skip connection used in one-third frame-rate models.

a sparse transition matrix $\mathbf{A}$. The three states model the beginning, middle and end of the acoustic unit. Co-articulation effects are captured by using context-dependent phones (triphones).

More unconventional topologies have recently been experimented with as as consequence of using one-third frame-rate modelling to speed up decoding (as used in Lattice-Free MMI (LF-MMI) discussed below in Section 2.3.1). Features from three adjacent time-steps are spliced at the input and processed only once (Sak et al., 2015a). Using standard frame lengths of 10ms, the minimum duration of a three-state HMM is 30ms. Modelling with a one third frame-rate requires being able to traverse the triphone in a single (effectively 30ms) frame. Hadian et al. (2018c) experimented with a range of topologies, but converged upon the 2-stage skip model shown in Figure 2.3 which is also used in the original work on LF-MMI (Povey et al., 2016).

The amount of time spent in a triphone HMM model has been discussed in-depth previously for HMM-GMM systems. We revisit the discussion here in light of the new topologies in modern models. A common criticism of HMMs has been that the probability of the time spent in a single state is geometric in time with respect to its self-loop probability, $a_{j,j}$ (Gales and Young, 2008)[4]. In practice, however, a side-effect of a multiple state model is a more realistic duration distribution, particularly if states are tied as a consequence of clustering. In this case, the sum of geometrically distributed variables becomes a negative binomial distribution (Bilmes, 2006), as shown in Figure 2.4. Interestingly, the skip-connection in the two-state topology results again in a geometric duration distribution. However, Gales and Young (2008) noted that duration modelling becomes less important with improved output distributions. Indeed, Hadian et al. (2018c) saw only relatively minor increases in WER when experimenting with a topology consisting of just a single state (with a self-loop).

[4]There has been a history of work in HMM duration modelling dating back to Ferguson (1980) and Levinson (1986). See also hidden semi-Markov models (Russell and Moore, 1985).

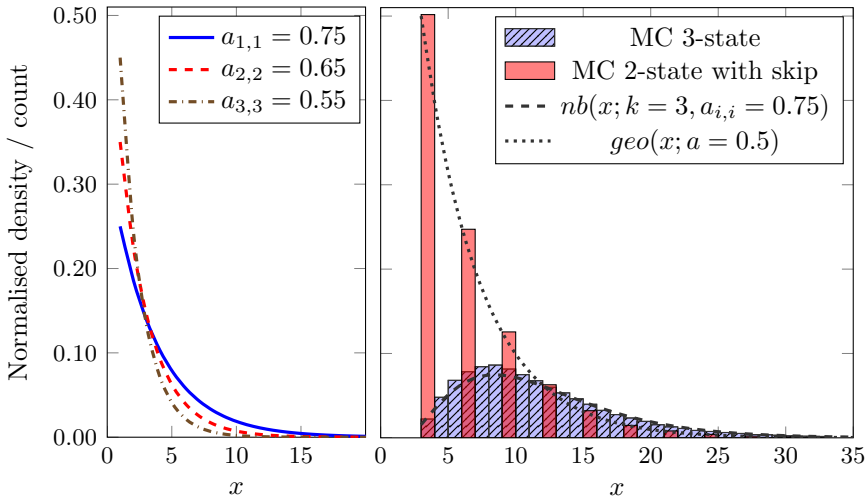


Figure 2.4: Probability of remaining in a state, $a_{i,i}^{t-1}(1-t)$, with self-transition probability $a_{i,i}$ (left); Monte Carlo estimate (right) of duration in three and two-state HMMs. The two-state skip HMM has all transition probabilities set to 0.5 by standard practice, and those for the three-state model were extracted from the position-dependent phone `AY_B`. The transition probabilities were extracted from monophone models trained on AMI as used in Chapter 8. The negative binomial distribution corresponds to three states each with a self-loop probability 0.75; as is standard for initialisation with Kaldi. The geometric distributions correspond to the duration of a single HMM state with a self-loop.

2.1.1. Gaussian Mixture Models

Mixtures of Gaussian distributions may be used as the emission densities, $b_j(O_t)$, of the HMM states, as depicted in Figure 2.5. All models used in later experiments will use alignments obtained from an HMM-GMM system, and the following review will highlight standard components in the pipeline of Kaldi (Povey et al., 2011), used throughout the thesis.

Gaussian mixtures are required because the data is often multi-modal – particularly lower Mel-Frequency Cepstral Coefficients (MFCCs), shown in Figure 2.6. Diagonal covariance matrices are used to avoid a dramatic expansion of parameters to estimate. This in turn requires uncorrelated features, for which the final transform in MFCC extraction by design is the decorrelating Discrete Cosine Transform (DCT). Linear Discriminant Analysis (LDA) is normally applied to spliced features (concatenated frames) to further decorrelate and reduce dimensionality (Batlle, Nadeu, and Fonollosa, 1998; Brown, 1987). Diagonal covariance GMMs are still able to model covariance, although perhaps ineffectively (Axelrod et al., 2005) and at the expense of their modelling capacity for multi-modal feature distributions (Gales, 1999). A model-space transform known as semi-tied covariance modelling, or Maximum Likelihood Linear Transform (MLLT), is therefore used to share one or more full covariance matrices across all GMM components (Gales, 1999).

The models are (in Kaldi) trained using Viterbi training as an approximation to full

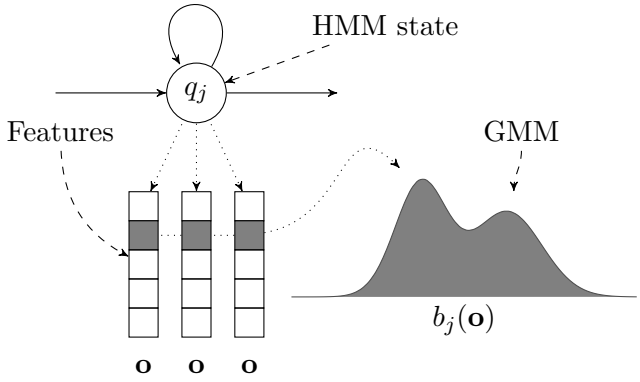


Figure 2.5: GMMs can be used to model the distribution of feature vectors aligned to a particular state j, here represented by the emission density $b_j(\mathbf{o})$.

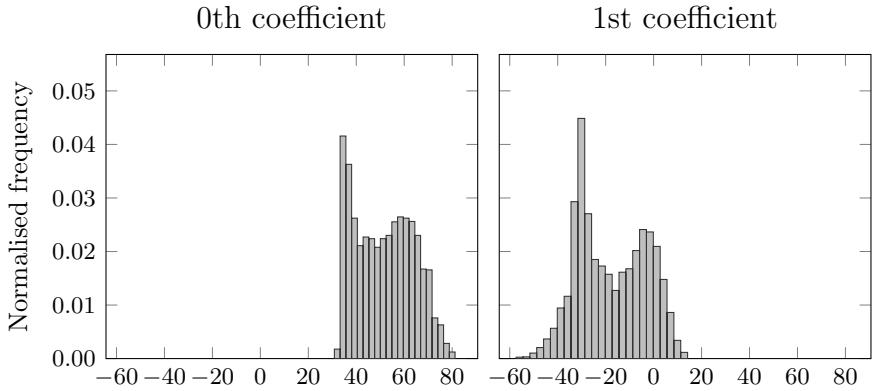


Figure 2.6: Distribution of the first two MFCCs from the TIMIT (Garofolo et al., 1993) training data set for frames aligned to the first HMM state representing the phone `/dh/`.

Baum-Welch (Baum et al., 1970). Later stages use Speaker-Adaptive Training (SAT) (Anastasakos et al., 1996), which folds in speaker adaptation (in this case Constrained Maximum Likelihood Linear Regression (CMLLR), see Section 2.4) during training, allowing the model to focus on modelling phonological variations.

2.1.2. Neural Network Models

Neural networks have predominately replaced GMMs in hybrid acoustic models for ASR. This is a result of improved neural network architectures, training procedures, and GPU acceleration that enable suitable models yielding significant improvements to WER. Additionally, the ability of neural networks to use multiple frames of context (in addition to delta features) helps compensate for the conditional independence assumption of HMMs (as well as possibly neural network depth, as discussed by Ravuri and Wegmann, 2016).

Among the most common architectures for hybrid acoustic models are Time-Delay Neural Networks (TDNNs) (Waibel et al., 1989) which will be discussed further below. These may be considered functionally equivalent to 1-dimensional Convolutional Neural

Networks (CNNs) across time. 2-dimensional convolutions may also be used, considering the time-frequency representation of speech as an image (Abdel-Hamid et al., 2014; Sainath et al., 2013b). An alternative way to model temporal context is to use Recurrent Neural Networks (RNNs) (Robinson, 1994; Saon et al., 2014; Vinyals, Ravuri, and Povey, 2012), particularly Long Short-Term Memory (LSTM) models (Hochreiter and Schmidhuber, 1997; Sak, Senior, and Beaufays, 2014) to help alleviate the problem of vanishing gradients over time (Bengio, Simard, and Frasconi, 1994). A recent development is the use of self-attention to build purely attention-based models, such as the transformer (Wang et al., 2020). These have the advantage of being able to connect arbitrary parts of a sequence, and they are easy to parallelise. Most of the above models use Rectified Linear Units (ReLUs) as activation functions.

Many variations of the above architectures exist: CNNs or TDNNs are often combined with RNN architectures in the higher layers (Xiong et al., 2017). Zhang et al. (2016) included highway connections in deep LSTM models to control for vanishing gradients because of depth. Yu et al. (2016) incorporated a layer-wise attention mask and residual connections (He et al., 2016) into a TDNN architecture known as *LACE*. It is difficult to conclude which model architecture that works best. For example, Xiong et al. (2017) found an LSTM, a CNN with residual connections, and a combined CNN-LSTM all to perform similarly, with perhaps the LSTM model slightly better. These different model architectures, however, typically combine well for ensembling (Saon et al., 2015, 2017; Woodland et al., 2015; Xiong et al., 2017).

The neural network models are typically trained using alignments from a preceding HMM-GMM system, although there are approaches to flat-start training with HMM-DNN models (Hadian et al., 2018b; Senior et al., 2014; Zhang and Woodland, 2014). The models are normally built upon MFCC or Filterbank (FBANK) features (see Section 2.2), using several frames of context. It is worth noting, however, that it is common to use a dimensionality-preserving LDA transform as part of normalisation, so that more discriminative directions in feature-space have higher variance[5]. Since LDA is invariant to linear transforms of the data, and since the only difference between MFCCs and FBANK features is the DCT, a linear transformation, the choice of MFCCs or FBANK features becomes arbitrary (unless the features are compressed). There has also been work suggesting to simplify feature extraction given the number of linear transformations and model invariance properties (Yu and Waibel, 2000). More recently, models have been trained on raw-waveforms in the time-domain (e. g. Hoshen, Weiss, and Wilson, 2015), aspects of which will be explored in Chapter 8.

The neural network outputs are used as estimates of the emission probabilities $b_j(\mathbf{o}_t) = p(\mathbf{o}_t \mid q_t = j)$ for the HMM states by dividing the neural network outputs, $p(i \mid \mathbf{o}_t)$, by the class prior, $p(i)$, to create scaled likelihoods $p(\mathbf{o}_t \mid q_t = j)/p(\mathbf{o}_t)$. This is illustrated in Figure 2.7. We are modelling context-dependent phones (triphones).

[5]Kaldi applies some additional non-linear scaling to further de-emphasise the less discriminative directions (Povey, Zhang, and Khudanpur, 2014).

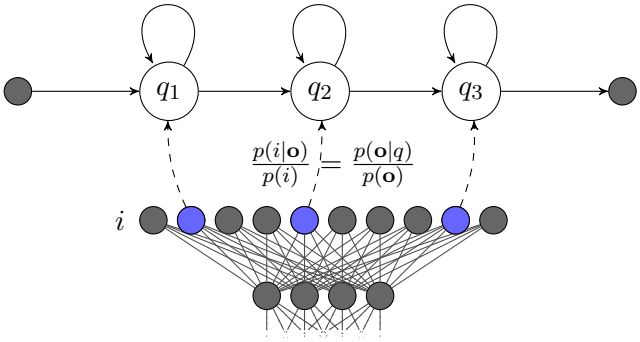


Figure 2.7: Illustration of an HMM-DNN hybrid model. Neural network outputs replace GMMs as emission probabilites for the HMM states.

Since we are unlikely to observe data for each possible logical triphone, we cluster and tie parameters from the individual HMM states, often using decision trees. The neural network output targets correspond to these tied HMM states. There can be a large number of physical states to model and consequently the softmax layer often contains a large fraction of the total number of parameters in a model. Models may be trained using the cross-entropy objective function, often followed by a pass of sequence-discriminative training using criteria such as Maximum Mutual Information (MMI), state-level Minimum Bayes Risk (sMBR) or Minimum Phone Error (MPE) (Veselý et al., 2013; Wang and Sim, 2011). Alternatively, models may be trained from scratch using e. g. LF-MMI (Povey et al., 2016). Sequence discriminative training will be discussed in Section 2.3.

A common architecture used throughout the thesis is TDNNs (Waibel et al., 1989). As noted above, these are mostly functionally equivalent to 1-dimensional CNNs, but with different terminology and modelling conventions. In ASR, the models typically use small kernels with large dilation rates (Peddinti, Povey, and Khudanpur, 2015) and a large number of feature maps. This corresponds to the number of spliced frames for a layer, splice context (with sub-sampling), and units in TDNN terminology. Figure 2.8 illustrates a TDNN architecture. Each layer processes a wider temporal context. The combination of sub-sampling (dilation) with parameter-tying (kernels) makes TDNNs efficient and small in terms of the total number of parameters.

Povey et al. (2018) introduced Factored Time-Delay Neural Networks (TDNN-Fs) which are used often in this thesis. The idea is to train from scratch using factored weight matrices. This builds upon work using weight matrix compression with the Singular Value Decomposition (SVD), whereby models are compressed by factoring the weight matrices into parts using the SVD and zeroing out the smallest singular values. It turns out that it is difficult to train with this factorisation from scratch, and it is therefore normally applied after an initial training pass, followed by fine-tuning (Prabhavalkar et al., 2016; Xue, Li, and Gong, 2013). Povey et al. (2018), however, worked out a way to train such a factorised representation from scratch. The proposed method was to approximate the properties of the SVD during training by

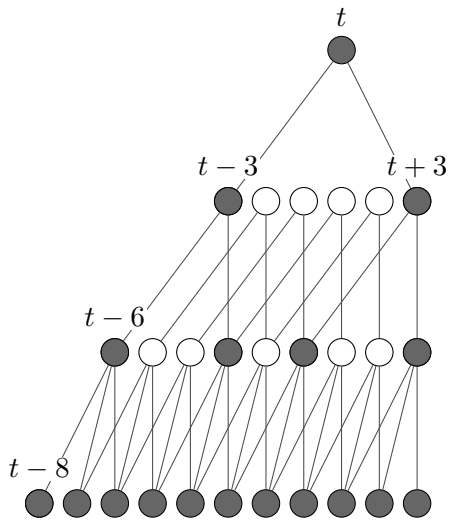


Figure 2.8: Illustration of a TDNN with splice-contexts and sub-sampling (dilation), equivalent to $[-2, 0], \{-3, 0\}, \{-3, +3\}$. Only the gray neurons require evaluation at time t.

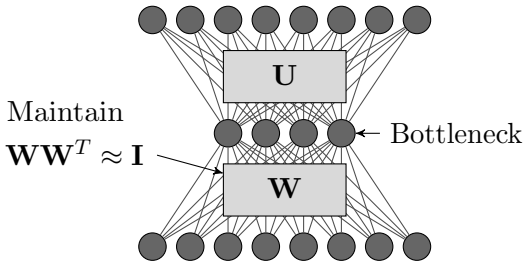


Figure 2.9: TDNN-F layers are trained from scratch with a matrix factorisation in which one of the matrices is semi-orthogonal.

enforcing "semi-orthogonality"[6] in one of the factors through the objective function. By adding $\mathrm{Tr}(\mathbf{QQ}^T)$ to the objective function, where $\mathbf{Q} \triangleq \mathbf{WW}^T - \mathbf{I}$, the parameter matrix $\mathbf{W}$ is in effect moved towards an orthonormal matrix, such that $\mathbf{WW}^T \approx \mathbf{I}$. Rather than using this technique to compress models, Povey et al. (2018) showed empirically that this model topology reduces the WER when using a similar total parameter budget to a normal TDNN model. Figure 2.9 illustrates factored weight matrices with a bottleneck layer.

2.2. Features

Features for GMM or DNN systems should ideally reduce dimensionality while preserving phone discrimination. A standard choice are MFCCs (Davis and Mermelstein, 1980) which provides a low-dimensional encoding of typically 40 dimensions or less. These are extracted by first computing frequency-domain representations by taking the Fast Fourier Transform (FFT) of windowed time-domain signals. The resulting spectra are

[6] "Semi-orthogonality" is used in Povey et al. (2018) to denote orthonormality applied to rectangular matrices, i. e. $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ or $\mathbf{WW}^T = \mathbf{I}$.

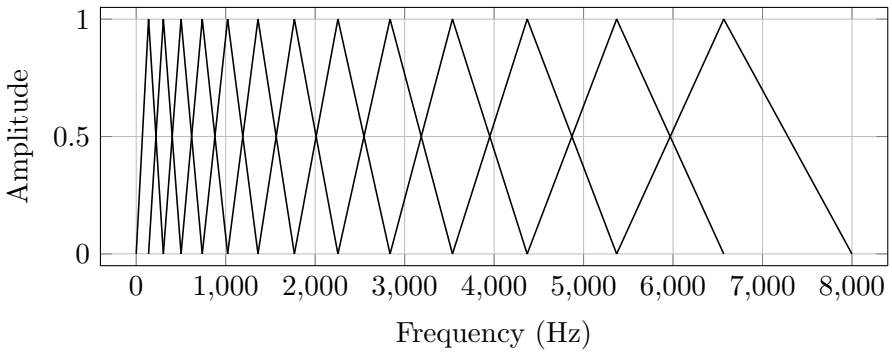


Figure 2.10: The human ear is not equally sensitive to all frequency bands. The mel-filterbank uses the mel scale (Stevens, Volkmann, and Newman, 1937) to model diminishing sensitivity to changes in pitch as frequency increases.

passed through the mel-filterbank (Davis and Mermelstein, 1980), shown in Figure 2.10, in order to emulate the frequency sensitivity of the human ear, followed by taking the log of each output value, to emulate human sensitivity to signal level[7]. At this point we can extract Mel filterbank coefficients (Abdel-Hamid et al., 2012; Deng et al., 2013), which has become popular for training neural networks, in particular for CNNs with 2-dimensional kernels, as the spectral energies are still local (Abdel-Hamid et al., 2014, 2012). MFCC extraction additionally obtains a second spectrum of the log-spectra output of the filterbank, usually using the DCT as mentioned above. The coefficients of the resulting *cepstrum* are mostly decorrelated, making them suitable for diagonal covariance modelling with GMMs, discussed above. The lower coefficients correspond to features that help discriminate phones, such as vocal tract shape. Traditionally, GMMs use 12-dimensional features (with 1 energy feature), and more recently neural networks often use 40 dimensions. Finally, to capture non-stationary features of the speech signal, we may extract regression coefficients (Furui, 1986) (delta features), from the change between successive coefficients. These have the additional benefit that they relax the assumption of conditional independence in HMMs (Gales and Young, 2008).

It can be argued that hand-crafted features like MFCCs may lose relevant information to word discrimination due to a choice of preprocessing or to their low dimensionality (Jaitly and Hinton, 2011; Sheikhzadeh and Deng, 1994). Neural network based models, sufficient data, and increased computational power allows most, if not all, of the feature processing pipeline to be left to a neural feature extractor. This can then learn task-dependent features (Sainath et al., 2013a). Promising results have been observed with CNNs (or TDNNs) on top of raw time-domain input (Hoshen, Weiss, and Wilson, 2015; Palaz, Collobert, and Doss, 2013; Sainath et al., 2015). Chapter 8 will experiment with raw time-domain modelling using a particular parameterisation of the CNN filters known as SincNet (Ravanelli and Bengio, 2018).

[7]In general, traditional feature pipelines are motivated from psychoacoustic phenomena – Perceptual Linear Prediction (PLP) coefficients (Hermansky, 1990) similarly pass spectra through the related Bark scale (Zwicker, 1961), and uses cubic-root amplitude compression to account for level.

2.3. Sequence-discriminative training

The standard cross-entropy training of the neural networks in the hybrid systems is discriminative at the frame-level. Speech is, however, inherently a sequence classification problem. It is possible to train to discriminate across sequences, in which the criterion is the direct misclassification of the hypotheses with the true reference. Since the 0-1 loss function is not differentiable, a number of other discriminative criteria have been applied in the literature. Examples include the Minimum Bayes Risk (MBR) critera that minimise the expected error at some granularity. For instance, the sMBR criterion (Gibson, 2008; Gibson and Hain, 2006; Povey et al., 2008) is a sum over the posteriors from hypotheses weighted by the state-level edit-distance. Another criterion that we will make use of more in this thesis is MMI, which aims to maximise the mutual information between the reference word sequence, $\mathbf{W}_{\text{ref}}$, and the model output given some observation sequence $\mathbf{O}$ (Bahl et al., 1986; Brown, 1987; Kapadia, Valtchev, and Young, 1993). This turns out to be correlated with the minimum expected sentence error (Yu and Deng, 2016). Since the language model is fixed, the MMI criterion effectively becomes the posterior of the correct word sequence:

$$\mathcal{F}_{\text{MMI}}(\theta) = \frac{1}{M}\sum_{m=1}^{M} \log P(\mathbf{W}_{\text{ref}}^{m} \mid \mathbf{O}^{m}; \theta) \tag{2.3}$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \log \frac{P(\mathbf{O}^{m} \mid \mathbf{W}_{\text{ref}}^{m}; \theta)^{\kappa} P(\mathbf{W}_{\text{ref}}^{m})}{\sum_{\mathbf{W}} P(\mathbf{O}^{m} \mid \mathbf{W}; \theta)^{\kappa} P(\mathbf{W})}, \tag{2.4}$$

where M is the number of utterances, θ indicate the model parameters, and κ is a weighting term between the acoustic and language models (Povey, 2005; Schluter and Macherey, 1998). The weighting factor κ is required because maximum likelihood training with HMMs tend to overstate the posterior probabilities since the HMM assumptions do not hold. For MMI, applying the factor to the AM or the inverse-factor to the LM is not equal: with LM scaling one particular hypothesis tends to dominate (Woodland and Povey, 2002). By instead applying it to the AM hypotheses, we can smooth the posteriors and make less likely hypotheses more confusable (when $\kappa \to 0$).

In Equation 2.4, by increasing the numerator through adjusting the model parameters, the reference model sequence is made more likely; and by decreasing the denominator, competing sequences are made less likely. As we will see in Section 3.3, when differentiating the MMI objective with respect to a neural network output i, the respective error signal backpropagated to the neural network becomes:

$$\mathbf{e}_t(i) = \kappa(\gamma_t^{\text{NUM}}(i) - \ddot{\gamma}_t^{\text{DEN}}(i)), \tag{2.5}$$

where $\gamma_t^{\text{NUM}}(i)$ and $\ddot{\gamma}_t^{\text{DEN}}(i)$ are the posteriors of being in state i at time t for the numerator and denominator, respectively, $\ddot{\gamma}(\cdot)$ indicates multiple paths, and κ is the

weighting factor from above. The numerator may consist of only a single reference hypothesis with state labels $\mathbf{Q} = [q_1, \ldots, q_T]$ obtained through forced alignment. In this case it is sufficient to compute the indicator function: $\gamma_t^{\text{NUM}}(i) = \kappa\delta_{i;q_t}$. The posterior from the denominator (and the numerator if it contains multiple hypotheses) can be obtained using the forward-backward algorithm. The numerator and denominator are usually represented using lattices (Kingsbury, 2009; Normandin, Lacouture, and Cardin, 1994) encoded as WFSTs which will be discussed in Chapter 3. In practice, it is normally infeasible to compute the denominator over all possible sequences. Instead, the most likely competing paths are computed by recognising the data with an existing cross-entropy trained model to generate a suitable lattice of hypotheses. A weak LM may be used to generate more varied hypotheses (Povey, 2005). Alternatively, a phone-based graph may be used instead, as discussed below.

2.3.1. Lattice-Free MMI

The standard method to train neural networks with sequence-discriminative criteria is to first train a model with an initial pass of cross-entropy and then to use that model to generate denominator lattices for further training with e. g. MMI (Veselý et al., 2013; Wang and Sim, 2011). To avoid this time-consuming first step, Povey et al. (2016) proposed Lattice-Free MMI (LF-MMI), which bypasses the need for the denominator lattice altogether by replacing it with an utterance-agnostic language-model graph. Using LF-MMI, Povey et al. (2016) demonstrated up to an 8% relative improvement in WER over previous cross-entropy trained systems followed by sequence-discriminative training with the state-level Minimum Bayes Risk (sMBR) criterion (Gibson and Hain, 2006; Povey et al., 2008). Note that since we can train from scratch using LF-MMI, the posteriors are well-calibrated, and the optimal value for the weighting factor, κ, in Equation 2.4 above, is close to 1.

To make such a denominator computationally feasible, Povey et al. (2016) use a 4-gram phone-level, rather than word-level, LM, trained from alignments of a preceding GMM system. To keep the graph size to a minimum, there is no smoothing, interpolation or back-off. The forward-backward computation is implemented on GPUs. To further reduce complexity, the model outputs at one third of the frame rate, with an amended topology that can be traversed in a single frame (as discussed in Section 2.1). A mixture of regularisation methods are required to control for overfitting, such as dropout and a cross-entropy multi-task loss (Povey et al., 2016).

The numerator is now also a phone-level graph representing alternative pronunciations, and is computed using the forward-backward algorithm. In order to extract suitable fixed-size chunks for training, the graph needs to be acyclic so that it can be topologically sorted by time. The self-loops can then be expanded within some tolerance, and the final graph can be sorted and split into chunks for training. Within each chunk it is possible to re-introduce cycles, which results in smaller (no longer acyclic) graphs

that are faster to prepare with similar final error rates (Hadian et al., 2018a). Hadian et al. (2018b) have since proposed an extension to the LF-MMI framework that enables flat-start training with neural networks, without the need for alignments from a GMM system.

We noted above that the numerator now can represent alternative pronunciations, or in other words, multiple hypotheses. This is possible with LF-MMI since we have a sum over all possible paths in the denominator through the phone-level graph. Consequently, all numerator paths are counted for in the denominator. In previous implementations of MMI this was not necessarily the case which could lead to spikes in the objective function. There are examples of semi-supervised training (Manohar et al., 2018) and unsupervised adaptation (Klejch et al., 2019) that make effective use of this capability for multiple hypotheses. We will call this *lattice supervision* (see Section 3.3), referring to the word level lattices prior to numerator graph creation. Lattice supervision will be particularly useful for lightly supervised training, presented in Chapter 5, where we will demonstrate a technique to combine lattice-based semi-supervised training with possibly inaccurate transcriptions. In particular, we will show that we obtain WER improvements by using a lattice compared to using a single path.

2.4. Adaptation and normalisation

Mismatch between training and testing conditions can be detrimental to system performance. There is a large amount of research on adaptation and normalisation techniques to alleviate such mismatch. We will focus on the methods applicable to the thesis. These methods may be categorised into those acting in the model-space, feature-space, or through auxiliary features, but many may have analogous transforms in the other space. A typical example for HMM-GMM models is Maximum Likelihood Linear Regression (MLLR) (Leggetter and Woodland, 1995) which operates on the GMM means and variances and its constrained variant, CMLLR (Gales, 1998) which has a corresponding feature-level transform. This is used for SAT training of HMM-GMM models throughout the thesis to obtain alignments for downstream neural network models.

For neural network model adaptation, one possibility is to simply fine-tune the entire model on adaptation data, potentially with cross-validation. If the target labels are inaccurate, such as may be the case with a first-pass recognition, it is important to constrain the parameters in some manner, or to reduce the total number of parameters to adjust. Otherwise it is possible to overfit to incorrect adaptation targets. This leads to methods that adapt only a subset of the parameters, often by inserting hidden layers (Gemello et al., 2007; Li and Sim, 2010; Neto et al., 1995), which may be further constrained to just the diagonal of the layer matrix, as is the case with Learning Hidden Unit Contributions (LHUC) (Swietojanski and Renals, 2014):

$$\mathbf{h}^{(l)} = s(\mathbf{r}^{(l-1)}) \odot \mathbf{h}^{(l-1)}, \tag{2.6}$$

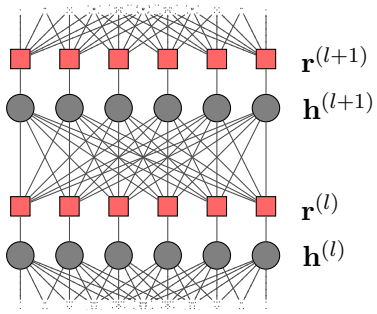


Figure 2.11: An LHUC layer consists of scalars (red squares) that are multiplied with the activations from the previous layer.

where $\mathbf{h}^{(l-1)}$ is the output of the previous layer l, $\mathbf{r}^{(l-1)}$ is a vector of LHUC scalars and s is an optional non-linearity or squashing function. This is illustrated in Figure 2.11. LHUC turns out to be surprisingly robust to both inaccurate targets, number of parameters and training speed: it can typically be applied robustly to all layers with a high learning rate (e.g. 0.8) (Swietojanski, Li, and Renals, 2016). A number of variations upon LHUC exist, such as subspace-LHUC (Samarakoon and Sim, 2016), which reduces the number of parameters per speaker by a matrix-vector product, and Bayesian-LHUC (Xie et al., 2019), which obtains a full posterior over LHUC parameters. We will make comparisons to LHUC in Chapter 8.

When an auxiliary feature for a speaker s, $\mathbf{z}_s$, is used it affects the layer through a bias:

$$\mathbf{h}^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)} + \mathbf{b}_s^{(l)}), \tag{2.7}$$

where $\mathbf{b}_s^{(l)} = \mathbf{U}^{(l)}\mathbf{z}_s$, $\mathbf{U}$ and $\mathbf{W}$ are weight matrices, and $\mathbf{b}$ is the standard bias. Such features may be speaker codes (Abdel-Hamid and Jiang, 2013), bottleneck features (Liu, Zhang, and Hain, 2014) or i-vectors (Dehak et al., 2010; Saon et al., 2013), the latter of which will be used in Chapter 6. While normally used as auxiliary features in this manner, these features can also be used as independently estimated features for subspace-LHUC (Samarakoon and Sim, 2016), or embedded within a feature transformation matrix (Samarakoon and Sim, 2015). i-vectors (Dehak et al., 2010) are estimated using means from GMMs trained on the features. Specifically, an extracted i-vector, $\boldsymbol{\lambda}$, represents coordinates in a (total variability) subspace that models the difference between speaker-specific GMM means, $\mathbf{m}_{SD}$, and means from a background GMM, $\mathbf{m}_{SI}$:

$$\mathbf{m}_{SD} = \mathbf{m}_{SI} + \mathbf{T}\boldsymbol{\lambda}, \tag{2.8}$$

where $\mathbf{T}$ is the total variability matrix. Used in neural network models they often yield up to 1-2% absolute improvements in WER with 100-dimensional vectors (Saon et al., 2013). They are included by standard in most Kaldi recipes. An advantage of i-vectors (as well as most bottleneck features and speaker codes) is that they are estimated without transcriptions. However, they may do worse on unseen speakers

than a baseline without i-vectors (Liu, Karanasou, and Hain, 2015). Chapter 6 will demonstrate a technique to factorise i-vectors, producing bottleneck features that are more robust to speaker or environment mismatch.

In Chapter 8 we will study an analogy to a feature-space technique known as Vocal Tract Length Normalisation (VTLN) (Lee and Rose, 1996). This technique was proposed to control for varying vocal tract length in speakers by estimating a, typically piecewise linear, warping function with a single warping factor parameter that is applied to the triangular filterbank in MFCC extraction (Figure 2.10). Such warping functions are illustrated in Figure 2.12. Piecewise linear functions are suitable since vocal tract length affects the formants in a near linear manner (Pitz and Ney, 2005). The warping factors may be estimated by a grid-search, whereby the model is iteratively re-trained with the optimal warp factors until they converge (Lee and Rose, 1996). Apart from being a time-consuming process, comparing likelihoods with effectively different models given the warp applied, ideally requires computing a Jacobian compensation term which is not trivial (for further discussions see Povey et al., 2011; Uebel and Woodland, 1999). VTLN also has a model-space analogue, a linear transform which may be estimated by analytical (e. g. Pitz and Ney, 2005) or data-driven (e. g. Kim et al., 2004; Uebel and Woodland, 1999) means. In this space there are related methods also using a single tunable parameter, but that forgo any explicit relation to frequency warping, such as the exponential transform (Povey, Zweig, and Acero, 2011).

In mismatched conditions the application of VTLN at training or test-time can have great impact on error rates. Giuliani and Gerosa (2003) saw improvements from 39.68% to 32.35% WER when recognising child speech with an adult speech GMM model with and without VTLN. VTLN cannot, however, fully bridge the gap to their model trained on child speech from scratch, which in that case obtained 22.7% WER. Chapter 8 will investigate a similar idea to VTLN, but with the filterbank embedded within the first layer of a neural network, and with no constraint on the warping function.

2.5. Summary

We have briefly reviewed hybrid acoustic models, along with common techniques that will be used throughout the thesis. Specifically, we discussed neural network architectures (TDNNs), feature extraction, the LF-MMI criterion and the ability to use lattice supervision, and touched upon techniques for model-based neural adaptation (LHUC), auxiliary feature-based adaptation (i-vectors) and feature normalisation (VTLN).

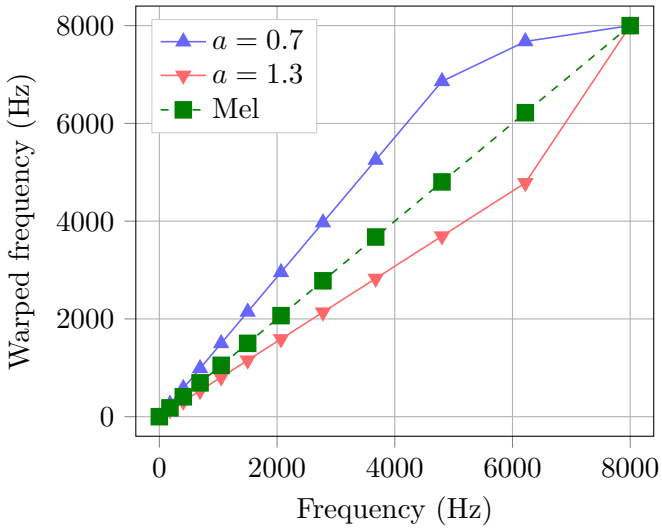


Figure 2.12: Example of two piece-wise linear scaling functions for VTLN with arbitrary warping parameters for demonstration. Produced using the implementation details from Kaldi (Povey et al., 2011). $\alpha = 1$ would equal the mel frequency curve of a traditional filterbank for MFCC extraction (Figure 2.10).

Chapter 3

Lattices for decoding and supervision

In the previous chapter we briefly discussed how lattices may be used to represent the numerator and denominator in sequence-discriminative training. ASR systems in general make great use of lattices to represent alternative hypotheses. The nodes in a lattice represent points in time, and the arcs represent (typically word-level) hypotheses. Acoustic and language model scores may be attributed to each arc. An example word-level lattice is shown in Figure 3.1. Lattices are used to store the results of decoding without the redundancy of n-best lists. Further, they enable re-scoring with higher order language models or other knowledge sources (e. g. Mangu, Brill, and Stolcke, 2000), on-the-fly manipulation without re-running the decoder (e. g. Bell et al., 2017), lattice supervision for semi-supervised training (e. g. Huang and Hasegawa-Johnson, 2010; Manohar et al., 2018), among many other applications. Encoding these lattices as Weighted Finite State Transducers (WFSTs) enables a host of well-defined operations (see Mohri, Pereira, and Riley, 2008). A canonical example of a WFST is the HCLG decoding graph (Table 3.1), which will be introduced below. Another is the manipulation of lattice supervision, which will be an important idea in Chapter 5. We first review concepts of WFSTs that are relevant to this thesis, then briefly discuss the HCLG decoding graph and the process of generating a hypothesis lattice through lattice generation. Lastly, we discuss the effect of lattice supervision during training.

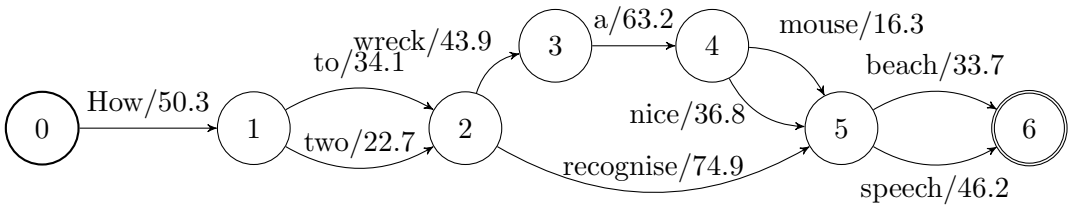


Figure 3.1: Example of a lattice with combined model scores encoded as a WFST.

3.1. Finite State Transducers

A Weighted Finite State Transducer (WFST) is an automaton that transduces a legal input string into an output string. Weights on each transition along the path determine

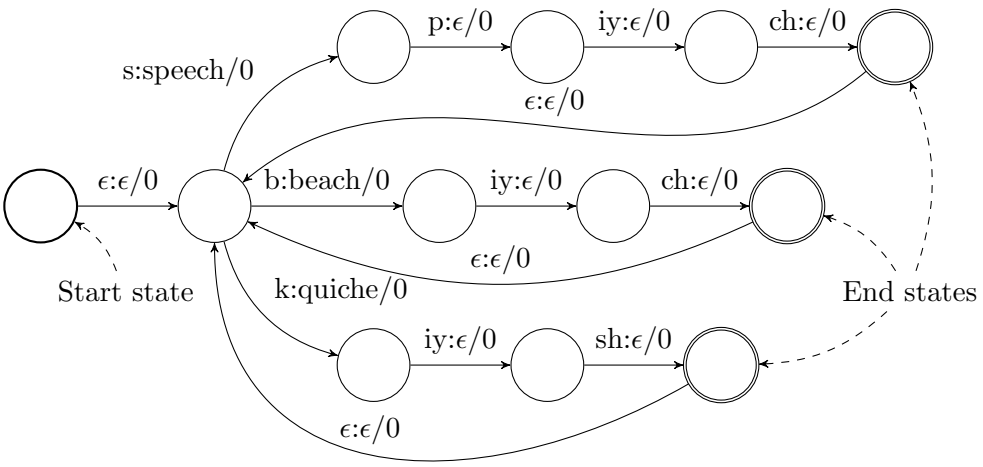


Figure 3.2: Example of a WFST, here a toy lexicon. Input and output labels are separated by colons (:), and weights after slashes (/) (in this case all 0). ϵ-arcs consume no inputs or produce no outputs. This example also demonstrates the union between multiple automata, as well as Kleene closure: the union operation has combined individual transducers for each word into having a shared start state, with a new ϵ-arc to avoid repeatedly incurring any start state cost; the Kleene closure is shown by the arcs from the end states back to the start.

a total cost of the operation. An example WFST is shown in Figure 3.2. This is a toy lexicon transducing sequences of phones into words. ϵ output labels produce no output, so the input {`s`, `p`, `iy`, `ch`} will produce the word `speech`. Such a lexicon can easily be constructed from a list of words and their pronunciations: We first construct individual WFSTs for each word, and then we join them using the *union* and *closure* operations which we will describe below. If input and output labels are identical, then the automaton is instead known as a weighted finite state *acceptor* (WFSA). A typical acceptor is a language model, or grammar, as we will see below, that consumes an input string if there is a match.

Natural operations on a transducer include finding the "best", or "shortest", path between two states. In order to define this we will introduce some standard notation using general binary operators $\oplus$ and $\otimes$: the total cost accrued on a path, $\boldsymbol{\pi} = [a_1, \ldots, a_n]$, is $w(\boldsymbol{\pi}) = w(a_1) \otimes w(a_2) \otimes \cdots \otimes w(a_n)$, for each arc a in the path. To compare paths, and compute the best path cost over a finite set of possible paths R, $w(R)$, we write $w(R) = w(\boldsymbol{\pi}_1) \oplus w(\boldsymbol{\pi}_2) \oplus \cdots \oplus w(\boldsymbol{\pi}_m)$, for all possible paths $\pi \in R$ between two states. More succinctly, the best path cost may be written (Mohri, 2002):

$$w(R) = \oplus_{\boldsymbol{\pi} \in R} w(\boldsymbol{\pi}). \tag{3.1}$$

This notation is very general, so that we can choose appropriate operations given the problem at hand, but using a common computational framework. To choose the operations, we choose a semiring, where a semiring defines an algebraic structure with

certain properties[1] (see e. g. Hori and Nakamura, 2013). It is defined by a 5-tuple: $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$, where $\mathbb{K}$ is the domain, $\oplus$ is the additive operator, $\otimes$ is the multiplicative operator, $\bar{0} \in \mathbb{K}$ is the additive identity, and $\bar{1} \in \mathbb{K}$ is the multiplicative identity. For example, we may define a semiring over probabilities: $([0, 1], \max, \times, 0, 1)$, with which we can now write the best path cost as:

$$w(R) = \max_{\boldsymbol{\pi} \in R} w(\boldsymbol{\pi}), \tag{3.2}$$

where $w(\boldsymbol{\pi}) = w(a_1) \times w(a_2) \times \cdots \times w(a_n)$. The best path is the path with the maximum probability, and the probabilities are multiplied.

The one we require in this thesis, and is typically used for ASR applications, is the tropical semiring: $(\mathbb{R} \cup \{+\infty\}, \min, +, +\infty, 0)$, where the best path is the minimum cost path, and the costs are summed along a path:

$$w(R) = \min_{\boldsymbol{\pi} \in R} w(\boldsymbol{\pi}), \tag{3.3}$$

where $w(\boldsymbol{\pi}) = w(a_1) + w(a_2) + \cdots + w(a_n)$.

Next we define some key operations. The inner workings of some depend on the semiring that the finite state automaton admits. We refer to Hori and Nakamura (2013) and Mohri, Pereira, and Riley (2008) for more complete treatments, and more precise mathematical definitions.

Composition combines different levels of representation by cascading multiple transducers. For example, below we will see the composition of four individual automata that make up a decoding graph commonly referred to as the *HCLG*: $H \circ C \circ L \circ G$ (see Table 3.1). Output labels in one transducer are matched with input labels in the next. The combined weight is the sum of the matching paths. For acceptors, composition is equivalent to intersection: since input and output label pairs in this case are identical, the composition of two acceptors will find the shared paths between the two.

Determinisation produces a deterministic automaton which has no transitions leaving a state sharing the same input label. In other words, if there are multiple possible paths for a given input string, determinisation combines them into a single path, summing their weights to create an equivalent automaton. The choice of semiring determines the interpretation of the sum.

Minimisation produces an equivalent transducer with the fewest possible number of states and transitions.

Kleene closure allows an automaton to match patterns 0 or more times by introducing ϵ-arcs from final to input states, and making the initial state also a final state.

[1] Well known examples are probabilities (semiring) and square matrices (ring).

Component	Input labels	Output labels
H	Context-dep. HMM states	Context-dep. phones
C	Context-dep. phones	Phones
L	Phones	Words
G	Words	Words

Table 3.1: Components in the HCLG and their respective input and output labels.

This is equivalent to the Kleene star operation in regular expressions (Kleene, 1951).

Union combines automata by creating a new initial state that branches out to the original initial states of each automaton with ϵ-arcs. Any previous initial state costs are moved to these arcs.

Concatenation combines automata in series, creating new ϵ-arcs where one automaton joins the other.

Inverse switches the input with the output labels on each arc. The structure and the costs are not changed.

Reverse reverses the states of an automaton. The initial state becomes the new final state.

Projection creates an acceptor from a transducer by retaining only the input or the output labels.

Chapter 5 will make use of most of these operations.

3.2. Decoding graph and lattice generation

The various components of a hybrid ASR system, specifically a context-dependent phone-based HMM system, can be encoded as a single WFST. This is typically denoted as an HCLG decoding graph (Mohri, Pereira, and Riley, 2008), which is the composition of the HMM structure (H), the phonetic context dependency (C), the lexicon (L), and a grammar or LM (G). The first three are transducers, whilst G is an acceptor. Table 3.1 shows the input and output labels for each component. The resulting HCLG has as inputs context dependent HMM states, and as outputs words.

To demonstrate the use of the HCLG decoding graph, we will consider a particular utterance u. We would like to use that utterance in combination with the HCLG to find the most likely corresponding sequence of words. A component that is missing from the HCLG is the neural network (in a hybrid system). We will therefore build a Weighted Finite State Acceptor (WFSA) for the utterance, U, which will encode the likelihoods

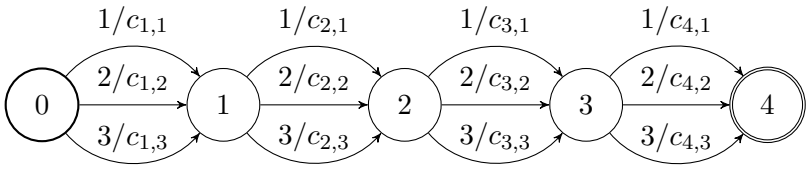


Figure 3.3: WFSA U for a four-frame utterance with corresponding model costs, $c_{n,s}$, for frame n and state s. In a real system the number of outgoing arcs per state will equal the number of tied states in the system.

of the acoustic model for each frame. To build it, we evaluate each frame with the acoustic model, and record the likelihoods for every state. For neural network acoustic models, this amounts to performing a forward pass given the current feature frame, and recording all the output probabilities. These probabilities represent context-dependent states, and become parallel arcs between (automaton) states in the FST for the current frame, as illustrated in Figure 3.3. The hypothesis space of that utterance given the ASR model then becomes

$$S = U \circ HCLG, \tag{3.4}$$

where the most likely hypothesis for U corresponds to the best path of S.

In toolkits such as Kaldi, S is never constructed explicitly, as it amounts to copying the HCLG for each frame in U. Instead, Kaldi directly produces a pruned version of S, P_S, by employing a variation of the token-passing algorithm (Young, Russell, and Thornton, 1989; Young et al., 2002) for Viterbi decoding, with on-the-fly pruning. Since we ultimately care about the word sequence, and P_S has context-dependent states as inputs and words as outputs, we project on the output words. A modified determinisation algorithm follows, which is quite involved, and not relevant to the thesis. The final result is a lattice of hypotheses with no duplicate paths, and where the best path matches the best path in P_S. See Povey et al. (2012) for further details, as well as Chen et al. (2019) for incremental determinisation, and Chen et al. (2018) for an extension to GPU-based decoding. The lattice of hypotheses can be further used in various decoding methods such as Minimum Bayes Risk (MBR) decoding which aims to minimise risk measures based on WER (see e. g. Xu et al., 2011).

3.3. Lattices as supervision

A method to include a notion of uncertainty about the labels directly into training is to use a lattice for supervision. Multiple hypotheses in the supervision encode uncertainty about the accuracy of each hypothesis. This is useful when the supervision is inaccurate, such as when adapting to first pass targets (Klejch et al., 2019), training in a semi-supervised manner (Manohar et al., 2018), or when using subtitles as labels (Fainberg et al., 2019b). This will be demonstrated in Section 5.2. Discriminative criteria such as MMI are particularly sensitive to inaccurate labels (Mathias, Yegnanarayanan, and

Fritsch, 2005; Yu et al., 2010).

Lattice supervision is not a new idea. It has previously been used with HMM-GMM systems for modelling multiple pronunciations (Hain, 2002; Povey, 2005), although using a single pronunciation variant per word could be more successful (Hain, 2002). Huang and Hasegawa-Johnson (2010) and Manohar, Povey, and Khudanpur (2015) used lattice entropy minimisation for semi-supervised training, for GMM and DNN systems, respectively. Lattice supervision has been beneficial for training with LF-MMI: Manohar, Povey, and Khudanpur (2017) fused crowd-sourced transcripts into confusion networks (Mangu, Brill, and Stolcke, 2000), Manohar et al. (2018) used recognition lattices in a semi-supervised setup, and Klejch et al. (2019) demonstrated the advantage of lattice supervision in test-time applications.

We will see how including multiple hypotheses for supervision encodes uncertainty in those hypotheses. It is due to the posterior of an HMM state being affected by multiple states occurring at the same timestep t. It is instructive to look at how it affects the error backpropagated to the neural network. Recall the MMI criterion from Section 2.3:

$$\begin{aligned}\mathcal{F}_{MMI}(\theta) &= \log p(\mathbf{W}^m_{\text{ref}} \mid \mathbf{O}^m; \theta) \\ &\approx \log \frac{p(\mathbf{O}^m \mid \mathbf{W}^m_{\text{ref}}; \theta)^\kappa \, P(\mathbf{W}^m_{\text{ref}})}{\sum_{\mathbf{W}} p(\mathbf{O}^m \mid \mathbf{W}; \theta)^\kappa \, P(\mathbf{W})}\end{aligned} \tag{3.5}$$

where $\mathbf{W}^m_{\text{ref}}$ denotes a reference word sequence for utterance m, $\mathbf{O}^m$ is the corresponding sequence of acoustic observations over T_m frames, θ denotes the model parameters, and κ is a scaling factor (see Section 2.3). To connect the equation more closely with the HMM state sequence, we will use $\mathbf{Q}^w$ to denote the sequence of states (over T_m frames) that corresponds to a particular word sequence $\mathbf{W}$, which allows us to re-express Equation 3.5 as:

$$\mathcal{F}_{MMI}(\theta) = \log \frac{p(\mathbf{O}^m \mid \mathbf{Q}^m_{\text{ref}}; \theta)^\kappa \, P(\mathbf{W}^m_{\text{ref}})}{\sum_{\mathbf{W}} p(\mathbf{O}^m \mid \mathbf{Q}^w; \theta)^\kappa \, P(\mathbf{W})}. \tag{3.6}$$

The derivative of the numerator of $\mathcal{F}_{MMI}(\theta; \mathbf{O}^u, \mathbf{W}^u)$ with respect to the neural network scaled log likelihoods, $\log p(\mathbf{o}^m_t \mid i)$, for neural network output i, provides the following error signal component from the numerator at time t:

$$\frac{\delta}{\delta \log p(\mathbf{o}^m_t \mid i)} \log p(\mathbf{O}^m \mid \mathbf{Q}^m_{\text{ref}}; \theta)^\kappa P(\mathbf{W}^m_{\text{ref}}) = \kappa \delta_{i;q^m_t} = \boldsymbol{\gamma}^{\text{NUM}}_{mt}(i). \tag{3.7}$$

The contribution from the numerator supervision for output i is, in other words:

$$\kappa \delta_{i;q^m_t} = \begin{cases} \kappa & \text{if } q_t = i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

The magnitude of the error contribution from the supervision is always the same.

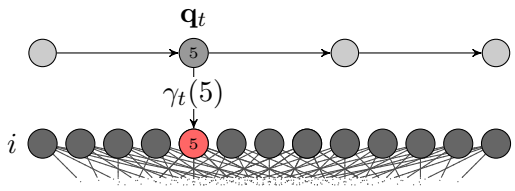


Figure 3.4: With a supervision lattice in which there is only a single path, with state labels obtained through forced alignment, the error contribution from the numerator (denominator not shown) is only non-zero for the corresponding output at time t.

Here we have assumed that the state labels are obtained through forced alignment with the Viterbi algorithm[2] as in other work (see e. g. Veselý et al., 2013). We could also use the forward-backward algorithm to obtain state posteriors for the numerator. The complete error signal at time t for an output i is

$$\mathbf{e}_t^m(i) = \kappa(\boldsymbol{\gamma}_{mt}^{\mathrm{NUM}}(i) - \ddot{\boldsymbol{\gamma}}_{mt}^{\mathrm{DEN}}(i)), \tag{3.9}$$

where $\ddot{\boldsymbol{\gamma}}_{mt}^{\mathrm{DEN}}(i)$ is the contribution from the denominator from multiple hypotheses[3]. Using a lattice for supervision means to include a sum over sequences (hypotheses), m', in the numerator of Equation 3.5, similar to the denominator. The updated criterion, where the superscript with two dots indicates lattice supervision, is:

$$\begin{aligned} \ddot{\mathcal{F}}_{MMI}(\theta; \mathbf{O}^u, \mathbf{W}^u) &= \log p(\mathbf{W}^m \mid \mathbf{O}^m; \theta) \\ &= \log \frac{\sum_{m'} p(\mathbf{O}^m \mid \mathbf{Q}^{m'}; \theta)^\kappa \, P(\mathbf{W}^{m'})}{\sum_{\mathbf{W}} p(\mathbf{O}^m \mid \mathbf{Q}^w; \theta)^\kappa \, P(\mathbf{W})}. \end{aligned} \tag{3.10}$$

The derivative of the numerator of Equation 3.10 now becomes:

$$\frac{\delta}{\delta \log p(\mathbf{o}_t^m | i)} \log \sum_{m'} p(\mathbf{O}^m \mid \mathbf{Q}^{m'}; \theta)^\kappa \, P(\mathbf{W}^{m'}) \tag{3.11}$$

$$= \kappa \frac{\sum_{m': q_t = i} p(\mathbf{O}^m \mid \mathbf{Q}^{m'}; \theta)^\kappa P(\mathbf{W}^{m'})}{\sum_{m'} p(\mathbf{O}^m \mid \mathbf{Q}^{m'}; \theta)^\kappa P(\mathbf{W}^{m'})} = \kappa \ddot{\boldsymbol{\gamma}}_{mt}^{\mathrm{NUM}}(i), \tag{3.12}$$

which is the constant κ times the posterior of being in state i at time t. It can be computed in the same manner as the denominator using the forward-backward algorithm. As noted in Section 2.3.1, this is the default operation for LF-MMI.

We can visualise the consequence of lattice supervision. First, if the supervision consists of a single reference, with state labels obtained through forced alignment, then the error contribution from the numerator at a particular time-step only affects a single neural network output, shown in Figure 3.4. In contrast, the consequence of lattice supervision is effectively a smoothing of the errors, distributed across multiple network outputs that correspond to concurrent states in the lattice, as illustrated in Figure 3.5.

[2] This was the standard approach for MMI training with Kaldi prior to LF-MMI.
[3] See Yu and Deng (2016, Chapter 8) for the derivation.

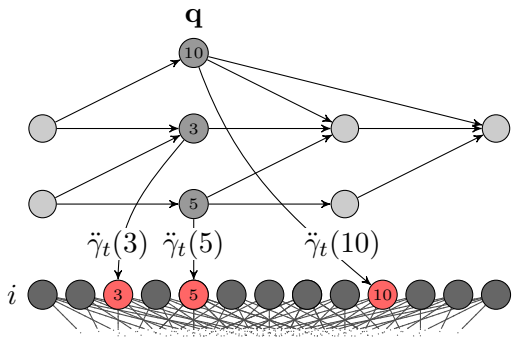


Figure 3.5: With lattice supervision, multiple numerator error signals (denominator not shown) contribute from different states in a supervision lattice corresponding to neural network outputs (bottom row). The supervision lattice may be implemented as a graph (e. g. LF-MMI).

The practical result is that such a lattice is useful if we know that the data is inaccurate. Specifically,

1. if there is a correct transcription in the lattice, then including other paths only reduces the impact of that transcription;

2. if there is an inaccurate transcription in the lattice, then including other hypotheses reduces the impact of that erroneous data point.

Consequently, this is particularly useful for semi-supervised training where we adapt to hypotheses generated with a seed model. An example of the effect is shown in Table 3.2[4] where we perform test-time adaptation using the MGB corpus (Chapter 4). A TDNN-F model was adapted using LF-MMI to episodes from broadcast media. The first pass decode supervision was rescored with a 4-gram LM, as well as the final decoded output.

The impact of lattice supervision is most pronounced when adapting all parameters, which also provided the best result. Using only the best path when adapting all parameters yields almost no gain (-1%). When only adapting a subset of the parameters with LHUC (Section 2.4) the results are less dependent upon the type of supervision, but does not perform as well as adapting all parameters with lattice supervision. We will see a considerable effect of lattice supervision when dealing with light supervision in Chapter 5.

3.4. Summary

Lattices are a common construct in ASR systems, used to compactly store decoding hypotheses, as well as for supervision. They are typically encoded as WFSTs. The use of lattices for supervision effectively incorporates a notion of uncertainty. Chapter 5 will make use of this by manipulating supervision using WFST operations, on lattices generated from the decoding hypotheses of a seed model.

[4]Table 4 in Klejch et al. (2019) was contributed by the author of this thesis.

Method	WER (%)
Baseline	19.9
All parameters – Lattice supervision	19.2
All parameters – Best path	19.7
LHUC – Lattice supervision	19.4
LHUC – Best path	19.5

Table 3.2: Test-time adaptation results adapting to entire episodes in the longitudinal eval data of the MGB corpus using all the parameters or LHUC (Section 2.4). The baseline model is from Chapter 4: a TDNN-F model trained with LF-MMI on the MGB training set with MER=40, using i-vectors. Results from Klejch et al. (2019).

Chapter 4

Data

Exploring the challenges identified in Chapter 1 requires experiments using a variety of corpora. These have been selected for their individual characteristics in order to focus on particular problems, such as adaptation to the domain, the speaker, the available labels, and environmental noise, taking into account the homogeneity (or lack thereof) of these factors. This chapter aims to provide context and detail to the data, presenting state-of-the-art error rates from the literature. We start by recalling the challenges and hence justifying the choice of each corpus.

Light supervision. How may inaccurate transcripts be used effectively during training? The MGB corpus (Bell et al., 2015) and data from the Scottish Parliament (ScotParl; see Section 4.2) both have inaccurate transcriptions, and they represent realistic, real-world data. We chose to study lightly-supervised adaptation from a seed model trained on MGB news to a test set from ScotParl.

Factorised adaptation. How can feature representations be factorised, such that they can be combined in novel combinations at test time? We would like a well-controlled setup of two factors that we can manipulate independently. We chose WSJ (Paul and Baker, 1992) which contains clean speech from a wide variety of speakers. In Chapter 6 we combine it with noise sources from the DEMAND (Thiemann, Ito, and Vincent, 2013) database. This becomes similar to the data in the Aurora (e. g. Parihar et al., 2004) and CHiME (e. g. Barker et al., 2015) challenges, but with more control of the variety in environmental noise through DEMAND.

Longitudinal learning. When data is ephemeral, what is required for a model to improve in a longitudinal setting? To study longitudinal learning we require data from the same domain that progresses in some natural manner. The MGB challenge (Bell et al., 2015) included a development set for exactly this purpose. One of the series in this set contains 11 episodes from the same broadcast programme, and is a suitable choice to study adaptation over time.

Domain adaptation. What is a parameter-efficient way to adapt a background model to new speaker characteristics, such as child speech? We chose to study

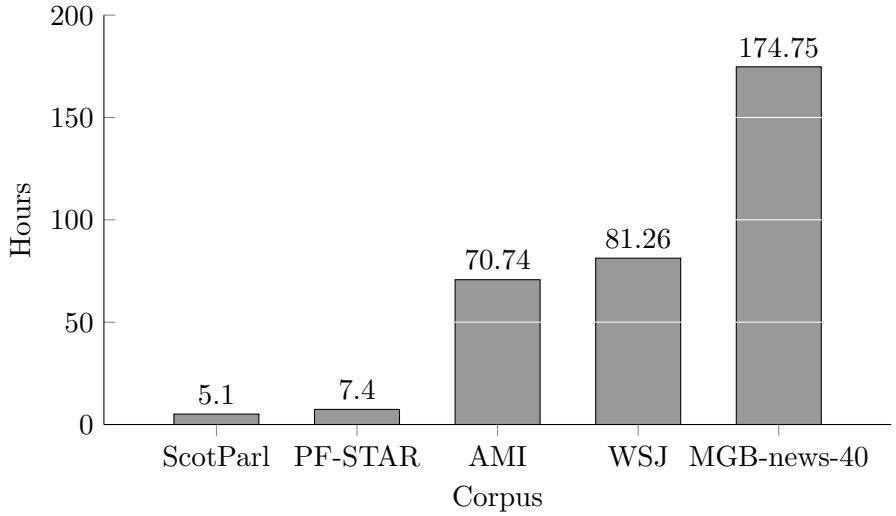


Figure 4.1: Amount of training data in the particular subsets of the corpora used in experiments. Only the ScotParl adaptation set is shown.

this in the case of adapting adult speech models to child speech. These large acoustic differences are interesting from an acoustic modelling perspective. It is also a highly practical application. Using the English AMI corpus (Carletta, 2007) as baseline data provides reasonably difficult adult speech scenarios with good labels. There are not many English child speech corpora to choose from. We chose PF-STAR (Batliner et al., 2005) because of access and previous experience with this corpus (Fainberg et al., 2016).

The remainder of this chapter provides an overview of the MGB, ScotParl, AMI, WSJ and PF-STAR corpora. For each corpus we have included example transcripts in Appendix B. The amount of training data from each of these corpora that we later use in experiments is shown in Figure 4.1.

4.1. Multi-Genre Broadcast corpus

The MGB corpus (Bell et al., 2015) contains speech from broadcast TV shows from the British Broadcasting Corporation (BBC) over the period 1 April 2008 to 19 May 2008. It is used in Chapter 5 as a seed model for lightly supervised experiments; and in Chapter 7 for longitudinal adaptation.

The corpus has a large variety of series accompanied by metadata identifying each series' genre. These are displayed in Figure 4.2 along with the amount of training data within each genre. It is not clear how best to make use of the genre labels. From the submitted systems in the MGB challenge (Bell et al., 2015), most trained without explicitly incorporating genre labels, the average results across episodes from different genres range from about 16% to 51% WER (Bell et al., 2015). Table 4.1 shows experiments in training and decoding HMM-GMM models across genres. We have computed the difference between the observed error rate and the expected error

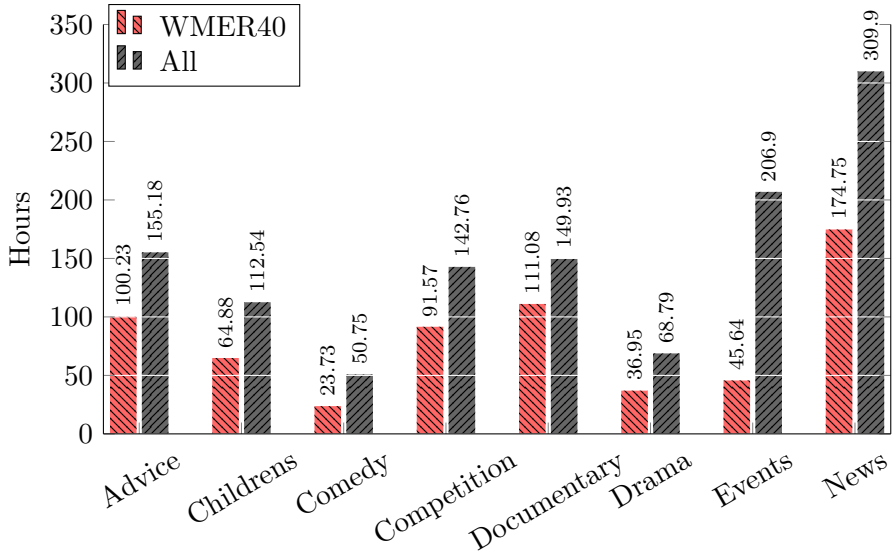


Figure 4.2: Hours in the MGB corpus by genre, considering all data in a genre or data subsetted by a word-level MER threshold of 40.

if the two axes, training and decoding genres, were independent (see figure caption for details). It shows whether a combination does better than the difficulty of that genre would imply. For example, the table shows that training on `drama` and testing on `comedy` performs surprisingly well, considering the average performance having trained on `drama` for other genres. Clearly, some genres seem well matched, e. g. `advice` and `documentary`. Yet, the delineation into genres may not be optimal, as suggested by work on discovering domains in the data (Doulaty et al., 2015).

The transcriptions in MGB stem from subtitles which were originally created through a "re-speaking" process. In re-speaking, a trained operator listens to audio while simultaneously speaking clearly into an ASR engine in order to generate close to real-time subtitles. As a result, these transcriptions often make simplifications and paraphrase the true speech (see Section B.1 for examples). Consequently, the data may require filtering or some selection method due to the varying label quality. A standard approach for the MGB corpus is to filter by a Matching Error Rate (MER) threshold.

The MER is the standard edit-distance metric computed between the transcriptions and a lightly-supervised decode and alignment (Bell et al., 2015; Long et al., 2013). It is computed either at the word or phone-level, and the lightly-supervised decode is obtained by recognising the data with a first-pass model (or some other seed model) and an LM biased towards the transcriptions (see Lamel, Gauvain, and Adda, 2002). If the transcriptions are the true verbatim transcriptions, then the MER computed at the word level is simply the standard WER. When the transcriptions may have errors with respect to the true speech, the MER may be considered a proxy for transcription quality that we can use to filter utterances where the MER exceeds a chosen threshold. Since the lightly-supervised decode is biased towards the transcriptions, any error between the decode and the transcriptions is an indication that the seed acoustic model disagrees

	advice	childrens	comedy	competition	documentary	drama	events	news
advice	-4.4	1.8	1.8	-0.37	-0.022	0.21	1.5	-0.54
childrens	0.95	-5.7	0.36	1.1	0.18	0.57	2.5	-0.043
comedy	3	-1.1	-5.7	1	2	-2.3	-0.48	3.6
competition	-1.3	0.54	0.91	-3.4	0.3	1.4	1.9	-0.32
documentary	-1.3	1.2	2.1	-0.1	-2.6	2	1.1	-2.4
drama	0.98	1.2	-2	1.2	0.44	-5.8	1.5	2.4
events	2.4	1.6	0.12	1.2	0.3	1.2	-7.8	1
news	-0.32	0.43	2.4	-0.66	-0.65	2.8	-0.21	-3.7

Table 4.1: Decoding across genres with MER 30, where the y axis indicates training sets and the x axis on which set the model was decoded. The table shows absolute differences from expected values of WER given the assumption that the two axes, training genres and test genres, are independent. That is, from a table of WERs, we compute the marginals for each axis and then obtain the expected WERs by normalising the outer product of the marginals. The final values are the expected WERs subtracted from the true WERs.

with the transcription, suggesting a true error. Existing MERs are included with the MGB corpus, computed with a lightly-supervised decode (for details on the setup see Bell et al., 2015). The effect of filtering using MER is demonstrated below, and will be discussed further in Chapter 5 on lightly supervised training. In the MGB challenge, most teams chose to subset data using word MER thresholds set in the range 30% to 50%. This is a reasonable range to obtain a suitable amount of data before the returns start to diminish. In the thesis we consistently filter MGB utterances that exceed a word-level MER of 40%.

We use two included development sets from MGB: `dev.full` contains 47 unique shows totalling 19.5 hours; `eval.long` is a set consisting of two series with a total of 19 episodes and 10 hours, designed for testing longitudinal algorithms. The best submitted hybrid system to the challenge obtained an error rate on `dev.full` of 24.9% with a 4-gram LM, and a model-combination of multiple hybrid and tandem systems obtained 21.8% (Woodland et al., 2015). The TDNN-F LF-MMI model topology used throughout the thesis (`Kaldi-1` in Appendix A) obtains 26.8% WER with a 4-gram LM biased towards the training transcriptions (see Section 5.4) and MBR decoding (Xu et al., 2011, see also Section 3.2), having trained on MER 40 filtered data with speed perturbation (Ko et al., 2015).

As an example of the effect of filtering, Figure 4.3 shows the error rates and numbers of hours selected for various choices of MER on the MGB dataset. Models were trained from scratch on MGB news data, using LF-MMI (`Kaldi-1` in Appendix A). The

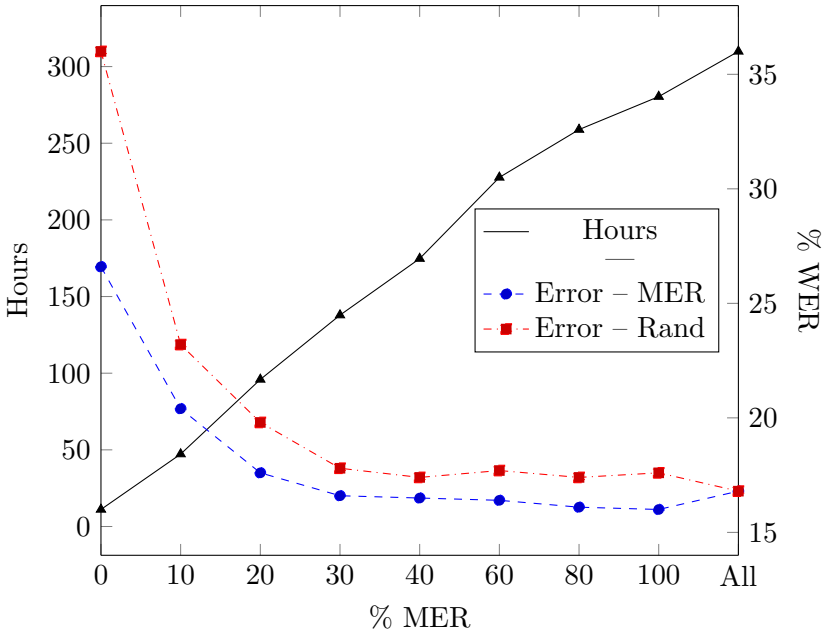


Figure 4.3: The effect of the choice of MER on the MGB corpus. Random selection matches the number of hours for the corresponding selection by MER.

models are evaluated on the news subset of `dev.full` (5 shows with total speech of 1.76 hours). Also included are randomly selected training subsets with their numbers of hours matching their corresponding MER-filtered counterpart. Clearly, using MER as a filtering metric outperforms random selection. A benefit of filtering using MER is in this case a reduction in training time[1] whilst maintaining similar WERs as to using the majority of the data. Note that there are 30 hours of utterances with MERs greater than 100 in the news training set, and when including these ('All'), the error rate increases compared to setting MER to 100. This likely explains why random selection does not match performance with MER filtering set to 100: selected utterances with very high MER affect random selection even with larger amounts of data. Table 4.2 further shows that, for a similar computational budget, it is better to choose an appropriate setting for MER, than to use all the data. In addition, using a very constrained set of data, yet of high quality (MER set to 20), does rather well with data augmentation, i. e. using speed perturbation (Ko et al., 2015).

4.2. Scottish Parliament

The Scottish Parliament Corporate Body publishes videos[2] of parliamentary sessions, including reports, question and answer sessions, and committee hearings. The data is used in Chapter 5 as adaptation data in a lightly-supervised scenario.

The accompanying subtitles from the official record are typically inaccurate with

[1] Although it may be possible to train for fewer epochs on more data without a noticeable drop in performance.

[2] As of writing there are 6,881 videos: `https://www.youtube.com/user/ScottishParl`

MER	Speed perturbation	Training Hours	WER (%)
20	No	95.9	17.6
20	Yes	$95.9 \times 3 \approx 288$	16.6
100	No	280.41	16.0
100	Yes	$280.41 \times 3 \approx 841$	16.4
All	No	309.90	16.8

Table 4.2: A comparison of using all MGB news data compared to a filtered subset with speed perturbation. Tested on `dev.full` filtered for news. For similar computational budgets, using MER set to 20 with speed perturbation is about as effective as using all the data; yet the best result is obtained with MER set to 100.

Dataset	WER (%)	Ins	Del	Sub
ScotParl Test (6.8 hours)	40.63	2.16	28.7	9.7

Table 4.3: WER (%) between aligned and segmented references and corresponding transcriptions for the test set (6.8 hours) of the Scottish Parliament data.

respect to the true speech, as demonstrated in Table 4.3, where we have computed the error rate between the subtitles and the verbatim transcriptions on the test set. This presents a similar problem as with the MGB corpus above with transcription quality. Empirically, large amounts of paraphrasing can be observed in the data (see Section B.2 for examples).

The data was scraped to produce roughly 289 hours, split by recording into train and test sets. The training set consists of 282 hours and is used to provide the baseline here for reference. 5.1 hours is randomly selected from the training set as adaptation data which will be used in Chapter 5 (with a model trained on different data). This subset consists of 1300 utterances across 374 speakers. On average the utterances contain 30 words each. The transcribed test set contains 6.8 hours of audio across 40 speakers. The total unique vocabulary in the true speech in the test data is 4,977 words. The vocabulary follows a typical Zipfian distribution with the 5 most common words[3] being `think`, `Scotland`, `government`, `care` and `people`, in that order. A large number of vocal ticks contribute to the number of deletions in Table 4.3. Apart from work contained herein[4], it is not clear that work using this data has been published elsewhere at the time of writing.

With the `Kaldi-1` model structure (Appendix A), excluding i-vectors and speed perturbation, we obtain on the test set WERs of 24.6% and 22.8% with 3-gram and 4-gram LMs, respectively. The LMs correspond to the non-biased MGB background LM from Section 5.4.

[3]Excluding stop words and vocal tics such as `um`.
[4]Including work from Chapter 5 which appears in Fainberg et al. (2019b).

Paper	Features	Model	WER (%)
Panayotov et al. (2015)	MFCC+CMLLR	DNN+pnorm	3.92
Amodei et al. (2016)	-	Humans	5.03
Amodei et al. (2016)	Log-spectrograms	E2E CNN+RNN	3.10
Chan and Lane (2015)	MFCC+CMLLR	CNN-DNN-BLSTM	3.50
Hadian et al. (2018b)	MFCC (+SP)	TDNN-LSTM E2E LF-MMI	3.00

Table 4.4: Results on the WSJ `eval92` set from the literature. The result from Amodei et al. (2016) used external training data ($\sim$ 12000 hours). SP stands for speed perturbation (Ko et al., 2015).

4.3. WSJ

The Wall Street Journal (WSJ) corpus (Paul and Baker, 1992) contains read speech from the newspaper of the same name, and is well studied in the speech literature (see Section B.3 for example transcriptions). This data is used in Chapter 6 with environmental noise from DEMAND as data for experiments factorising speakers and environments.

The WSJ corpus has served as a benchmark for ASR systems since its introduction: see Table 4.4 for some recent results on the `eval92` test set. It has also been used with external noise in the Aurora and CHiME challenges (e. g. Barker et al., 2015; Parihar et al., 2004). The `si284` training set contains 282 speakers over a total of $\sim$81 hours. There are about 17,000 unique utterances that are repeated twice with different speakers. The `eval92` and `dev93` test sets contain about 0.7 and 1 hours across 8 and 10 speakers, respectively, with mostly unique utterances. With the 6-layer TDNN model (`Kaldi-2` in Appendix A) we obtain 6.72% on `eval92` and 10.36% on `dev93` using a 3-gram LM trained on the training data.

4.4. AMI

The Augmented Multiparty Interaction (AMI) corpus (Carletta, 2007) consists of 70 hours of training data from fictitious design team meetings (see Section B.4 for example transcriptions). The data is used in Chapter 8 to build adult speech baseline models. The speakers in the AMI corpus speak English, although they speak many different dialects and there are also many non-native English speakers. In the meeting scenarios, short utterances are common, particularly because of affirmatives (e.g. "yeah"), as shown in Figure 4.4. There are multiple streams of audio, including far-field microphone arrays, individual lapel microphones, and Individual Head-Mounted Microphone (IHM). The latter, IHM, is exclusively used here. Some results from literature are shown in Table 4.5, with the best performing Kaldi recipe obtaining 18.9% WER on the dev set

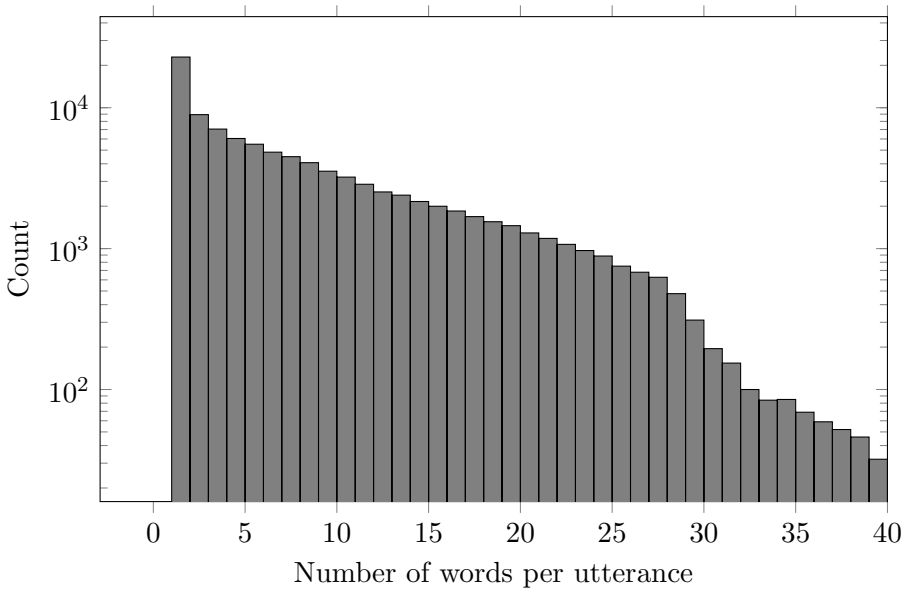


Figure 4.4: Histogram of the number of words per utterance in the training set for AMI. Note the logarithmic y-axis – many utterances in AMI are affirmatives.

Paper	Features	Model	Params	WER (%)
Renals and Swietojanski (2014)	FBANK+Δ+ΔΔ	DNN	∼ 30M	25.5
Platen, Zhang, and Woodland (2019)	FBANK	DNN+4gm	∼ 2M	28.3
Platen, Zhang, and Woodland (2019)	RAW×3	CNN×3+4gm	∼ 3M	27.2
Kaldi chain	MFCC(+SP)	TDNN	21.4	
Kaldi chain	MFCC(+SP+Reverb)	TDNN-LSTM	18.9	

Table 4.5: Results on the AMI dev set with IHM data from the literature. Platen, Zhang, and Woodland (2019) use three inputs across different time-spans of the same waveform. All models use ReLU activations.

with speed perturbation and reverberated data[5]. The far-field streams are used often for research on distant speech recognition, and are far more challenging. Renals and Swietojanski (2014) obtained 46.0% WER using a multi-channel far-field stream with beamforming and a large CNN-ReLU model, compared to 24.9% with the same model topology and the IHM stream. Chapter 8 will explore SincNet raw-waveform models trained om AMI, which as a baseline obtains the same 28% error rate with either a standard MFCC-based model, or a SincNet model using a flat initial filter distribution (see `Keras-1` and `Keras-2` in Appendix A).

[5] `https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/RESULTS_ihm`

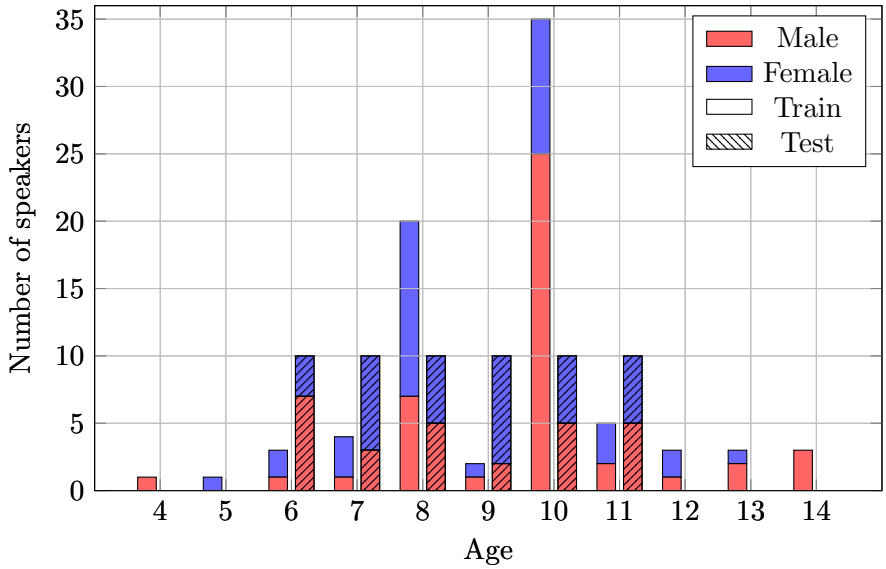


Figure 4.5: The distribution of age and gender of speakers in the PF-STAR training and test sets.

4.5. PF-STAR

The British English PF-STAR (Batliner et al., 2005) corpus is used in Chapter 8 as child speech adaptation data. It contains 14 hours of data of read children's speech, of which ~ 7.4 and ~ 4.7 hours are allotted for training and test sets (no speaker overlap), with the remaining in miscellaneous development sets, such as an evaluation set for speaker adaptation where utterances for the same speakers are split into `eval/adapt` and `eval/test`. The corpus contains children aged four to fourteen, but with the majority of children aged eight and ten, as shown in Figure 4.5.

From 921 unique prompts, 40 are shared between the training and test sets, making up roughly 11% and 7% of the unique prompts in each of those sets, respectively. The remaining prompts do not match, but are similar in style and content (see Section B.5 for example transcripts). That includes sequences of numbers and random nouns. Care should be taken when designing experiments or making claims on state-of-the-art results. Fainberg et al. (2016) obtained an error rate of 29% on the test set using the trigram language model from MGB (see Section 5.4). The model consisted of six 1024-dimensional feedforward layers (no convolutions) with sigmoid non-linearities, and used Restricted Boltzmann Machine (RBM) pre-training (Salakhutdinov and Hinton, 2009). In the work in Chapter 8 we bias the language model on purpose, as we are only interested in the acoustic model. That model obtains 20.46% on PF-STAR (see `Keras-2` in Appendix A).

4.6. A note on scoring

For key results we perform significance testing using the matched pairs test from the NIST Scoring Toolkit (National Institute of Standards and Technology (NIST), 2007). The matched pairs test compares errors on matching speech segments, or utterances, from two systems recognising the same test set. The null hypothesis is that the average difference of errors between the two systems, across all segments, is zero (Gillick and Cox, 1989). Occasionally we may also run the probability of improvement metric proposed by Bisani and Ney (2004) which is computed as the fraction of bootstrap samples that improve the error rate.

4.7. Summary

The next chapters will make use of these datasets to explore the challenges identified in Chapter 1. Specifically, in Chapter 5 we will use the datasets with inaccurate labels, MGB and ScotParl, to study a lightly supervised technique that can make use of these labels. The clean WSJ corpus is suitable to combine with environmental noise to produce data for experiments on factorised adaptation in Chapter 6. The longitudinal development set in the MGB corpus will be used to study longitudinal training in Chapter 7. Finally, AMI and PF-STAR will be used to explore adaptation from adult speech to child speech in raw-waveform models in Chapter 8.

Chapter 5

Lightly-supervised training

While the majority of speech data "in-the-wild" is untranscribed, there are many sources that naturally include some variety of transcription – although inaccurate with respect to the true speech. However, many training paradigms assume verbatim transcriptions (and are sensitive to errors). This chapter addresses the challenge from Chapter 1 on how best to make use of inaccurate transcriptions, which is commonly termed lightly-supervised training. This term encompasses situations in which the text data contain errors with respect to the speech and when text data may be correct but the timings are incorrect. There are other training paradigms that use no transcriptions, known as semi-supervised training. However, we believe that there is a benefit to incorporating possibly inaccurate text and the information that it contains. This chapter reviews lightly-supervised methods and presents experiments comparing the use of light supervision over a purely semi-supervised approach. A new method is proposed for lightly-supervised training using lattice supervision and sequence-discriminative training with the LF-MMI criterion, which effectively combines the benefit of a lattice-based semi-supervised method with light supervision from transcriptions.

5.1. Introduction

ASR systems are ideally trained on accurate transcripts that match the audio, but obtaining manual transcriptions is expensive. There is, however, a large amount of available data in some domains that has inaccurate or partial transcripts. Examples include traditional broadcast data (Bell et al., 2015; Driesen and Renals, 2013; Graff, 2002; Long et al., 2013), YouTube data (Liao, McDermott, and Senior, 2013), medical data (Mathias, Yegnanarayanan, and Fritsch, 2005), and children's speech (Nicolao, Sanders, and Hain, 2018). The MGB corpus and data from the Scottish Parliament, introduced in Chapter 4, are examples of data with inaccurate transcripts used in this thesis.

Training directly on possibly inaccurate transcripts may, as we will see, may lead to suboptimal results. On the other hand, semi-supervised training would discard the potentially valuable transcripts altogether. There is therefore a growing body of work on *lightly-supervised* methods that aim to make best use of partial or inaccurate transcriptions for training acoustic models. Most of these methods make use of automatic

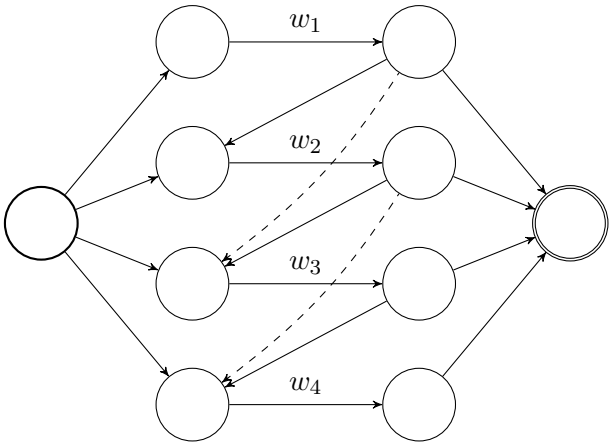


Figure 5.1: A sequence-net (without dashed connections). With dashed connections it is a skip-net allowing skips of one word at a time. Factor transducers can be seen as a general framework encompassing both of these architectures.

transcription hypotheses of the data, generated by decoding with a seed model together with a Language Model (LM) biased towards the domain of the transcriptions or just the transcriptions themselves (Braunschweiler, Gales, and Buchholz, 2010; Chan and Woodland, 2004; Driesen and Renals, 2013; Lamel, Gauvain, and Adda, 2002; Long et al., 2013). The hypotheses can then be compared to the transcriptions and the result of the comparison can be used to *filter* the transcriptions (or occasionally the hypotheses) at various levels of granularity. Alternatively, the hypotheses can be used in some *combination* or error correction algorithm with respect to the transcriptions.

Filtering in this context was first introduced by Lamel, Gauvain, and Adda (2002) and was based on segment-level matching with biased LMs. This approach was later applied to discriminative training (Chan and Woodland, 2004). Typically, segments are filtered given a matching error rate threshold at the word (WMER) or phone level (PMER) between the transcriptions and the biased decode (Bell et al., 2015; Braunschweiler, Gales, and Buchholz, 2010; Lanchantin et al., 2016; Long et al., 2013), as seen in Chapter 4 for the MGB corpus. Methods operating at finer levels of granularity often include selecting *islands* of consecutive words with zero string edit distance (Driesen and Renals, 2013; Liao, McDermott, and Senior, 2013; Mathias, Yegnanarayanan, and Fritsch, 2005; Nguyen and Xiang, 2004), or to select words based on binary classifiers that are trained to choose between words in the hypotheses and words in the transcriptions (Li, Akita, and Kawahara, 2015). Another approach is to consider the match of two special transducers between the text and the recognised output: a *skip-net* which allows word-skips, and a *sequence net* which does not allow word-skips, shown in Figure 5.1). Training data can then be selected if the alignments from each transducer are equal (Driesen and Renals, 2013; Stan, Bell, and King, 2012). Alternatively, contiguous strings of reliable segments can be found by decoding with a factor transducer in place of G in the $HCLG$ (Bell and Renals, 2015; Moreno and Alberti, 2009), as shown in Figure 5.2.

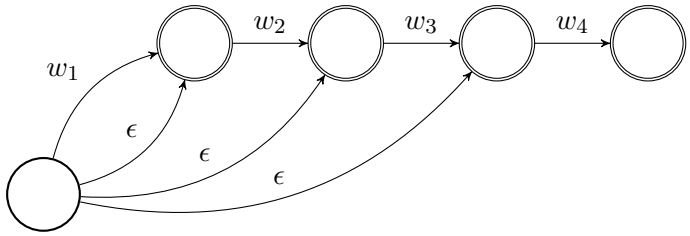


Figure 5.2: A factor transducer. Word-skip connections may optionally be added (Bell and Renals, 2015).

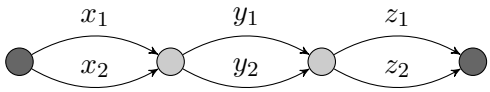


Figure 5.3: Example of a confusion network (sometimes known as a sausage lattice) (Mangu, Brill, and Stolcke, 2000).

Combination approaches, on the other hand, aim to maintain as much data as possible through correcting or combining the hypotheses with the transcriptions. Long et al. (2013) proposed a word-level combination scheme that uses ROVER (Fiscus, 1997) to select a sequence of words from reference transcriptions with the corresponding hypothesis lattices. Words in the reference that occur in the lattices are given a high score to force the selection of that word. A similar approach was used by van Dalen et al. (2015) for hypotheses from crowd sourced data. Manohar, Povey, and Khudanpur (2017) propose a different approach that combines four transcripts into a confusion network for training (see Figure 5.3). In Chen, Lamel, and Gauvain (2004) the authors align the transcription to a sausage lattice version of the hypotheses (i. e. a confusion network), and select words at each arc depending on the posterior probabilities of the words and the match with the aligned word. Venkataraman et al. (2004) proposed to align the data using a robust alignment procedure based on transducers that allow for words to be skipped, and certain insertions, for which transition probabilities are estimated empirically on held-out data. The resulting best path is used as training data. A related approach was taken by Nicolao, Sanders, and Hain (2018), in which they propose to use a transducer to model variations in children's speech, implemented as the grammar in the $HCLG$ for decoding to create improved transcriptions. In Olcoz, Saz, and Hain (2016) the authors propose to correct for word-boundary errors and insertions in a lightly supervised alignment. They compare alignments and their confidences from different acoustic models, and include a model specifically trained to detect insertions.

The benefit of the combination techniques is that they can maintain more data than through filtering. Filtering may perhaps be most appropriate given large amounts of data, but false rejects could be an issue because it may bias the selection away from difficult data. New acoustic models can be trained on the filtered or corrected transcriptions, and sometimes the process is repeated, yielding increasing model improvements (Nicolao, Sanders, and Hain, 2018; Stan, Bell, and King, 2012).

Outside of the ASR literature, there is a wide variety of work on training neural networks with inaccurate labels (e. g. Dehghani et al., 2018; Goldberger and Ben-Reuven, 2016; Sukhbaatar et al., 2015). Applying these techniques to ASR is not necessarily straightforward: unmodified they would operate at the frame-level, which does not necessarily correlate with improvements in WERs; and some techniques use another dense layer on top of the softmax layer, which for large-vocabulary ASR may prove too computationally expensive.

In contrast to the lightly supervised approaches above, semi-supervised training requires no manual transcriptions for new data, but generates hypotheses using a seed model, the choice of which impacts the quality of the hypotheses. In state-of-the-art approaches using the sequence-discriminative LF-MMI objective (see Povey et al., 2016, and Section 2.3.1), the decoding lattices are maintained and used as lattice supervision, effectively encoding the uncertainty of the hypotheses by the width of the lattice as we saw in Chapter 3 (see also Klejch et al., 2019; Manohar et al., 2018). As we mentioned earlier and as also remarked in related literature Manohar et al. (2018), this is beneficial for sequence-discriminative training which is sensitive to the accuracy of the supervision (Mathias, Yegnanarayanan, and Fritsch, 2005; Yu et al., 2010).

A new method for lightly supervised training is proposed in Section 5.2. This will make effective use of lattice supervision discussed in Section 3.3. Preliminary experiments on a possible extension to this work, making use of paraphrasing phenomena, are presented in Section 5.3. The contribution of the new method is three-fold:

- First, typical lightly supervised techniques produce linear transcription hypotheses on which to train. Yet, state-of-the-art semi-supervised, sequence-discriminative training techniques benefit strongly from lattice supervision which encodes the uncertainty of the data (Klejch et al., 2019; Manohar et al., 2018). Our experiments demonstrate that lightly-supervised training where the lattice supervision is generated with a biased LM can substantially improve WERs.

- Second, Long et al. (2013) showed that, instead of filtering, it is possible to combine inaccurate transcripts with a biased decode lattice to create an improved best path transcription on which to train. However, the output is a best path, not a lattice. Manohar, Povey, and Khudanpur (2017) demonstrated an algorithm that combines individual transcripts into a confusion network lattice for training, although this approach does not combine with a hypothesis lattice. Our proposed method combines the transcriptions and a hypothesis lattice, while, crucially, maintaining a lattice for supervision. This encodes uncertainty where the transcriptions and lattices disagree, whilst maintaining a narrow lattice where they do agree. Up to a 17.5% relative reduction in WER is obtained in experiments with respect to a semi-supervised baseline, or up to 12.5% with respect to a lightly-supervised best path baseline.

- Finally, our experiments suggest that the proposed method compensates for a large number of deletions in the ASR output. An alternative method is proposed, in which deletions are reduced by rewarding insertions when *generating the supervision lattices*. This reduces WERs by up to 13% relative. The combined improvements from the above ideas yield up to 20% relative WER reductions on broadcast data from the Scottish Parliament, adapting from a model trained on BBC news broadcasts from the MGB corpus (Section 4.1).

5.2. Lattice combination

This section presents a new method to combine lattices with inaccurate transcripts. The lattices will typically be the output of a decode with a biased LM, and the transcriptions are, in the following experiments, subtitles from broadcast data. As in Long et al. (2013), we make the assumption that if a word in the transcription is present in the lattice, then the word is likely correct; but also that a transcript word not occurring in the lattices is wrong. Seen from the view of the transcriptions we would like to substitute with hypotheses from the lattice when this is the case. Viewed from the lattice of hypotheses, we want to collapse the lattice onto words in the transcription when possible. This provides a narrow lattice where we believe it should be confident, and wide otherwise. Consequently, if the transcriptions are not at all present in the lattice, then the lattice is kept in its entirety. An example output of the algorithm is shown in Figure 5.4.

To perform the combination, we require a linear transducer, R, of a transcription, and a hypothesis lattice H, for a particular utterance (assuming an existing utterance segmentation), projected on words. All weights in the hypothesis lattice H are discarded. By creating an edit transducer, $E_{R,H}$, that allows for insertions $(\epsilon : w)$, deletions $(w : \epsilon)$, and substitutions $(w_i : w_j)$, across all the words in the complete vocabulary of R and H, the lattices are composed in the following order:

$$T = R \circ E_{R,H} \circ H. \tag{5.1}$$

An edit transducer can trivially be constructed by making a *flower automaton*, where each arc represents a specific insertion, deletion or substitution as shown in Figure 5.5. There are more efficient implementations in terms of complexity, as will be discussed below, but the results of composition are the same.

As the goal is to maximise the number of correct words in the lattice, not to minimise the number of edits, the standard costs for E are not used. Instead, every edit cost is set to 0, apart from matches $(w_k : w_k)$ which are set to -1. The path with the most correct words will then have the most negative total cost, i. e. the shortest path. Recall from Chapter 3 that the shortest path cost can be written as $\oplus_\pi w(\pi)$, where $w(\pi)$ is the cost of path π, and that we have the following relations in the min tropical semiring: $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}) \triangleq (\mathbb{R} \cup \{+\infty\}, \min, +, +\infty, 0)$. We retain only the paths with

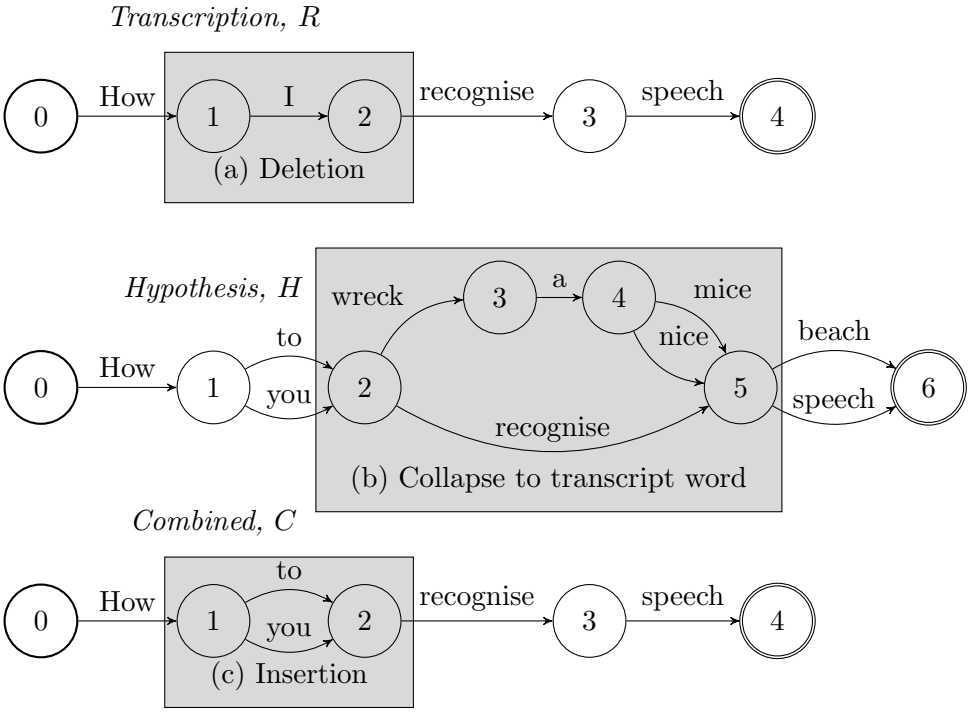


Figure 5.4: Example of the outcome of the combination algorithm. Words in the transcript not present in the hypotheses are deleted or substituted, while confusions in the hypotheses are collapsed onto transcript words where they exist.

that minimum cost, or some multiple of it, by pruning the transducer with a threshold t times the shortest path cost:

$$t \otimes [\oplus_\pi w(\pi)]\,, \tag{5.2}$$

where the sum is over all paths π in $R \circ E_{R,H} \circ H$. The pruning factor t is set to 0 ($\bar{1}$ in the tropical semiring) in the experiments below, unless noted otherwise. Since only matches have non-zero costs, sections in the hypothesis lattices without matches are kept with multiple paths. This is shown in Figure 5.6, given the example above in Figure 5.4.

To obtain the final combined lattice, the pruned transducer is first projected onto the output. This retains any substitutions made by the hypothesis lattice, and deletes words in R that were not present in H. Finally, epsilons are removed, and the result is determinised, and minimised. The complete set of operations are[1]:

$$C = \min(\det(\text{rmeps}(\text{proj}(\text{prune}(\underbrace{R \circ E_{R,H} \circ H}_{T}))))). \tag{5.3}$$

To use the combined transducer C as supervision, the resulting grammar is used

[1]Our notation here with C as the combined transducer is overloaded with the transducer for phonetic context dependency in the $HCLG$ decoding graph (Section 3.2). In this chapter C always refers to the combined transducer.

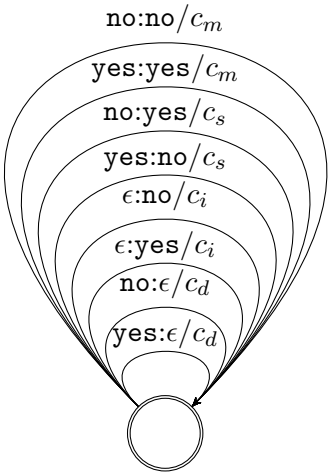


Figure 5.5: An example of an edit transducer for the vocabulary `yes`, `no`, with insertion costs c_i, substitution costs c_s, deletion costs c_d and matching costs c_m. Normally, for e. g. computing error rates, the edit costs are $c_i = c_s = c_d = 1$ and the match costs $c_m = 0$. For this algorithm we instead set edits to have costs $c_i = c_s = c_d = 0$, and matches cost $c_m = -1$.

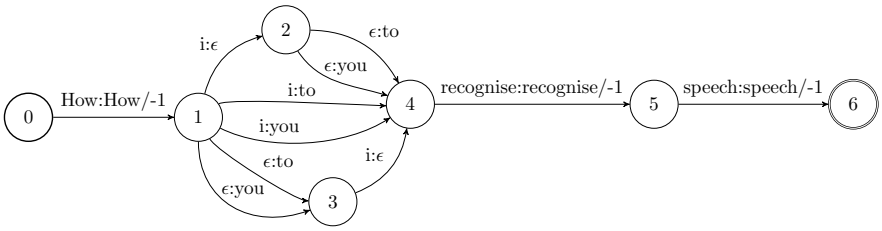


Figure 5.6: Combined lattice after pruning, prior to projection on output labels, epsilon removal and determinisation.

to compile new training graphs which are aligned to the data in order to add acoustic costs. LM costs may be reintroduced by composing it with G, the standard LM.

The edit transducer $E_{R,H}$ is not particularly efficient in that it needs to store $(|V| + 1)^2 - 1$ transitions for a vocabulary V, and the resulting search space is quadratic in the length of the input. This could be mitigated with factored transducers, three-way compositions or using a rho-matcher (see e. g. OpenFST, 2017). Note, however, that the computation time spent during lattice combination is negligible compared to the decode pass required to create H. A high level view of using lattice combination in practice is shown in Figure 5.7.

Shown in Table 5.1 are the expected WERs over the lattices with respect to the true transcriptions in the test set. That is, the errors are weighted given multiple aligned hypotheses to a particular reference word. The table indicates a considerable improvement with the combined lattices, and demonstrates the inaccuracy of the transcriptions. This can likely be attributed to a reduction in lattice depth where the captions are correct. Figure 5.8 shows histograms of lattice-depths per-frame, demonstrating a clear shift to more narrow lattices when using lattice combination.

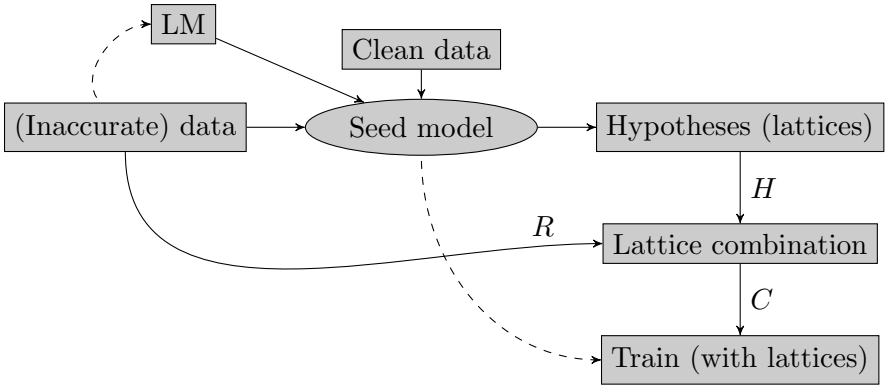


Figure 5.7: Lattice combination procedure in practice, with or without a biased LM when generating supervision.

Supervision	Lattice	Best path
Transcriptions (R)	-	40.63
Decode-bias (H)	35.51	32.74
Lattice combination-bias (C)	26.40	25.21

Table 5.1: Expected WERs (%) over lattices with aligned and segmented references on the test set of the Scottish Parliament Data. Best path indicates a comparison of the reference to the most probable path in the lattice, or just the transcriptions themselves. Compared to the transcriptions, biasing or using the lattice combination method reduces expected errors considerably.

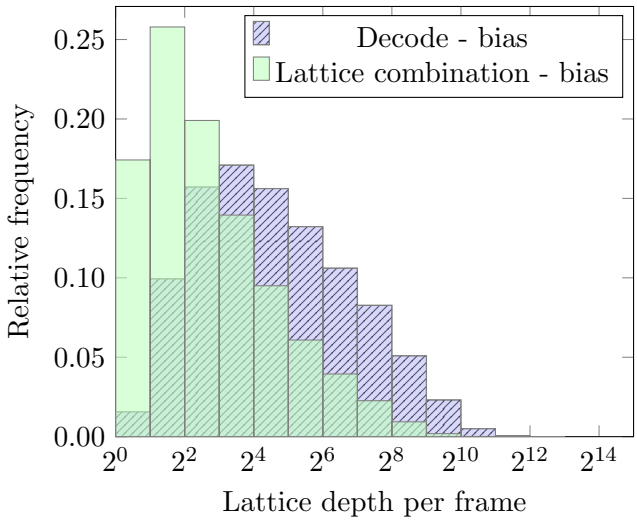


Figure 5.8: Histograms of lattice-depths per frame, i. e. on average how many arcs cross each frame. With lattice-combination, a significant fraction of frames have a lattice depth of 1, where words in the transcriptions matched words in the hypothesis lattices.

5.3. Paraphrasing

In much data with inaccurate transcriptions, the inaccuracies stem from paraphrasing. For example, in data from the Scottish Parliament (see Section 4.2), speakers often deviate from a formally prepared script, often opting for more casual language. There are also miscellaneous changes in the written record. In other words, the distinction between the surface form and semantics of natural language and the meanings they represent, become particularly evident with lightly supervised data. In broadcast media in general, the actual words spoken compared to the subtitles convey the same message, but with different surface forms. In Japanese, the written and spoken forms may differ substantially (Hori, Willett, and Minami, 2003). In machine translation, paraphrasing makes evaluation particularly difficult, leading to measures such as BLEU (Papineni et al., 2002).

The lattice combination algorithm presented thus far works by assuming a subset of correct words matching between the transcriptions and the ASR output. It improves upon semi-supervised training by matching sequences in the transcriptions with the hypotheses, and using the result as supervision. If the substitution errors in the transcriptions (with respect to the reference) are paraphrases, then incorporating knowledge of possible paraphrases of the data during lattice combination may help to further match paths in the hypothesis lattice. As observed in Section 3.3 on lattice supervision, if the true label is known, a narrow lattice is preferred. In situations with large amounts of paraphrasing, can we improve the result of the algorithm by introducing paraphrases into the transcriptions? An extension to the lattice combination algorithm is proposed below, using lexical paraphrases learned from data or extracted from an existing database.

Algorithm

We propose to augment the transcriptions with new paths containing possible paraphrases, and then to use these augmented transcriptions in the lattice combination algorithm. However, we would like to prioritise matching words between the original transcription and the hypotheses. Matching with the augmented transcriptions containing paraphrases should therefore occur after an initial lattice combination composition. This prioritises words in the transcriptions, and only applies the paraphrases to potentially remaining confusables. Previously unmatched hypothesis paths in the lattices become targets for potential improvements using the paraphrases. Conversely, if we were to apply paraphrases to the transcriptions, prior to lattice combination, we may unnecessarily widen the resulting combined lattice when the ASR errors match the paraphrases and the lattice should have been entirely collapsed onto the transcription.

We require an edit transducer containing only substitutions of the paraphrases, P, the lattice C (a lattice combination of H and R), and the transcriptions R. A second

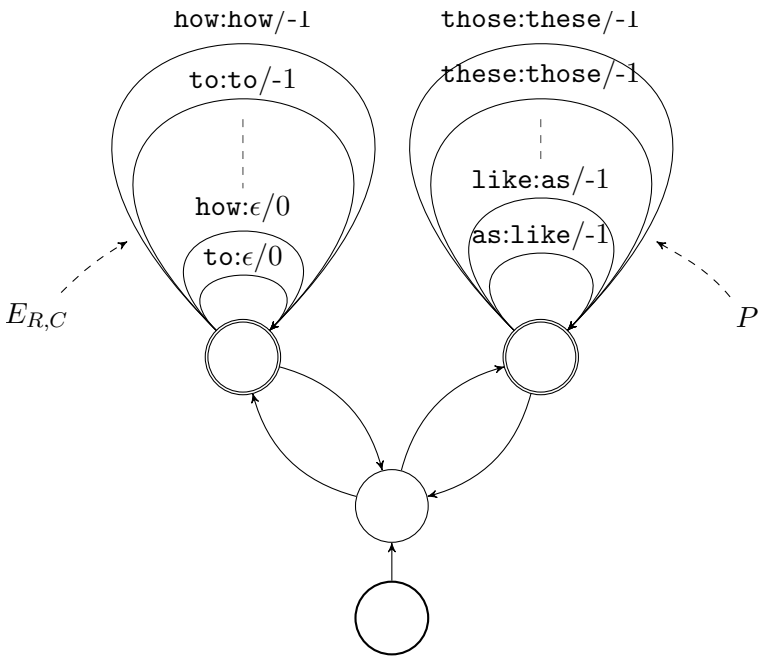


Figure 5.9: Illustration of the union, and subsequent closure, of $E_{R,C}$ and P, where $E_{R,C}$ is the edit transducer between the reference transcription and the previous result of lattice combination, and P is the paraphrase transducer that only contains word-substitutions. Paraphrases and matches have costs -1.

pass of lattice combination is performed using the union of the paraphrases and the edit transducer between the transcriptions and the already combined transducer, $E_{R,C}$:

$$T_p = R \circ (E_{R,C} \cup P)^* \circ C, \tag{5.4}$$

where * denotes the Kleene Closure operation (see Section 3.1). Figure 5.9 illustrates the Finite State Transducer (FST). Here we have used the transcriptions R two times: once to compute C (Equation 5.3), and then again with the paraphrases. As noted above, we make sure to complete C first, to prioritise words in the transcriptions.

T_p then undergoes the same steps as above to produce C from T in order to create C_p for use as supervision:

$$C_p = \min(\det(\text{rmeps}(\text{proj}(\text{prune}(T_p))))). \tag{5.5}$$

The success of this approach hinges upon obtaining appropriate paraphrases for P, the amount of paraphrasing that occurs between the audio and the captions in the data, and on the hypotheses. A limitation of this approach is also that it only allows for unigram, or word-level, paraphrases, i. e. the most limited form of paraphrasing.

There is a significant amount of work in the literature on paraphrase extraction. Much of this derives from work in Machine Translation (MT), using a second *pivot* language on parallel texts (Bannard and Callison-Burch, 2005). Distributional methods have been used to build paraphrase clusters (Pereira, Tishby, and Lee, 1993) and

Supervision	Lattice	Best path
Transcriptions (R)	-	40.63
Lattice combination-bias (T)	26.40	25.21
Lattice combination-bias-pp (T_p)	26.39	25.21

Table 5.2: Expected WERs (%) over lattices with aligned and segmented references on the test set of the Scottish Parliament Data. Best path indicates a comparison of the reference to the most probable path in the lattice, or just the transcriptions themselves. A second pass using paraphrases (pp) from PPDB provides nearly no reductions in expected WERs.

paraphrastic language models (Liu, Gales, and Woodland, 2014) from single bodies of text. Barzilay and McKeown (2001) propose to obtain paraphrases between multiple English translations of the same books.

For the purposes of the experiments below, an existing paraphrase database, PPDB[2] (Ganitkevitch, Durme, and Callison-Burch, 2013), is used. Dedicated paraphrase extraction is suggested as further work, and also relies on having suitable data. Lexical paraphrases that are annotated as equivalence relations are extracted from the large English database in PPDB, producing 35,735 paraphrase pairs. These are doubled to enable paraphrases in both directions. The paraphrases are used to directly create a flower transducer with cost -1 on each arc, as described above.

Table 5.2 shows experiments on the test set that demonstrate very subtle reductions in expected WERs over the lattices with respect to the true transcriptions, when using a second pass lattice combination with paraphrases. It may be that the technique would prove more promising on different data, with domain-specific paraphrases, and with changes to the algorithm to allow for phrase-level paraphrases.

5.4. Experimental setup

The baseline model is trained on news data selected from the MGB corpus (Bell et al., 2015), i. e. data with the metadata genre "news" (see Section 4.1). The news data is filtered using Word-Matching Error Rate (WMER) 40% with respect to the lightly supervised decode that is included in the MGB challenge. The resulting data consists of 179 hours across 545 shows, and approximately 15,600 (unlinked) speakers. This data is in a similar domain to our chosen adaptation data (below), and has fairly accurate labels (given the low WMER). It provides a realistic seed model for the adaptation experiments.

We use 5 hours of data from the Scottish Parliament as adaptation data (Section 4.2). This consists of 1300 utterances across 374 speakers. On average the utterances contain 30 words each. The test set contains 6.8 hours of audio across 40 speakers. The

[2] `http://paraphrase.org`

accompanying subtitles are inaccurate, as demonstrated in Table 5.1. Empirically one can observe large amounts of paraphrasing in the data.

Model

The baseline model is trained using Kaldi (Povey et al., 2011), and is based upon an existing Kaldi recipe for Switchboard[3], as also used in Chapter 4 for the MGB results for reference. This is a TDNN-F model (see Section 2.1.2) with 12 layers, each with 1280 units (apart from the penultimate layer), and bottleneck dimensions of 256. Interleaving the layers are ReLU activations, batchnorm and dropout layers. We train on alignments obtained from a standard HMM-GMM system like that described in Section 2.1.1, that matches the parameters set out in the MGB challenge (Bell et al., 2015). The model is trained with speed-perturbed (Ko et al., 2015) MGB news data for 8 epochs. The background trigram LM is trained on 640 million words of BBC subtitle text, and is restricted to the top 150,000 word types.

Biased n-gram LMs are estimated on the adaptation data, interpolating with the background MGB LM with a weight of 0.7. All models are evaluated with the background LM. During semi-supervised training, we train for 3 epochs with an initial learning rate of 5×10^{-5}. The lattice combination is implemented[4] using Kaldi (Povey et al., 2011) and OpenFST (Allauzen et al., 2007).

5.5. Results

Baseline results adapting to the raw transcriptions or in a semi-supervised manner are shown in Section 5.5.1. Results with the lattice combination are presented in Section 5.5.2, and with biased LMs in Section 5.5.3. Section 5.5.4 presents experiments with an alternative method to control for deletions in the supervision. Finally, Section 5.5.5 presents results with the paraphrase transducer.

5.5.1. Baseline model

The baseline model trained on MGB news data achieves 30.0% WER on the Scottish Parliament test data, as shown in Table 5.3. Adapting using the unfiltered transcriptions as supervision increases the error rate to 33.2%. This is expected given the high error rate of the transcriptions with the true reference (Table 5.1). The large proportion of deletion errors suggests that the transcriptions have failed to account for words present in the audio.

In contrast, adapting in a semi-supervised manner, having generated supervision with a decode of the data, improves results. Primarily the number of deletions have dropped, which indicates that the semi-supervised supervision has filled in deletions

[3] `Kaldi-1` in Section A.3.

[4] The implementation is available on `github.com/jfainberg/lattice_combination`.

Method	WER (%)	Sub	Del	Ins
Baseline (MGB)	30.0	15.8	11.3	3.0
Transcriptions	33.2	11.0	20.5	1.7
Semisup	28.6	11.7	14.7	2.1
Semisup-best path	28.8	12.9	13.8	2.1
Transcriptions MER 40	27.5	12.1	13.0	2.3
Transcriptions-exp MER 40	27.4	12.6	12.3	2.5
Lattice combination	23.6	10.7	10.7	2.3
Lattice combination-best path	25.2	12.2	10.2	2.7

Table 5.3: Results (%) with lattice combination and standard semi-supervised using the ScotParl adaptation set and evaluated on the ScotParl test set (Section 4.2). The baseline was trained on news data from the MGB corpus (Section 4.1). Transcriptions(-exp) MER refers to a subset of the transcriptions that have been filtered with respect to the MERs with best-path transcripts from the semi-supervised lattices, or across the lattices (expected MER).

that were absent in the transcriptions. Training using the best path is worse than using the entire lattice, which is consistent with the literature (Klejch et al., 2019; Manohar et al., 2018).

Filtering the transcripts given MERs between a best-path of the semi-supervised lattices or across the lattices (expected MER) improves upon the semi-supervised results, while introducing one hyperparameter. A dependence on the MER threshold is shown in Figure 5.10. In this case, a smaller amount of data, but higher quality, provides better results than a purely semi-supervised lattice-based approach. Note that the transcriptions necessarily have a lattice-depth of 1.

5.5.2. Lattice combination

The last rows of Table 5.3 show the results of the lattice combination method compared with semi-supervised and filtering approaches. The combined approach reduces WERs up to 17.5% relative to `Semisup`, and 13.9% relative to the filtered result with MER 40, with in both cases a substantial drop in deletion and some substitution errors. As discussed above, the key difference between the combined supervision and the semi-supervised approach is that the decoded lattices typically contain multiple confusable hypotheses where they match the transcriptions, while the combined lattices will have a lattice depth of 1 in these cases. This is reflected in the average lattice depth of the supervision lattices: 21.2 for the combined lattices compared to 78.1 in the original hypotheses. Additionally, the gap between best path and lattice supervision for the combined approach is larger, suggesting that it is benefiting from uncertainty encoded by the wide lattices where it was joined with the original decode. Figure 5.10 further compares lattice-combination applied after filtering with MER, demonstrating that

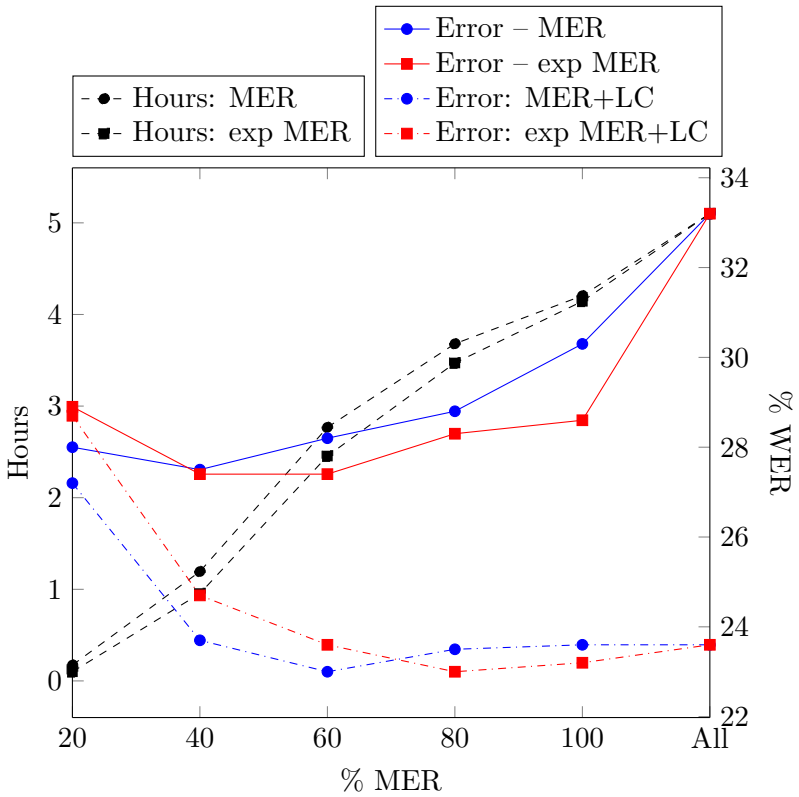


Figure 5.10: ScotParl adaptation results with transcriptions filtered with varying thresholds with respect to the best path (MER) or the lattice (exp. MER) of a (non-biased) semi-supervised recognition pass, as well as the lattice combination (LC) result on top of the filtered transcriptions.

Method	WER (%)	Sub	Del	Ins
Baseline (MGB)	30.0	15.8	11.3	3.0
Transcriptions (no bias)	33.2	11.0	20.5	1.7
Transcriptions-bias-MER 40	27.5	12.6	12.3	2.6
Transcriptions-bias-exp MER 40	26.8	12.0	12.6	2.3
Semisup-bias	26.8	11.7	13.1	2.0
Semisup-bias-best path	26.6	12.0	12.5	2.1
Lattice combination-bias	23.3	10.8	10.2	2.4
Lattice combination-bias-best path	25.0	12.0	10.2	2.8

Table 5.4: Results (%) with an LM biased to the ScotParl adaptation data, evaluated on the ScotParl test set (Section 4.2). The baseline was trained on news data from the MGB corpus (Section 4.1).

the use of lattice-combination is robust to poor transcriptions. In contrast, filtering transcriptions leads to increased WERs with higher MER thresholds.

5.5.3. Biased language model

The results in Table 5.4 demonstrate the benefit of including a biased LM when generating lattice supervision. For all methods shown, it reduces the WER by up to 10% relative, compared to Table 5.3, benefiting both lattice and best path supervision. Unsurprisingly, the standard semi-supervised method seems to benefit more from biasing the LM than the combined method[5]. Our combined method, however, also continues to work even with a biased LM. The results with MER filtering now demonstrate that the standard lightly-supervised filtering approach with an appropriate threshold matches the performance of the biased semi-supervised result. It no longer improves upon the semi-supervised result, suggesting that the two approaches obtain similar benefit from the light supervision in the transcriptions. The sweep of MER for the biased LM case is shown in Figure 5.11, showing similar trends to the non-biased case of Figure 5.10.

[5]Technically, having used the biased LM this is no longer strictly semi-supervised, but in effect lightly-supervised.

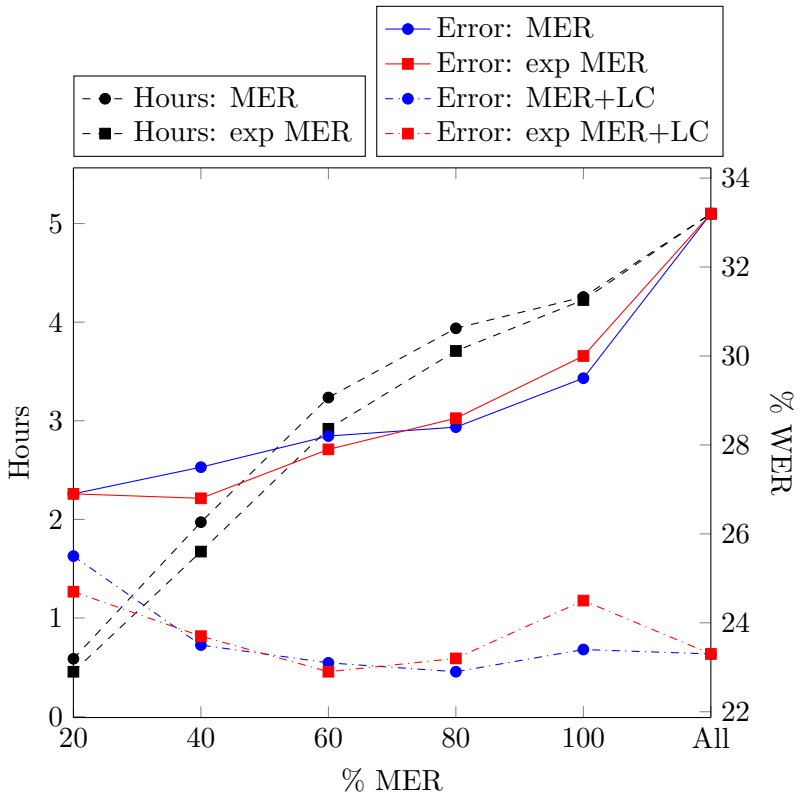


Figure 5.11: ScotParl adaptation results with transcriptions filtered with varying thresholds with respect to the best path (MER) or the lattice (exp. MER) of a biased semi-supervised recognition pass, as well as the lattice combination (LC) result on top of the filtered transcriptions.

5.5.4. Controlling for deletions

The results shown thus far indicate that the seed model is inclined to delete, which has affected the generation of supervision lattices. Deletion errors account for more than half of the errors when using `Semisup` supervision. In contrast, the `Combined` method appears to control for deletions, likely because it has prioritised words over pauses in the combination. We may consider whether there is another option to compensate for a tendency to delete. A proposal to achieve this is to penalise deletions (or reward insertions) in the HCLG decoding graph (see Section 3.1) when generating the lattices. This is implemented by subtracting a constant from every word output label in the HCLG graph. Note that a deletion penalty on the final decode to tune the ratio of insertions to deletions is no longer current practice. Indeed, a penalty on the final decode was not helpful on this data in preliminary experiments. The proposal is instead to include a penalty for a specific type of training, and crucially only during the generation of supervision. The final model is decoded in the standard fashion.

The effect of including a deletion penalty is shown in Figure 5.12. All models benefit from the penalty, with the least effect on those trained with the combined lattices, since these already controlled for deletions to some extent. The detailed results with the optimal penalty for our experiment are shown in Table 5.5. By including this penalty for supervision generation, the WERs after adaptation drop, along with the number of deletion errors, by up to 13% relative. The combined method now yields 22.8% WER, a small improvement upon the previous result (Table 5.4). It still improves upon standard semi-supervised training, but the difference is now less. However, tuning the deletion penalty hyperparameter is time-consuming, as it requires entire passes of decoding to generate supervision. The non-penalised combined result is close to the best overall result.

With a deletion penalty of 3 the lattices grow very large, increasing disk usage and the time to generate training examples, by several orders of magnitude. This is reflected in the average lattice depths, which for the semi-supervised lattices is now 418.0 (originally 78.1), compared to 10.7 (originally 21.2) for the combined lattices. Note that the lattice depth for the combined lattices has actually reduced, since the transcriptions are likely to match to more words, as more words are present in the hypothesis lattices.

Figure 5.13 shows that increasing the pruning threshold, t, from Equation 5.2, yields increasingly worse supervision for each choice of deletion penalty. The additional maintained paths do not improve upon the strictest criteria of only keeping the minimum cost path.

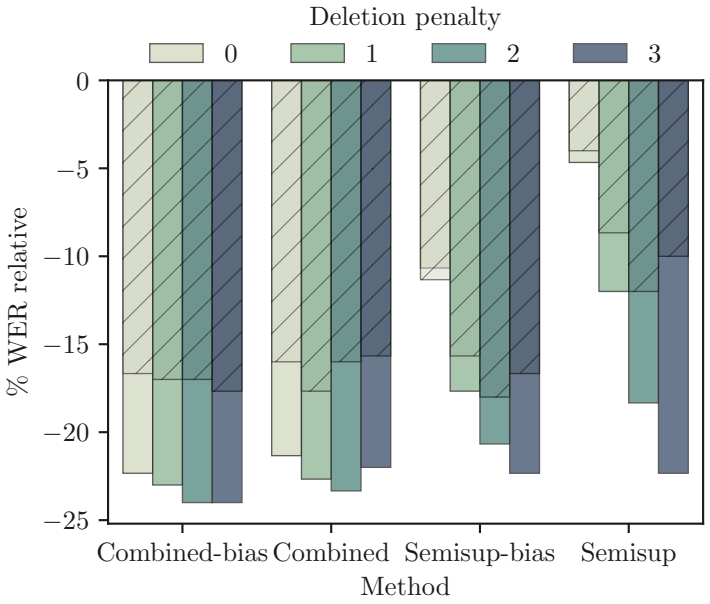


Figure 5.12: Results WER (%) on Scottish Parliament data with respect to the baseline (30% WER) for increasing deletion penalties when generating supervision lattices. The overlaying, hashed, bars represent the performance when using the corresponding best path supervision. The effect of the deletion penalty is more pronounced for the standard semi-supervised results, while the combination method is more robust to deletions in the supervision.

Method	WER (%)	Sub	Del	Ins
Baseline	30.0	15.8	11.3	3.0
Transcriptions	33.2	11.0	20.5	1.7
Semisup-bias	23.3	11.2	9.4	2.7
Semisup-bias-best path	25.0	12.8	8.4	3.9
Semisup	25.8	12.7	9.6	3.5
Semisup-best path	27.0	13.8	9.1	4.1
Lattice combination-bias	22.8	10.9	9.0	2.9
Lattice combination-bias-best path	24.7	11.9	9.9	2.8
Lattice combination	23.4	12.1	7.8	3.5
Lattice combination-best path	25.3	12.9	9.0	3.4

Table 5.5: Results WER (%) using a deletion penalty of 3 when generating supervision lattices. Transcriptions are not affected. The difference between using biased lattice combination compared to biased semi-supervised training is statistically significant with $p < 0.001$. The use of a biased supervision lattice compared to a best path is similarly significant, as well as the use of biasing for lattice combination. See Section 4.6 for details on significance testing.

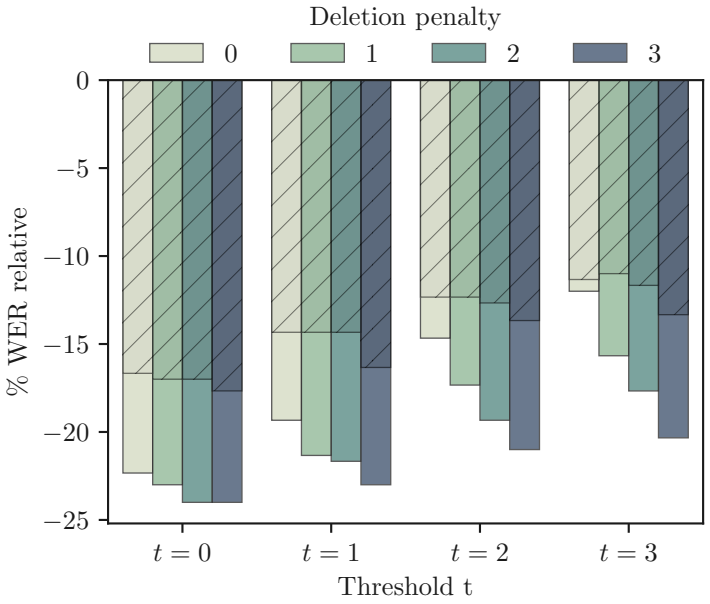


Figure 5.13: Results on Scottish Parliament data using different pruning thresholds, t, along with deletion penalties when generating supervision as in Figure 5.12. Increasing the threshold consistently reduces the WER reduction.

5.5.5. Using external paraphrase data

Table 5.6 shows results after a second pass of lattice combination with the paraphrase transducer from Section 5.3. For a deletion penalty of 0 there is no improvement when using lattice supervision, but with a penalty of 3 there is a modest improvement in WER from 22.8 to 22.6, which is the best result obtained in this chapter. This suggests that the additional hypotheses generated with the high deletion penalty provides more words for the paraphrase transducer to match with. Note that using a paraphrase transducer does improve best path supervision with 0 deletion penalty by a small amount, suggesting that the second combination corrected a few possibly incorrect paths.

5.6. Conclusions and future work

A method has been proposed for lightly supervised training: to combine inaccurate transcriptions with the decoded hypothesis lattices of a seed model. We have shown that this method enables the robust use of inaccurate transcripts during training, exceeding a semi-supervised method. Specifically, the proposed method produced an improvement upon a purely semi-supervised approach by up to 17.5% relative. Biasing the background language model to the data was found to substantially improve both the semi-supervised training and the lattice combination technique. The use of a deletion penalty during hypothesis lattice generation was effective when using a seed model that was prone to delete. The selection of the deletion penalty was selected empirically, balancing the

Method	Penalty	WER	Sub	Del	Ins
Baseline	-	30.0	15.8	11.3	3.0
Transcriptions	-	33.2	11.0	20.5	1.7
Lattice combination-bias	0	23.3	10.8	10.2	2.4
Lattice combination-bias-best path	0	25.0	12.0	10.2	2.8
Lattice combination-bias-pp	0	23.4	10.3	10.8	2.3
Lattice combination-bias-pp-best path	0	24.7	11.7	10.5	2.5
Lattice combination-bias	3	22.8	10.9	9.0	2.9
Lattice combination-bias-best path	3	24.7	11.9	9.9	2.8
Lattice combination-bias-pp	3	22.6	10.5	9.5	2.7
Lattice combination-bias-pp-best path	3	24.7	11.9	9.9	2.9

Table 5.6: Results with lattice combination with paraphrasing (pp) on Scottish Parliament data. The improvement with paraphrases on top of biased lattice combination with deletion penalty -3 is statistically significant with $p < 0.05$ (see Section 4.6).

increased number of hypotheses with rapidly growing lattice sizes. Finally, a two-pass approach making use of external, lexical paraphrases provided marginal improvements when combined with a high deletion penalty. The combined use of the above ideas produced a relative WER reduction of about 21% (from 28.6 to 22.6) with respect to the semi-supervised result.

Possible improvements to the algorithm include improved efficiency in the FST operations, particularly in the construction, and composition, of the edit transducer. Incorporating lexical paraphrases in an effort to extend the above work was not as successful, yielding only marginal reductions in WERs. It may be that this framework is more effective in a different scenario where certain paraphrases are well-known. There is also much scope in dedicated paraphrase extraction when in-domain data is available, as well as exploring ways to learn to use distributional methods or borrowing from machine translation literature. Finally, a key limitation is the use of lexical paraphrases, rather than phrases. Future work could explore a modification to the algorithm that enables the use of (possibly weighted) paraphrases. Bearing in mind that the number of possible paraphrases may considerably expand, an important aspect will likely be to obtain a small set of suitable paraphrases for the task at hand.

Chapter 6

Factorised representations

Adaptation of acoustic models is typically considered in terms of a single factor, usually a speaker. Learning a joint transform to a group of factors may further reduce the WER. However, explicitly modelling each factor independently, such as a speaker and an environment, yields additional practical benefits. This chapter addresses the challenge from Chapter 1 of how feature representations may be factorised, such that they can be combined in novel combinations at test time. That is, we may have observed a particular speaker and environment at training time, but never together. If we can learn independent representations for this speaker and this environment, then we can directly make use of these at test time when they occur together. We propose a method to factorise i-vectors into speaker and environment transforms using neural networks. Experiments show improvements using the proposed factorisation approach, both when one of two factors is unknown at test time, and when existing representations were extracted in mismatched conditions.

6.1. Introduction

Acoustic mismatch between training and test conditions may significantly affect acoustic models for speech recognition. Adaptation to speakers has proven particularly effective in reducing WER (Gales, 1998; Neto et al., 1995; Saon et al., 2013; Swietojanski, Li, and Renals, 2016). There are, however, other acoustic factors that affect the speech signal in addition to the speaker. Modelling these factors, such as environments, may reduce acoustic mismatch and can yield additional reductions in WER (Karanasou et al., 2014; Khokhlov et al., 2019; Saz and Hain, 2017; Seltzer and Acero, 2011; Swietojanski, Li, and Renals, 2016). Particularly, learning a single transform to a joint combination of factors, such as a speaker and an environment combination, can reduce WERs over speaker adaptation alone (Swietojanski, Li, and Renals, 2016). This, however, requires sufficient adaptation data for each combination of factors. Further, it does not make efficient use of existing information of, for example, the same speaker seen in a different environment than the present. When using transforms estimated in mismatched conditions, performance typically degrades (Karanasou et al., 2014).

A number of studies have investigated methods to adapt to each factor with separate transforms (e. g. Gales, 2001; Karanasou et al., 2014; Seltzer and Acero, 2011; Seo,

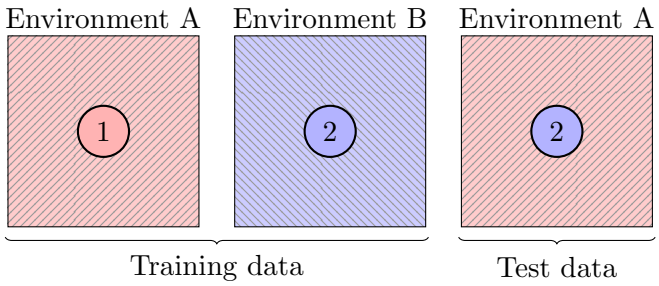


Figure 6.1: If speaker and environment transforms are independent, then a speaker transform for speaker 2, and an environment transform for environment A, may be combined to reduce acoustic mismatch in a novel combination at test time (speaker 2 and environment A).

Kang, and Seltzer, 2014; Wang and Gales, 2013, and also Table 6.1). For example, Swietojanski, Li, and Renals (2016) linearly interpolate transforms obtained using LHUC (see Section 2.4). Speaker transforms were estimated from clean training data, and environment transforms from pooling data across speakers for a particular environment, excluding the speaker in question to avoid a joint transform. The authors found that transforms estimated jointly to a combination of factors perform best, but a combination of separate transforms closely follow, and improve upon solely using a speaker transform. Li, Huang, and Gong (2014) modelled factors as additional inputs to the softmax layer of a neural network acoustic model, where each input has its own weight matrix dependent upon a particular factor as input. They use a noise factor which is estimated using frames from the current utterance. Seltzer and Acero (2011) proposed to cascade speaker and environment transforms using CMLLR. Each transform is optimised alternately with suitable data, similar to the approach above. In experiments with HMM-GMM models, they find that the method enables the use of the same speaker transform in multiple environments without degradation. This suggests that the speaker transforms are independent from the environments. More generally, if the adaptation transforms for each factor are independent, then they can be combined in novel combinations not seen in the data, and existing transforms may be reused and recombined. An example scenario is shown in Figure 6.1. If they are, however, not independent, then the nuisance information from irrelevant speakers or environments may limit improvements, as we will see below.

Independence may occur implicitly when estimating the adaptation transforms if the nuisance factor is evenly distributed within the data. This is, however, an assumption not always found in real data and explicit approaches to obtaining independent transforms are sometimes required. One explicit approach makes use of the observation that, if speaker and environment transformations are independent, then their first derivatives (i. e. differentiating one with respect to the other) should be zero (Karanasou et al., 2014; Wang and Gales, 2013). This has been shown using constrained optimisation for cluster adaptive training (Wang and Gales, 2013) and i-vectors (Karanasou et al., 2014). In

Paper	Method
Swietojanski, Li, and Renals (2016)	Interpolated LHUC
Karanasou et al. (2014)	Factorised i-vectors
Seo, Kang, and Seltzer (2014)	Orthogonal subspaces
Wang and Gales (2013)	Factorised cluster adapative training
Gales (2001)	Product of MLLR transforms
Seltzer and Acero (2011)	Cascade of CMLLR transforms

Table 6.1: Related approaches in the literature.

the latter paper 10% relative reductions in WERs are obtained with factorised i-vectors over normal speaker i-vectors on a perturbed set of the WSJ corpus (Section 4.3). A somewhat similar approach is taken by Seo, Kang, and Seltzer (2014), where they obtain independent CMLLR-style transforms by projecting adaptation transforms onto orthogonal, factor-dependent subspaces. They show a 7.5% WER reduction from an un-adapted baseline adapting to speakers using environment-independent transforms on Aurora4 (Parihar et al., 2004), compared to a WER increase of up to 24% relative when using transforms estimated in mismatched environments.

This chapter investigates the use of neural networks to factorise adaptation transforms. This is achieved by extracting bottleneck features from networks trained to classify speakers or environments given speaker or environment i-vectors, respectively (Section 6.2). The overall goal of the chapter is similar to that of i-vector factorisation (Karanasou et al., 2014). A key difference is that Karanasou et al. (2014) factorise the i-vectors during extraction by means of optimising under constraints, while this chapter demonstrates the possibility of factorising i-vectors after extraction. This opens the possibility of factorising any feature representation, since the process is not tied to the specifics of i-vector extraction.

Results are shown on the WSJ corpus (Section 4.3) perturbed using the Diverse Environments Multichannel Acoustic Noise Database (DEMAND) (Thiemann, Ito, and Vincent, 2013). To avoid implicit factorisation during i-vector extraction at test time, we explicitly avoid balancing environments across speakers and speakers across environments (Section 6.3). We first show that environment information in the factorised speaker representations is significantly reduced, while maintaining speaker information, and similarly with speaker information in environment representations (Section 6.4.1). We then show improvements when either speaker or environment information is absent at test time (Section 6.4.2). Lastly, we experiment with the situation in which we reuse speaker and environment transforms which have been estimated in mismatched conditions and where there is no adaptation data in the given combination (Section 6.4.3). For this experiment, the factorised representations are shown to be particularly important.

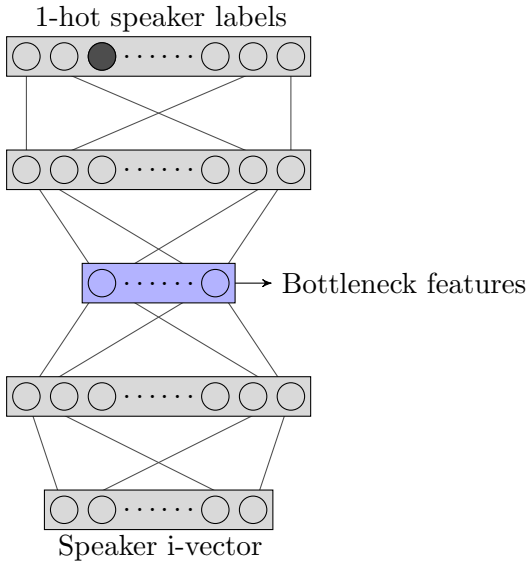


Figure 6.2: Bottleneck feature extractor. When learning to classify speakers the bottleneck features are not required to maintain possible nuisance information present in the speaker i-vectors pertaining to the environment.

6.2. Multi-condition neural networks

Recall from Section 2.4 that i-vectors, $\boldsymbol{\lambda}_f$, represent the difference between factor-specific (typically speaker) GMM means, $\mathbf{m}_f$, and means from a background GMM, $\mathbf{M}$, by the following relationship:

$$\mathbf{m}_f = \mathbf{M} + \mathbf{T}\boldsymbol{\lambda}_f \tag{6.1}$$

where $\mathbf{T}$ is the total variability matrix. We will consider speaker and environment i-vectors, $\boldsymbol{\lambda}_s$ and $\boldsymbol{\lambda}_e$, where in each case the i-vector is estimated from data pooled across a speaker (regardless of the environment), or across an environment (regardless of the speaker), respectively. (There will be certain exceptions to this in the experiments below.) As shown in Section 2.4, auxiliary features in the input to a neural network affect the next layer through the bias. When including a second i-vector for environment, there will be a second i-vector dependent bias term:

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b} + \mathbf{U}_s\boldsymbol{\lambda}_s + \mathbf{U}_e\boldsymbol{\lambda}_e) \tag{6.2}$$

where $\mathbf{W}\mathbf{x} + \mathbf{b}$ is the standard affine function of the input features, $\mathbf{x}$, and $\mathbf{U}_s$ and $\mathbf{U}_e$ are the weight matrices corresponding to each i-vector.

To factorise the i-vectors we experiment with using feedforward neural networks for classification. Specifically, the networks have as inputs either speaker or environment i-vectors and as outputs the corresponding speaker or environment classes, respectively. This is illustrated in Figure 6.2. This tests the notion that by learning to classify one factor, other nuisance factors are implicitly normalised out in the hidden representations since they are not relevant to the classification task. The extracted bottleneck features,

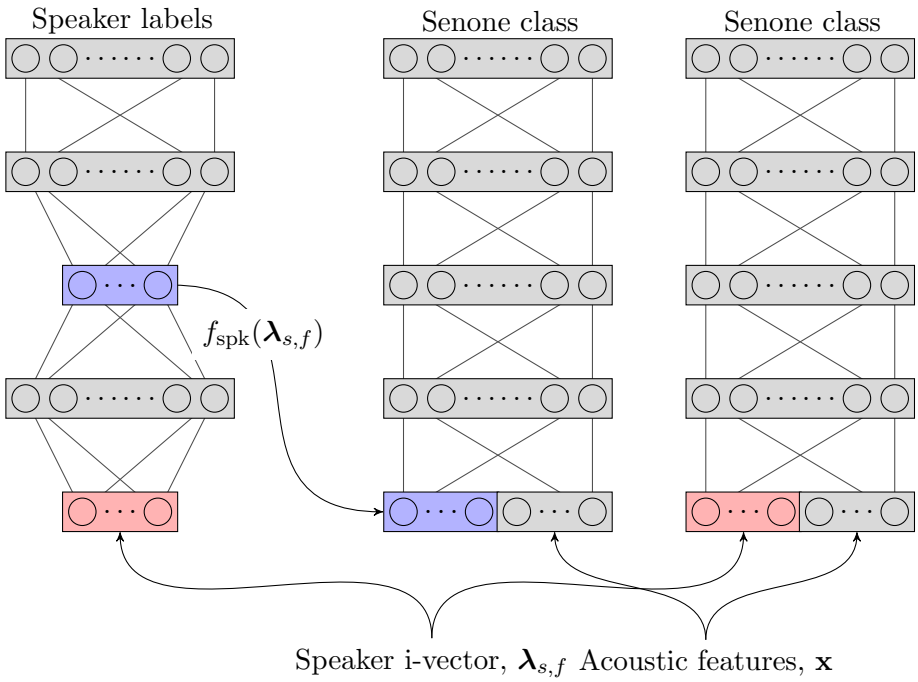


Figure 6.3: The original speaker i-vectors $\boldsymbol{\lambda}_{s,f}$, with possible nuisance factor f, are either concatenated directly with the acoustic features $\boldsymbol{x}$ (right), or first passed through a network that has been trained to classify speakers (left), represented by the function $f_{\text{spk}}(\boldsymbol{\lambda}_{s,f}) \approx \boldsymbol{\lambda}_s$. The resulting bottleneck features are then used in place of the normal i-vectors (middle).

`bn`, should mostly embody information only about a particular speaker or a particular environment. The bottleneck features are concatenated with the acoustic features in place of the standard i-vectors, and used to train the neural network acoustic model. At test time, the i-vectors, `iv`, are factorised by passing them through the existing networks, generating bottleneck features. Figure 6.3 shows the use of either normal i-vectors or the bottleneck features.

6.3. Experimental setup

Experiments are performed with the WSJ corpus (Paul and Baker, 1992) containing 282 speakers (Section 4.3). Noise sources are obtained from the Diverse Environments Multichannel Acoustic Noise Database (DEMAND) (Thiemann, Ito, and Vincent, 2013). This provides 18 recordings of environments ranging from residential environments to offices and transportation as shown in Table 6.2. Each noise recording is made with a 16 microphone array at 48kHz. Channel 1 of the array was arbitrarily selected to add noise to the single channel speech. The Signal-to-Noise Ratio (SNR) for each combination is set to 0 dB, so that both speakers and environments are treated equally, and the results are more easily interpretable. Note that no clean examples are included in the perturbed dataset.

Environments are randomly chosen for each training utterance. For test utterances

Category	Environments
Domestic	Kitchen, Living Room, Washing
Nature	Field, Park, River
Office	Hallway, Meeting, Office
Public	Cafeteria, Restaurant, Station
Street	Cafe, Public square, Traffic
Transportation	Bus, Car, Metro

Table 6.2: Environments in DEMAND.

the environments are applied to utterances in an unbalanced manner. Otherwise there is a risk of learning implicit orthogonality between the factors, since each speaker will have seen an even distribution of environments. A correlated set of environments per speaker is ensured by for each speaker, s, sampling a probability vector $\mathbf{p}_s \mid \alpha$:

$$\mathbf{p}_s \sim \mathrm{Dir}(\alpha) \tag{6.3}$$

where $\mathbf{p}_s \in \mathbb{R}^n$, n is the number of environments and α is the concentration parameter of a symmetric Dirichlet distribution. Then, for each utterance, u, from speaker s, an environment $k_u^{(s)}$ is sampled with respect to a categorical distribution on $\mathbf{p}_s$:

$$k_u^{(s)} \sim \mathrm{Cat}(\mathbf{p}_s) \tag{6.4}$$

This procedure ensures that speakers and environments are not independent, i. e.

$$p(s \mid n) \neq p(s), \quad p(n \mid s) \neq p(n) \qquad \forall s, n \tag{6.5}$$

such that both speaker and environment i-vectors do not see an even balance of the other factor.

For values less than 1.0 the distributions will be highly peaked, while $\alpha = 1$ provides a flat Dirichlet distribution, effectively a 17-dimensional uniform simplex. In the limit where $\alpha \to \infty$, the distribution over environments in $\mathbf{p}_s$ will be uniform. We chose $\alpha = 0.75$ as a reasonable compromise, i. e. each speaker is seen in several environments, but the distributions are distinctly non-uniform. Figure 6.4 shows an example of sampling a vector from a 2-simplex with $\alpha = 0.75$.

The i-vectors are obtained as typical in Kaldi (see e. g. Peddinti et al., 2015), with online and offline extraction for training and test data, respectively. Specifically, to obtain a suitable variety of i-vectors during training, each speaker (or environment) is split into "sub-speakers" that each have a maximum of two utterances from the original speaker. i-vectors are then extracted in an online fashion using only frames prior to the current frame within a sub-speaker. For our setup this means that the training time i-vectors will in effect represent joint speaker-environment transforms,

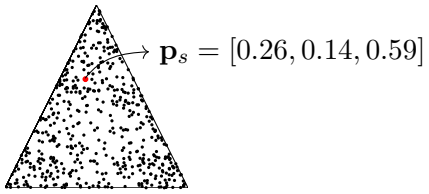


Figure 6.4: Sampling a vector $\mathbf{p}_s$ from a symmetric Dirichlet 2-simplex with concentration $\alpha = 0.75$. The dots represent random samples. As $\alpha \to 0$ they would concentrate at the edges, conversely, as $\alpha \to \infty$ they will move to the centre of the triangle.

due to the short time-span of the features used for extraction. During decoding the i-vectors are estimated in an offline fashion for higher quality representations. Normally this is split into subspeakers with 60 seconds minimum per speaker, but for more easily interpretable results a single i-vector is extracted across all the data for a given speaker or environment.

For the final experiment (Section 6.4.3) this procedure changes. We address the situation where speaker and environment i-vectors are extracted in mismatched conditions. To obtain results that are not biased towards particular environments the test sets now consist of an even balance of environments. The procedure for i-vector extraction is then as follows: For each possible speaker and environment combination, i. e. $\{s \in S, n \in N_s\}$, where N_s denotes the environments that have occurred with speaker s, we hold out each pair (s, n) in turn. The speaker i-vector for the heldout pair is then determined by extracting an i-vector from the speaker seen in a different environment, e. g. $\boldsymbol{\lambda}_{s,n'}$ where n' is chosen such that $n' \in N_s$ and $n' \neq n$. Similarly, for the environment vector, an i-vector $\boldsymbol{\lambda}_{s',n}$ is extracted, where s' is chosen such that $s' \in S_n$ and $s' \neq s$. The i-vectors will now also contain information from mismatched factors s' and n'. The idea is then to factorise these vectors by generating bottleneck features using the neural networks, denoted by $f_{\{\texttt{spk},\texttt{env}\}}(\cdot)$, where the desired output is new representations that pertain only to a speaker or an environment:

$$f_{\texttt{spk}}(\boldsymbol{\lambda}_{s,n'}) \approx \boldsymbol{\lambda}_s \tag{6.6}$$

$$f_{\texttt{env}}(\boldsymbol{\lambda}_{s',n}) \approx \boldsymbol{\lambda}_n \tag{6.7}$$

HMM - GMM acoustic models are trained with the Kaldi toolkit and hybrid neural network models with the `nnet3` package, using the standard WSJ recipe[1]. Specifically, monophone and triphone models are first trained on top of 13 MFCCs with delta and double deltas. Training continues on top of 40-dimensional features from LDA, and MLLT transformations. The final training stage includes speaker adaptive training using CMLLR.

The neural networks[2] are 6-layer TDNNs (Section 2.1.2) with p-norm activations (Zhang et al., 2014a) ($p = 2$), and input and output dimensions set to 2000

[1] `github.com/kaldi-asr/kaldi/tree/master/egs/wsj/s5`

[2] See `Kaldi-2` in Appendix A.

Model	dev93	eval92	dev93d	eval92d
GMM	12.06	7.76	49.25	42.96
NN	10.36	6.72	48.17	41.52

Table 6.3: Baseline results WER (%) with models trained on WSJ and no i-vectors. Test sets postfixed with *D* are noisy versions.

Model	dev93	eval92	dev93d	eval92d
GMM	13.52	9.50	32.77	25.09
NN	11.38	7.81	22.70	15.95

Table 6.4: Baseline results WER (%) with models trained on WSJ+DEMAND and no i-vectors. Test sets postfixed with *D* are noisy versions.

and 250, respectively. The splice indexes are $\{\pm 4\}, [0], [\pm 2], [0], [\pm 4], [0]$. The networks are trained for 8 epochs with an initial learning rate set to 0.005, which is reduced exponentially to a tenth of the original rate. As discussed above we use online i-vectors (and corresponding bottleneck features) during training that are extracted each tenth frame.

For the bottleneck networks, independent feedforward networks are trained for speaker and environment classification, where each network has 282 or 18 output classes, respectively. The networks have three 500-dimensional layers with the exception of middle bottleneck layers the size of the original i-vectors. ReLU activations (Nair and Hinton, 2010) are used throughout except the final softmax layer. They are trained with RMSProp (Tieleman and Hinton, 2012) using the Keras deep learning toolkit (Chollet et al., 2015). A random subset of 10% of the data is held out as validation data. Training uses early stopping on the validation data with a patience of 2 epochs. After training, the i-vectors are passed through the network to generate bottleneck features – the factorised representations. Acoustic models are then trained as above using the bottleneck features in place of the i-vectors.

6.4. Results

Table 6.3 shows baseline results (without i-vectors) training with clean data and testing on clean and perturbed test sets denoted with the postfix *D*. Unsurprisingly, when the training and test conditions are not matched, the WERs are significantly higher. Training on matched, perturbed data significantly reduces errors on the perturbed test sets, as shown in Table 6.4. This is particularly evident for the neural network models, where the WERs on the perturbed sets drop by up to 62% relative, while increasing WERs on the non-perturbed sets by about 10%, or 1% absolute.

Features	Spk-class	Env-class
i-vectors speaker	85.9	97.4
i-vectors environment	69.6	99.2
bottleneck-features speaker	84.3	41.2
bottleneck-features environment	6.4	99.3

Table 6.5: Classification validation accuracy (%) from 100-dimensional vectors to speaker or environment categories on a held-out evaluation set.

6.4.1. Multi-condition training

To confirm that the respective conditions are indeed factored out, classification experiments are performed on the original i-vectors and the bottleneck features. This is a similar idea to the probing experiments on speaker embeddings such as x-vectors (Raj et al., 2019; Wang, Qian, and Yu, 2017). We train networks to classify speakers, or environments, from the i-vectors, or the bottleneck features. The networks have the same architectures and training procedures as the speaker/environment classification networks above, but with 500 units in each hidden layer. Table 6.5 shows classification accuracies on held-out data. As shown in the first two rows, the original i-vectors contain a large amount of information about the respective nuisance factors. In contrast, when training on top of the extracted bottleneck features, we observe large drops in accuracies of the nuisance class (e. g. 6.4% classifying speakers with environment bottleneck features, compared to 69.6% using the i-vectors). The differences in accuracies between the speaker and environment vectors may be due to the large difference in the number of speakers and environments (282 and 18, respectively). Crucially, the factorised representations still classify their respective classes with high accuracies. Figure 6.5 visualises the effect on the speaker representations using t-SNE (Maaten and Hinton, 2008). The original speaker i-vectors cluster into environments (colours), whereas there is no evident clustering with the bottleneck features.

6.4.2. Results with factorised representations

The results for using the factorised representations (bottleneck features) are shown in Table 6.6. When relying solely on speaker or environment i-vectors, the use of factorised representations provides about 12% relative reduction in WER on dev93d with 100-dimensional speaker i-vectors. Similarly, factorising the environment i-vectors reduces WER on eval92d by roughly 5.5% relative. It is interesting to note that reducing the speaker i-vector dimension to 30 improves WERs in the non-factorised case. The smaller dimension may lead to implicit factorisation. Note that the use of standard speaker or environment i-vectors actually increase the WER compared to the baseline in Table 6.4.

When concatenating speaker and environment i-vectors the opposite effect occurs:

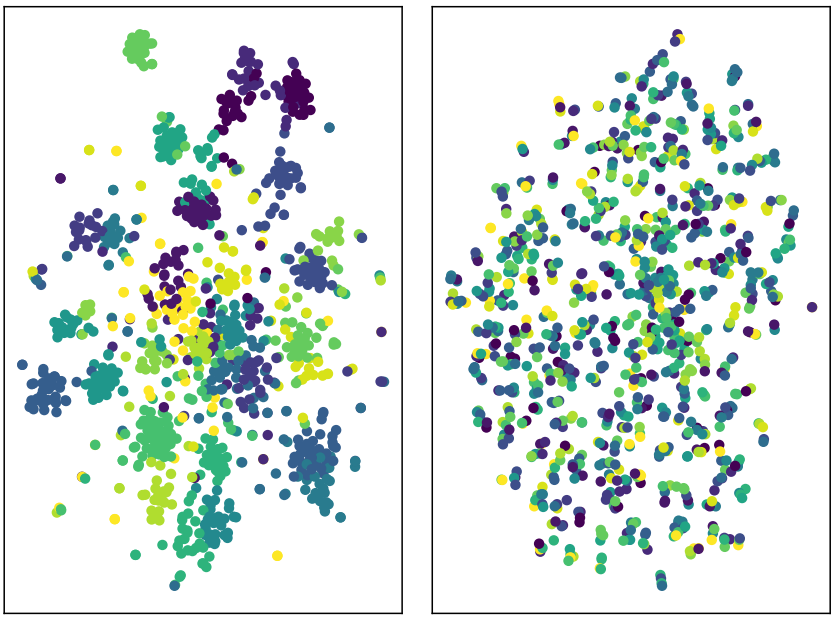


Figure 6.5: t-SNE (Maaten and Hinton, 2008) of 1000 sampled 30-dimensional speaker i-vectors, before (left) and after (right) factorising the i-vectors. The colours indicate the 18 environments in the data. The loss of evident clustering shows that the factorised representations have lost information about the environments.

Features	Spk	Env	Spk+Env
i-vectors (100)	23.27 / 16.59	23.21 / 16.27	20.39 / 13.50
i-vectors (30)	22.89 / 15.20	- / -	20.18 / 14.44
Bottleneck (100)	20.34 / 14.46	22.48 / 15.36	20.08 / 14.85
Bottleneck (30)	21.02 / 14.94	- / -	20.85 / 14.71

Table 6.6: WER (%) with i-vectors or bottleneck features (factorised representations), evaluated on WSJ+DEMAND with perturbed test sets (`dev93d` / `eval92d`). The baseline without i-vectors obtains 22.70% and 15.95% WER, respectively (Table 6.4). Speaker vector sizes are provided in parentheses. Statistical significance (see Section 4.6) was measured for i-vector to corresponding bottleneck pairs with $p < 0.001$ for all speaker representations except 30-dim for `eval92d` ($p = 0.08$); with $p < 0.05$ and $p < 0.01$ for environment representations on `dev93d` and `eval92d`, respectively; and with $p < 0.01$ for joint combinations for 100-dim representations on `eval92` only.

the factorised representations generally yield *increases* in WER. The non-factorised, 100 dimensional i-vectors provide the lowest WERs, whereas the lower dimensional vectors or the factorised representations yield higher WERs. This is unsurprising, and is likely due to the ability of the non-factorised vectors to make use of correlations between speakers and environments during training, and because each target combination is present in the data. This is, however, not always the situation. These results suggest that if the target combination is present in the training data, then the best practice implication is to use the original i-vectors in a speaker and environment combination. Below we experiment with the case in which the target combination is not present, and in that case we will see that it is better to use the factorised vectors.

We investigated the sparsity of the vectors and observed a drop in the average number of non-zero elements, or L_0 "norm" ($\sum_i |x_i|^0$, where $0^0 \equiv 0$). The 100-dimensional factorised speaker representations had an average norm of 32.1 on the development set, yet only 5 units were consistently turned off. This suggests that the learned representations are still making use of the majority of the dimensions, but only about a third at any one time, perhaps improving the match at test time.

6.4.3. Results for unseen combinations

This experiment addresses the situation when joint adaptation data is not available. Instead, one can reuse transforms estimated in a single, mismatched condition. The results are shown in Table 6.7. Previously, concatenating speaker and environment vectors did not produce improvements in WERs with the factorised representations (Table 6.6). This situation is now reversed, with up to 5% relative improvements. We believe that this is because there is no longer any implicit averaging across environments or speakers, and because the true, joint combination is not present. The improvements when using only speaker or environment representations are also more pronounced, with up to 21% relative for 100-dimensional speaker representations and 14-17% relative for the environment representations. There is now no possibility of implicit factorisation since the i-vectors are estimated in combination with exactly one mismatched condition. The results clearly demonstrate the requirement for factorisation when each factor has been estimated in strongly mismatched conditions.

6.5. Conclusions

In this chapter we have proposed a method that enables the factorisation of feature representations, such that they may be combined in novel combinations at test time in a robust manner, without worsening WERs. Specifically, we have demonstrated that we can successfully generate factorised representations from i-vectors using neural networks with a bottleneck layer. For speaker i-vectors we have shown up to roughly 12% relative improvements when the target environment is unknown and 5% when the

Features	Spk	Env	Spk+Env
i-vectors (100)	25.21 / 17.26	25.47 / 19.03	21.12 / 14.48
i-vectors (30)	24.71 / 17.19	- / -	21.29 / 15.33
Bottleneck (100)	19.88 / 14.44	21.71 / 15.68	20.06 / 14.32
Bottleneck (30)	19.71 / 14.80	- / -	19.57 / 14.19

Table 6.7: Heldout experiments. i-vector and bottleneck feature WER (%) results on WSJ+DEMAND with perturbed test sets (`dev93d` / `eval92d`) with an equal distribution of environments for each speaker. Speaker vector sizes are given in parentheses. Statistical significance (see Section 4.6) was measured for i-vector to corresponding bottleneck pairs with $p < 0.001$ for all speaker and all environment representations; and for joint combinations with $p < 0.05$ and $p < 0.001$ for 100- and 30-dim on `dev93d`, and not significant and $p < 0.05$ for 100-dim and 30-dim on `eval92d`.

target speaker is unknown, in cases where the use of standard i-vectors (either speaker or environment vectors) may in fact increase the WER. This situation is even more pronounced when the i-vectors are extracted in directly mismatched conditions. In this case the use of factorised speaker or environment representations yield up to a 21% relative WER reduction compared to the corresponding i-vector.

A disadvantage with the presented approach is that we lose correlations between known speaker and environment combinations at training time, yielding lower frame accuracy. This is reflected in the WERs when using both speaker and environment i-vectors that were extracted across many combinations (`Spk+Env`). In future work we would like to investigate the speaker adaptive i-vector approach presented by Miao, Zhang, and Metze (2014) which uses an i-vector specific sub-network that could possibly learn correlations between factors and thus might help mitigate this effect.

An advantage of the proposed approach is that it is technically independent of the feature representation. There is consequently much scope to explore the use of other representations, such as LHUC (Swietojanski, Li, and Renals, 2016) or speaker codes (Abdel-Hamid and Jiang, 2013). With neural methods to obtain representations, with architectures similar to x-vector extraction (Snyder et al., 2017), we can envision possibly combining the classification networks with the acoustic neural network model and training both objectives jointly. A limitation of the current method is that there is no explicit drive towards removing a factor from the bottleneck features: we are relying on this implicitly in the speaker/environment classification networks. Other architectures could embody this explicitly through for example adversarial learning (Shinohara, 2016). Finally, a disadvantage of the experiments shown in this chapter is the use of an artificial corpus. Future work should aim to experiment with more realistic data such as the MGB corpus (Bell et al., 2015, see also Section 4.1). The technique proposed in this chapter is not directly applicable to these corpora, since we do not have labels for the individual factors, nor is it clear what those factors should be. One approach may be to discover or disentangle such factors using unsupervised methods. We will discuss future

work further in Chapter 9.

Chapter 7

Longitudinal training

There are growing privacy and copyright concerns with the use of data, both in personal devices and in the broadcast domain (see e. g. Zimmeck et al., 2016). Data may therefore be time-limited, meaning that we can only use it to train a model for a limited amount of time before we subsequently lose access. Consider for example speech from a user of a smart phone application for which it may be necessary to erase the recorded speech at regular intervals. If we repeatedly obtain data from a particular domain in this manner, we would expect a model to continuously improve in that domain, in a scenario we called *longitudinal learning*. This chapter addresses the question from Chapter 1 on how well models improve in a longitudinal setting, and what implications this setting has on active learning. We first conduct experiments to observe the impact of training in a longitudinal manner. The experiments suggest that we can improve the WER by averaging successive models obtained with an effectively cyclical learning rate. We then discuss the impact of the longitudinal setting on active learning, and propose to combine active learning with semi-supervised training, showing that this improves upon either technique on its own.

7.1. Introduction

Incremental training of models in machine learning is an expanding field with a large variety of problems and solutions. One area of research is lifelong learning, which may be defined as the ability to learn to perform new tasks (e. g. the number of classification targets are not fixed), without forgetting previously learned tasks (Parisi et al., 2019). This may be alleviated by, for example, memory-based replay mechanisms (Robins, 1993), dynamic architectures (Rusu et al., 2016; Yoon et al., 2018) or regularisation (Li and Hoiem, 2017).

In ASR, life-long learning has a different connotation, as the classification targets would normally stay fixed (within a language). Model adaptation (Section 2.4) to a new domain can be effective, although maintaining performance on the seed data is usually not a concern. Ghorbani, Khorram, and Hansen (2019) identified the scenario in which we prefer to maintain a single model that can adapt to a new domain and still perform well in the previous domain, which they labelled *domain expansion*. The techniques they explored are similar to those used for life-long learning above.

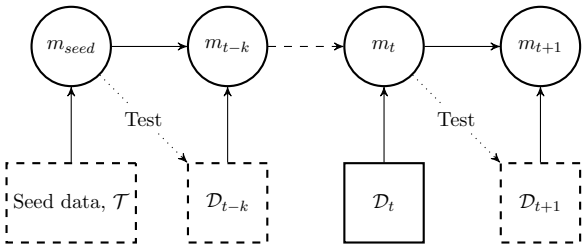


Figure 7.1: Illustration of the longitudinal learning problem with ephemeral data. Circles represent models, and boxes represent data. $\mathcal{D}_{t+1}$ represents data at test time, and data in dashed boxes are unavailable for training purposes at time t.

Our use of *longitudinal learning* is somewhat different, in that we consider improving a model incrementally within the same domain, and do not expect catastrophic forgetting to be a major problem. Consider for example the Masterchef TV series used in experiments in this chapter, where many of the recording situations will be largely the same throughout, with recurring characters. This is in contrast to models adapting between domains, such as adult speech to child speech, or between broadcast television genres.

Specifically, we want to obtain a model at time t, m_t, from one or more previous models $m_{1,\ldots,t-1}$, to improve the WER on future data $\mathcal{D}_{t+1}$, using only present data $\mathcal{D}_t$, where we assume past, present and future data are drawn from the same domain. This is illustrated in Figure 7.1. Note that we do not adapt to the test-data, but always to the data just preceding it, in order to measure the improvement in the domain as we observe more data. The key constraint is the lack of access to previous data: if this data was available, a reasonable strategy would be to retrain on all the pooled data thus far. The intuition is that, as we see more data from a particular domain, we expect a model to perform better on new data from that domain, even when we no longer have access to past data. This particular scenario has real-world examples. We may, for example, envisage a child speech model being repeatedly updated with new data, but due to privacy concerns we cannot store that data for any length of time. User-data from smart-devices may similarly be time-constrained. In the experiments of this chapter, we will consider a TV series for which an out-of-domain model is expected to increasingly improve on episodes from that series, as new episodes air.

We will approach this study from two aspects. First, the key difference to the batch case where we may pool all previous data, is that we cannot shuffle data across time. We may consider each successive model as specialising on the last episode for which it had data. We therefore look at the effects of combining successive models to compensate. Ensembling model posteriors from individually trained models is a common tool to building competitive ASR systems for challenges (e. g. Woodland et al., 2015). A different approach is to combine the weights of individual models, which is performed with success in Kaldi when training models in parallel (Povey, Zhang, and Khudanpur, 2014). An interesting aspect of longitudinal training is that, depending on the learning

rate schedule for each model, it may resemble the use of cyclical learning rates (Loshchilov and Hutter, 2017; Smith, 2017) where each cycle is considered one longitudinal time-step. These schedules have been shown to suit posterior ensembling or model averaging of the same model, with model instances obtained throughout training in techniques known as snapshot ensembling (Huang et al., 2017), Fast Geometric Ensembling (FGE) (Garipov et al., 2018) and Stochastic Weight Averaging (SWA) (Izmailov et al., 2018). In other words, these works suggest that with a natural longitudinal learning schedule, both posterior ensembling as in FGE and model averaging as in SWA should be beneficial to performance. We propose to make use of this observation, and explore posterior ensembling and model averaging of models extracted at every longitudinal time-step.

Second, it may be that we have the budget to annotate a small portion of the data. A significant amount of annotation is wasteful since we are unable to keep the audio. However, even with small amounts, it is interesting to observe the effect on training. In ASR there has been demonstrated success in using active learning in a variety of tasks, the majority of these using confidence measures given a seed model (e. g. Drugman, Pylkkönen, and Kneser, 2016; Long et al., 2018; Nallasamy, Metze, and Schultz, 2012). As above, we consider active learning on data from the current time-step, while testing on the next. The typical form of active learning requires including the original data in each training pass with the newly acquired data, to control for the fact that the data selected for active learning is not i.i.d. (Sugiyama and Kawanabe, 2012). Since we are unable to include previous data in the longitudinal setting, we must find an alternative method to maintain a suitable distribution of data points. We propose to include the remaining data from an episode as semi-supervised training data. This is an idea that has previously been applied to various classifiers within problems such as emotion recognition (Zhang et al., 2014b), spoken language understanding (Tur, Hakkani-Tür, and Schapire, 2005), image classification (Li, Wang, and Tang, 2013) and text classification (McCallumzy and Nigamy, 1998).

In the rest of this chapter, we first discuss posterior ensembling and model averaging in Section 7.2. In Section 7.3 we formalise active learning, and discuss how we include any remaining data as semi-supervised training data. Section 7.4 presents the experimental setup, and Section 7.5 the results.

7.2. Posterior ensembling and model averaging

As noted above, if we reset the learning rate for each training episode in a longitudinal learning setup, this setup resembles the use of cyclical learning rates in work on FGE (Garipov et al., 2018) and SWA (Izmailov et al., 2018). The success in these works motivates the use of posterior ensembling and weight averaging, which we describe next.

If we denote the softmax output of model m given test input $\mathbf{x}$ as $\mathbf{h}_m(\mathbf{x})$, then the

posterior ensemble of N models is the average of the softmax outputs from each model:

$$\mathbf{h}_{\text{ens}}(\mathbf{x}) = \frac{1}{N} \sum_{m=t-N}^{t} \mathbf{h}_m(\mathbf{x}), \tag{7.1}$$

where t is the current time-step, and N is the total number of previous models to ensemble. Given that we are interested in using this approach in a longitudinal setting, we average models starting from the current model at time t, and include N models backwards in time until model $m = t - N$. The resulting averaged output of the ensemble is passed to the decoder in the normal fashion. We can consider averaging the posteriors as averaging in *output space*.

Weight averaging is similar, but averaging takes place in *weight space*:

$$\mathbf{w}_{\text{avg}} = \frac{1}{N} \sum_{m=t-N}^{t} \mathbf{w}_m, \tag{7.2}$$

where $\mathbf{w}_m$ represents the entire set of weights for model m. Rather than computing the average repeatedly each time we train a new model and N grows by one, we can compute the cumulative average:

$$\mathbf{w}_{\text{avg}}^{(t+1)} = \frac{\mathbf{w}_{t+1} + t\mathbf{w}_{\text{avg}}^{(t)}}{t+1}. \tag{7.3}$$

An advantage of weight averaging is that we are not required to perform N forward passes of the data, compared to posterior ensembling.

Huang et al. (2017) argue that the best models with which to perform posterior ensembling are those that sit within disjoint, isolated minima, because they make significantly different predictions of the same data. In a method they call *snapshot ensembling*, the models are obtained by training using cyclical learning rates with cycles over a long time-span (20-40 epochs). A model "snapshot" is extracted each time the learning rate reaches a minimum, corresponding to a new minimum, before the next cycle begins. Similarly, for typical practice in ASR, we see effective ensembles when using entirely different, independently trained models (e. g. Woodland et al., 2015). For a single architecture, the implication is that we should expect increased errors in the interpolation in weight space between two models to be ensembled. This is perhaps sufficient, but not necessary, for a reasonable ensemble. For example, with FGE (Garipov et al., 2018), models are constructed from a single architecture in a similar fashion to snapshot ensembles, but using cyclical learning rates over a much shorter time-span (2-4 epochs) in order to find minima that are connected by paths of low-error. The resulting models, however, produce less diverse predictions than snapshot ensembling, but larger ensembles can be built with the same computational budget. This scenario is the most similar to longitudinal learning, if each cycle corresponds to a time-step (i. e. an episode), and we adapt for a few epochs each time. Figure 7.2 shows

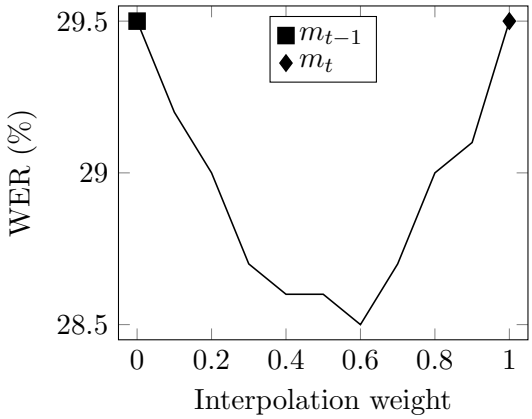


Figure 7.2: Example of convex combination of two models along a line in weight space, evaluated on longitudinal data from MGB corresponding to $\mathcal{D}_{t+1}$ for models successively adapted to $\mathcal{D}_{t-1}$ and $\mathcal{D}_t$. In this case both models produce the same error on the test data, but the interpolation between them shows that their average would provide a lower WER.

the interpolation of two successive models obtained in the longitudinal experiments below.

Model averaging in weight space is different to posterior ensembling, in that it requires the interpolation between models to yield reduced error rates. It is standard procedure in Kaldi when combining models trained in parallel using multiple GPUs (Povey, Zhang, and Khudanpur, 2014). Through empirical experiments on image recognition tasks, Izmailov et al. (2018) demonstrate that the use of cyclical learning rates (over short spans like in FGE), produce intermediate models at various points of the learning trajectory that move in high-performance regions close to an optimal model. The optimal model can be approximated by averaging individual models along the trajectory. They show this by producing an error surface from affine combinations of models extracted from each cycle. We will conduct a similar experiment in the longitudinal setting below (see Figure 7.6). The authors further observe that when a sequence of models are close together in weight space, posterior ensembling and parameter averaging should have similar properties and yield similar results. In other words we should expect models obtained with FGE to be more suitable for averaging than those obtained using snapshot ensembling.

7.3. Active learning

The goal of active learning is to select data from an unlabeled sample set to be transcribed in the most efficient manner possible. By efficient we mean with the least cost and manual effort – typically given some constraining budget – that produces the most informative subset for the training of a model. Consequently, a successful algorithm will select the most informative samples given the task or model at hand, and with the least redundancy. Specifically, we are looking for a set function $f\colon 2^{\mathcal{S}} \to \mathbb{R}$ that returns

Algorithm 7.1: Active learning with semi-supervised training

Require: $\mathcal{T}, k, \mathcal{V}_i$ for episodes $i \in \{1 \dots N\}$
1: Train seed model M_0 using transcribed set $\mathcal{T}$
2: **for** $i \in \{1 \dots N\}$ **do**
3: $\quad \mathcal{S} = \emptyset$
4: $\quad$ **while** $|\mathcal{S}| \leq k$ **do**
5: $\quad\quad s^* = \arg\max_{s \in \mathcal{V}_i} f(s \mid M_{i-1})$
6: $\quad\quad \mathcal{S} = \mathcal{S} \cup \{s^*\}$
7: $\quad$ **end while**
8: $\quad$ Transcribe $\mathcal{S}$
9: $\quad$ Produce hypotheses for untranscribed set $\mathcal{U} = \{u \in \mathcal{V}_i \setminus \mathcal{S}\}$
10: $\quad$ Adapt model M_{i-1} using $\mathcal{S} \cup \mathcal{U}$ to produce M_i
11: **end for**

a score, or overall uncertainty, for a subset $\mathcal{S}$. Given a pool of data, $\mathcal{V}$, we can then extract the subset $\mathcal{S}^*$ under a cardinality constraint k:

$$\mathcal{S}^* = \arg\max\{f(\mathcal{S} \mid \mathcal{M}) \colon |\mathcal{S}| \leq k, \mathcal{S} \subset \mathcal{V}\}. \tag{7.4}$$

Often, $f(\mathcal{S} \mid M)$ is defined as the uncertainty (e. g. 1-confidence) that a seed model M places on new samples. The seed model may be produced by training on already transcribed seed data, $\mathcal{T}$. The overall goal is then to optimise performance on the combined set $\mathcal{S} \cup \mathcal{T}$. For adaptation in the longitudinal setting, we may not have access to previous data or the seed data. Hence, we may be required to update the model on $\mathcal{S}$ only. We will see in the experiments below that this is a poor strategy because it tends to bias the model to the selected data, since the samples in $\mathcal{S}$ are not i.i.d. (Sugiyama and Kawanabe, 2012). Instead, we propose to include the remaining data from that episode by using that data in a semi-supervised manner. This is shown in Algorithm 7.1.

Adapting to first-pass targets in a semi-supervised manner works well, even when some targets are inaccurate (e. g. adapting with LHUC or lattice-supervision; Section 2.4). Hence, when selecting samples to label, it is most useful to select those that would produce the least reduction in errors, or that would possible increase the errors, with unsupervised techniques. We would therefore like to find the samples that not only produce the largest changes to a model, but also which are most likely labelled wrong. A standard measure is to compute confidence scores from lattices (Kemp and Schaaf, 1997; Wessel, Macherey, and Schluter, 1998). In this work, as typical in Kaldi (Povey et al., 2011), we compute posterior probabilities from lattices obtained through MBR decoding (Xu et al., 2011, see also Section 3.2). We experiment with generating confidences using both unigram and trigram language models.

7.4. Experimental setup

The baseline model matches that used in Chapter 5. Specifically, it is trained on news data selected from the MGB corpus (Bell et al., 2015) with MER set to 40% (Section 4.1). As adaptation data we use the longest sequence of episodes in the `eval.long` set of MGB, which corresponds to 11 chronological episodes from the TV show MasterChef, with roughly 30 minutes per episode. When selecting utterances for active learning we select the corresponding words from the oracle data with word-level timings obtained through forced alignment.

Model

The acoustic[1] and language models also match those used in Chapter 5. Briefly, the acoustic model is a 12-layer TDNN-F model using ReLU activations. We adapt each model for three epochs using the learning rate schedules below. The language model is an unbiased MGB trigram model trained on 640 million words of BBC subtitle text. For confidence score generation we also compute a unigram model from the same data.

Learning rates

We experiment with various learning rate schedules illustrated in Figure 7.3. Each is based on the same exponentially decaying schedule:

$$l_i \exp\left(\frac{i}{I}\log(\frac{l_f}{l_i})\right), \tag{7.5}$$

where l_i is the initial learning rate, l_f is the final learning rate, and I is the total number of iterations. We use this in the following schedules:

- Decay: this matches the batch learning rate schedule across episodes, i. e. we consider one exponentially decaying schedule across all episodes, and the final learning rate for each episode matches the initial learning rate for the next.
- Warm: in this case the exponential schedule is reset to the initial learning rate at the beginning of each episode. This is similar to cyclical learning rates.
- Warm+Decay: this is a combination of both of the above, where the initial learning rate for each episode resets to the corresponding decay learning rate for that episode, but the final learning rate is the same across each episode.

For each schedule we use an initial learning rate of 0.00005, which is the final learning rate of the seed model. As the final learning rate we use 0.00001 in all experiments. We will also experiment with a three times higher initial learning rate of 0.00015.

[1] `Kaldi-1` in Section A.3

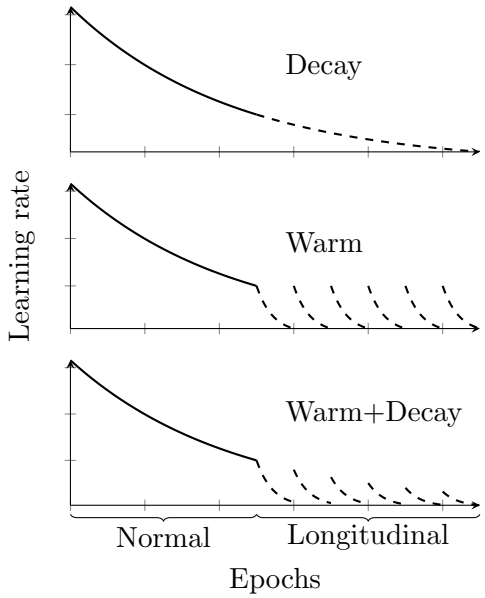


Figure 7.3: Learning rate schedules: decaying to match batch training (top), with warm restarts (middle), with warm restarts and decay (bottom).

7.5. Results

We first demonstrate potential improvements in the longitudinal setup using oracle data and different learning rate schedules in Section 7.5.1. We then look at the possible gains by ensembling or averaging models over time in Section 7.5.2 and Section 7.5.3, respectively. Next, in Section 7.5.4, we experiment with longitudinal adaptation with unsupervised data and observe that the deletion penalty proposed in Chapter 5 is important. In Section 7.5.5 we look at the possible benefits of including small amounts of transcribed data using active learning in the longitudinal setting, when previous data is not available. Finally, in Section 7.5.6 we show that the combination of active learning and semi-supervised training is mutually beneficial.

7.5.1. Learning rate schedules with oracle data

Figure 7.4 shows the results with different learning rate schedules, given the number of episodes adapted to so far (x-axis). Figure 7.5 shows the same results averaged over time. Batch represents the case when we can pool previous data.

The choice of learning rate schedules does impact performance to a subtle degree, but both "warm" schedules perform better than the "decay" schedule that mimics batch learning. Note that the decaying schedules require knowing a priori the number of future episodes[2]. The warm schedules are natural in the longitudinal setting, but resemble cyclical learning rates used in other work (Garipov et al., 2018; Huang et al., 2017; Izmailov et al., 2018; Loshchilov and Hutter, 2017). As noted above, Izmailov

[2]Alternatively we could choose to decay given the performance on a held-out validation set, or for every fixed number of training steps.

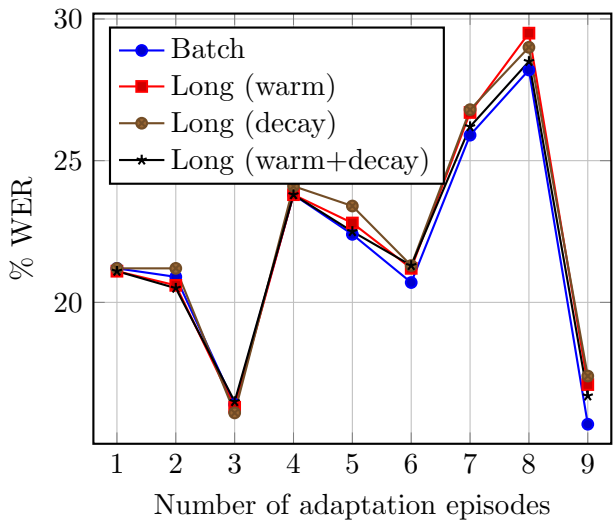


Figure 7.4: WERs when adapting to a given number of episodes chronologically in batch or longitudinal mode. The episodes are of different difficulty, and the differences between the methods are relatively small (see Figure 7.5). Evaluated on MasterChef episodes from `eval.long` of MGB.

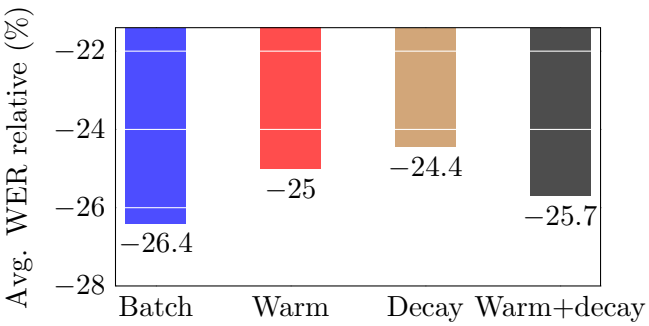


Figure 7.5: Average relative WER reduction per episode compared to unadapted (29.33%) for longitudinal setting with oracle supervision. Evaluated on MasterChef episodes from `eval.long` of MGB.

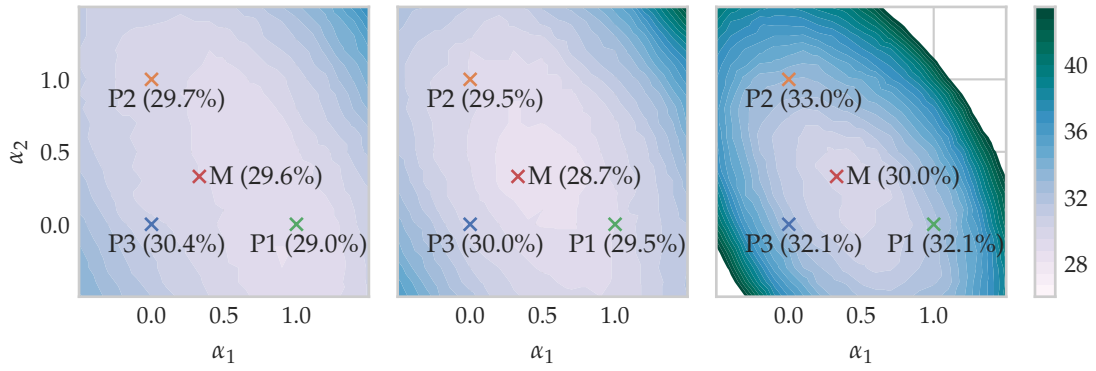


Figure 7.6: Test error surfaces (WER %) of affine combinations (weight averaging) of models trained on the three previous episodes (P1, P2 and P3) and tested on the following episode. That is, what is shown is the test error computed over the affine hull of the three models: each point represents the weight-space combination $\alpha_1 \text{P1} + \alpha_2 \text{P2} + \alpha_3 \text{P3}$, where $\alpha_3 = 1 - \alpha_1 + \alpha_2$ since the coefficients must sum to 1. M is the model created by averaging with equal contributions ($\alpha_1 = \alpha_2 = \alpha_3 = 1/3$). **Left:** Decay schedule, **Middle:** "Warm" schedule with the initial learning rate 0.00005, **Right:** "Warm" with a high initial learning rate of 0.00015. In this case the previous and current model perform similarly on the test set for the next episode, but their slight differences mean that they obtain better performance when combined.

et al. (2018) and Garipov et al. (2018) argue that models stemming from such schedules are well-suited for parameter averaging or ensembling.

In Figure 7.6 we plot test error surfaces for a selection of models by taking affine combinations between three models in weight space and computing the error on the following test set. The plots suggest that these succeeding models between episodes may explore the periphery of a central point of an optimum with respect to the next episode. This is similar to the observation made by Izmailov et al. (2018) in a similar experiment, who also noted that this suggests that both posterior ensembling and averaging should work well. We note that these surfaces represent one instance of one slice of a high-dimensional space, and we should therefore not draw any bold conclusions. Even with a high learning rate in the warm schedule we do not observe increasing error between the models. In this particular case the warm schedules seem to place their average in a more optimal position, despite the latest model with the decay schedule obtaining a better WER on its own (29.0% against 29.5%).

7.5.2. Posterior ensembling

We experiment with ensembling by averaging the posteriors from the models produced during longitudinal training. Table 7.1 and Table 7.2 show ensembles obtained using the warm learning rate schedule, with the normal and high initial learning rates. If we choose the best cumulative combination of models for each episode, then we obtain on average 21.64% and 21.6% WER, respectively. This compares to 22.13% (normal) and 23.3% (high) without ensembling. Hence, on average the possible gain with oracle

Seed	1	2	3	4	5	6	7	8	9
26.2	21.2								
27.7	**20.6**	21.0							
25.0	**16.3**	16.9	17.3						
31.0	23.8	23.8	**23.3**	23.3					
30.3	22.8	**22.6**	22.8	22.9	22.8				
28.8	21.2	21.3	**21.1**	21.2	21.4	21.2			
34.9	26.7	26.1	26.4	26.3	26.4	**26.0**	26.1		
35.6	29.5	28.4	28.3	28.5	28.3	28.3	28.4	**28.2**	
24.5	17.1	16.3	16.4	15.9	15.6	**15.5**	15.7	15.6	15.6

Table 7.1: Posterior ensembling with the warm learning rate schedule (initial learning rate 0.00005). Each row represents a different test episode, and each column represents the cumulative number of models combined. For example, in the first row we have only seen a single episode, so there is only one possible model (no combination); in the second row we are able to combine with the most recent model to produce a combination of two models; and by the last row we have adapted nine times, and can combine nine models in total. The seed model was trained on news data from MGB, and the remaining training and all evaluation was made using MasterChef episodes from `eval.long` from MGB. The average error without ensembling is 22.1% (second column), the best possible is 21.6%, and ensembling all previous models is 21.9% (diagonal).

model selection is small (2-3% relative). The high initial learning rate models obtain poorer initial performance (23.3), but produce similar results after combination, and nearly always with combining all possible models. This is in line with the observation by Huang et al. (2017) that in some cases the cyclical learning rate schedule leads to worse performance when using a model on its own, but that the diversity of the ensembles makes up for the difference. On average, the difference between the best results of the ensembles with different initial learning rates is small. One strategy could be to use a higher initial learning rate, and then combine all possible models at every time step. This corresponds to the diagonal in the tables, and would produce an average error of 21.62%. The caveat is that each added model is another expensive computation to obtain posteriors.

Seed	1	2	3	4	5	6	7	8	9
26.2	21.4								
27.7	21.6	**20.8**							
25.0	17.1	**16.7**	16.8						
31.0	24.9	24.1	23.2	**22.6**					
30.3	23.9	23.2	23.3	22.9	**22.6**				
28.8	22.1	20.9	20.4	20.6	20.2	**20.0**			
34.9	29.3	28.2	27.3	26.4	26.4	26.3	**26.1**		
35.6	31.3	29.7	29.7	29.6	29.0	28.9	28.7	**28.4**	
24.5	18.3	17.5	17.0	16.2	**15.8**	15.8	15.9	15.8	15.9

Table 7.2: Posterior ensembling with the warm learning rate schedule (initial learning rate 0.00015). Each row represents a different test episode, and each column represents the cumulative number of models combined. For example, in the first row we have only seen a single episode, so there is only one possible model (no combination); in the second row we are able to combine with the most recent model to produce a combination of two models; and by the last row we have adapted nine times, and can combine nine models in total. The seed model was trained on news data from MGB, and the remaining training and all evaluation was made using MasterChef episodes from `eval.long` from MGB. The average error without ensembling is 23.3% (second column), the best possible is 21.6%, and ensembling all previous models is 21.6% (diagonal)

7.5.3. Weight averaging

The posterior ensembling results above, along with the analysis in Section 7.5.1, suggest that these models should be well suited for weight averaging. As noted, Izmailov et al. (2018) argue that when a sequence of models are close together in weight space, posterior ensembling and parameter averaging should have similar properties and yield similar results. Table 7.3 and Table 7.4 indeed show similar results to the above, and in some cases (shaded blue) we see the best results having combined the same models as with posterior ensembling (Table 7.1). Again, the learning rate schedule with the more aggressive initial learning rate (0.00015) obtains worse errors to begin with, but produces similar results when averaged, with a more predictable pattern: we can choose to always average every possible model (diagonal), and obtain an average WER of 21.7% compared to 23.3% without averaging for the models in Table 7.4. This is nearly the same error as with posterior ensembling above, but as noted without the computational overhead.

Seed	1	2	3	4	5	6	7	8	9
26.2	21.2								
27.7	**20.6**	21.1							
25.0	**16.1**	16.9	17.5						
31.0	23.8	23.7	23.4	**23.2**					
30.3	**22.8**	23.0	22.8	22.9	23.0				
28.8	21.2	21.3	**21.1**	21.3	21.4	21.7			
34.9	26.7	**26.1**	26.8	26.4	26.5	26.4	26.6		
35.6	29.5	28.6	28.8	28.8	28.6	28.7	28.5	**28.4**	
24.5	17.1	16.4	16.4	16.1	15.6	**15.2**	15.5	15.5	15.7

Table 7.3: Weight averaging with the warm learning rate schedule (0.00005). Each row represents a test episode, and each column represents the cumulative number of models that have been averaged. Blue cells match the best cells for posterior ensembling in Table 7.1. The average error without model averaging (first column) is 22.1%, the best possible is 21.63%, and averaging all previous models (diagonal) is 22.04%. The seed model was trained on news data from MGB, and the remaining training and all evaluation was made using MasterChef episodes from `eval.long` from MGB.

Seed	1	2	3	4	5	6	7	8	9
26.2	21.4								
27.7	21.6	**20.7**							
25.0	17.1	16.9	**16.6**						
31.0	24.9	24.2	23.0	**22.7**					
30.3	23.9	23.5	23.6	23.1	**22.9**				
28.8	22.1	20.8	20.6	20.7	20.5	**20.5**			
34.9	29.3	28.2	27.2	26.5	26.4	**26.1**	26.3		
35.6	31.3	29.9	29.4	29.1	28.7	28.6	28.2	**28.2**	
24.5	18.3	17.7	16.9	16.7	16.4	**15.9**	16.0	16.0	16.0

Table 7.4: Weight averaging with the warm learning rate schedule (0.00015). Each row represents a test episode, and each column represents the cumulative number of models that have been averaged. Blue cells match the best cells for posterior ensembling in Table 7.2. The average error without model averaging (first column) is 23.3%, the best possible is 21.67%, and averaging all previous models (diagonal) is 21.7%. The seed model was trained on news data from MGB, and the remaining training and all evaluation was made using MasterChef episodes from `eval.long` from MGB.

7.5.4. Semi-supervised training

The experiments so far has used oracle targets. We next show how much more difficult it is to adapt to the first pass targets. We find it critical to include a deletion penalty in the HCLG (as in Chapter 5) when generating supervision lattices. Table 7.5 shows the effect of the deletion penalty. A penalty of -2 obtains the lowest WER and we use this value for the remaining experiments. In Figure 7.7 we compare batch and longitudinal semi-supervised training to the oracle case (oracle numbers from Figure 7.4), as well as the result when adapting from the seed model to each individual episode ("single"). Longitudinal trails batch similarly for both the oracle and unsupervised case. With semi-supervised training, the gap between batch and longitudinal diminishes, and both significantly outperform adapting anew to each episode.

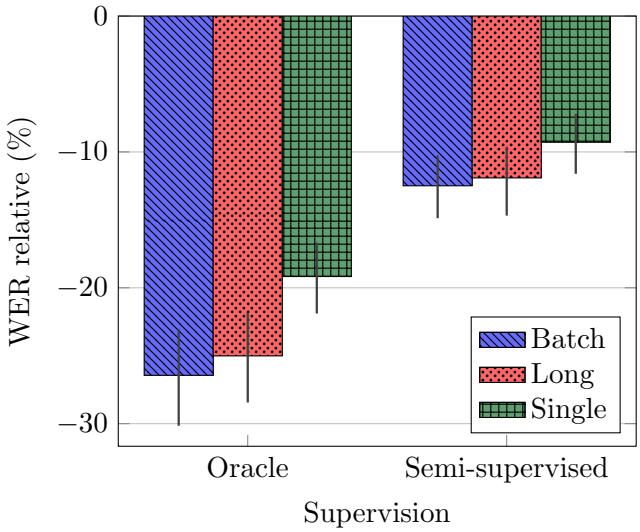


Figure 7.7: Average relative WERs across episodes compared to the unadapted baseline (29.33%) when adapting in batch, longitudinal, or single mode, for either semi-supervised (penalty -2, lattices generated with baseline model) or oracle data. The improvements from single to long and from long to batch are statistically significant (see Section 4.6) with $p < 0.001$ for both oracle and semi-supervised, with the exception of long to batch for the oracle experiment ($p = 0.070$).

Penalty	WER (%)	ins	sub	del
0	29.4	2.0	15.2	12.1
-1	27.4	2.1	14.2	11.1
-2	26.4	2.4	14.5	9.5
-3	28.3	3.4	15.0	9.9

Table 7.5: Effect of deletion penalty in HCLG when creating supervision lattices for adaptation, having trained in a longitudinal, semi-supervised manner. Decoded with non-penalised HCLG on MasterChef episodes from `long.eval` of the MGB corpus.

7.5.5. Active learning

We discussed above the implications of selecting data for active learning when we are unable to pool that data with the original training data. In this experiment we look at the effect of using either confidence selection or random selection of the data for active learning. We can gradually move between a random and lowest confidence setting, by using the following selection procedure: Sample a set of k elements from all the data in episode $\mathcal{V}_i$ and select the least confident sample, s^*, from the set, repeat by sampling k elements from $\mathcal{V}_i \setminus \{s^*\}$ until the cardinality constraint is met. Setting $k = 1$ will then equal random sampling, while $k \geq |\mathcal{V}_i|$ equals lowest confidence selection.

Figure 7.8 shows the relative change in WER compared to the unadapted model, for varying choices of k. The results show that as we move from a sampling strategy based on random selection to confidence selection, the results deteriorate. This may be attributed to the random selection choosing data points independently from the data, while confidence selection biases the selection.

As a further analysis Figure 7.9 shows the fraction of the most common speakers in test episode 10 when adapting to episode 9 using a random or lowest confidence strategy. Since confidence selection does not sample evenly from the adaptation data distribution, it rarely samples from the most common speaker in the data which has high confidence.

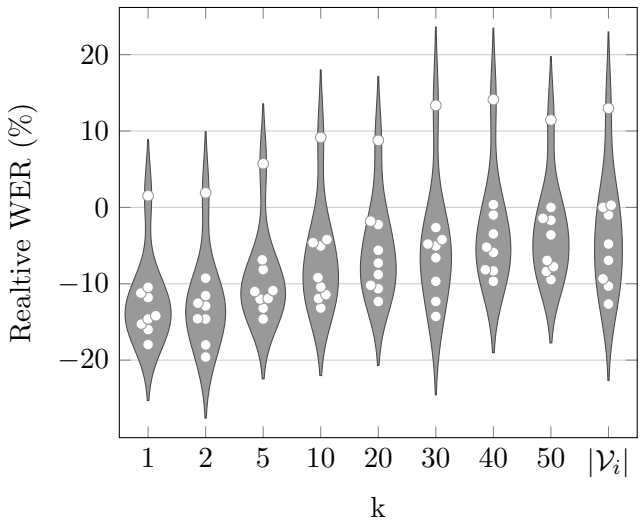


Figure 7.8: Effect of varying k, where $k = 1$ equals random selection and $k = |\mathcal{V}_i|$ equals confidence selection. Each circle represents an individual episode, with some perturbation along the x-axis to aid visibility. The episodes are of varying difficulty, hence the spread across the y-axis for each choice of k. Evaluated on MasterChef episodes from `long.eval` of the MGB corpus.

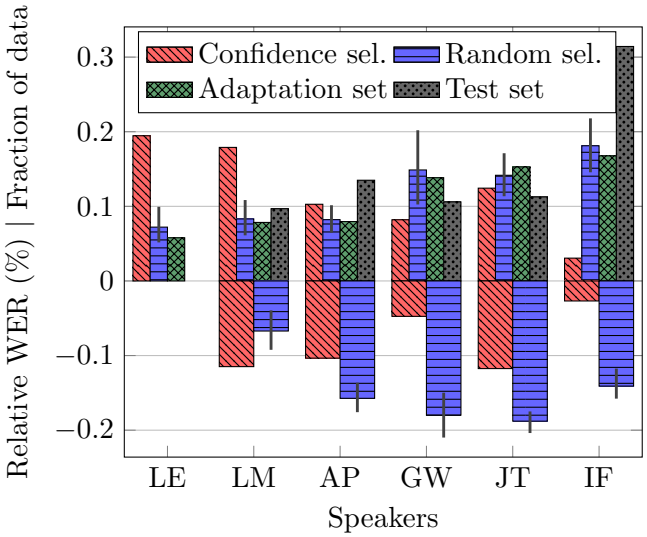


Figure 7.9: Relative improvements by speaker for confidence and random selection between two MasterChef episodes from `long.eval` of the MGB corpus. The positive y-axis represents the fraction of that speaker in data selected by confidence, by random, or the total amount of that speaker in the adaptation set and in the test set. The negative y-values represent the relative WER improvement having used confidence or random selection. For example, for speaker IF, confidence selection selects a very small amount of data, but the test set feature that speaker quite considerably. The random selection selects about the same amount of that speaker in the adaptation set. The result is a much improved relative performance compared to confidence selection for that speaker.

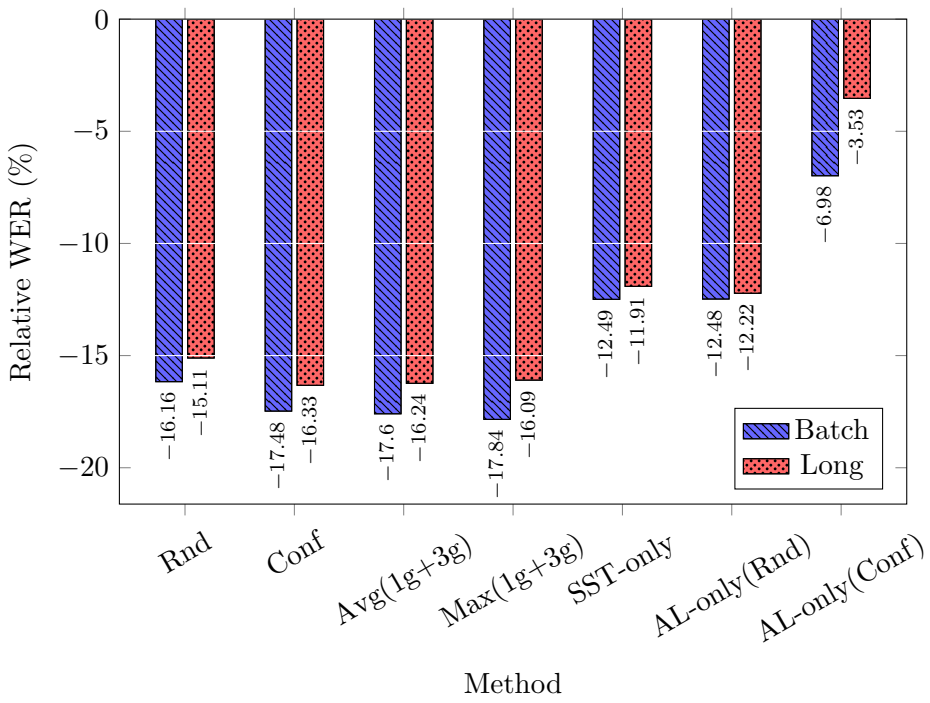


Figure 7.10: Comparison of average relative WERs improvement (w.r.t. 29.33%) for different sampling strategies to select 10% of data for active learning (AL). The last two items are the results using only active learning (no semi-supervised (SST) data) for comparison. Evaluated on MasterChef episodes from `long.eval` of the MGB corpus. Statistical significance (see Section 4.6) was measured with $p < 0.001$ for the following pairs: AL-only(Conf)→AL-only(Rnd)/SST-only, AL-only(Rnd)/SST-only→AL+SST(Rnd)/AL+SST(Conf); and with $p < 0.01$ for AL+SST(Rnd)→AL+SST(Conf); for both batch and longitudinal experiments.

7.5.6. Active learning with semi-supervised training

We now include semi-supervised training on the remaining data after filtering for active data (random and lowest confidence). Figure 7.10 compares the combined use of active learning of 10% data with semi-supervised training (SST) on the remaining data, for different sampling strategies. It also includes semi-supervised training on all data, and active learning without semi-supervised training. The figure shows that active learning combined with semi-supervised training improve significantly over both semi-supervised and active learning on their own. Additionally, we now see that confidence selection improves upon random selection, as expected. In addition to standard confidence selection, the figure also shows the results when using a combination of confidences extracted using a unigram and a trigram LM: the first sorts utterances based on the average of both confidences, and the second sorts based on the max of each confidence. Neither strategy has any significant effect on the results.

7.6. Conclusions

In this chapter we have explored some traits of longitudinal learning: we have seen that continuous training is feasible and we have shown how to perform active learning in this setting. Specifically, we first saw that when using an effectively cyclical learning rate schedule, the difference between longitudinal training and the batch case diminishes: with a decay schedule mimicking that of the batch case, longitudinal improved 24.4% WER relative over the baseline, compared to a 26.4% relative improvement with batch. With a warm restart for each episode (cyclical), the relative improvement across episodes was 25%, or 25.7% with the warm and decay schedules combined. Posterior ensembling provided improvements, and one strategy would be to train with a high initial learning rate and combine all models produced at a given time. This yielded similar improvements to that observed with snapshot or FGE ensembling with image recognition (e. g. Garipov et al., 2018), with reductions in the range of $2 - 7\%$ relative compared to no ensembling, depending on the learning rate used to restart each cycle. With weight averaging we observed similar improvements to posterior ensembling, without the computational overhead, as also observed by Izmailov et al. (2018). We then discussed the use of active learning in the longitudinal setting, where we observed that it is important to include data points from semi-supervised training to avoid biasing the model. Specifically, while semi-supervised training of all the data in the longitudinal setting improved 11.9% WER relative upon the baseline, active learning of 10% of the data chosen by confidence selection only reduced errors by 3.5% relative. Instead, a random selection strategy of 10% data produced similar improvements to semi-supervised training with all of the data. However, the combination of active learning using confidence selection, and semi-supervised training on the remaining data yielded the best results with 16.3% relative over the baseline.

Chapter 8

Adaptation with raw waveform models

As we discussed briefly in Section 2.2, ASR models have long benefited from physiologically motivated feature extraction, such as MFCCs. There has been recent research efforts into learning feature representations from raw waveforms, typically using additional neural network layers. This presents new possibilities for model-based adaptation to acoustic factors at the feature level. The last question from Chapter 1 was to find a parameter efficient and robust method to adapt a model to new speaker acoustics. This chapter explores the possibilities of adaptation using an efficient parameterisation of a neural feature extractor.

8.1. Introduction

A key component to improving ASR performance is to reduce mismatch between the acoustic model and test data, by explicit adaptation or normalisation of acoustic factors (see Section 2.4). Methods such as Vocal Tract Length Normalisation (VTLN) (Lee and Rose, 1996), which aims to mitigate large variations in individual speakers' acoustics, scales the filterbank in standard feature extraction. There has, however, been a growing interest in reducing the amount of hand-crafted feature extraction that is required for acoustic modelling of speech (Palaz, Collobert, and Doss, 2013; Ravanelli and Bengio, 2018; Sainath et al., 2015; Takeda, Nakadai, and Komatani, 2018; Tüske et al., 2014). The motivations to learn part, or all, of the feature extractor range from aiding interpretability (Ravanelli and Bengio, 2018; Tüske et al., 2014), to obtaining more optimal representations for the task at hand (Sainath et al., 2013a). Jaitly and Hinton (2011), for example, argued that low-dimensional, hand-crafted features, such as MFCCs, may lose relevant information that is otherwise present in the original signals (e. g. phase if only the magnitude spectrum is used).

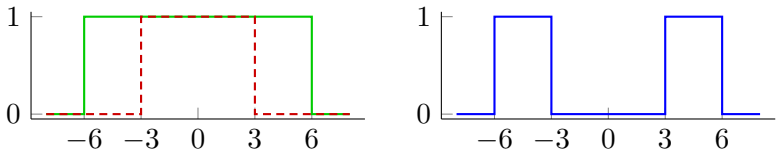
From raw time-domain waveforms, Convolutional Neural Networks (CNNs) have shown promising results (Hoshen, Weiss, and Wilson, 2015; Palaz, Magimai-Doss, and Collobert, 2015; Sainath et al., 2015). It has even been demonstrated that it is possible to learn band-pass beamformers from multi-channel raw waveforms (Hoshen, Weiss, and Wilson, 2015), and a feature extractor learned from raw frequency representations of

speech has been shown to outperform conventional methods (Ghahremani et al., 2018). The interpretability of the learned representations, however, is sometimes limited, and it is not always clear how to apply existing adaptation techniques. In a recent approach called SincNet, Ravanelli and Bengio (2018) propose to constrain the CNN filters learned from raw time-domain signals, by requiring each kernel to model a rectangular band-pass filter (other choices of filters have since been studied, e. g. Loweimi, Bell, and Renals, 2019). The authors show that using a constrained set of parameters representing positions and widths of filters yields improved efficiency, and that the filters are more easily interpretable.

This chapter proposes to make use of these characteristics for the adaptation of raw waveform acoustic models: we would like efficient, compact representations that are quick to estimate and cheap to store. We explore whether we can obtain this by adapting the cut-off frequencies, and the gains of the filters in SincNet. To our knowledge this filter has not previously been used for the purposes of adapting an existing model. The SincNet layer may be particularly well suited for speaker adaptation, as the lower layers closer to the input are known to carry more speaker information than the other layers (Mohamed, Hinton, and Penn, 2012; Swietojanski and Renals, 2014). Section 8.2.1 will show that adapting this parameterisation of the CNN filters has similarities to, and crucial differences from, VTLN, feature-space MLLR (CMLLR) (Gales, 1998), and LHUC (Swietojanski, Li, and Renals, 2016). VTLN, in particular, has been used to mitigate large variations in vocal tract length for the recognition of children's speech (Potamianos and Narayanan, 2003). The following experiments will show that adapting SincNet from adult to child speech yields VTLN-like scaling functions of the filter frequencies.

There are related approaches that aim to learn, and update filterbanks on top of e.g. raw spectra (Sainath et al., 2013a; Sainath et al., 2015; Seki et al., 2018; Seki, Yamamoto, and Nakagawa, 2017). As argued in those papers, fixed filterbanks may not be an optimal choice for a particular task. Sailor and Patil (2016) indeed show that a convolutional RBM model learns different centre frequencies depending on the task at hand. The work herein is perhaps most closely related to Seki et al. (2018), who proposed to adapt a filterbank composed of differentiable functions such as Gaussian or Gammatone filters. They demonstrated more than 7% relative reductions in WER when adapting to speakers in a spontaneous Japanese speech transcription task. The present work differs in that it adapts the SincNet layer, which operates on raw waveforms, rather than power spectra.

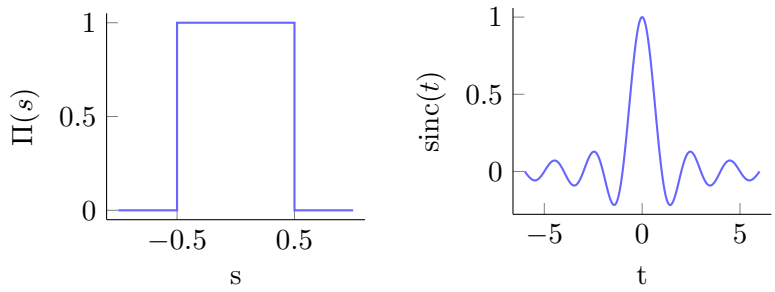
We start by reviewing the formulation of SincNet in Section 8.2. Section 8.3 presents the experimental setup, with a SincNet model and an MFCC based model for comparison. In Section 8.4 we present the results, adapting adult speech models to child speech, either as domain adaptation or by speaker. We find that adapting the SincNet layer considerably improves performance of the previously mismatched model. Section 8.5 concludes the chapter.



(a) Two low-pass filters.

(b) Band-pass filter: $\Pi_2 - \Pi_1$.

Figure 8.1: Constructing a band-pass filter from two rectangular low-pass filters, with $f_1 = 3$, and $f_2 = 6$, $\Pi(\frac{t}{2f_1})$, $\Pi(\frac{t}{2f_2})$.



(a) $\Pi(s)$

(b) $\hat{\Pi}(t) = \text{sinc}(t)$

Figure 8.2: The Fourier transform of a rectangular function is a sinc-function.

8.2. SincNet

The idea of SincNet (Ravanelli and Bengio, 2018) is to parameterise the CNN filters using rectangular band-pass filters in place of standard CNN filters for raw-waveform acoustic models. A rectangular band-pass filter is constructed by subtracting two rectangular low-pass filters from each other, as illustrated in Figure 8.1, where the ideal rectangular filter is:

$$\Pi(s) = \text{rect}(s) = \begin{cases} 0, & \text{if } |s| \geq \frac{1}{2} \\ 1, & \text{if } |s| < \frac{1}{2}. \end{cases} \tag{8.1}$$

Further, the Fourier transform of a rectangular function is a sinc function,

$$\hat{\Pi}(t) = \text{sinc}(t) = \sin(t)/t, \tag{8.2}$$

as shown in Figure 8.2.

Hence, a rectangular filter with lower and upper cut-off frequencies f_l and f_u has the following time-domain representation, represented as the difference between two low-pass filters:

$$g[n, f_u, f_l] = 2f_u \,\text{sinc}(2\pi f_u n) - 2f_l \,\text{sinc}(2\pi f_l n). \tag{8.3}$$

Consequently, the number of parameters per filter is reduced from having to model every tap of each filter (i. e. the filter length) to only having to model two: the cut-off

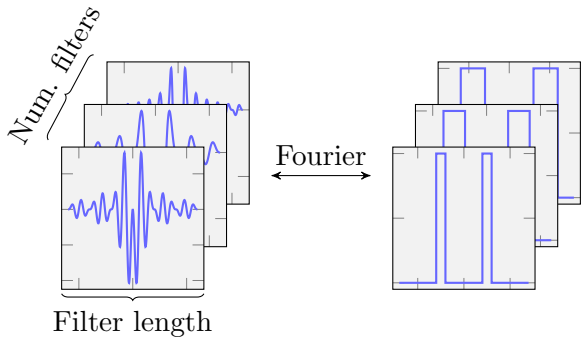


Figure 8.3: Bandpass filterbank in SincNet.

frequencies of the filters, regardless of the filter length. A set of filters becomes a learnable filterbank of approximately rectangular filters. Figure 8.3 shows the corresponding time- and frequency-domain filters of the constructed filterbank.

As in Ravanelli and Bengio (2018), Hamming windows (Oppenheim and Schafer, 1975) are used to smooth discontinuities towards the edges:

$$w[n] = 0.54 - 0.46\cos(\frac{2\pi n}{L}), \tag{8.4}$$

where L is the filter length. Consequently, the final forward pass for speech input $x[n]$ with one filter is[1]:

$$y[n] = x[n] \star g_w[n, f_u, f_l] = x[n] \star w[n]\, g[n, f_u, f_l]. \tag{8.5}$$

An example of learned filters are shown in Figure 8.4.

A related method by Seki, Yamamoto, and Nakagawa (2017) replaced the standard Mel-filterbank during feature extraction of MFCCs with differentiable Gaussian filters on top of power spectra, enabling the learning of centre frequencies, bandwidths and gain. SincNet also learns a filterbank, but in the time-domain on raw waveform features. For SincNet, Ravanelli and Bengio (2018) chose not to explicitly model the gain of each filter, as it can be readily learned by later parts of the neural network. Experiments later in this chapter will experiment with adapting the filter gains to speakers.

8.2.1. Relationship with VTLN, fMLLR and LHUC

A learnable filterbank has close relationships with other well-known methods, as also previously highlighted by Seki, Yamamoto, and Nakagawa (2017). In this chapter we suggest to update the SincNet filterbank for each speaker. This strongly resembles VTLN (Lee and Rose, 1996), which aims to compensate for varying vocal tract lengths

[1] $\star$ (star) denotes cross-correlation, compared to $*$ (asterisk) for convolution. The difference is a filter flip which does not affect the learning of the weights. PyTorch (and most toolkits) implement convolution as cross-correlation:
`https://pytorch.org/docs/stable/nn.html#convolution-layers`.

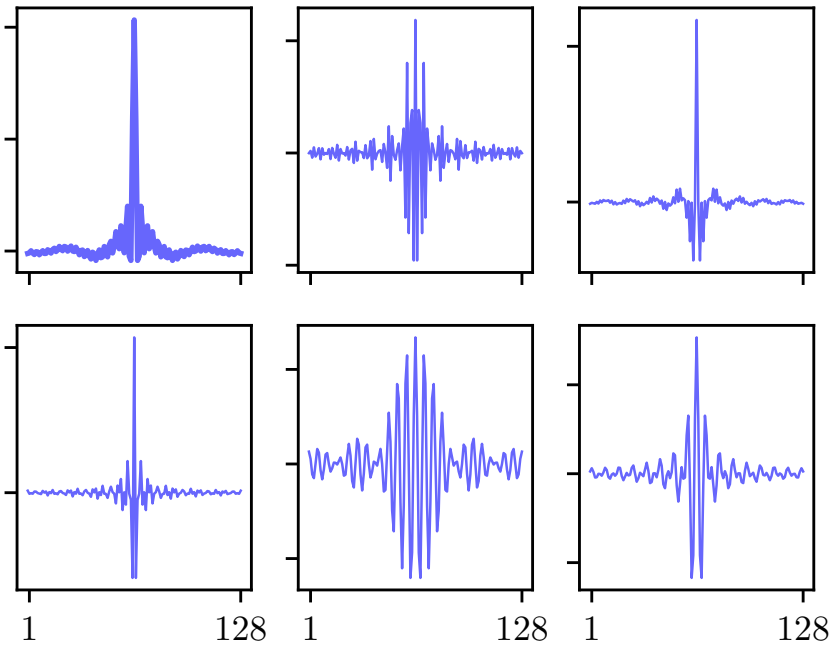


Figure 8.4: Examples of learned bandpass filters in the time-domain.

among speakers (see Section 2.4). It accomplishes this by scaling, or warping, the centre frequencies of the filters in the Mel-filterbank. Consequently, adapting the parameters of the SincNet layer resembles VTLN with a few key differences:

1. SincNet operates in the time-domain, and uses corresponding rectangular filters rather than triangular filters as in the Mel-filterbank;

2. VTLN typically uses a scaling function that is assumed to be piece-wise linear with a single slope parameter, α (as shown in Figure 2.12), whilst if adapting SincNet, the effective learned scaling functions are less constrained.

3. The slope parameter α is typically determined with a grid search (although, there exist more sophisticated methods such as gradient search (Panchapagesan and Alwan, 2006)). With SincNet we can learn the scaling function using gradient descent.

In the original SincNet formulation (Ravanelli and Bengio, 2018), the gains of the filters are held fixed. Downstream layers can learn to scale the contributions of the filters. However, the filter gains may be suitable targets for adaptation for which we would like to attribute importance to the output of individual filters with a small number of parameters. This has similarly been done with learnable filterbanks in traditional feature extraction pipelines (Seki et al., 2018). We also briefly note that if we were to scale the gain of each filter, then this would be structurally similar to a version of feature-space MLLR (CMLLR) (Gales, 1998) with a diagonal matrix and no bias, or

similarly to LHUC (Swietojanski, Li, and Renals, 2016) which scales the output of each neuron by a scalar $r^{(i)}$ for filter i:

$$h^{(i)}[n] = r^{(i)} \sum_{l=0}^{L-1} x[l]\, g_w^{(i)}[n-l], \tag{8.6}$$

where $h^{(i)}$ is the layer output for filter i, x is the raw waveform input, and g_w is the windowed filter function from Equation 8.5. Clearly, we can view the scalars, $\mathbf{r}^{(i)}$, as either scaling the features, the gain of the filters, or the output of the layer.

8.3. Experimental setup

The baseline models are built using the AMI corpus (Carletta, 2007), which contains about 70 hours of training data from fictitious design team meetings (see Section 4.4 for further details and Section B.4 for example transcripts). HMM-GMM systems are trained with the Individual Head-Mounted Microphone (IHM) stream using Kaldi (Povey et al., 2011) following the recipe for AMI[2].

Child speech from the British English PF-STAR corpus (Batliner et al., 2005) is used as adaptation data, which in total consists of roughly 14 hours of data of read children's speech (see Section 4.5). The children are aged between 4–14, with the majority being 8–10 years old. The data contains a fair amount of mispronunciation and hesitation, making recognition challenging (see Section B.5 for example transcripts). It is clearly mismatched to the AMI data, in terms of what is spoken, the speaking style, and the acoustics of the speakers (see e. g. Fainberg et al., 2016).

SincNet acoustic model

The neural network acoustic model[3] is detailed in Table 8.1. The first layer consists of 40 Sinc filters, each with length 129 as has been used previously in a speech recognition task (Loweimi, Bell, and Renals, 2019). We experimented with different methods of initialising the upper and lower frequencies of each filter as follows:

1. Mel-scale as in Ravanelli and Bengio (2018): the lower frequencies of the filters are linearly interpolated between the corresponding mels of f_{min} and f_{max}, with $f_u[i] = f_l[i-1]$;
2. Uniformly at random in the same range: $f_l \sim \mathcal{U}(f_{min}, f_{max})$, and $f_u[i] = f_l[i-1]$. When sorted by centre frequency, this scheme in effect becomes linear (e. g. $f_u[i] = a f_u[i-1]$ for some constant a);
3. Flat with $f_l = f_{min}$ and $f_u = f_{min} + b$ for each filter (randomness is induced by the layers above).

[2] `github.com/kaldi-asr/kaldi/tree/master/egs/ami`
[3] Corresponding to `Keras-2` in Appendix A.

#	Type	Dim	Size	Dil	Params
1	SincConv	40	129	-	80
-	MaxPooling	-	3	-	
2	BN(ReLU(Conv))	800	2	1	68,000
-	MaxPooling	-	3	-	
3	BN(ReLU(Conv))	800	2	3	1,284,000
-	MaxPooling	-	3	-	
4	BN(ReLU(Conv))	800	2	6	1,284,000
-	MaxPooling	-	3	-	
5	BN(ReLU(Conv))	800	2	9	1,284,000
-	MaxPooling	-	2	-	
6	BN(ReLU(Conv))	800	2	6	1,284,000
7	ReLU(Conv)	800	1	1	640,800
8	Softmax(Conv)	3976	1	1	3,184,776

Table 8.1: SincNet model topology. In total there are 9,029,656 parameters (including BatchNorm).

For each scheme we set $f_{min} = 30\,\text{Hz}$ and $f_{max} = sr/2 - (f_{min} + b)$, where sr is the sampling rate (16 kHz in our experiments), and $b = 50$ is the minimum bandwidth. The remaining layers consist of six 1-D convolution layers with ReLUs, each with 800 units. Kernel sizes and dilation rates are shown in Table 8.1. BatchNorm (BN) layers (Ioffe and Szegedy, 2015) are interspersed throughout. The final softmax layer outputs to 3,976 tied states.

The models are trained using Adam (Kingma and Ba, 2015) with a batch size of 256 and a learning rate of 0.0015, unless noted otherwise. The waveforms are sampled as in Loweimi, Bell, and Renals (2019) and Ravanelli and Bengio (2018): 200 ms windows with a shift of 10 ms, i.e. the input size to the network is $16,000 * 0.200 = 3200$. The models are implemented and trained using Keras (Chollet et al., 2015) and Tensorflow (Abadi et al., 2016). Decoding and scoring is performed using Kaldi (Povey et al., 2011). The experimental code is publicly available[4].

Standard acoustic model

A comparison model[5] was built on standard 40-dimensional MFCCs with cepstral mean normalisation. Excluding the SincNet layer, the topology otherwise matches that described above and in Table 8.1. Empirically it was important to train this model for slightly longer – good results were obtained after 9 epochs with early-stopping, instead of 6. The remaining training schedule and decoding setup was identical. A crucial difference from the corresponding Kaldi recipe is using cepstral mean normalisation

[4] `https://github.com/jfainberg/sincnet_adapt`.
[5] Corresponding to `Keras-1` in Appendix A.

instead of LDA and no speed perturbation. Cepstral mean normalisation was required to have comparable unadapted performance on PF-STAR.

To perform VTLN, a standard grid-search evaluation on `eval/test` from PF-STAR was used to select a warping factor in the range 0.70 to 1.25 with a step size of 0.01.

Language model

As the acoustic and language models for AMI are greatly mismatched to PF-STAR, we interpolate the standard AMI language model based on AMI and Fisher (Cieri, Miller, and Walker, 2004) data, with the training data from PF-STAR. This is similar to other literature working with PF-STAR (Dubagunta, Kabil, and Doss, 2019). As noted in Section 4.5, however, there is some overlap in the sentences between training and test sets for PF-STAR, i.e. training a LM on the training set causes some data leakage. For this chapter this is not critical, given that the main interest is the acoustic model mismatch. Without the biased LM, the combined effect of a mismatched LM, and a mismatched AM, produced WERs greater than 90% in preliminary experiments.

The LM is a 3-gram model with Kneser-Ney discounting, estimated on the PF-STAR training set using the SRILM toolkit (Stolcke, 2002). This is interpolated with the AMI model, giving the latter a weight of 0.7. The vocabulary is restricted to the top 150k word types from an interpolated 1-gram model. Finally, the interpolated model is pruned with a threshold of 10^{-7}.

8.4. Results

The results using models trained on AMI are shown in Table 8.2. The various initialisation schemes produce quite similar WERs, but, perhaps surprisingly, the Mel-initialisation performs least well. The differences are, however, less than 2% relative. Overall, these results are roughly 5% absolute worse than those produced with cross-entropy systems in the corresponding Kaldi recipe for AMI. A key difference may be the use of speed perturbation for data augmentation (Ko et al., 2015). The models are slow to train, and improving training speed is suggested as an area for future work.

Figure 8.5 demonstrates how the initialisation schemes lead to different final responses in the filters. The flat initialisation is essentially forced to change significantly, otherwise each filter would extract identical information. After training it begins to approximate a Mel-like, non-linear curve. This is in line with similar research (Sailor and Patil, 2016; Sainath et al., 2015). It was noted in Section 8.3 that the uniform initialisation effectively creates a linear initialisation. It remains largely linear after training, with some shifts in higher frequencies, and changes to bandwidths: it already covers the relevant frequency range and consequently does not need to change as much as the uniform initialisation. It is perhaps surprising that it does not move towards a non-linear relationship between the centre frequencies like most auditory filterbanks. Note also

Model	Eval	Dev
SincNet - Mel	30.6	28.0
SincNet - Flat	30.2	28.0
SincNet - Uni	30.3	27.9
SincNet - Mel (fixed)	30.5	27.9
MFCC CMN	31.0	28.0

Table 8.2: Baseline results (%) on AMI having initialised the SincConv layer using either flat, Mel or uniform initialisation. Differences are subtle, with Mel-initialisation performing least well. Also shown is a system trained on MFCC features with cepstral mean normalisation (CMN).

that each response is markedly different, yet the corresponding WERs are similar. Both of these phenomena may be explained by the ability of the downstream network to learn to use the extracted features in different ways (Seki et al., 2018). Figure 8.6 demonstrates that using a larger number of filters than 40 (e.g. 128), yields no benefit to WERs; instead, the filter bandwidths become quite erratic. The model trained from a flat initialisation are used for the remaining experiments.

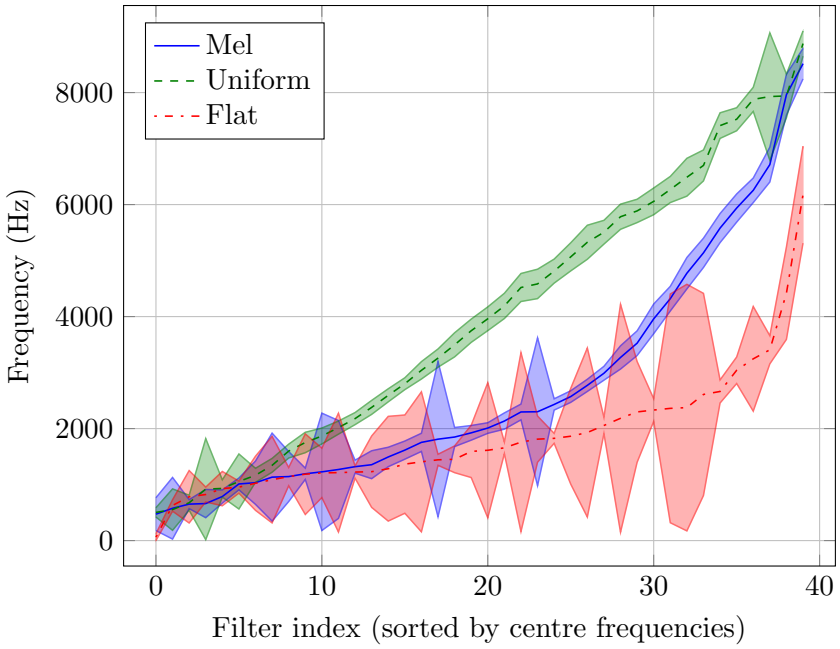


Figure 8.5: Upper and lower learned frequencies per filter with different initialisation schemes, after six epochs of training on AMI. In contrast to Mel and Uniform, Flat is forced to change in order to extract different information in each filter.

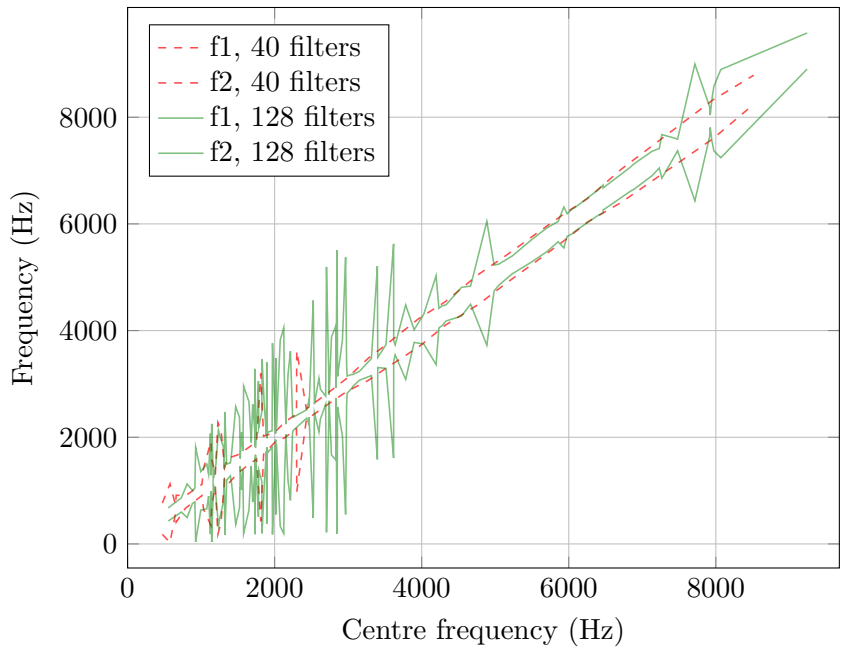


Figure 8.6: Cut-off frequencies per filter for filterbanks with 40-filters or 128-filters after training with a Mel-initialised filterbank, plotted against their respective centre frequencies.

Model	WER (%)
SincNet-AMI	68.19
SincNet-PF-STAR	20.46
SincNet-AMI Adapt+BatchNorm	29.90
SincNet-AMI Adapt–BatchNorm	31.65
MFCC-AMI CMN	60.20
MFCC-AMI CMN VTLN ($\alpha = 0.84$)	37.06

Table 8.3: Results WER (%) on the PF-STAR test set, after having adapted the SincConv layer in the AMI model for a single epoch. A model trained from scratch on the PF-STAR training set is shown for reference. The models have 40 filters that were initialised flat.

8.4.1. Domain adaptation to children's speech

We next investigate supervised domain adaptation of the SincConv layer from AMI to PF-STAR (from adult speech to child speech). As shown in Table 8.3, the AMI model is initially highly mismatched with PF-STAR, with a WER of 68.19%, which aligns with what is expected from the literature (Potamianos and Narayanan, 2003). A model trained from scratch on PF-STAR is included as reference, which obtains 20.46% WER. Adapting the SincConv layer of the AMI model for a single epoch to the training set of PF-STAR reduces the error rate to 31.65%.

Also shown is the effect of updating the statistics of the BatchNorm layers. Freezing the BatchNorm layers demonstrates that the primary improvement comes from adapting the 80 parameters in the SincConv layer. The BatchNorm layers are frozen in all experiments that follow. This experiment shows that it is possible to effectively adapt a very small number of parameters in the model, improving the out-of-domain model by over 50% relative, and coming within 12 percentage points of a model trained from scratch with all 9M parameters. Adapting the SincConv layer amounts to adapting less than 0.0009% of the total number of parameters in the model (see Table 8.1). We also compare with the standard MFCC model, which unadapted performs better than the unadapted SincNet AMI model, with a WER of 60.20% compared to 68.19%. The VTLN grid search further improves the error rate to 37.06%, but does not match the adapted SincNet model which produced an error rate of 31.65%.

Figure 8.7 shows that adapting the SincConv layer shifts the upper frequency distribution of the filters, and their bandwidths. This is reflected in the corresponding VTLN function, shown in Figure 8.8. This suggests that the model has adjusted to higher frequency content in the children's speech data. Figure 8.9 shows average power spectra from the corresponding log-mel filterbank features of AMI and PF-STAR which supports this notion.

Note that the VTLN-like function (Figure 8.8) is nearly piecewise-linear; i. e. similar

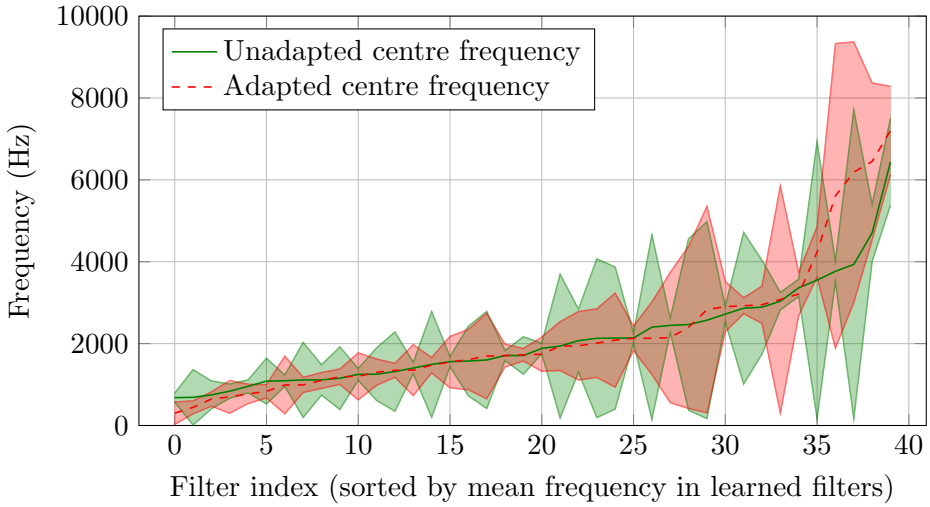


Figure 8.7: SincNet filters before and after domain adaptation to PF-STAR.

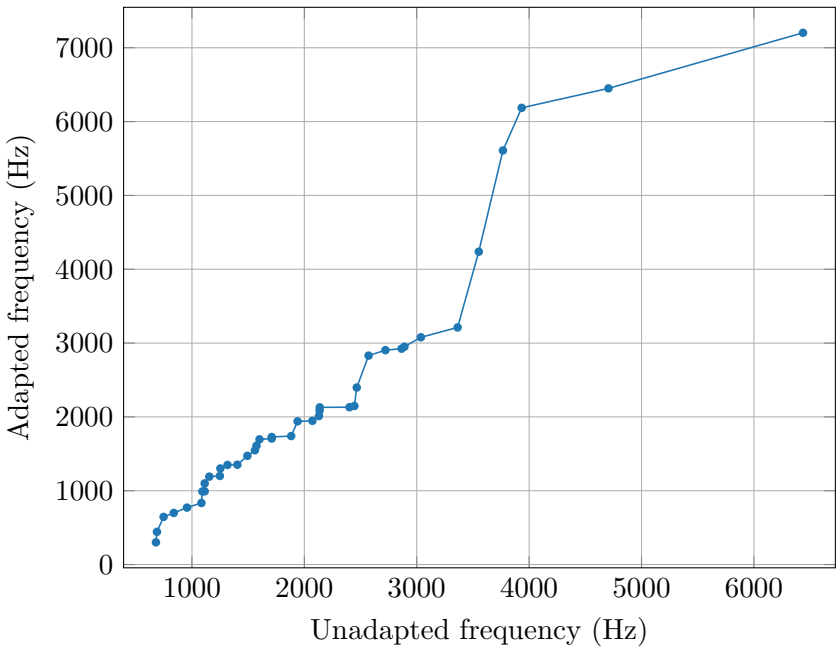


Figure 8.8: SincNet centre frequencies after domain adaptation, plotted against the unadapted frequencies (i. e. VTLN-like function).

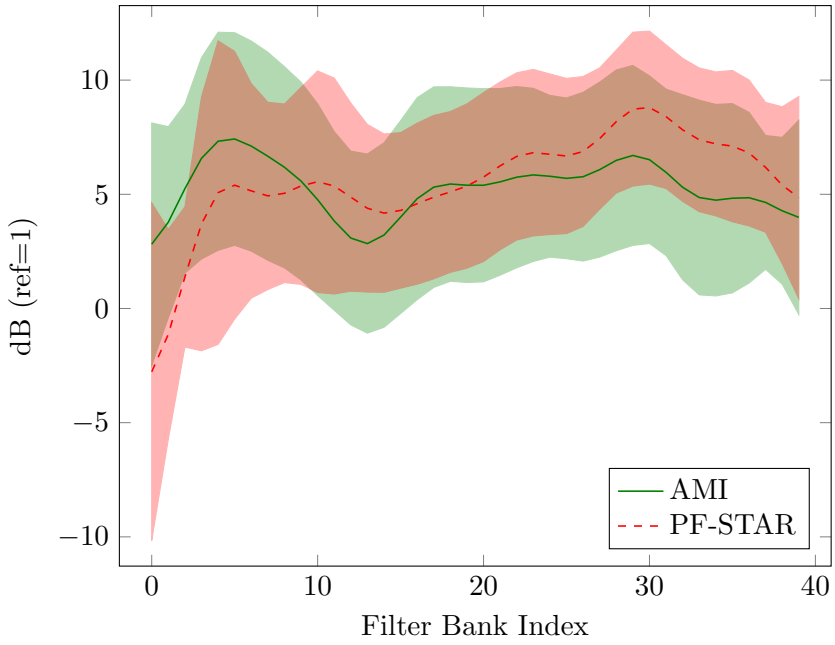


Figure 8.9: Average log-mel filterbank spectra for random subsets of AMI and PF-STAR data. The shaded region denotes plus-minus one standard deviation.

Params/utterances	0	1	2	3	20
Sinc	68.19	56.67	47.87	40.36	31.06
All–Sinc	68.19	56.70	47.42	35.89	21.34
All+Sinc	68.19	54.55	44.21	32.83	18.92

Table 8.4: Results WER (%) on the PF-STAR test set given the number of adaptation utterances when adapting from the AMI baseline model. ±Sinc indicates whether the SincNet layer was included in the trainable parameters. For small amounts of data, WERs are similar between adapting all parameters and adapting only the SincNet layer. As the amount of data increases, adapting all parameters yield larger improvements than adapting only the SincNet layer.

to the assumptions made during typical use of VTLN. However, it was here obtained through backpropagation instead of a grid-search or other methods.

Table 8.4 demonstrates the effect of the number of adaptation utterances. As the amount of data increases, adapting all parameters (excluding SincConv) produces lower error rates, as should be expected. The models begin to diverge at about three utterances (roughly 1 minute for PF-STAR).

8.4.2. Speaker adaptation

A more realistic, practical scenario, is to adapt to a few utterances obtained per speaker rather than per domain. In these experiments the AMI model is adapted to 12 individual speakers in PF-STAR's `eval/adapt` set, testing on the corresponding speakers in `eval/test`. The results are shown in Figure 8.10 which shows the evolution

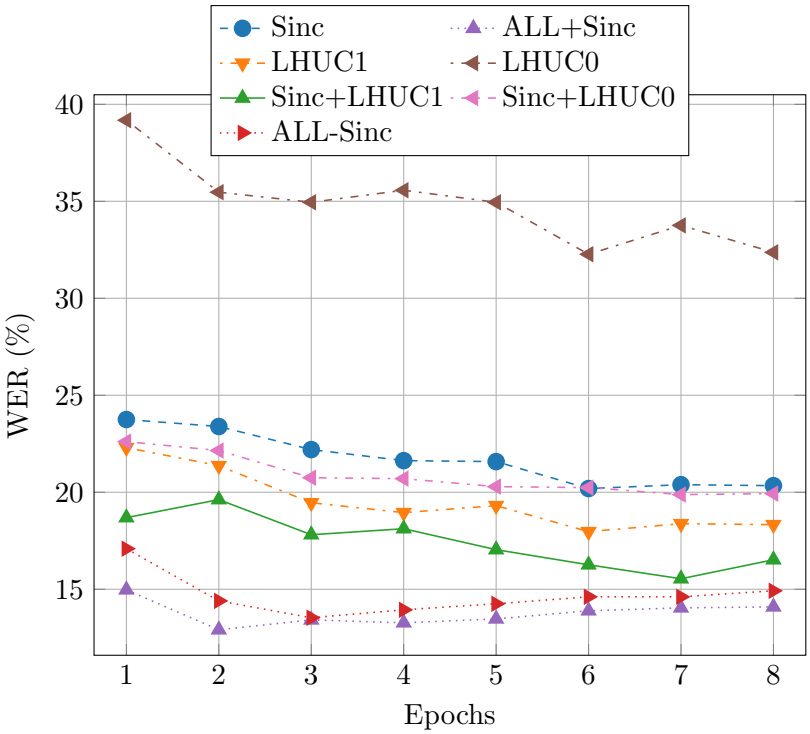


Figure 8.10: Speaker adaptation over epochs with various techniques. The unadapted model obtains 59.06% WER.

with the number of epochs of adaptation. LHUC0 indicates using LHUC on the output of the SincConv layer (40 parameters; equivalent to the gain of the filters), and LHUC1 is LHUC on the output of the first CNN layer (800 parameters). A learning rate of 0.8 is used for LHUC0 and LHUC1, as LHUC can characteristically use very large learning rates without overfitting (Klejch, Fainberg, and Bell, 2018; Swietojanski, Li, and Renals, 2016). When using LHUC1 in combination with SincConv, the standard 0.0015 learning rate is used for SincConv, but a multiplier of 500 for LHUC. However, this was not found beneficial for Sinc+LHUC0, for which the same learning rate was used for both sets of parameters.

The un-adapted WER is 59.06% on `eval/test`. Adapting the 80 parameters of the SincConv layer yields only slightly worse results than LHUC1 with 10 times fewer parameters. Interestingly, the two are complementary, as demonstrated by Sinc+LHUC1, and at best produces WERs similar to adapting all 9M parameters. ALL-Sinc and ALL+Sinc are more sensitive to overfitting as evident from the figure.

Adapting the gain of the filters (LHUC0) improves substantially over the unadapted model, but does not provide similar performance to any of the other approaches. One factor may be that the these parameters were fixed during the training of the baseline model as in Ravanelli and Bengio (2018), hence the rest of the network may have compensated by other means. It is, however, complementary with adapting the filterbank frequencies, with Sinc+LHUC0 slightly outperforming Sinc. A summary of the results after adapting for eight epochs is shown in Table 8.5.

Method	Adapted params	WER (%)
Unadapted	-	59.06
Sinc	80	20.34
LHUC0	40	32.37
Sinc+LHUC0	120	19.93
LHUC1	800	18.33
Sinc+LHUC1	880	16.52
ALL-Sinc	$\sim$ 9M	14.92
ALL+Sinc	$\sim$ 9M	14.09

Table 8.5: Results WER (%) adapting from the AMI baseline model to individual speakers in the PF-STAR `eval/adapt` set for 8 epochs. Tested on corresponding speakers in `eval/test`. ±Sinc indicates whether the SincNet layer was included in the trainable parameters. Adapting the SincNet layer yields large reductions in WERs in a parameter efficient manner. Adapting the SincNet layer and the LHUC parameters in the next layer is complementary. All results are statistically significant with $p < 0.001$ with respect to the unadapted baseline; the pairs LHUC1 / Sinc+LHUC1 and All-Sinc / All+Sinc are not statistically significant under the matched pairs test, but have probabilities of improvement of 99% and 94.5%, respectively (see Section 4.6).

Figure 8.11 shows VTLN-like functions obtained from the adapted SincConv layer to each speaker. There is a clear difference between each function, which is in line with what one might expect given the variability of the acoustics of children's data (Potamianos and Narayanan, 2003).

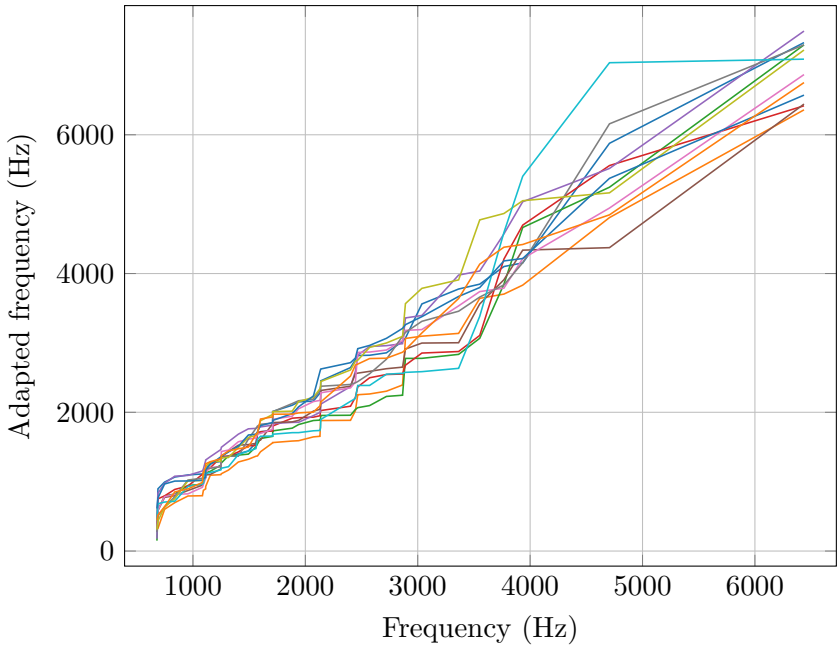


Figure 8.11: Corresponding VTLN scaling functions obtained from adapting the SincNet layer for individual speakers (colours). The majority of the scaling occurs in the higher frequencies.

8.4.3. Unsupervised speaker adaptation

Thus far we have demonstrated the utility of adapting the SincNet layer in a supervised adaptation setting. Next we show results for an identical setup as the speaker adaptation experiments in the previous section, but replacing the supervised adaptation targets with targets obtained through a recognition pass with the baseline model. With a high baseline error rate, and consequently inaccurate adaptation targets, there is a risk of overfitting to incorrect data. Techniques such as LHUC (see Section 2.4) are more robust to overfitting by limiting the number of adaptation parameters, or the expressivity of adaptation. Adapting the SincNet layer admits a different type of expressivity, also with a small number of parameters.

Figure 8.12 shows the results of adapting to first pass targets, with otherwise identical setups to the previous section, except that we have reduced the learning rate for All+Sinc and All-Sinc to 15e-5. With a baseline model with such a high error rate, it is unsurprising that adapting to such targets does not yield nearly as good performance as adapting to the oracle targets above. However, in contrast to the oracle results, it is clear that adapting all parameters is no longer the best option, showing clear signs of overfitting with an increasing number of epochs, after an initial improvement in the first epoch. This may be controlled by early-stopping on a held-out validation set. However, note that adapting the 80 parameters in the SincNet layer yields lower error rates, without obvious signs of overfitting – demonstrating a similar robustness as that is known for LHUC (see Swietojanski, Li, and Renals (2016)). Further, perhaps due to the different expressivity of the different techniques, we observe again that the adaptation

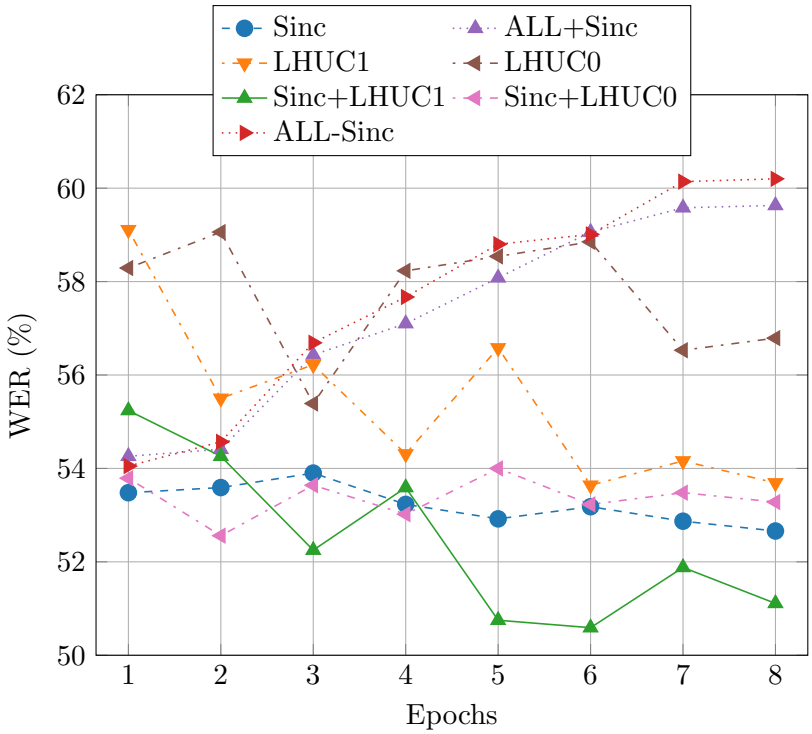


Figure 8.12: Unsupervised speaker adaptation over epochs with various techniques. The unadapted model obtains 59.06% WER.

of LHUC and SincNet parameters is complementary. Table 8.6 summarises the results.

8.5. Conclusions

This chapter proposed to adapt a SincNet layer as a parameter-efficient method which proved robust to large changes in speaker acoustics with first-pass tarets. The experiments have shown that adapting the filterbank frequencies from raw waveforms with SincNet is extremely parameter efficient, obtaining substantial improvements in WERs with a fraction of the total model parameters on a children's speaker adaptation task. It is also complementary with the standard LHUC technique, producing results similar to adapting all 9 million model parameters (excluding the filterbank layer). In unsupervised speaker adaptation experiments, adapting the SincNet layer demonstrates a robustness to errors from a first pass decode, similar to that known for LHUC. The parameterisation of SincNet further affords interpretability during adaptation: for domain adaptation to children's speech, the layer learns to pay more attention to higher frequencies. Similarly for speaker adaptation, the change in the filter frequencies in effect resembles VTLN, producing individual scaling functions for each speaker. Finally, it was noted that adapting the gain is related to LHUC and CMLLR, and this proved complementary to adapting the filterbank frequencies.

Future work could explore the use of meta-learning (as in Klejch, Fainberg, and Bell, 2018) to learn filter-specific learning rates and explore the layer's response to other

Method	Adapted params	WER (%)
Unadapted	-	59.06
Sinc	80	52.66
LHUC0	40	56.79
Sinc+LHUC0	120	53.28
LHUC1	800	53.69
Sinc+LHUC1	880	51.11
ALL-Sinc (1 epoch)	$\sim$ 9M	54.36
ALL+Sinc (1 epoch)	$\sim$ 9M	54.26

Table 8.6: Unsupervised adaptation results WER (%) adapting from the AMI baseline model to individual speakers in the PF-STAR `eval/adapt` set for 8 epochs, except those that include all parameters, which were stopped after 1 epoch. Tested on corresponding speakers in `eval/test`. ±Sinc indicates whether the SincNet layer was included in the trainable parameters. All results are statistically significant with $p < 0.001$ with respect to the unadapted baseline; the addition of Sinc to LHUC0 and to LHUC1 are both significant with $p < 0.01$ (see Section 4.6).

factors such as noise. Finally, if we consider adapting the SincConv layer as a neural analogue of VTLN, one may envisage extending this to related techniques such as Vocal Tract Length Perturbation (VTLP) for data augmentation (Jaitly and Hinton, 2013).

Chapter 9

Conclusions

In Chapter 1 we commented on the increasing ubiquity of ASR technology in everyday applications. We noted that many challenges still remain, such as handling mismatch between training and test time conditions, or mismatch with transcriptions. In other words, we required methods of robustly learning acoustic representations when facing changing acoustic conditions, and inexact supervision. This meant both improving the use of mismatched or inaccurate training data as well as suitable updating methods. We identified four key challenges that we explored in the chapters that followed:

1. **Light supervision**. How may inaccurate transcripts be used effectively during training?

In Chapter 5 we showed that inaccurate transcripts can yield significant improvements to WERs in a robust manner, when combined in a particular way with semi-supervised training. Specifically, the agreement between hypotheses from semi-supervised training and the transcripts informs the model of the uncertainty in the supervision, and we were able to implement this core idea with a simple FST algorithm. This technique proved robust to errors in the transcripts, which otherwise increased WERs if trained on directly.

2. **Factorised adaptation**. How can feature representations be factorised, such that they can be combined in novel combinations at test time?

Even with perfect transcriptions we need to consider changing contexts, such as combinations of speakers and their environments. In Chapter 6 we demonstrated that neural networks trained to classify a particular factor, such as a speaker, implicitly and effectively factor out nuisance factors, such as an environment, in the internal representations of the network. The corresponding bottleneck features were shown to be robust when combined in novel combinations at test time, without performance degradation.

3. **Longitudinal learning**. When data is ephemeral, how well does a model improve with continuous training, and what are the implications for active learning?

With longitudinal training the context from which the data originates is continuously changing. In Chapter 7 we saw that restarting the learning rate schedules per longitudinal

time-step (or episode) helped bridge the gap in performance with the batch case with pooled data, and that a model trained in this way obtains performance just shy of the batch case. The corresponding models further combined well over time, either with posterior ensembling or weight averaging. To perform active learning in this setting, it was required to combine the transcribed low-confidence data with the remaining untranscribed data as in semi-supervised training.

4. **Adaptation**. What is a parameter-efficient, robust way to adapt a model to new speaker acoustics, such as child speech?

Updating a model to account for changing contexts may require compact representations. In Chapter 8 we demonstrated that the parameterisation of the SincNet layer lends itself well to adaptation from adult speech to child speech, with a transform that can be interpreted as adjusting for vocal tract length. Adapting this compact set of parameters proved very effective, and even more so when complemented with LHUC. Finally, its compact representation proved robust to errors in first-pass adaptation targets, yielding considerable improvements that still were complementary with LHUC.

9.1. Future work

Next we discuss possible extensions and ideas for further experiments with the methods introduced in the thesis.

Light supervision

Perhaps the most interesting avenue of future work is the use of light supervision in end-to-end models. We discuss this below, followed by ideas to improve the lattice combination algorithm itself.

Light supervision with end-to-end models. Semi-supervised and lightly-supervised training with end-to-end models is an open research topic. A direct application of existing techniques is to train on best path hypotheses obtained with hybrid systems (Li et al., 2019), for which the language model can be readily biased as in standard lightly-supervised training. Without involving a hybrid system, one approach is to filter based on confidence scores obtained using dropout (Dey et al., 2019). To bias the output with light supervision, one could use various language model fusion methods (see e. g. Chorowski and Jaitly, 2017; Kannan et al., 2018; Toshniwal et al., 2018). In general, fusion methods could be used to produce lightly-supervised hypotheses that could be filtered using MER. However, we should perhaps also consider ideas that do not have immediate correlates in previous work with hybrid models. For example, maybe light supervision with end-to-end models means to extract representations that are robust to alignment and transcription errors. In this vein Karita et al. (2018) propose to train end-to-end models in a semi-supervised manner on unpaired speech and text

datasets, by using a shared text-to-speech and text-to-text model. Another possible direction is to consider teacher-student methods with a confidence weight to dynamically choose between a teacher's posteriors and a transcription (Meng et al., 2019).

If we were to consider applying lattice combination directly, the application would depend on the architecture. Connectionist Temporal Classification (CTC) models can easily incorporate WFSTs both during training and decoding (Miao, Gowayyed, and Metze, 2015; Sak et al., 2015b; Soltau, Liao, and Sak, 2016). For example, Sak et al. (2015b) proposed to compute the forward-backward algorithm for a phone-based CTC model over a transducer $S \circ L \circ C$, where S encodes the output labels and their posteriors for each time frame (akin to U in Section 3.2), L is a linear transducer of the target label sequence, and C allows for repetitions and blank labels within the phone sequences. This is similarly possible for word-based models (Soltau, Liao, and Sak, 2016). It would be trivial to replace L with a lattice of supervision.

With encoder-decoder models, WFSTs in general feature less (although they are often used for contextual, or biased, decoding (Hall et al., 2015; Williams et al., 2018)) and due to the continuous hidden states, the models produce decoding trees, rather than lattices, with the exception of some particular decoder architectures with discrete hidden states that enable state recombination (Zapotoczny et al., 2019). If we were to obtain a hypothesis lattice, either from a hybrid seed-model or from such a decoder architecture, the question remains how to incorporate that lattice into training. Perhaps a lattice could be included within methods that attempt to directly minimise the expected WER (Prabhavalkar et al., 2018) (this also applies to CTC models, e. g. Graves and Jaitly, 2014); or individual paths could be sampled and used as labels, similar to the idea of sampling labels from an ensemble (Kahn, Lee, and Hannun, 2020) or using dropout (Dey et al., 2019), for semi-supervised training of encoder-decoder models.

Lattice combination efficiency. We noted in Chapter 5 that the lattice combination algorithm in its current implementation could become unwieldy with large vocabularies, specifically that the number of transitions in the edit-transducer grows with the square of the size of the vocabulary. In most applications this would not pose a problem, because the vocabulary can be restricted to the union of the transducers that are combined. We also noted that the time during lattice combination was negligible compared to generating hypothesis lattices. However, should it be required, it would be reasonably straightforward to factor the edit-transducer into two components in such a way that the transitions grow linearly with the vocabulary size.

Improved use of paraphrases. We did not observe any particular improvements using a 2-pass lattice combination setup with the transcriptions augmented by paraphrases. This may partly be attributed to a poor match of paraphrases to the data, and partly to the fact that our implementation only allows paraphrases at the word level. There are therefore two interesting paths to extend this work. The first is to collect, or extract, better paraphrases that more closely suit the data at hand. There is a wide range of existing literature to explore, with different techniques given the available

data, such as using a pivot language on parallel texts (Bannard and Callison-Burch, 2005), obtaining paraphrases between multiple variations of the same text (Barzilay and McKeown, 2001), or distributional methods on a single body of text (Liu, Gales, and Woodland, 2014; Pereira, Tishby, and Lee, 1993). A pitfall to keep in mind is that introducing new words into the transcription may result in lattice combination choosing ASR errors in the hypothesis lattice, should the introduced words match. However, this is arguably unlikely to occur at any scale if the paraphrases are sufficiently different phonetically to the original transcriptions. The second path of extension is to modify the algorithm to handle phrases, and not just individual words. Paraphrases at the phrasal level may prove much more powerful, and are also arguably more common. Theoretically it would be possible to include these directly into the existing framework, but care would need to be taken to avoid a dramatic expansion of computational requirements. A core part of such work may therefore focus on how to obtain a suitable, but compact, set of paraphrases for the task at hand.

Factorised adaptation

Key directions of research include improving the method of extraction, and to enable the use in more realistic applications.

Improved factorised representations. A limitation of the proposed method in Chapter 6 is that it yields orthogonality only implicitly. As we saw in the experiments, the nuisance factors were not completely factored out of the resulting representations. However, the results were still promising, and it would be interesting to see whether further improvements could be obtained. One approach could be to apply an adversarial objective, using e. g. a Gradient Reversal Layer (GRL) (Ganin and Lempitsky, 2015; Ganin et al., 2016). Shinohara (2016) used a GRL with a multi-task objective to encourage invariance to noise conditions in the internal representations of an acoustic model. Another avenue of interesting research is to look at work done on disentangling factors of variation using, for example, autoencoders (Cheung et al., 2015; Makhzani et al., 2016). With autoencoders we could feed environment (or other nuisance factors) class labels into the decoder. The latent representations produced by the encoder would then be required to model the remaining variation in the data, and these could possibly serve as factorised representations that could be used for training acoustic models.

Towards realistic applications. The proposed method requires labels for both factors of variation (speakers and environments). More realistic datasets will not contain such labels. Ideally, we would like to adapt using factorised representations on data such as the MGB corpus. If multi-modal data is available, then one possible idea is to consider cross-modal disentangling (Nagrani et al., 2020), which makes use of correlations between video and audio content to learn disentangled embeddings. Alternatively, there is work on unsupervised disentangling from a single modality, such as *InfoGAN* (Chen et al., 2016), which makes use of a Generative Adversarial Network (GAN) (Goodfellow

et al., 2014) to learn latent representations of the data, but with a modification to the objective that encourages meaningful representations. An advantage to unsupervised learning of the factors, is that the delineation into two axes of variation by speakers and environments is not necessarily optimal from an acoustic modelling perspective. In this framework it is also trivial to add additional possible factors to the model.

Longitudinal learning

Many aspects of longitudinal learning are still unexplored. Of particular interest is observing the evolution over longer time-spans. We then propose to break down longitudinal learning into studying individual factors across time.

Experiments over longer time-spans. We observed in Chapter 7 that the differences between the batch case, where we could pool previous data, and longitudinal, were not particularly large. However, the results suggested that these differences grew with time. In our experiments we had chosen the TV show with the largest number of episodes from the MGB corpus, yet it would be interesting to go much beyond this. One possibility would be to use TED talks[1] and experiment with longitudinal training over a long time span. Such data would also provide a useful means of analysis: we expect the model to improve given that the scenario is fairly static with one host, one speaker and an audience in the same room from talk to talk. Yet, the identity of the speaker changes, while the host (usually) does not. In the longer time-span, large differences may emerge between the batch case and longitudinal training. As we have noted, it is possible to see the difference as simply due to the ability to shuffle data across time. It would be interesting to better quantify this difference and to look for principled ways to correct for it.

Longitudinal learning for individual factors. We experimented with longitudinal adaptation to episodes as a whole. However, it may be interesting to instead learn representations for individual factors across time. For example, we may observe speakers come and go, some recurring and some occurring only once. Using unsupervised speaker diarisation, it may be interesting to incrementally update representations (e. g. LHUC or i-vectors; Section 2.4) for each factor independently. Moreover, we may use ideas from Chapter 6 on factorised adaptation: it may be that the same acoustic environment occurs repeatedly across time, while the speakers change.

Adaptation

Below we present ideas for further experiments and to improve the model in general.

Exploring the relationship to VTLN. We discussed how the SincNet layer is similar to a neural, unconstrained version of VTLN. Building on this analogue, it would be interesting to explore even more compact parameterisations, such as the use of a single warping parameter as in standard VTLN. Another interesting line of

[1] `www.ted.com`.

work is to consider data augmentation with the SincNet layer, taking inspiration from VTLP (Jaitly and Hinton, 2013).

Improved training. Perhaps a limitation of SincNet for adaptation is that adaptation only occurs on an input layer. Skip-transitions (He et al., 2016) could perhaps be used to more directly impact higher layers. It would also be interesting to explore how fast different filters in the filterbank need to adapt. For example, we could apply meta-learning to learn filter-specific learning rates (Klejch, Fainberg, and Bell, 2018). This may perhaps also benefit unsupervised adaptation with SincNet. Finally, we could pretrain the SincNet and surrounding layers to create an improved feature extractor with a technique similar to `wav2vec` (Schneider et al., 2019), which trains a raw-waveform model to distinguish future samples from distractor samples.

Looking ahead

We started this thesis with four research questions, or challenges. In light of the results and the above discussion, we can propose new questions that build on the work presented. First, it can be interesting, and of large practical benefit, to consider how external knowledge may be applied to light supervision, such as meta-data or contextual information (e. g. Aleksic et al., 2015) that may help inform lightly-supervised training. Second, as we discussed above, we can consider how we can discover and untangle factors of variation in data without explicit supervision such as a priori environment classes. This is a key component to using factorised representations in many real applications. But perhaps the key challenge is to combine these ideas, along with improved longitudinal training and efficient raw-waveform modelling. That could culminate into a system that efficiently and robustly adapts over time, that makes the best use of any supervision available, and that can make effective use of factors of variations interleaving over time.

Appendix A

Models

Throughout the thesis we may refer to certain models as `Kaldi-X` or `Keras-X`, where `X` indicates the model ID. The following sections provides a brief overview of each model and provides the recipe URLs (if any) at the time of writing. The models labelled `Kaldi` are trained in Kaldi (Povey et al., 2011), the ones labelled `Keras` are trained using Keras (Chollet et al., 2015) and Tensorflow (Abadi et al., 2016).

Optimisers

We did not discuss optimisers in Chapter 2, but below we list the optimisers used in the experiments. Kaldi uses its own variation of *natural gradient* (Amari, 1998) which may convergence faster than standard Stochastic Gradient Descent (SGD) (Povey, Zhang, and Khudanpur, 2014). Adam (Kingma and Ba, 2015) is an algorithm for adaptive learning rates based on the first and second moments of the gradients.

A.1. Kaldi-Libri

This is a standard TDNN-F model for Librispeech, mentioned in Chapter 2, shown in Table A.1. It obtains 8.76% WER on `test-other`. The recipe is available here: `github.com/kaldi-asr/kaldi/blob/master/egs/librispeech/s5/local/chain/tuning/run_tdnn_1d.sh`.

Feature	Details
Input	40 dim. LDA + 100 dim. i-vectors
Layers	16 x TDNN-F+BN 1536 units w/ 160 bottleneck
Splicing	Most layers $[-3, +3]$
Activations	ReLU
Dropout	Schedule: 0.5 at 20%, 0 at 50%
Objective	LF-MMI + CE
Optimiser	Natural gradient

Table A.1: `Kaldi-Libri` model details. Minor details not shown.

A.2. Kaldi-SWBD

This is a TDNN-F+LSTM model for Switchboard, mentioned in Chapter 2, shown in Table A.2. It obtains 8.8% WER on the Switchboard portion of the `eval2000` test set, using 4-gram rescoring. The recipe is available here: `github.com/kaldi-asr/kaldi/blob/master/egs/swbd/s5c/local/chain/tuning/run_tdnn_lstm_1l.sh`.

Feature	Details
Input	40 dim. LDA + 100 dim. i-vectors
Layers	7 x TDNN interspersed by 3 x LSTM, 1024 units
Splicing	Most layers $[-3, 0, +3]$
Activations	ReLU
Dropout	Schedule: 0.5 at 20%, 0 at 50%
Objective	LF-MMI + CE
Optimiser	Natural gradient

Table A.2: `Kaldi-SWBD` model details. Minor details not shown.

A.3. Kaldi-1

This is a standard TDNN-F model for Switchboard, used in Chapter 5 for training using the Scottish Parliament, shown in Table A.4. It uses unconstrained examples (see Section 2.3.1). It obtains 28.6% WER with a biased 4-gram MGB LM on the MGB `dev.full` set, and 22.8% WER with a non-biased 4-gram MGB LM on the Scottish Parliament test set (see Chapter 4). The recipe is available here: `https://github.com/kaldi-asr/kaldi/blob/master/egs/swbd/s5c/local/chain/tuning/run_tdnn_7p.sh`.

Feature	Details
Input	40 dim. LDA + 100 dim. i-vectors
Layers	11 x TDNN-F+BN 1280 units w/ 256 bottleneck
Splicing	Layers 1-4 $[-1, +1]$; 5-11 $[-3, +3]$
Activations	ReLU
Dropout	Schedule: 0.5 at 20%, 0 at 50%
Objective	LF-MMI + CE
Optimiser	Natural gradient

Table A.3: `Kaldi-1` model details. Minor details not shown.

A.4. Kaldi-2

This is a standard TDNN model for WSJ, used in Chapter 6, shown in Table A.4. It obtains 10.36% on `dev93` and 6.72% on `eval92` with a 3-gram LM trained on WSJ training data. The recipe corresponds to most default parameters of this script at the time of writing (see table for exceptions):
`https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/steps/nnet3/train_tdnn.sh`.

Feature	Details
Input	13 dim. MFCC + 100 dim. i-vectors
Layers	6 x TDNN p-norm, in=2000, out=250
Splicing	$[4, 4], [0], \{-2, 2\}, \{-4, 4\}, [0]$
Activations	p-norm (p=2)
Dropout	None
Objective	CE
Optimiser	Natural gradient

Table A.4: `Kaldi-2` model details. Minor details not shown.

A.5. Keras-1

This is a TDNN topology used for the AMI result in Chapter 4, shown in Table A.5 and illustrated in **??**. It is trained using Keras and Tensorflow, but is similar to the AMI Kaldi recipe:
`https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/local/nnet3/run_tdnn.sh`. The Keras setup is part of the work on domain and speaker adaptation with SincNet (Ravanelli and Bengio, 2018) in Chapter 8, which is available here:
`https://github.com/jfainberg/sincnet_adapt`.

Feature	Details
Input	40 dim MFCC
Layers	5 x 800 dim TDNN
Kernel sizes	129, 2, 2, 2, 2, 2, 1
Dilations	0, 1, 3, 6, 9, 6, 1
Activations	ReLU
Dropout	None
Objective	CE
Optimiser	Adam

Table A.5: `Keras-1` model details. Minor details not shown.

A.6. Keras-2

This is a TDNN SincNet (Ravanelli and Bengio, 2018) topology used for PF-STAR in Chapter 8, shown in Table A.6. It is trained using Keras and Tensorflow, but is similar to the AMI Kaldi recipe without the SincNet layer:
`https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/local/nnet3/run_tdnn.sh`. The Keras setup is part of the work on domain and speaker adaptation with SincNet (Ravanelli and Bengio, 2018) in Chapter 8, which is available here:
`https://github.com/jfainberg/sincnet_adapt`.

Feature	Details
Input	Raw waveform 200ms
Layers	SincNet (40) + 5 x 800 dim TDNN+BN+MaxPool
Kernel sizes	129, 2, 2, 2, 2, 2, 1
Dilations	0, 1, 3, 6, 9, 6, 1
Activations	ReLU
Dropout	None
Objective	CE
Optimiser	Adam

Table A.6: `Keras-2` model details. Minor details not shown.

Appendix B

Example transcripts

This chapter provides some example transcripts from the test sets of data presented in Chapter 4.

B.1. Multi-Genre Broadcast corpus

As discussed in Section 4.1, the MGB corpus Bell et al., 2015 contains subtitles as transcriptions. In the examples below we have provided both the subtitles (top) and the verbatim transcriptions (bottom).

73.17% MER

```
I just can't believe this this is amazing you look great you
   really you look I thought you were in America I thought you'd
   moved to look at you with your suit and I'm in a suit we're in
    a suit.
---
This is amazing, you [inaudible] great you really, you know I
   thought you were in America I thought you'd moved to, look at
   you in your suit and I'm in a suit and we're in a suit.
```

45.45% MER

```
One of the most important things in my life is music.
---
The most important things in my life is music.
```

21.43% MER

```
But remember the mountain will try its utmost to shake you from
   the beam.
---
But remember the mountain will try its utmost to shake you from
   the beam.
```

B.2. Scottish Parliament

As above with the MGB corpus, and as discussed in Section 4.2, the ScotParl dataset includes subtitles as transcriptions. In the examples below we have provided both the subtitles (top) and the verbatim transcriptions (bottom).

```
First item on our agenda is to decide whether to take in item
   four
---
First item on our agenda is the decision whether to take agenda
   item four in private
```

```
Agree to make no recommendation
---
Is it agreed that we make no recommendations thank you very much
   um
```

```
Graeme will that by the senior phase is a three year experience
   it is not helpful I know that you are telling me things I know
    I am asking a specific question [...]
---
Graeme do you wanna come in come in Mr Scott on that I think the
   senior phase by design is a three year experience so it's not
   helpful please I know that you're telling me things I know I'm
    asking you a specific question [...]
```

B.3. Wall Street Journal

The WSJ corpus (Paul and Baker, 1992), discussed in Section 4.3, contains read speech from utterances extracted from the newspaper of the same name. Below are some examples from the `eval92` test set.

```
As with the rest of the regime however their ideology became
   contaminated by the germ of corruption.
```

```
The company declined to comment on the results.
```

```
Even the ebullience on the floor of the New York Stock exchange
   was tempered by a few gloomy thoughts.
```

B.4. AMI

We have selected utterances that contain more than one or two words (such as `yeah`, `hmm`, and `ok`). See Section 4.4 for a discussion on the utterance lenghts in the AMI corpus (Carletta, 2007).

```
Wonder how much of the meetings is talking about the stuff at the
    meetings.
```

```
If our company is if it is easily recognisable that our company
   made it.
```

```
Is it everybody is going to evaluate or just the market okay.
```

B.5. PF-STAR

We noted in Section 4.5 that the PF-STAR corpus (Batliner et al., 2005) has somewhat artificial utterances. Below we include three examples: lists of numbers, lists of three-word phrases, and list of nouns.

```
One one six four five three five zero eight zero three three four
    seven seven [...]
```

```
Confusion of sounds draw a red book I can do anything it's got
   long pink hair my birthday is in july [...]
```

```
Kid tree dog water cup car book get man girl sister day time
   noise town boat [...]
```

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283 (cit. on pp. 98, 116).

Abdel-Hamid, O. and Jiang, H. (2013). "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code". In: *Proc. ICASSP*. IEEE, pp. 7942–7946 (cit. on pp. 19, 72).

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G., and Yu, D. (2014). "Convolutional neural networks for speech recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.10, pp. 1533–1545 (cit. on pp. 12, 15).

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., and Penn, G. (2012). "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition". In: *Proc. ICASSP*. IEEE, pp. 4277–4280 (cit. on p. 15).

Aleksic, P., Ghodsi, M., Michaely, A., Allauzen, C., Hall, K., Roark, B., Rybach, D., and Moreno, P. (2015). "Bringing contextual information to google speech recognition". In: *Proc. Interspeech*, pp. 468–472 (cit. on p. 115).

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). "OpenFst: A general and efficient weighted finite-state transducer library". In: *International Conference on Implementation and Application of Automata*, pp. 11–23 (cit. on p. 52).

Amari, S.-I. (1998). "Natural gradient works efficiently in learning". In: *Neural Computation* 10.2, pp. 251–276 (cit. on p. 116).

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). "Deep speech 2: End-to-end speech recognition in english and mandarin". In: *Proc. ICML*, pp. 173–182 (cit. on p. 37).

Anastasakos, T., McDonough, J., Schwartz, R., and Makhoul, J. (1996). "A compact model for speaker-adaptive training". In: *Proc. ICSLP*. Vol. 2. IEEE, pp. 1137–1140 (cit. on p. 11).

Axelrod, S., Goel, V., Gopinath, R. A., Olsen, P. A., and Visweswariah, K. (2005). "Subspace constrained Gaussian mixture models for speech recognition". In: *IEEE Transactions on Speech and Audio Processing* 13.6, pp. 1144–1160 (cit. on p. 10).

Bahl, L., Brown, P., De Souza, P., and Mercer, R. (1986). "Maximum mutual information estimation of hidden Markov model parameters for speech recognition". In: *Proc. ICASSP*. Vol. 11. IEEE, pp. 49–52 (cit. on p. 16).

Bannard, C. and Callison-Burch, C. (2005). "Paraphrasing with Bilingual Parallel Corpora". In: *Proc. ACL*, pp. 597–604 (cit. on pp. 50, 113).

Barker, J., Marxer, R., Vincent, E., and Watanabe, S. (2015). "The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines". In: *Proc. ASRU*, pp. 504–511 (cit. on pp. 31, 37).

Barker, J., Vincent, E., Ma, N., Christensen, H., and Green, P. (2013). "The PASCAL CHiME speech separation and recognition challenge". In: *Computer Speech & Language* 27.3, pp. 621–633 (cit. on p. 1).

Barker, J., Watanabe, S., Vincent, E., and Trmal, J. (2018). "The fifth 'CHiME' speech separation and recognition challenge: dataset, task and baselines". In: *Proc. Interspeech*, pp. 1561–1565 (cit. on p. 1).

Barzilay, R. and McKeown, K. R. (2001). "Extracting paraphrases from a parallel corpus". In: *Proc. ACL*, pp. 50–57 (cit. on pp. 51, 113).

Batliner, A., Blomberg, M., D'Arcy, S., Elenius, D., Giuliani, D., Gerosa, M., Hacker, C., Russell, M., Steidl, S., and Wong, M. (2005). "The PF_STAR Childrens Speech Corpus". In: *Proc. Interspeech*, pp. 2761–2764 (cit. on pp. 32, 39, 97, 123).

Batlle, E., Nadeu, C., and Fonollosa, J. A. (1998). "Feature decorrelation methods in speech recognition. A comparative study". In: *Proc. ICSLP* (cit. on p. 10).

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *The Annals of Mathematical Statistics* 41.1, pp. 164–171 (cit. on p. 11).

Bell, P. et al. (2015). "The MGB Challenge: Evaluating multi-genre broadcast media recognition". In: *Proc. ASRU*. IEEE, pp. 687–693 (cit. on pp. 1, 31–34, 41, 42, 51, 52, 72, 80, 121).

Bell, P., Fainberg, J., Lai, C., and Sinclair, M. (2017). "A System for Real Time Collaborative Transcription Correction". In: *Proc. Interspeech*, pp. 817–818 (cit. on p. 22).

Bell, P. and Renals, S. (2015). "A system for automatic alignment of broadcast media captions using weighted finite-state transducers". In: *Proc. ASRU*. IEEE, pp. 675–680 (cit. on pp. 42, 43).

Bengio, Y., Simard, P., and Frasconi, P. (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166 (cit. on p. 12).

Bilmes, J. A. (2006). "What HMMs can do". In: *IEICE Transactions on Information and Systems* 89.3, pp. 869–891 (cit. on pp. 8, 9).

Bisani, M. and Ney, H. (2004). "Bootstrap estimates for confidence intervals in ASR performance evaluation". In: *Proc. ICASSP*. Vol. 1. IEEE, pp. I–409 (cit. on p. 40).

Braunschweiler, N., Gales, M. J., and Buchholz, S. (2010). "Lightly supervised recognition for automatic alignment of large coherent speech recordings". In: *Proc. Interspeech*, pp. 2222–2225 (cit. on p. 42).

Brown, P. F. (1987). "The Acoustic-Modeling Problem in Automatic Speech Recognition". In: *Ph. D. Thesis, Carnegie Mellon University* (cit. on pp. 10, 16).

Carletta, J. (2007). "Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus". In: *Language Resources and Evaluation* 2, pp. 181–190 (cit. on pp. 1, 32, 37, 97, 122).

Chan, H. Y. and Woodland, P. (2004). "Improving broadcast news transcription by lightly supervised discriminative training". In: *Proc. ICASSP*. Vol. 1. IEEE, pp. I–737 (cit. on p. 42).

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Proc. ICASSP*. IEEE, pp. 4960–4964 (cit. on p. 7).

Chan, W. and Lane, I. (2015). "Deep recurrent neural networks for acoustic modelling". In: *arXiv preprint arXiv:1504.01482* (cit. on p. 37).

Chen, L., Lamel, L., and Gauvain, J.-L. (2004). "Lightly supervised acoustic model training using consensus networks". In: *Proc. ICASSP*. Vol. 1. IEEE, pp. I–189 (cit. on p. 43).

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *Proc. NIPS*, pp. 2172–2180 (cit. on p. 113).

Chen, Z., Luitjens, J., Xu, H., Wang, Y., Povey, D., and Khudanpur, S. (2018). "A GPU-based WFST decoder with exact lattice generation". In: *Proc. Interspeech*, pp. 2212–2216 (cit. on p. 26).

Chen, Z., Yarmohammadi, M., Xu, H., Lv, H., Xie, L., Povey, D., and Khudanpur, S. (2019). "Incremental Lattice Determinization for WFST Decoders". In: *Proc. ASRU*. IEEE, pp. 1–7 (cit. on p. 26).

Cheung, B., Livezey, J. A., Bansal, A. K., and Olshausen, B. A. (2015). "Discovering hidden factors of variation in deep networks". In: *Proc. ICLR* (cit. on p. 113).

Chollet, F. et al. (2015). *Keras*. `https://github.com/keras-team/keras` (cit. on pp. 68, 98, 116).

Chorowski, J., Bahdanau, D., Cho, K., and Bengio, Y. (2014). "End-to-end continuous speech recognition using attention-based recurrent NN: First results". In: *NIPS Workshop on Deep Learning* (cit. on p. 7).

Chorowski, J. and Jaitly, N. (2017). "Towards Better Decoding and Language Model Integration in Sequence to Sequence Models". In: *Proc. Interspeech*, pp. 523–527 (cit. on p. 111).

Cieri, C., Miller, D., and Walker, K. (2004). "The Fisher Corpus: a Resource for the Next Generations of Speech-to-Text". In: *LREC*, pp. 69–71 (cit. on p. 99).

Davis, S. and Mermelstein, P. (1980). "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal processing* 28.4, pp. 357–366 (cit. on pp. 14, 15).

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2010). "Front-end factor analysis for speaker verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4, pp. 788–798 (cit. on p. 19).

Dehghani, M, Mehrjou, A, Gouws, S, Kamps, J, and Schölkopf, B (2018). "Fidelity-Weighted Learning". In: *Proc. ICLR* (cit. on p. 44).

Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., et al. (2013). "Recent advances in deep learning for speech research at Microsoft". In: *Proc. ICASSP*. IEEE, pp. 8604–8608 (cit. on p. 15).

Dey, S., Motlicek, P., Bui, T., and Dernoncourt, F. (2019). "Exploiting semi-supervised training through a dropout regularization in end-to-end speech recognition". In: *Proc. Interspeech*, pp. 734–738 (cit. on pp. 111, 112).

Doulaty, M., Saz, O., and Hain, T. (2015). "Data-Selective Transfer Learning for Multi-Domain Speech Recognition". In: *Proc. Interspeech*, pp. 2897–2901 (cit. on p. 2).

Doulaty, M., Saz, O., Ng, R. W., and Hain, T. (2015). "Latent Dirichlet Allocation Based Organisation of Broadcast Media Archives for Deep Neural Network Adaptation". In: IEEE, pp. 130–136 (cit. on p. 33).

Driesen, J. and Renals, S. (2013). "Lightly supervised automatic subtitling of weather forecasts". In: *Proc. ASRU*. IEEE, pp. 452–457 (cit. on pp. 41, 42).

Drugman, T., Pylkkönen, J., and Kneser, R. (2016). "Active and Semi-Supervised Learning in ASR: Benefits on the Acoustic and Language Models". In: *Proc. Interspeech*, pp. 2318–2322 (cit. on pp. 2, 76).

Dubagunta, S. P., Kabil, S. H., and Doss, M. M. (2019). "Improving Children Speech Recognition through Feature Learning from Raw Speech Signal". In: *Proc. ICASSP*. IEEE, pp. 5736–5740 (cit. on p. 99).

Fainberg, J., Bell, P., Lincoln, M., and Renals, S. (2016). "Improving Children's Speech Recognition Through Out-of-Domain Data Augmentation". In: *Proc. Interspeech*, pp. 1598–1602 (cit. on pp. 32, 39, 97).

Fainberg, J., Klejch, O., Loweimi, E., Bell, P., and Renals, S. (2019a). "Acoustic Model Adaptation from Raw Waveforms with SincNet". In: *Proc. ASRU*. IEEE, pp. 897–904 (cit. on p. 6).

Fainberg, J., Klejch, O., Renals, S., and Bell, P. (2019b). "Lattice-Based Lightly-Supervised Acoustic Model Training". In: *Proc. Interspeech*, pp. 1596–1600 (cit. on pp. 5, 26, 36).

Fainberg, J., Renals, S., and Bell, P. (2017). "Factorised Representations for Neural Network Adaptation to Diverse Acoustic Environments". In: *Proc. Interspeech*, pp. 749–753 (cit. on p. 6).

Ferguson, J. (1980). "Variable duration models for speech". In: *Proc. Symposium on the Application of Hidden Markov Models to Text and Speech* (cit. on p. 9).

Fiscus, J. G. (1997). "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)". In: *Proc. ASRU*. IEEE, pp. 347–354 (cit. on p. 43).

Furui, S. (1986). "Speaker-independent isolated word recognition using dynamic features of speech spectrum". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34.1, pp. 52–59 (cit. on p. 15).

Gales, M. (2001). "Acoustic factorisation". In: *Proc. ASRU*. IEEE, pp. 77–80 (cit. on pp. 61, 63).

Gales, M. J. (1998). "Maximum likelihood linear transformations for HMM-based speech recognition". In: *Computer Speech & Language* 2, pp. 75–98 (cit. on pp. 18, 61, 93, 96).

— (1999). "Semi-tied covariance matrices for hidden Markov models". In: *IEEE Transactions on Speech and Audio Processing* 7.3, pp. 272–281 (cit. on p. 10).

Gales, M. and Young, S. (2008). "The application of hidden Markov models in speech recognition". In: *Foundations and Trends® in Signal Processing* (cit. on pp. 9, 15).

Ganin, Y. and Lempitsky, V. (2015). "Unsupervised domain adaptation by backpropagation". In: *Proc. ICML*, pp. 1180–1189 (cit. on p. 113).

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). "Domain-adversarial training of neural networks". In: *Journal of Machine Learning Research* 17.59, pp. 1–35 (cit. on p. 113).

Ganitkevitch, J., Durme, B. V., and Callison-Burch, C. (2013). "PPDB: The Paraphrase Database". In: *Proc. HLT-NAACL*, pp. 758–764 (cit. on p. 51).

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). "Loss surfaces, mode connectivity, and fast ensembling of DNNs". In: *Proc. NIPS*, pp. 8789–8798 (cit. on pp. 76, 77, 81, 83, 91).

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. (1993). "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1". In: *NASA STI/Recon technical report n* (cit. on p. 11).

Gemello, R., Mana, F., Scanzio, S., Laface, P., and De Mori, R. (2007). "Linear hidden transformations for adaptation of hybrid ANN/HMM models". In: *Speech Communication* 49, pp. 827–835 (cit. on p. 18).

Ghahremani, P., Hadian, H., Lv, H., Povey, D., and Khudanpur, S. (2018). "Acoustic Modeling from Frequency Domain Representations of Speech". In: *Proc. Interspeech*, pp. 1596–1600 (cit. on p. 93).

Ghorbani, S., Khorram, S., and Hansen, J. H. (2019). "Domain Expansion in DNN-based Acoustic Models for Robust Speech Recognition". In: *Proc. ASRU*. IEEE, pp. 107–113 (cit. on p. 74).

Gibson, M. (2008). "Minimum Bayes risk acoustic model estimation and adaptation". PhD thesis. University of Sheffield (cit. on p. 16).

Gibson, M. and Hain, T. (2006). "Hypothesis spaces for minimum Bayes risk training in large vocabulary speech recognition". In: *Proc. Interspeech*, pp. 2406–2409 (cit. on pp. 16, 17).

Gillick, L. and Cox, S. (1989). "Some statistical issues in the comparison of speech recognition algorithms". In: *Proc. ICASSP*. Vol. 1. IEEE, pp. 532–535 (cit. on p. 40).

Giuliani, D. and Gerosa, M. (2003). "Investigating recognition of children's speech". In: *Proc. ICASSP*. Vol. 2. IEEE, pp. II–137 (cit. on p. 20).

Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992). "SWITCHBOARD: Telephone speech corpus for research and development". In: *Proc. ICASSP*. Vol. 1, pp. 517–520 (cit. on p. 7).

Goldberger, J. and Ben-Reuven, E. (2016). "Training deep neural-networks using a noise adaptation layer". In: *Proc. ICLR* (cit. on p. 44).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). "Generative adversarial nets". In: *Proc. NIPS*, pp. 2672–2680 (cit. on p. 113).

Graff, D. (2002). "An overview of Broadcast News corpora". In: *Speech Communication* 1-2, pp. 15–26 (cit. on p. 41).

Graves, A. and Jaitly, N. (2014). "Towards end-to-end speech recognition with recurrent neural networks". In: *Proc. ICML*, pp. 1764–1772 (cit. on pp. 7, 112).

Hadian, H., Povey, D., Sameti, H., Trmal, J., and Khudanpur, S. (2018a). "Improving LF-MMI Using Unconstrained Supervisions for ASR". In: *Proc. SLT*. IEEE, pp. 43–47 (cit. on p. 18).

Hadian, H., Sameti, H., Povey, D., and Khudanpur, S. (2018b). "End-to-end Speech Recognition Using Lattice-free MMI". In: *Proc. Interspeech*, pp. 12–16 (cit. on pp. 12, 18, 37).

— (2018c). "Flat-start single-stage discriminatively trained HMM-based models for ASR". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.11, pp. 1949–1961 (cit. on p. 9).

Hain, T. (2002). "Implicit pronunciation modelling in ASR". In: *ISCA Tutorial and Research Workshop (ITRW) on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology (PMLA)*, pp. 129–134 (cit. on p. 27).

Hall, K., Cho, E., Allauzen, C., Beaufays, F., Coccaro, N., Nakajima, K., Riley, M., Roark, B., Rybach, D., and Zhang, L. (2015). "Composition-based on-the-fly rescoring for salient n-gram biasing". In: *Proc. Interspeech*, pp. 1418–1422 (cit. on p. 112).

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition". In: *Proc. CVPR*, pp. 770–778 (cit. on pp. 12, 115).

Hermansky, H. (1990). "Perceptual linear predictive (PLP) analysis of speech". In: *the Journal of the Acoustical Society of America* 87.4, pp. 1738–1752 (cit. on p. 15).

Hochreiter, S. and Schmidhuber, J. (1997). "Long short-term memory". In: *Neural Computation* 9, pp. 1735–1780 (cit. on p. 12).

Hori, T. and Nakamura, A. (2013). "Speech recognition algorithms using weighted finite-state transducers". In: *Synthesis Lectures on Speech and Audio Processing* 9.1, pp. 1–162 (cit. on p. 24).

Hori, T., Willett, D., and Minami, Y. (2003). "Paraphrasing spontaneous speech using weighted finite-state transducers". In: *Proc. SSPR* (cit. on p. 49).

Hoshen, Y., Weiss, R. J., and Wilson, K. W. (2015). "Speech acoustic modeling from raw multichannel waveforms". In: *Proc. ICASSP*, pp. 4624–4628 (cit. on pp. 12, 15, 92).

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017). "Snapshot ensembles: Train 1, get m for free". In: *Proc. ICLR* (cit. on pp. 76, 77, 81, 84).

Huang, J.-T. and Hasegawa-Johnson, M. (2010). "Semi-supervised training of gaussian mixture models by conditional entropy minimization". In: *Proc. Interspeech*, pp. 1353–1356 (cit. on pp. 22, 27).

Ioffe, S. and Szegedy, C. (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proc. ICML*. Ed. by F. Bach and D. Blei. Vol. 37, pp. 448–456 (cit. on p. 98).

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. (2018). "Averaging Weights Leads to Wider Optima and Better Generalization". In: *Proc. UAI*, pp. 876–885 (cit. on pp. 76, 78, 81, 83, 85, 91).

Jaitly, N. and Hinton, G. E. (2013). "Vocal tract length perturbation (VTLP) improves speech recognition". In: *Proc. ICML*. Vol. 117 (cit. on pp. 109, 115).

Jaitly, N. and Hinton, G. (2011). "Learning a better representation of speech soundwaves using restricted Boltzmann machines". In: *Proc. ICASSP*. IEEE, pp. 5884–5887 (cit. on pp. 15, 92).

Kahn, J., Lee, A., and Hannun, A. (2020). "Self-training for end-to-end speech recognition". In: *Proc. ICASSP*. IEEE, pp. 7084–7088 (cit. on p. 112).

Kannan, A., Wu, Y., Nguyen, P., Sainath, T. N., Chen, Z., and Prabhavalkar, R. (2018). "An analysis of incorporating an external language model into a sequence-to-sequence model". In: *Proc. ICASSP*. IEEE, pp. 5824–5828 (cit. on p. 111).

Kapadia, S., Valtchev, V., and Young, S. J. (1993). "MMI training for continuous phoneme recognition on the TIMIT database". In: *Proc. ICASSP*. Vol. 2. IEEE, pp. 491–494 (cit. on p. 16).

Karanasou, P., Wang, Y., Gales, M. J., and Woodland, P. C. (2014). "Adaptation of deep neural network acoustic models using factorised i-vectors". In: *Proc. Interspeech*, pp. 2180–2184 (cit. on pp. 2, 61–63).

Karita, S., Watanabe, S., Iwata, T., Ogawa, A., and Delcroix, M. (2018). "Semi-Supervised End-to-End Speech Recognition". In: *Proc. Interspeech*, pp. 2–6 (cit. on p. 111).

Kemp, T. and Schaaf, T. (1997). "Estimating confidence using word lattices". In: *Proc. Eurospeech* (cit. on p. 79).

Khokhlov, Y., Zatvornitskiy, A., Medennikov, I., Sorokin, I., Prisyach, T., Romanenko, A., Mitrofanov, A., Bataev, V., Andrusenko, A., Korenevskaya, M., et al. (2019). "R-vectors: New technique for adaptation to room acoustics". In: *Proc. Interspeech*, pp. 1243–1247 (cit. on p. 61).

Kim, D., Umesh, S, Gales, M., Hain, T, and Woodland, P. (2004). "Using VTLN for broadcast news transcription". In: *Proc. Interspeech*, pp. 1953–1956 (cit. on p. 20).

Kincaid, J. (2018). *The State of Automatic Speech Recognition: Q&A with Kaldi's Dan Povey*. URL: `https://medium.com/descript/the-state-of-automatic-speech-recognition-q-a-with-kaldis-dan-povey-c860aada9b85` (visited on 01/11/2020) (cit. on p. 7).

Kingma, D. P. and Ba, J. (2015). "Adam: A Method for Stochastic Optimization". In: *Proc. ICLR* (cit. on pp. 98, 116).

Kingsbury, B. (2009). "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling". In: *Proc. ICASSP*. IEEE, pp. 3761–3764 (cit. on p. 17).

Kleene, S. C. (1951). *Representation of events in nerve nets and finite automata*. Tech. rep. Rand Project Air Force Santa Monica CA (cit. on p. 25).

Klejch, O., Fainberg, J., and Bell, P. (2018). "Learning to Adapt: A Meta-learning Approach for Speaker Adaptation". In: *Proc. Interspeech*, pp. 867–871 (cit. on pp. 105, 108, 115).

Klejch, O., Fainberg, J., Bell, P., and Renals, S. (2019). "Lattice-based unsupervised test-time adaptation of neural network acoustic models". In: *arXiv preprint arXiv:1906.11521* (cit. on pp. 6, 18, 26, 27, 29, 30, 44, 53).

Ko, T., Peddinti, V., Povey, D., and Khudanpur, S. (2015). "Audio augmentation for speech recognition". In: *Proc. Interspeech*, pp. 3586–3589 (cit. on pp. 34, 35, 37, 52, 99).

Lamel, L., Gauvain, J.-L., and Adda, G. (2002). "Lightly supervised and unsupervised acoustic model training". In: *Computer Speech & Language* 16.1, pp. 115–129 (cit. on pp. 2, 33, 42).

Lanchantin, P., Gales, M. J., Karanasou, P., Liu, X, Qian, Y, Wang, L, Woodland, P. C., and Zhang, C (2016). "Selection of Multi-Genre Broadcast Data for the Training of Automatic Speech Recognition Systems." In: *Proc. Interspeech*, pp. 3057–3061 (cit. on pp. 2, 42).

Lee, L. and Rose, R. C. (1996). "Speaker normalization using efficient frequency warping procedures". In: *Proc. ICASSP*. IEEE, pp. 353–356 (cit. on pp. 20, 92, 95).

Lee, S., Potamianos, A., and Narayanan, S. (1999). "Acoustics of children's speech: Developmental changes of temporal and spectral parameters". In: *The Journal of the Acoustical Society of America* 105.3, pp. 1455–1468 (cit. on p. 1).

Leggetter, C. J. and Woodland, P. C. (1995). "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models". In: *Computer Speech & Language* 9.2, pp. 171–185 (cit. on p. 18).

Levinson, S. E. (1986). "Continuously variable duration hidden Markov models for automatic speech recognition". In: *Computer Speech & Language* 1.1, pp. 29–45 (cit. on p. 9).

Li, B., Sainath, T. N., Pang, R., and Wu, Z. (2019). "Semi-supervised training for End-to-End models via weak distillation". In: *Proc. ICASSP*. IEEE, pp. 2837–2841 (cit. on p. 111).

Li, B. and Sim, K. C. (2010). "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems". In: *Proc. Interspeech*, pp. 526–529 (cit. on p. 18).

Li, J., Huang, J. T., and Gong, Y. (2014). "Factorized adaptation for deep neural network". In: *Proc. ICASSP*. IEEE, pp. 5537–5541 (cit. on p. 62).

Li, M., Wang, R., and Tang, K. (2013). "Combining semi-supervised and active learning for hyperspectral image classification". In: *Proc. CIDM*, pp. 89–94 (cit. on p. 76).

Li, S., Akita, Y., and Kawahara, T. (2015). "Automatic Lecture Transcription Based on Discriminative Data Selection for Lightly Supervised Acoustic Model Training". In: *IEICE Transactions on Information and Systems* 8, pp. 1545–1552 (cit. on p. 42).

Li, Z. and Hoiem, D. (2017). "Learning without forgetting". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.12, pp. 2935–2947 (cit. on p. 74).

Liao, H., McDermott, E., and Senior, A. (2013). "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription". In: *Proc. ASRU*. IEEE, pp. 368–373 (cit. on pp. 41, 42).

Liu, X., Gales, M. J., and Woodland, P. C. (2014). "Paraphrastic language models". In: *Computer Speech & Language* 28.6, pp. 1298–1316 (cit. on pp. 51, 113).

Liu, Y., Karanasou, P., and Hain, T. (2015). "An investigation into speaker informed DNN front-end for LVCSR". In: *Proc. ICASSP*. IEEE, pp. 4300–4304 (cit. on p. 20).

Liu, Y., Zhang, P., and Hain, T. (2014). "Using neural network front-ends on far field multiple microphones based speech recognition". In: *Proc. ICASSP*. IEEE, pp. 5542–5546 (cit. on p. 19).

Long, Y., Gales, M. J., Lanchantin, P. K., Liu, X., Seigel, M. S., and Woodland, P. C. (2013). "Improving lightly supervised training for broadcast transcription". In: *Proc. Interspeech*, pp. 2187–2191 (cit. on pp. 33, 41–45).

Long, Y., Ye, H., Li, Y., and Liang, J. (2018). "Active Learning for LF-MMI Trained Neural Networks in ASR". In: *Proc. Interspeech*, pp. 2898–2902 (cit. on p. 76).

Loshchilov, I. and Hutter, F. (2017). "SGDR: Stochastic gradient descent with warm restarts". In: *Proc. ICLR* (cit. on pp. 76, 81).

Loweimi, E., Bell, P., and Renals, S. (2019). "On Learning Interpretable CNNs with Parametric Modulated Kernel-based Filters". In: *Proc. Interspeech*, pp. 3480–3484 (cit. on pp. 93, 97, 98).

Maaten, L. v. d. and Hinton, G. (2008). "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov, pp. 2579–2605 (cit. on pp. 69, 70).

Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. (2016). "Adversarial Autoencoders". In: *Proc. ICLR* (cit. on p. 113).

Mangu, L., Brill, E., and Stolcke, A. (2000). "Finding consensus in speech recognition: word error minimization and other applications of confusion networks". In: *Computer Speech & Language* 14, pp. 373–400 (cit. on pp. 22, 27, 43).

Manohar, V., Hadian, H., Povey, D., and Khudanpur, S. (2018). "Semi-supervised training of acoustic models using lattice-free MMI". In: *Proc. ICASSP*. IEEE, pp. 4844–4848 (cit. on pp. 2, 18, 22, 26, 27, 44, 53).

Manohar, V., Povey, D., and Khudanpur, S. (2015). "Semi-supervised maximum mutual information training of deep neural network acoustic models". In: *Proc. Interspeech*, pp. 2630–2634 (cit. on p. 27).

— (2017). "JHU Kaldi system for Arabic MGB-3 ASR challenge using diarization, audio-transcript alignment and transfer learning". In: *Proc. ASRU*. IEEE, pp. 346–352 (cit. on pp. 27, 43, 44).

Mathias, L., Yegnanarayanan, G., and Fritsch, J. (2005). "Discriminative training of acoustic models applied to domains with unreliable transcripts". In: *Proc. ICASSP*. IEEE, pp. I–109 (cit. on pp. 26, 41, 42, 44).

McCall, W. G. and Craig, C. (2009). "Same-Language-Subtitling (SLS): Using subtitled music video for reading growth". In: *EdMedia+ Innovate Learning*. Association for the Advancement of Computing in Education (AACE), pp. 3983–3992 (cit. on p. 2).

McCallumzy, A. K. and Nigamy, K. (1998). "Employing EM and pool-based active learning for text classification". In: *Proc. ICML*, pp. 359–367 (cit. on p. 76).

Meng, Z., Li, J., Gaur, Y., and Gong, Y. (2019). "Domain adaptation via teacher-student learning for end-to-end speech recognition". In: *Proc. ASRU*. IEEE, pp. 268–275 (cit. on p. 112).

Miao, Y., Gowayyed, M., and Metze, F. (2015). "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding". In: *Proc. ASRU*. IEEE, pp. 167–174 (cit. on p. 112).

Miao, Y., Zhang, H., and Metze, F. (2014). "Towards speaker adaptive training of deep neural network acoustic models". In: *Proc. Interspeech*, pp. 2189–2193 (cit. on p. 72).

Mohamed, A.-r., Hinton, G., and Penn, G. (2012). "Understanding how Deep Belief Networks perform acoustic modelling". In: *Proc. ICASSP*. IEEE, pp. 4273–4276 (cit. on p. 93).

Mohri, M. (2002). "Semiring frameworks and algorithms for shortest-distance problems". In: *Journal of Automata, Languages and Combinatorics* 7.3, pp. 321–350 (cit. on p. 23).

Mohri, M., Pereira, F., and Riley, M. (2008). "Speech recognition with weighted finite-state transducers". In: *Springer Handbook of Speech Processing*, pp. 559–584 (cit. on pp. 22, 24, 25).

Moreno, P. J. and Alberti, C. (2009). "A factor automaton approach for the forced alignment of long speech recordings". In: *Proc. ICASSP*. IEEE, pp. 4869–4872 (cit. on p. 42).

Nagrani, A., Chung, J. S., Albanie, S., and Zisserman, A. (2020). "Disentangled speech embeddings using cross-modal self-supervision". In: *Proc. ICASSP*. IEEE, pp. 6829–6833 (cit. on p. 113).

Nair, V. and Hinton, G. E. (2010). "Rectified linear units improve restricted Boltzmann machines". In: *Proc. ICML*, pp. 807–814 (cit. on p. 68).

Nallasamy, U., Metze, F., and Schultz, T. (2012). "Active learning for accent adaptation in automatic speech recognition". In: *Proc. SLT*, pp. 360–365 (cit. on p. 76).

National Institute of Standards and Technology (NIST) (2007). *Speech Recognition Scoring Toolkit (SCTK)* (cit. on p. 40).

Neto, J., Almeida, L., Hochberg, M., Martins, C., Nunes, L., Renals, S., and Robinson, T. (1995). "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system". In: *Eurospeech*, pp. 2171–2174 (cit. on pp. 18, 61).

Nguyen, L. and Xiang, B. (2004). "Light supervision in acoustic model training". In: *Proc. ICASSP*. IEEE, pp. I–185 (cit. on p. 42).

Nicolao, M., Sanders, M., and Hain, T. (2018). "Improved acoustic modelling for automatic literacy assessment of children". In: *Proc. Interspeech*, pp. 1666–1670 (cit. on pp. 41, 43).

Normandin, Y., Lacouture, R., and Cardin, R. (1994). "MMIE training for large vocabulary continuous speech recognition". In: *Proc. ICSLP*, pp. 1367–1370 (cit. on p. 17).

Ofcom (2006). *Television Access Services Review* (cit. on p. 2).

Olcoz, J., Saz, O., and Hain, T. (2016). "Error Correction in Lightly Supervised Alignment of Broadcast Subtitles". In: *Proc. Interspeech*, pp. 2110–2114 (cit. on p. 43).

OpenFST (2017). *OpenFst Examples*. URL: `http://www.openfst.org/twiki/bin/view/FST/FstExamples` (visited on 04/09/2020) (cit. on p. 47).

Oppenheim, A. V. and Schafer, R. W. (1975). "Digital Signal Processing". In: *Prentice-Hall*, pp. 26–30 (cit. on p. 95).

Palaz, D., Collobert, R., and Doss, M. M. (2013). "Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks". In: *Proc. Interspeech*, pp. 1766–1770 (cit. on pp. 15, 92).

Palaz, D., Magimai-Doss, M., and Collobert, R. (2015). "Analysis of CNN-based speech recognition system using raw speech as input". In: *Proc. Interspeech*, pp. 11–15 (cit. on p. 92).

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). "Librispeech: an ASR corpus based on public domain audio books". In: *Proc. ICASSP*. IEEE, pp. 5206–5210 (cit. on pp. 7, 37).

Panchapagesan, S. and Alwan, A. (2006). "Multi-parameter frequency warping for VTLN by gradient search". In: *Proc. ICASSP*. IEEE, pp. 1181–1184 (cit. on p. 96).

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). "BLEU: a method for automatic evaluation of machine translation". In: *Proc. ACL*, pp. 311–318 (cit. on p. 49).

Parihar, N., Picone, J., Pearce, D., and Hirsch, H.-G. (2004). "Performance analysis of the Aurora large vocabulary baseline system". In: *European Signal Processing Conference*, pp. 553–556 (cit. on pp. 31, 37, 63).

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). "Continual lifelong learning with neural networks: A review". In: *Neural Networks* 113, pp. 54–71 (cit. on p. 74).

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Proc. Interspeech*, pp. 2613–2617 (cit. on p. 7).

Patel, S. (2016). *85 percent of Facebook video is watched without sound*. URL: `https://digiday.com/media/silent-world-facebook-video` (visited on 01/24/2020) (cit. on p. 2).

Paul, D. B. and Baker, J. M. (1992). "The design for the Wall Street Journal-based CSR corpus". In: *Proceedings of the workshop on Speech and Natural Language*, pp. 357–362 (cit. on pp. 31, 37, 65, 122).

Peddinti, V., Chen, G., Povey, D., and Khudanpur, S. (2015). "Reverberation robust acoustic modeling using i-vectors with time delay neural networks". In: *Proc. Interspeech*, pp. 2440–2444 (cit. on p. 66).

Peddinti, V., Povey, D., and Khudanpur, S. (2015). "A time delay neural network architecture for efficient modeling of long temporal contexts". In: *Proc. Interspeech*, pp. 3214–3218 (cit. on p. 13).

Perego, E., Del Missier, F., Porta, M., and Mosconi, M. (2010). "The cognitive effectiveness of subtitle processing". In: *Media psychology* 13.3, pp. 243–272 (cit. on p. 2).

Pereira, F., Tishby, N., and Lee, L. (1993). "Distributional clustering of English words". In: *Proc. ACL*, pp. 183–190 (cit. on pp. 50, 113).

Pitz, M. and Ney, H. (2005). "Vocal tract normalization equals linear transformation in cepstral space". In: *IEEE Transactions on Speech and Audio Processing* 13.5, pp. 930–944 (cit. on p. 20).

Platen, P. von, Zhang, C., and Woodland, P. (2019). "Multi-span acoustic modelling using raw waveform signals". In: *Proc. Interspeech*, pp. 1393–1397 (cit. on p. 38).

Poritz, A. B. (1988). "Hidden Markov models: A guided tour". In: *Proc. ICASSP*, pp. 8–13 (cit. on p. 8).

Potamianos, A. and Narayanan, S. (2003). "Robust recognition of children's speech". In: *IEEE Transactions on Speech and Audio Processing* 11.6, pp. 603–616 (cit. on pp. 93, 102, 106).

Povey, D. et al. (2011). "The Kaldi speech recognition toolkit". In: *Proc. ASRU*. IEEE (cit. on pp. 10, 20, 21, 52, 79, 97, 98, 116).

Povey, D. (2005). "Discriminative training for large vocabulary speech recognition". PhD thesis. University of Cambridge (cit. on pp. 16, 17, 27).

Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohamadi, M., and Khudanpur, S. (2018). "Semi-orthogonal low-rank matrix factorization for deep neural networks". In: *Proc. Interspeech*, pp. 3743–3747 (cit. on pp. 7, 13, 14).

Povey, D., Hannemann, M., Boulianne, G., Burget, L., Ghoshal, A., Janda, M., Karafiát, M., Kombrink, S., Motlíček, P., Qian, Y., et al. (2012). "Generating exact lattices in the WFST framework". In: *Proc. ICASSP*. IEEE, pp. 4213–4216 (cit. on p. 26).

Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., and Visweswariah, K. (2008). "Boosted MMI for model and feature-space discriminative training". In: *Proc. ICASSP*. IEEE, pp. 4057–4060 (cit. on pp. 16, 17).

Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., Wang, Y., and Khudanpur, S. (2016). "Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI". In: *Proc. Interspeech*, pp. 2751–2755 (cit. on pp. 7, 9, 13, 17, 44).

Povey, D., Zhang, X., and Khudanpur, S. (2014). "Parallel training of DNNs with natural gradient and parameter averaging". In: *arXiv preprint arXiv:1410.7455* (cit. on pp. 12, 75, 78, 116).

Povey, D., Zweig, G., and Acero, A. (2011). "Speaker adaptation with an exponential transform". In: *Proc. ASRU*. IEEE, pp. 158–163 (cit. on p. 20).

Prabhavalkar, R., Alsharif, O., Bruguier, A., and McGraw, L. (2016). "On the compression of recurrent neural networks with an application to LVCSR acoustic modeling for embedded speech recognition". In: *Proc. ICASSP*. IEEE, pp. 5970–5974 (cit. on p. 13).

Prabhavalkar, R., Sainath, T. N., Wu, Y., Nguyen, P., Chen, Z., Chiu, C.-C., and Kannan, A. (2018). "Minimum word error rate training for attention-based sequence-to-sequence models". In: *Proc. ICASSP*. IEEE, pp. 4839–4843 (cit. on p. 112).

Raj, D., Snyder, D., Povey, D., and Khudanpur, S. (2019). "Probing the information encoded in x-vectors". In: *Proc. ASRU*. IEEE, pp. 726–733 (cit. on p. 69).

Ravanelli, M. and Bengio, Y. (2018). "Speaker recognition from raw waveform with SincNet". In: *Proc. SLT*. IEEE, pp. 1021–1028 (cit. on pp. 5, 15, 92–98, 105, 119).

Ravuri, S. V. and Wegmann, S. (2016). "How neural network depth compensates for HMM conditional independence assumptions in DNN-HMM acoustic models". In: *Proc. Interspeech*, pp. 2736–2740 (cit. on p. 11).

Renals, S. and Swietojanski, P. (2014). "Neural networks for distant speech recognition". In: *Proc. HSCMA*. IEEE, pp. 172–176 (cit. on p. 38).

Robins, A. (1993). "Catastrophic forgetting in neural networks: the role of rehearsal mechanisms". In: *Proc. of The First New Zealand International Two-Stream Con-*

ference on Artificial Neural Networks and Expert Systems. IEEE, pp. 65–68 (cit. on p. 74).

Robinson, A. J. (1994). "An application of recurrent nets to phone probability estimation". In: *IEEE Transactions on Neural Networks*, pp. 298–305 (cit. on p. 12).

Royal National Institute for Deaf People (2008). *Annual Survey Report 2008* (cit. on p. 2).

Russell, M. and Moore, R. (1985). "Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition". In: *Proc. ICASSP*. IEEE, pp. 5–8 (cit. on p. 9).

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). "Progressive neural networks". In: *arXiv preprint arXiv:1606.04671* (cit. on p. 74).

Sailor, H. B. and Patil, H. A. (2016). "Novel unsupervised auditory filterbank learning using convolutional RBM for speech recognition". In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 12, pp. 2341–2353 (cit. on pp. 93, 99).

Sainath, T. N., Kingsbury, B., Mohamed, A.-r., and Ramabhadran, B. (2013a). "Learning filter banks within a deep neural network framework". In: *Proc. ASRU*. IEEE, pp. 297–302 (cit. on pp. 15, 92, 93).

Sainath, T. N., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. (2013b). "Deep convolutional neural networks for LVCSR". In: *Proc. ICASSP*. IEEE, pp. 8614–8618 (cit. on p. 12).

Sainath, T. N., Weiss, R. J., Senior, A., Wilson, K. W., and Vinyals, O. (2015). "Learning the speech front-end with raw waveform CLDNNs". In: *Proc. Interspeech*, pp. 1–5 (cit. on pp. 15, 92, 93, 99).

Sak, H., Senior, A., and Beaufays, F. (2014). "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: *Proc. Interspeech*, pp. 338–342 (cit. on p. 12).

Sak, H., Senior, A., Rao, K., and Beaufays, F. (2015a). "Fast and accurate recurrent neural network acoustic models for speech recognition". In: *Proc. Interspeech*, pp. 1468–1472 (cit. on p. 9).

Sak, H., Senior, A., Rao, K., Irsoy, O., Graves, A., Beaufays, F., and Schalkwyk, J. (2015b). "Learning acoustic frame labeling for speech recognition with recurrent neural networks". In: *Proc. ICASSP*. IEEE, pp. 4280–4284 (cit. on p. 112).

Salakhutdinov, R. and Hinton, G. (2009). "Deep Boltzmann machines". In: *Artificial Intelligence and Statistics*, pp. 448–455 (cit. on p. 39).

Samarakoon, L. and Sim, K. C. (2015). "Learning factorized feature transforms for speaker normalization". In: *Proc. ASRU*. IEEE, pp. 145–152 (cit. on p. 19).

— (2016). "Subspace LHUC for fast adaptation of deep neural network acoustic models". In: *Proc. Interspeech*, pp. 1593–1597 (cit. on p. 19).

Saon, G., Soltau, H., Nahamoo, D., and Picheny, M. (2013). "Speaker adaptation of neural network acoustic models using i-vectors". In: *Proc. ASRU*. IEEE, pp. 55–59 (cit. on pp. 19, 61).

Saon, G., Kuo, H.-K. J., Rennie, S., and Picheny, M. (2015). "The IBM 2015 English conversational telephone speech recognition system". In: *Proc. Interspeech*, pp. 3140–3144 (cit. on p. 12).

Saon, G., Soltau, H., Emami, A., and Picheny, M. (2014). "Unfolded recurrent neural networks for speech recognition". In: *Proc. Interspeech*, pp. 343–347 (cit. on p. 12).

Saon, G. et al. (2017). "English conversational telephone speech recognition by humans and machines". In: *Proc. Interspeech*, pp. 132–136 (cit. on p. 12).

Saz, O. and Hain, T. (2017). "Acoustic adaptation to dynamic background conditions with asynchronous transformations". In: *Computer Speech & Language* 41, pp. 180–194 (cit. on pp. 2, 61).

Schachter, R (1998). *Bayes-ball: The rational pasttime (for determining irrelevance and requisite information in belief networks and influence diagrams)* (cit. on p. 8).

Schluter, R. and Macherey, W. (1998). "Comparison of discriminative training criteria". In: *Proc. ICASSP*. IEEE, pp. 493–496 (cit. on p. 16).

Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). "wav2vec: Unsupervised pre-training for speech recognition". In: *Proc. Interspeech*, pp. 3465–3469 (cit. on p. 115).

Seki, H., Yamamoto, K., Akiba, T., and Nakagawa, S. (2018). "Rapid speaker adaptation of neural network based filterbank layer for automatic speech recognition". In: *Proc. SLT*. IEEE, pp. 574–580 (cit. on pp. 93, 96, 100).

Seki, H., Yamamoto, K., and Nakagawa, S. (2017). "A deep neural network integrated with filterbank learning for speech recognition". In: *Proc. ICASSP*. IEEE, pp. 5480–5484 (cit. on pp. 93, 95).

Seltzer, M. L. and Acero, A. (2011). "Separating speaker and environmental variability using factored transforms". In: *Proc. Interspeech*, pp. 1097–1100 (cit. on pp. 61–63).

Senior, A., Heigold, G., Bacchiani, M., and Liao, H. (2014). "GMM-free DNN acoustic model training". In: *Proc. ICASSP*. IEEE, pp. 5602–5606 (cit. on p. 12).

Seo, H., Kang, H.-G., and Seltzer, M. L. (2014). "Factored adaptation of speaker and environment using orthogonal subspace transforms". In: *Proc. ICASSP*. IEEE, pp. 3251–3255 (cit. on pp. 61, 63).

Sheikhzadeh, H. and Deng, L. (1994). "Waveform-based speech recognition using hidden filter models: Parameter selection and sensitivity to power normalization". In: *IEEE Transactions on Speech and Audio Processing* 2.1, pp. 80–89 (cit. on p. 15).

Shinohara, Y. (2016). "Adversarial Multi-Task Learning of Deep Neural Networks for Robust Speech Recognition". In: *Proc. Interspeech*, pp. 2369–2372 (cit. on pp. 72, 113).

Shivakumar, P. G., Potamianos, A., Lee, S., and Narayanan, S. (2014). "Improving speech recognition for children using acoustic adaptation and pronunciation modeling". In: *WOCCI*, pp. 15–19 (cit. on p. 1).

Sim, K. C., Narayanan, A., Misra, A., Tripathi, A., Pundak, G., Sainath, T. N., Haghani, P., Li, B., and Bacchiani, M. (2018). "Domain adaptation using factorized hidden layer for robust automatic speech recognition". In: *Proc. Interspeech*, pp. 892–896 (cit. on p. 2).

Smith, L. N. (2017). "Cyclical learning rates for training neural networks". In: *Proc. WACV*. IEEE, pp. 464–472 (cit. on p. 76).

Snyder, D., Garcia-Romero, D., Povey, D., and Khudanpur, S. (2017). "Deep Neural Network Embeddings for Text-Independent Speaker Verification". In: *Proc. Interspeech*, pp. 999–1003 (cit. on p. 72).

Soltau, H., Liao, H., and Sak, H. (2016). "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition". In: *Proc. Interspeech*, pp. 3707–3711 (cit. on p. 112).

Stan, A., Bell, P., and King, S. (2012). "A grapheme-based method for automatic alignment of speech and text data". In: *Proc. SLT*. IEEE, pp. 286–290 (cit. on pp. 42, 43).

Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). "A scale for the measurement of the psychological magnitude pitch". In: *The Journal of the Acoustical Society of America*, pp. 185–190 (cit. on p. 15).

Stolcke, A. (2002). "SRILM-an extensible language modeling toolkit". In: *Proc. ICSLP*, pp. 901–904 (cit. on p. 99).

Sugiyama, M. and Kawanabe, M. (2012). *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press (cit. on pp. 76, 79).

Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., and Fergus, R. (2015). "Training convolutional networks with noisy labels". In: *Proc. ICLR* (cit. on p. 44).

Swietojanski, P., Li, J., and Renals, S. (2016). "Learning hidden unit contributions for unsupervised acoustic model adaptation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.8, pp. 1450–1463 (cit. on pp. 19, 61–63, 72, 93, 97, 105, 107).

Swietojanski, P. and Renals, S. (2014). "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models". In: *Proc. SLT*. IEEE, pp. 171–176 (cit. on pp. 18, 93).

Takeda, R., Nakadai, K., and Komatani, K. (2018). "Multi-timescale Feature-extraction Architecture of Deep Neural Networks for Acoustic Model Training from Raw Speech Signal". In: *Proc. IROS*. IEEE/RSJ, pp. 2503–2510 (cit. on p. 92).

Thiemann, J., Ito, N., and Vincent, E. (2013). "The diverse environments multi-channel acoustic noise database: A database of multichannel environmental noise recordings".

In: *The Journal of the Acoustical Society of America* 5, pp. 3591–3591 (cit. on pp. 31, 63, 65).

Tieleman, T. and Hinton, G. (2012). "Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* (cit. on p. 68).

Toshniwal, S., Kannan, A., Chiu, C.-C., Wu, Y., Sainath, T. N., and Livescu, K. (2018). "A comparison of techniques for language model integration in encoder-decoder speech recognition". In: *Proc. SLT*. IEEE, pp. 369–375 (cit. on p. 111).

Tur, G., Hakkani-Tür, D., and Schapire, R. E. (2005). "Combining active and semi-supervised learning for spoken language understanding". In: *Speech Communication* 45.2, pp. 171 –186 (cit. on p. 76).

Tüske, Z., Golik, P., Schlüter, R., and Ney, H. (2014). "Acoustic modeling with deep neural networks using raw time signal for LVCSR". In: *Proc. Interspeech*, pp. 890–894 (cit. on p. 92).

Uebel, L. F. and Woodland, P. C. (1999). "An investigation into vocal tract length normalisation". In: *Proc. Eurospeech*, pp. 2527–2530 (cit. on p. 20).

Venkataraman, A., Stolcke, A., Wang, W., Vergyri, D., Zheng, J., and Gadde, V. R. R. (2004). "An efficient repair procedure for quick transcriptions". In: *Proc. Interspeech*, pp. 1961–1964 (cit. on p. 43).

Veselý, K., Ghoshal, A., Burget, L., and Povey, D. (2013). "Sequence-discriminative training of deep neural networks". In: *Proc. Interspeech*, pp. 2345–2349 (cit. on pp. 13, 17, 28).

Vincent, E., Barker, J., Watanabe, S., Le Roux, J., Nesta, F., and Matassoni, M. (2013). "The second 'CHiME' speech separation and recognition challenge: Datasets, tasks and baselines". In: *Proc. ICASSP*. IEEE, pp. 126–130 (cit. on p. 1).

Vincent, E., Watanabe, S., Nugraha, A. A., Barker, J., and Marxer, R. (2017). "An analysis of environment, microphone and data simulation mismatches in robust speech recognition". In: *Computer Speech & Language* 46, pp. 535–557 (cit. on p. 1).

Vinyals, O., Ravuri, S. V., and Povey, D. (2012). "Revisiting recurrent neural networks for robust ASR". In: *Proc. ICASSP*. IEEE, pp. 4085–4088 (cit. on p. 12).

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). "Phoneme recognition using time-delay neural networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3, pp. 328–339 (cit. on pp. 11, 13).

Wang, G. and Sim, K. C. (2011). "Sequential classification criteria for NNs in automatic speech recognition". In: *Proc. Interspeech*, pp. 441–444 (cit. on pp. 13, 17).

Wang, S., Qian, Y., and Yu, K. (2017). "What does the speaker embedding encode?" In: *Proc. Interspeech*, pp. 1497–1501 (cit. on p. 69).

Wang, Y. Q. and Gales, M. J. F. (2013). "An explicit independence constraint for factorised adaptation in speech recognition". In: *Proc. Interspeech*, pp. 1233–1237 (cit. on pp. 62, 63).

Wang, Y., Mohamed, A., Le, D., Liu, C., Xiao, A., Mahadeokar, J., Huang, H., Tjandra, A., Zhang, X., Zhang, F., et al. (2020). "Transformer-based acoustic modeling for hybrid speech recognition". In: *Proc. ICASSP*. IEEE, pp. 6874–6878 (cit. on pp. 7, 12).

Wessel, F., Macherey, K., and Schluter, R. (1998). "Using word probabilities as confidence measures". In: *Proc. ICASSP*. IEEE, pp. 225–228 (cit. on p. 79).

Williams, I., Kannan, A., Aleksic, P. S., Rybach, D., and Sainath, T. N. (2018). "Contextual speech recognition in end-to-end neural network systems using beam search". In: *Proc. Interspeech*, pp. 2227–2231 (cit. on p. 112).

Woodland, P. C., Liu, X., Qian, Y., Zhang, C., Gales, M. J., Karanasou, P., Lanchantin, P., and Wang, L. (2015). "Cambridge University transcription systems for the Multi-Genre Broadcast challenge". In: *Proc. ASRU*. IEEE, pp. 639–646 (cit. on pp. 2, 12, 34, 75, 77).

Woodland, P. C. and Povey, D. (2002). "Large scale discriminative training of hidden Markov models for speech recognition". In: *Computer Speech & Language* 16.1, pp. 25–47 (cit. on p. 16).

Xie, X., Liu, X., Lee, T., Hu, S., and Wang, L. (2019). "BLHUC: Bayesian learning of hidden unit contributions for deep neural network speaker adaptation". In: *Proc. ICASSP*. IEEE, pp. 5711–5715 (cit. on p. 19).

Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., and Stolcke, A. (2017). "The Microsoft 2017 conversational speech recognition system". In: *Proc. ICASSP*. IEEE, pp. 5934–5938 (cit. on p. 12).

Xu, H., Povey, D., Mangu, L., and Zhu, J. (2011). "Minimum Bayes Risk decoding and system combination based on a recursion for edit distance". In: *Computer Speech & Language* 4, pp. 802–828 (cit. on pp. 26, 34, 79).

Xue, J., Li, J., and Gong, Y. (2013). "Restructuring of deep neural network acoustic models with singular value decomposition". In: *Proc. Interspeech*, pp. 2365–2369 (cit. on p. 13).

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). "Lifelong learning with dynamically expandable networks". In: *Proc. ICLR* (cit. on p. 74).

Young, S. J., Russell, N., and Thornton, J. (1989). *Token passing: a simple conceptual model for connected speech recognition systems*. Cambridge University Engineering Department (cit. on p. 26).

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (2002). "The HTK book". In: *Cambridge university engineering department* (cit. on p. 26).

Yu, D. and Deng, L. (2016). *Automatic Speech Recognition*. Springer (cit. on pp. 16, 28).

Yu, D., Xiong, W., Droppo, J., Stolcke, A., Ye, G., Li, J., and Zweig, G. (2016). "Deep convolutional neural networks with layer-wise context expansion and attention". In: *Proc. Interspeech*, pp. 17–21 (cit. on p. 12).

Yu, H. and Waibel, A. (2000). "Streamlining the front end of a speech recognizer". In: *Proc. ICSLP*, pp. 353–356 (cit. on p. 12).

Yu, K., Gales, M., Wang, L., and Woodland, P. C. (2010). "Unsupervised training and directed manual transcription for LVCSR". In: *Speech Communication* 52.7, pp. 652–663 (cit. on pp. 27, 44).

Zapotoczny, M., Pietrzak, P., Łancucki, A., and Chorowski, J. (2019). "Lattice generation in attention-based speech recognition models". In: *Proc. Interspeech*, pp. 2225–2229 (cit. on p. 112).

Zavaliagkos, G., Siu, M.-H., Colthurst, T., and Billa, J. (1998). "Using untranscribed training data to improve performance". In: *Proc. ICSLP* (cit. on p. 2).

Zhang, C. and Woodland, P. C. (2014). "Standalone training of context-dependent deep neural network acoustic models". In: *Proc. ICASSP*. IEEE, pp. 5597–5601 (cit. on p. 12).

Zhang, X., Trmal, J., Povey, D., and Khudanpur, S. (2014a). "Improving deep neural network acoustic models using generalized maxout networks". In: *Proc. ICASSP*. IEEE, pp. 215–219 (cit. on p. 67).

Zhang, Y., Chen, G., Yu, D., Yaco, K., Khudanpur, S., and Glass, J. (2016). "Highway long short-term memory RNNs for distant speech recognition". In: *Proc. ICASSP*, pp. 5755–5759 (cit. on p. 12).

Zhang, Z., Coutinho, E., Deng, J., and Schuller, B. (2014b). "Cooperative learning and its application to emotion recognition from speech". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.1, pp. 115–126 (cit. on p. 76).

Zimmeck, S., Wang, Z., Zou, L., Iyengar, R., Liu, B., Schaub, F., Wilson, S., Sadeh, N., Bellovin, S., and Reidenberg, J. (2016). "Automated analysis of privacy requirements for mobile apps". In: *AAAI Fall Symposium Series* (cit. on pp. 3, 74).

Zwicker, E. (1961). "Subdivision of the audible frequency range into critical bands (Frequenzgruppen)". In: *The Journal of the Acoustical Society of America* 33.2, pp. 248–248 (cit. on p. 15).

van Dalen, R. C., Knill, K. M., Tsiakoulis, P., and Gales, M. J. (2015). "Improving multiple-crowd-sourced transcriptions using a speech recogniser". In: *Proc. ICASSP*. IEEE, pp. 4709–4713 (cit. on p. 43).