View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Edinburgh Research Archive

THE UNIVERSITY *of* EDINBURGH

This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e.g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.

This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

Learning to Adapt: Meta-Learning Approaches for Speaker Adaptation

Ondřej Klejch

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

Abstract

The performance of automatic speech recognition systems degrades rapidly when there is a mismatch between training and testing conditions. One way to compensate for this mismatch is to adapt an acoustic model to test conditions, for example by performing speaker adaptation. In this thesis we focus on the discriminative model-based speaker adaptation approach. The success of this approach relies on having a robust speaker adaptation procedure – we need to specify which parameters should be adapted and how they should be adapted. Unfortunately, tuning the speaker adaptation procedure requires considerable manual effort.

In this thesis we propose to formulate speaker adaptation as a meta-learning task. In meta-learning, learning occurs on two levels: a learner learns a task specific model and a meta-learner learns how to train these task specific models. In our case, the learner is a speaker dependent-model and the meta-learner learns to adapt a speaker-independent model into the speaker dependent model. By using this formulation, we can automatically learn robust speaker adaptation procedures using gradient descent. In the experiments, we demonstrate that the meta-learning approach learns competitive adaptation schedules compared to adaptation procedures with handcrafted hyperparameters.

Subsequently, we show that speaker adaptive training can be formulated as a meta-learning task as well. In contrast to the traditional approach, which maintains and optimises a copy of speaker dependent parameters for each speaker during training, we embed the gradient based adaptation directly into the training of the acoustic model. We hypothesise that this formulation should steer the training of the acoustic model into finding parameters better suited for test-time speaker adaptation. We experimentally compare our approach with test-only adaptation of a standard baseline model and with SAT-LHUC, which represents a traditional speaker adaptive training method. We show that the meta-learning speaker-adaptive training approach achieves comparable results with SAT-LHUC. However, neither the meta-learning approach nor SAT-LHUC outperforms the baseline approach after adaptation.

Consequently, we run a series of experimental ablations to determine why SAT-LHUC does not yield any improvements compared to the baseline approach. In these experiments we explored multiple factors such as using various neural network architectures, normalisation techniques, activation functions or optimisers. We find that SAT-LHUC interferes with batch normalisation, and that it benefits from an increased hidden layer width and an increased model size. However, the baseline model benefits from increased capacity too, therefore in order to obtain the best model it is still

favourable to train a speaker independent model with batch normalisation. As such, an effective way of training state-of-the-art SAT-LHUC models remains an open question.

Finally, we show that the performance of unsupervised speaker adaptation can be further improved by using discriminative adaptation with lattices as supervision obtained from a first pass decoding, instead of traditionally used one-best path transcriptions. We find that this proposed approach enables many more parameters to be adapted without overfitting being observed, and is successful even when the initial transcription has a WER in excess of 50%.

Lay Summary

Automatic speech recognition (ASR) systems have become widely used in recent years. They are used in many areas of daily life in applications like dictation, smart assistants, subtitling and many others. The adoption of ASR systems was enabled by recent improvements to their accuracy, which were obtained through improved training methods that allow researchers to leverage vast amounts of training data efficiently. However, the accuracy of ASR systems degrades rapidly when they are used in conditions that differ from the training conditions, for example when transcribing child speech or non-native speech with an ASR system trained on native adult speech. One way to compensate for this mismatch is to adapt ASR systems to the new unseen conditions, for example by adapting to unseen speaker characteristics. In this thesis we focus on making speaker adaptation reliable by carefully tuning its parameters. In the past, the tuning of speaker adaptation parameters required considerable manual effort. Here we propose to use an approach called learning to learn, also known as meta-learning, which allows us to automatically learn these speaker adaptation parameters with well studied optimisation techniques. We experimentally show that with the learning to adapt approach we can automatically learn speaker adaptation parameters that achieve superior performance to the manually tuned parameters. Furthermore, we study how the learning to adapt approach can be leveraged to train ASR models that are better suited for rapid adaptation to new speakers. We also explore ways of making speaker adaptation more robust in situations, where we only have access to a recording without any transcript.

Acknowledgements

First, I would like to thank my supervisors Steve Renals and Peter Bell for their support and advice during my studies. I am especially grateful to them for encouraging me to find my own research direction and for guiding me on the way. I want to thank Korin Richmond for his insightful comments on my research plans during yearly reviews. I would also like to thank my thesis examiners Sharon Goldwater and Khe Chai Sim for reading this thesis and for their feedback.

I am grateful to all my colleagues from CSTR for creating such a friendly working environment and for being my role-models. I would also like to thank my fellow PhD students, Joachim Fainberg, Joanna Rownicka, Sameer Bansel, Mihai Sorin Dobre, Craig Innes, Chau Luu, Abhirub Ghosh and Ben Krause for all the thought-provoking discussions we had. I am especially grateful to Joachim Fainberg for our daily apple breaks.

I also want to thank Pararth Shah, Larry Heck, Sourish Chaudhury and Joseph Roth for hosting me during my two internships at Google and for showing me how research ideas can be transformed into great products.

I would like to thank friends from the Edinburgh University Ballroom Dancing Society, especially Hilary and Tibor, for making Edinburgh my second home away from home.

I am grateful to my parents and the whole family for their support and encouragement during my studies.

Most of all, I want to thank my wife Martina who showed me the beauty of research and encouraged me to pursue PhD studies at the University of Edinburgh.

This work was partially supported by the EU H2020 projects SUMMA (grant agreement 688139) and ELG (grant agreement 825627), and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Air Force Research Laboratory (AFRL) contract #FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, AFRL or the U.S. Government. The U.S. Government is authorised to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

Table of Contents

Chapter 1

Introduction

In the past few years, the accuracy of automatic speech recognition (ASR) systems has improved dramatically. This progress was enabled mainly by access to larger training datasets, better techniques for training neural networks and sufficient computational power to train them on large training datasets. This improvement, together with the increased popularity of smart devices, caused an adoption of ASR in applications like dictation, smart assistants, subtitling and many others. However, there are still many problems that prevent ASR to become even more widely used. One of them is that the performance of ASR systems degrades rapidly when there is a mismatch between training and testing conditions. One method to alleviate this problem is to adapt the ASR system to the test-time conditions, for example to unseen speakers by performing speaker adaptation of an acoustic model.

In the past there has been a lot of research in speaker adaptation of acoustic models. Speaker adaptation techniques can be divided into three approaches. The first, the feature-space approach, performs speaker adaptation by estimating a transformation of input acoustic features that maximises the log-likelihood of the adaptation data. The second, the model-based approach, discriminatively adapts the neural network acoustic model parameters based on the adaptation data. In the third, the auxiliary feature-based approach, the acoustic model is provided with auxiliary features that inform it about speaker characteristics, which enables the acoustic model to better model speaker variability. We provide a comprehensive review of speaker adaptation methods in Chapter 3.

In this thesis we focus on discriminative model-based speaker adaptation which works as follows. To perform speaker adaptation we need some adaptation data, which consists of input acoustic features and corresponding labels. The acoustic model makes

predictions based on the input acoustic features. These predictions are then compared with provided labels in order to compute a loss value, which measures how good the predictions are. Subsequently, gradients of the loss are computed with respect to the parameters of the acoustic model. Finally, these parameters are updated in the opposite direction of the gradients to minimise the loss value with a predefined step size, which we call learning rate. This process is usually repeated for several steps to obtain the best results.

The biggest challenge of model-based speaker adaptation applied to neural network based acoustic models is that their modelling capacity makes adaptation prone to overfitting. This overfitting can happen in several ways. Firstly, if the amount of adaptation data is limited, the acoustic model is adapted only on examples for a few output classes, which might lead to catastrophic forgetting of unseen classes. Secondly, if the adaptation data is limited it might not be fully representative of the speaker characteristics and the acoustic model might overfit to a certain subset of speaker characteristics present in the adaptation data. Thirdly, the provided labels for the adaptation data might be erroneous, therefore it is important not to overfit to those errors.

To prevent the aforementioned overfitting issues, the adaptation procedure needs to be carefully tuned. First, we need to select which parameters of the acoustic model should be treated as speaker dependent. By limiting the numbers of speaker dependent parameters we can prevent overfitting, but on the other hand we are also limiting the expressivity of the adaptation function which might result in sub-optimal solutions. Second, we need to find a reliable learning rate schedule that will not overfit to the adaptation data, but at the same time maximise the performance of the adapted model. Third, we need to find a suitable loss function, for example by using a linear combination of multiple losses. Finally, since we usually apply the update rule several times, we need to determine an appropriate number of adaptation steps. In the past these hyperparameters of the adaptation procedure were tuned manually, which required considerable effort.

The main contribution of this thesis is that we show that speaker adaptation hyperparameters can be optimised automatically by formulating speaker adaptation as a meta-learning task. In meta-learning, learning is happening on two levels. In the context of speaker adaptation, a learner is learning a speaker-dependent acoustic model and a meta-learner is learning to adapt a speaker-independent model into a speaker dependent model. When both the learner and the meta-learner are differentiable, it is possible to train them with gradient descent methods, which allows the automatic

tuning of speaker adaptation hyperparameters. We describe meta-learning approaches in Chapter 4 and in Chapter 5 we experimentally show that they achieve competitive results compared to popular speaker adaptation techniques .

Furthermore, we show that speaker adaptive training (SAT) can be also formulated as a meta-learning task (Section 4.3). Traditionally, speaker adaptive training is used to remove speaker variance from the data such that the canonical model can focus only on modelling the phonological variability. In neural network based models this can be achieved by maintaining a copy of speaker dependent parameters, which remove the speaker variance, for each speaker. However, this approach does not allow speaker adaptive training of the whole acoustic model. Due to memory constraints, it is not possible to maintain a copy of all the parameters for each speaker and we also usually do not have enough data to train the whole model in a speaker adaptive way. Our formulation of speaker adaptive training differs. Instead of removing the speaker variance from the data, we embed gradient based speaker adaptation directly into the training of the acoustic model. We hypothesise that this should steer the training process to find parameters suitable for test-time speaker adaptation. Also, it allows speaker adaptive training of the whole model. Unfortunately, in our experiments in Chapter 6 we find that model-based speaker adaptive training, both traditional and meta-learning based, does not yield any improvements compared to the baseline acoustic models. Consequently, we analyse why model-based speaker adaptive training does not work with the current state-of-the-art models in Chapter 7. In particular, we analyse it by altering various settings, such as the neural network architecture, hidden layer width, model size, normalisation techniques or optimisers. We find that changing these parameters did not improve the performance of the adapted SAT models compared to the adapted baseline acoustic model.

Finally in Chapter 8, we address an issue common to unsupervised speaker adaptation, where some of the labels provided with the adaptation data might be erroneous. We show that the issue with erroneous labels can be mitigated by using lattices, which encode uncertainty present in the first pass decoding, instead of one best paths for computation of the loss using the Lattice-free MMI framework.

1.1 Declaration of Content

The thesis consists of research from the following publications:

- Klejch, O., Fainberg, J., and Bell, P. (2018). Learning to adapt: a meta-learning approach for speaker adaptation. In *Interspeech*.
- Klejch, O., Fainberg, J., Bell, P., and Renals, S. (2019b). Speaker adaptive training using model agnostic meta-learning. In *ASRU*.
- Klejch, O., Fainberg, J., Bell, P., and Renals, S. (2019a). Lattice-based unsupervised test-time adaptation of neural network acoustic models. *arXiv preprint arXiv:1906.11521*.

Furthermore, during my PhD studies I also worked extensively on punctuation prediction of ASR transcripts. We decided not to include this work in this thesis in order to make the thesis more coherent. This work was published in the following publications:

- Klejch, O., Bell, P., and Renals, S. (2016). Punctuated transcription of multi-genre broadcasts using acoustic and lexical approaches. In *SLT*.
- Klejch, O., Bell, P., and Renals, S. (2017). Sequence-to-sequence models for punctuated transcription combining lexical and acoustic features. In *ICASSP*.

I was also involved in several projects within and without the university which resulted in the following publications:

- Liepins, R., Germann, U., Barzdins, G., Birch, A., Renals, S., Weber, S., van der Kreeft, P., Bourlard, H., Prieto, J., Klejch, O., et al. (2017). The SUMMA platform prototype. In *Software Demonstrations ACL*.
- Tsunoo, E., Klejch, O., Bell, P., and Renals, S. (2017). Hierarchical recurrent neural network for story segmentation using fusion of lexical and acoustic features. In *ASRU*.
- Roth, J., Chaudhuri, S., Klejch, O., Marvin, R., Gallagher, A., et al. (2019). AVA-ActiveSpeaker: An audio-visual dataset for active speaker detection. *arXiv preprint arXiv:1901.01342*.
- Fainberg, J., Klejch, O., Renals, S., and Bell, P. (2019b). Lattice-based lightly-supervised acoustic model training. In *Interspeech*.
- Fainberg, J., Klejch, O., Loweimi, E., Bell, P., and Renals, S. (2019a). Acoustic model adaptation from raw waveforms with SincNet. In *ASRU*.

Chapter 2

Automatic Speech Recognition

Automatic speech recognition (ASR) systems transcribe input audio into a sequence of words. Thanks to recent advancements, mainly in hardware, training data availability and training methods, ASR surpassed the usability threshold which caused its adoption in many areas of everyday life. ASR systems are used in a range of applications such as dictation, dialogue systems for smart assistants, subtitling and many others. In this chapter we describe how ASR works. We especially focus on hybrid DNN-HMM acoustic models (Bourlard and Morgan, 1993, 1994; Hinton et al., 2012; Yu and Deng, 2016) which we have used throughout this thesis.

The goal of automatic speech recognition (ASR) is to find the most likely word sequence $\hat{W} = w_1, \ldots, w_L$ of length L that corresponds to an input audio signal, from which a sequence of acoustic features $X = x_1, \ldots, x_T$ of length T is extracted. In the statistical modelling approach we use a posterior $P(W \mid X)$ to find the most likely sequence of words $\hat{W}$

$$\hat{W} = \arg\max_{W} P(W \mid X). \tag{2.1}$$

Nowadays, there are two predominant approaches for modelling $P(W \mid X)$. The first models the posterior directly using a neural network, this approach is sometimes called end-to-end. The most common examples of this approach are Connectionist Temporal Classification (CTC) (Graves, 2016), the Recurrent Neural Network Transducer (RNN-T) (Graves, 2012) and sequence-to-sequence models (Chorowski et al., 2015; Chan et al., 2016). The alternative, more traditional approach, is the modular approach, which decomposes the posterior into several models using Bayes' rule:

$$P(W \mid X) = \frac{P(X \mid W) P(W)}{P(X)}. \tag{2.2}$$

Therefore, the most likely sequence of words $\hat{W}$ can be found as

$$\hat{W} = \arg\max_{W} \frac{P(X \mid W) P(W)}{P(X)}. \tag{2.3}$$

Since we are looking only for the most likely sequence of words $\hat{W}$, we ignore $P(X)$ in the denominator:

$$\hat{W} = \arg\max_{W} \underbrace{P(X \mid W)}_{\text{AM}} \underbrace{P(W)}_{\text{LM}}. \tag{2.4}$$

As a result, the posterior is decomposed into two models: $P(X \mid W)$, which we call an *acoustic model* (AM), and $P(W)$, which we call a *language model* (LM).

Both approaches have their advantages and disadvantages. The main benefit of the end-to-end approach is that it optimises the posterior directly. Also, since it is just a single neural network, it is much easier to deploy end-to-end models into production, especially when we want to run ASR on-device. The main benefit of the modular approach is that it allows the training of models on different datasets. We typically train the acoustic model on audio data with corresponding transcripts and we train the language on a significantly larger text corpora. This modularisation also allows to adapt the ASR system to different domains just by updating the language model. Moreover, the modular approach allows us to use pronunciation modelling which is very important in some languages, especially in languages without any written form. The main disadvantage of the modular approach is that an improvement in one of its components does not necessarily imply an improved performance of the overall system. This problem can be mitigated by using sequence discriminative training criteria that take the language model into account during training of the acoustic models. These criteria include maximum mutual information (MMI) (Bahl et al., 1986; Valtchev et al., 1997), minimum phone error (MPE) (Povey, 2005), state-level minimum Bayes risk (sMBR) (Kaiser et al., 2000; Gibson and Hain, 2006; Povey and Kingsbury, 2007).

In this thesis we focus on the traditional modular approach that uses both acoustic and language models. The ASR pipeline of the modular approach, as illustrated in Figure 2.1, traditionally consists of four components:

1. *feature extraction* extracts acoustic features X from the input audio signal A by converting the time-domain signal into frequency domain acoustic features such as Mel filter bank (Fbank) coefficients (Deng et al., 2013), Mel-frequency cepstral coefficients (MFCC) (Davis and Mermelstein, 1980) or Perceptual linear predictions (PLP) (Hermansky, 1990) features.

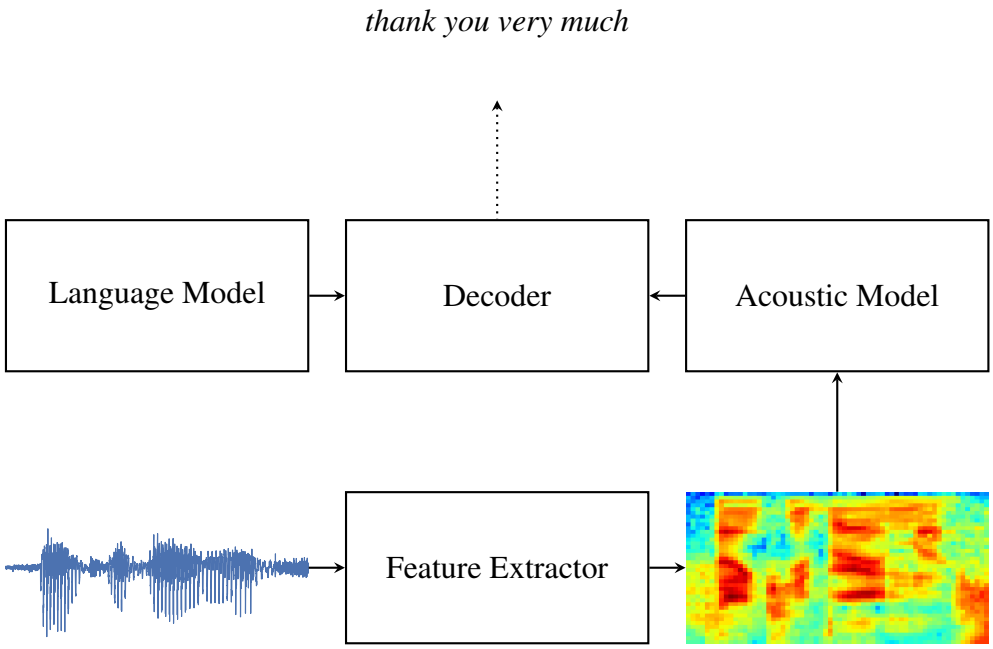

Figure 2.1: The ASR pipeline consists of four components. A *feature extractor* converts an input audio signal into a variable length sequence of acoustic features. These acoustic features are fed to an *acoustic model* that produces an AM score. A *language model* then produces an LM score for hypothesised sequence of words. A *decoder* combines these two scores to find the most likely sequence of words $\hat{W}$.

2. an *acoustic model* $P(X \mid W)$ computes the scores for a given sequence of acoustic features. We describe acoustic models in more detail in the rest of this chapter.

3. a *language model* $P(W)$ that estimates the score as the prior probability for hypothesised sequences of words. In ASR the language models are typically implemented as count-based n-gram language models. Recently, neural network language models have been used as well (Bengio et al., 2003; Mikolov et al., 2010), especially to rescore n-best lists.

4. a *decoder* combines scores from the acoustic and the language model to find the most likely sequence of words $\hat{W}$ in the hypothesis space.

Note that in some approaches some components are merged into a single model. As we mentioned before, the end-to-end models combine the acoustic model and the language model into a single neural network (Graves, 2012; Chorowski et al., 2015; Chan et al., 2016). Some approaches also combine feature extraction and the acoustic

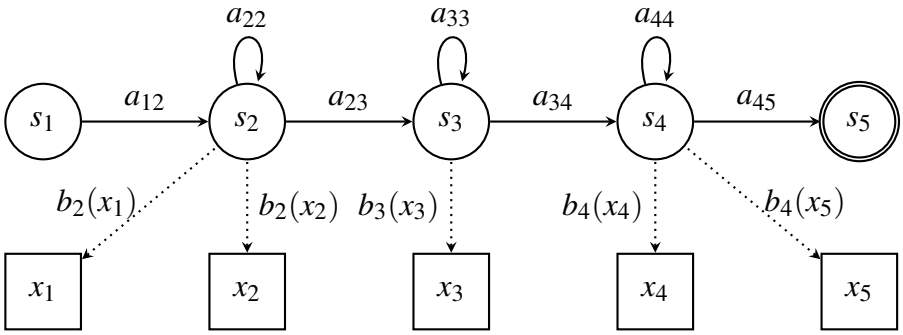

Figure 2.2: An illustration of a five state hidden Markov model (HMM) with transition probabilities a_{ij} and output observation probabilities $b_i(\cdot)$ that is used to model phones in ASR. Note that state s_1 and state s_5 are non-emitting states and they are only used for the construction of composite models by concatenating models of single speech units.

model into a single model which allows the model to learn a task specific feature extractor from raw waveforms (Sainath et al., 2015; Palaz et al., 2015; Golik et al., 2015).

2.1 Acoustic Modelling

The goal of the acoustic model is to model the likelihood $P(X\,|\,W)$. It does so typically by modelling speech units with continuous density hidden Markov models (HMM) (Rabiner, 1989; Gales et al., 2008) that can deal with the variable-length nature of speech units. In order to model the whole utterance, a composite model is created for the utterance by concatenating a sequence of models corresponding to the speech units present in the utterance. We typically model the speech units with a five state left-to-right HMM topology, as illustrated in Figure 2.2, with transition probabilities $\{a_{ij}\}$ and output observation probabilities $\{b_i(\cdot)\}$. Note that the first and the last state of the HMM are non-emitting and they are used only to construct the composite model, therefore we sometimes also call these models three state left-to-right HMMs. The HMM works as follows: at every time step t the model transitions from its current state s_t to the next state s_{t+1} with the transition probability $a_{s_t s_{t+1}}$ and outputs the acoustic vector x_t with the probability $b_{s_t}(x_t)$. The likelihood $P(X\,|\,W)$ is evaluated as

$$P(X\,|\,W) = \sum_{\pi} P(\pi, X\,|\,W)\,, \tag{2.5}$$

where $\pi = s_0, \ldots, s_T$ is a valid state sequence through the composite model and

$$P(\pi, X \mid W) = a_{s_0 s_1} \prod_{t=1}^{T} b_{s_t}(x_t) a_{s_t s_{t+1}}. \tag{2.6}$$

In GMM-HMM systems the output observation probabilities $\{b_i(\cdot)\}$ are modelled by Gaussian mixture models (GMM).

$$b_i(x) = \sum_{m=1}^{M} c_{im} \mathcal{N}(x; \mu^{(im)}, \Sigma^{(im)}), \tag{2.7}$$

where c_{im} are mixing coefficients satisfying the condition

$$\sum_{m=1}^{M} c_{im} = 1 \tag{2.8}$$

and $\mu^{(im)}$ is a mean vector and $\Sigma^{(im)}$ is a covariance matrix, typically diagonal. The transition probabilities and the output observation probabilities are estimated on the training data with the Baum-Welch algorithm (Baum et al., 1970; Juang et al., 1986) which might be considered a special case of the Expectation-Maximisation algorithm (Dempster et al., 1977). For more information about GMM-HMM systems we refer to Gales et al. (2008).

The base speech unit used in ASR is the *phone*. When we consider phones without their context, we call them *monophones*. We model words by creating a composite model for each word by concatenating HMMs for a sequence of phones that is given by a pronunciation dictionary. Note that each word can be uttered in multiple ways. In English we typically use 40 phones. However, phones can be produced differently depending on their context, which lead to using context dependent phones. Typically, we use *triphones* that model a central phone in the context of its preceding and its following phone. However, this results in a rapid increase of HMM states that need to be modelled, for example in English we would need to model $3 \times 40^3 = 192000$ states, which is impractical, because we usually do not have enough training data to estimate GMMs for all states. Therefore, the states of HMMs for triphones are clustered and tied to reduce their number (Young et al., 1994). We call the tied states *senones* (Hwang and Huang, 1992).

2.2 Hybrid DNN-HMM Acoustic Models

In hybrid DNN-HMM acoustic models the output observation probabilities $\{b_i(\cdot)\}$ are modelled with a single neural network (Figure 2.3) (Bourlard and Morgan, 1994; Seide

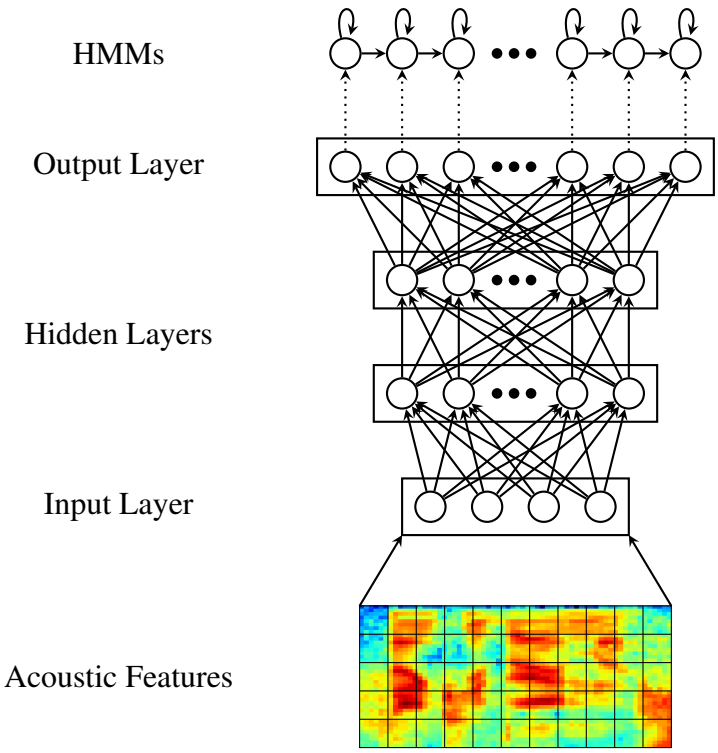

Figure 2.3: A diagram of the feed forward neural network with two hidden layers and a context window of 11 frames predicting senone posterior probabilities.

et al., 2011). The neural network is discriminatively trained to model the conditional posterior $P(S|X)$ for all senones S. In order to be able to use the output of the network for decoding, we need to use Bayes' rule to compute the likelihoods $P(X|S)$ as

$$P(X|S) = \frac{P(S|X)P(X)}{P(S)}. \tag{2.9}$$

Since $P(X)$ does not affect the decoder in Equation 2.4 we can ignore it and we can use scaled likelihoods $\hat{P}(X|S)$ (Bourlard and Morgan, 1994) that are obtained by dividing the posterior probabilities $P(S|X)$ by the senone priors $P(S)$

$$P(X|S) \propto \hat{P}(X|S) = \frac{P(S|X)}{P(S)}. \tag{2.10}$$

The priors $P(S)$ are estimated from the training data, by diving the senone specific counts C_s obtained from the training alignments with the total number of frames C

$$P(s) = \frac{C_s}{C}. \tag{2.11}$$

2.3 Neural Network Architecture

The architecture of the neural network in the hybrid DNN-HMM acoustic model (Figure 2.3) consists of three parts: an *input layer*, a sequence of *hidden layers* and an *output layer*.

The *input layer* accepts a stack of acoustic features for several consecutive frames, denoted x, and projects them into a hidden representation $h_0 \in \mathbb{R}^n$ with an affine transformation with parameters $W_0 \in \mathbb{R}^{m \times n}$ and $b_0 \in \mathbb{R}^n$

$$h_0 = W_0 x + b_0, \tag{2.12}$$

where is m the dimension of the input vector and n is the dimension of the hidden representation. The input layer is usually fused into the first hidden layer and trained jointly with the rest of the network. Alternatively, the parameters of the input layer can be estimated separately and then they can be kept fixed during the training of the neural network. For example, in Kaldi (Povey et al., 2011), a stack of 5 acoustic frames is fed into the input layer and the parameters of the affine transformation are estimated using Linear Discriminant Analysis (LDA) (Batlle et al., 1998; Duda et al., 2012). One important consequence of using LDA as the input layer is that the input layer implicitly performs global mean and variance normalisation of the input features, which helps the training of the neural network.

The hidden representation h_0 is then processed with a sequence of *hidden layers*. Each hidden layer usually consists of three components: an *affine transformation* with parameters $W_i \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}^n$, an *activation function* and an optional *normalisation*. In the feed-forward neural networks, the affine projection is applied only to the output of the previous layer for a given time-step to produce pre-activations z_i

$$z_i = W_i h_{i-1} + b_i. \tag{2.13}$$

In Time-delayed neural networks (TDNN) (Waibel et al., 1989; Lang et al., 1990; Peddinti et al., 2015), the affine projection is applied to a stack of hidden representations corresponding to several time-steps (Figure 2.4). In fact, TDNNs correspond to one-dimensional convolutional neural networks (LeCun et al., 1995; Abdel-Hamid et al., 2012), where the convolutions are applied in the time-domain. Note that feed forward neural networks are a special case of TDNNs.

Subsequently, a non-linear *activation function* is applied element-wise on the result of the affine projection. The most commonly used activation functions are

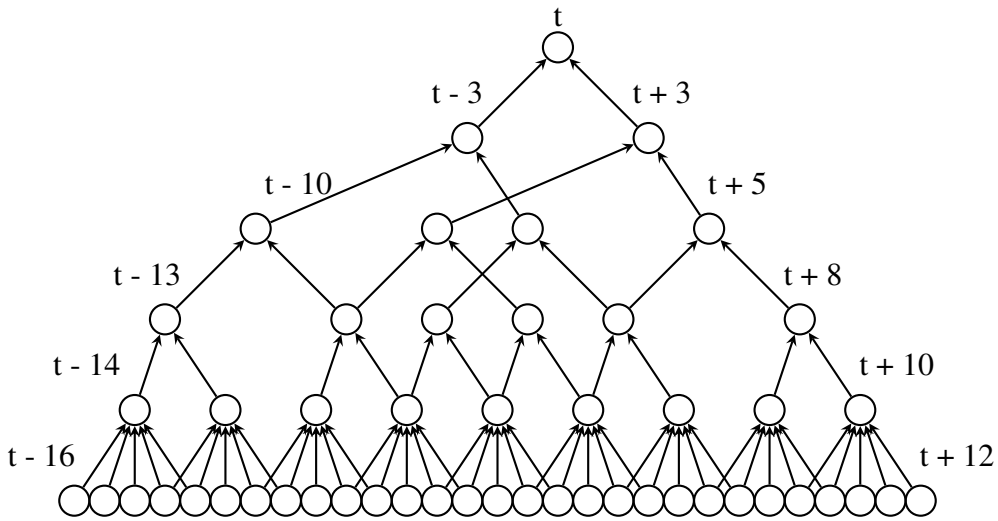

Figure 2.4: A diagram of a Time delayed neural network with a recursive structure defined by offsets $\{-2,-1,0,1,2\},\{-1,2\},\{-3,3\},\{-7,3\},\{-3,3\}$. For example, in order to make a prediction at a time-step t we need the hidden activations of the penultimate layer for the time-steps $t-3$ and $t+3$, *and* to produce *those* we need the hidden activations of the previous layer for the time-steps $t-3-7$, $t-3+3$, $t+3-7$ and $t+3+3$. This process is applied recursively until we reach the acoustic input features. As can be seen from the diagram, this results in a subsampled architecture where we do not have to compute all the hidden activations to produce the prediction for time-step t.

- the *sigmoid*

$$\sigma(z) = \frac{1}{1+e^{-z}}, \tag{2.14}$$

- the *hyperbolic tangent*

$$\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}, \tag{2.15}$$

- and the *rectified linear unit* (Nair and Hinton, 2010)

$$\text{relu}(z) = \max(0,z). \tag{2.16}$$

These activation functions are used to introduce non-linearities into the network, because otherwise the network would be just a linear network, whose layers can be fused into a single affine transformation.

Finally, the hidden representations can be normalised with *Batch Normalisation* (Ioffe and Szegedy, 2015), *Layer Normalisation* (Ba et al., 2016) or other normalisation techniques (Salimans and Kingma, 2016; Ioffe, 2017; Ulyanov et al., 2016).

Normalising hidden representations can significantly speed up training convergence. This is because it normalises the distributions of inputs for each hidden layer. This is important because even when we have normalised features as an input to the neural network, the distribution of activations can become skewed as the input features are passed through several hidden layers. Also normalisation enables the usage of large learning rates (Ioffe and Szegedy, 2015). This is due to the fact that normalisation together with the ReLU activation function (or linear activation function if we normalise preactivations) makes the parameters of the layer scale invariant, e.g. the output of a layer with parameters A, b is equal to the output of a layer with parameters pA, pb for $p > 0$. However, corresponding gradients differ by a factor $\frac{1}{p^2}$, such that $\nabla_{pA}\mathcal{L} = \frac{1}{p^2}\nabla_A L$. This makes the step size of the update dependent on the scale of the weights and as the training progresses the weights become larger and the actual step size becomes smaller. As a result, the neural network has the ability to tune its effective learning rate on its own (Arora et al., 2019).

Batch Normalisation (Ioffe and Szegedy, 2015) produces a normalised hidden representation h_i' from a hidden representation h_i as

$$h_i' = \gamma \frac{h_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta, \tag{2.17}$$

where μ is an estimated mean of hidden representations h_i, σ^2 is an estimate of their variance; and γ and β are trainable parameters. During training, these quantities are computed on a batch of hidden activations for N inputs as

$$H_i = \left\{h_i^1, \ldots, h_i^N\right\}, \tag{2.18}$$

such that

$$\mu_B = \frac{1}{N}\sum_{j=1}^{N} h_i^j \tag{2.19}$$

and

$$\sigma_B^2 = \frac{1}{N}\sum_{j=1}^{N} (h_i^j - \mu)^2. \tag{2.20}$$

During inference, a global mean μ_G and global variance σ_G^2 are used for Batch Normalisation. They can either be computed on the whole training dataset or they can be estimated as running mean statistics during training.

The statistics computed on the batches can be significantly different from the global statistics depending on how the batches are sampled (for example if we include only one speaker in a single batch compared to having multiple speakers in a single batch).

Therefore, Ioffe (2017) proposed a technique called Batch Renormalisation, which applies the same statistics both during training and inference. Batch renormalisation normalises hidden activations h_i during training in the following way:

$$h_i' = \gamma \left(r \frac{h_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + d \right) + \beta, \tag{2.21}$$

where r and d are computed during the forward pass as follows:

$$r = \frac{\sqrt{\sigma_B^2 + \varepsilon}}{\sqrt{\sigma_G^2 + \varepsilon}}, \tag{2.22}$$

$$d = \frac{\mu_B - \mu_G}{\sqrt{\sigma_G^2 + \varepsilon}} \tag{2.23}$$

and are treated as constants during the backward pass. By substituting Equations 2.22 and 2.23 into Equation 2.21 we obtain

$$h_i' = \gamma \frac{h_i - \mu_G}{\sqrt{\sigma_G^2 + \varepsilon}} + \beta, \tag{2.24}$$

which is identical to performing Batch Normalisation during inference, therefore the normalisation works in the same way during both training and inference.

In Layer Normalisation (Ba et al., 2016) mean and variance statistics are not computed on the batch but are computed for each example in the batch separately, therefore the statistics do not change between training and inference. The mean and variance for activations h_i are computed as

$$\mu = \frac{1}{D} \sum_{j=1}^{D} [h_i]_j \tag{2.25}$$

and

$$\sigma^2 = \frac{1}{D} \sum_{j=1}^{D} ([h_i]_j - [\mu]_j)^2 \tag{2.26}$$

respectively, where D is the dimensionality of the i-th hidden layer and $[h_i]_j$ denotes the j-th element of the hidden representation h_i. Note that in Kaldi (Povey et al., 2011) a similar approach, which only normalises the variance σ^2, is used.

Finally, the output of the last hidden layer h_n is fed into the *output layer* with parameters W_o and b_o, which computes the posteriors $P(S|X)$ as:

$$P(S|X) = \text{softmax}(z_o), \tag{2.27}$$

where

$$z_o = W_o h_n + b_o \tag{2.28}$$

and the softmax activation is used to obtain a normalised probability distribution as

$$P(S = s \,|\, X) = [\text{softmax}(h_o)]_s = \frac{e^{[x]_s}}{\sum_{i=1}^{C} e^{[x]_i}}. \tag{2.29}$$

The neural network is trained with gradient descent (Rumelhart et al., 1986) with the cross-entropy loss function. The target senones are obtained by force-aligning the training data with a GMM-HMM model. The network can be trained with various optimisers such as Stochastic Gradient Descent (SGD) (Bishop, 2006), Adam (Kingma and Ba, 2014) or Natural Gradient (Povey et al., 2014). More information about neural network architectures and neural network training can be found in Jurafsky and Martin (2019, Chapter 7) or Goodfellow et al. (2016).

Chapter 3

Speaker Adaptation

In the previous chapter we described the basics of automatic speech recognition. The performance of ASR models, similarly as for all other statistical machine learning models, deteriorates quickly when a distribution of the testing data differs from the distribution of the training data. One way to remedy this problem is to adapt the ASR system to new conditions found in the test data. For example, we can adapt the ASR system to unseen speakers via speaker adaptation. Even though both acoustic model and language model speaker adaptation have been explored in the past (Neto et al., 1995; Gales, 1998; Woodland, 2001; Stolcke, 2001; Gretter and Riccardi, 2001; Bacchiani and Roark, 2003; Gangireddy et al., 2016), we focus solely on speaker adaptation of the acoustic model in this thesis. Speaker adaptation of an acoustic model $f(x;\theta)$ is performed as follows. First, we need to obtain some adaptation data

$$\mathcal{D}^{\text{adapt}} = \left\{ (x_j^{\text{adapt}}, y_j^{\text{adapt}}) | j \in \{1 \dots n\} \right\}, \tag{3.1}$$

which consists of n tuples of input acoustic features x_j^{adapt} and corresponding labels y_j^{adapt}. These labels can be obtained either from the reference transcript (*supervised speaker adaptation*) or from transcriptions produced by some seed ASR system (*unsupervised speaker adaptation*). The adaptation data might either be some enrolment utterance or it can be a part of the recording that we want to transcribe. We use the adaptation data $\mathcal{D}^{\text{adapt}}$ to adapt the ASR model $f(x;\theta)$ to improve its performance on the test data $\mathcal{D}^{\text{test}}$.

In this chapter we will review techniques for speaker adaptation that are traditionally divided into three approaches: *feature-space approaches*, *model-based approaches* and *auxiliary feature approaches*. We only review speaker adaptation techniques that are applicable to DNN-HMM acoustic models; for an overview of speaker

adaptation of GMM-HMM models we refer to Woodland (2001). We also review speaker adaptive training of DNN-HMM models that applies speaker adaptation techniques during the training of the acoustic model in order to remove speaker variability from the training data allowing the canonical acoustic model to focus solely on modelling phonological variations.

3.1 Feature-Space Speaker Adaptation

First of the approaches is the Feature-space speaker adaptation that performs speaker adaptation by applying some transformation to the input acoustic features. Probably the simplest method is cepstral mean and variance normalisation (CMVN) which normalises cepstral features to have zero mean and unit variance for each speaker. The normalised features x' can be obtained as follows:

$$x' = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}, \tag{3.2}$$

where μ is a cepstral mean, σ^2 is a cepstral variance and ε is a small constant used to ensure numerical stability. The cepstral mean and variance statistics are computed on the corresponding speaker data, which might be some enrolment utterance, a current utterance or a whole recording. The effectiveness of CMVN, and also all other methods, depends on the amount and quality of the data that we use to compute these statistics. For example, estimating them on short recordings or recordings with a lot of silence can be harmful for the performance of the acoustic model.

Another feature-space speaker adaptation method is Vocal Tract Length Normalisation (VTLN) (Andreou, 1994; Lee and Rose, 1996). This technique is based on the observations that the vocal tract length, which varies a lot in the population (children and females usually have shorter vocal tract than males (Lee and Rose, 1996)), significantly affects formant frequencies of vowels causing the performance of the speaker independent model to degrade. Therefore, a speaker specific warping factor of the frequency-axis is applied to minimise this mismatch between speakers. The warping factor α can be estimated from formant positions which correlate with vocal tract length (Wakita, 1977; Eide and Gish, 1996) or it can be estimated with a line-search to maximise the likelihood of the adaptation data.

The last feature-space speaker adaptation method that we will cover in this thesis is an affine transformation with parameters $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ that adapts the input

features as

$$x' = Ax + b. \tag{3.3}$$

There are two major approaches for the estimation of the parameters A and b. The first, Constrained Maximum Likelihood Linear Regression (cMLLR) sometimes also called Feature-space Maximum Likelihood Linear Regression (fMLLR), uses the EM algorithm (Dempster et al., 1977) to estimate these parameters to maximise the likelihood of the adaptation data. cMLLR belongs to a wider family of Maximum Likelihood Linear Regression (MLLR) speaker adaptation methods developed for GMM-HMM models (Gales, 1998). The second, Linear Input Network (LIN) (Neto et al., 1995) estimates the parameters discriminatively with gradient descent.

3.2 Model-Based Speaker Adaptation

Model-based speaker adaptation uses the adaptation data $\mathcal{D}^{\text{adapt}}$ to update the acoustic model $f(x;\theta)$, herein represented as a neural network, to allow the acoustic model to adapt to differences in the adaptation data (Neto et al., 1995). The acoustic model is adapted by discriminatively adapting the parameters θ using gradient descent with a suitable objective function. Model-based adaptation methods can be split into two groups. The first group adapts the whole acoustic model or some of its layers (Liao, 2013; Yu et al., 2013; Huang and Gong, 2015). The second group employs linear transformations to transform input features x, hidden activations h or outputs y of the acoustic model. These transformations are called Linear Input Network (LIN) (Neto et al., 1995), Linear Hidden Network (LHN) (Gemello et al., 2007) and Linear Output Network (LON) (Li and Sim, 2010) respectively. These transforms can be parameterised with a transformation matrix $A \in \mathbb{R}^{n \times n}$ and a bias $b \in \mathbb{R}^n$. The transformation matrix A is initialised as an identity matrix and the bias b is initialised as a zero vector prior to the speaker adaptation. The adapted hidden activations then become

$$h' = Ah + b. \tag{3.4}$$

Neural network acoustic models have a large modelling capacity, but the adaptation of too many parameters might lead to overfitting to the adaptation data. The overfitting can manifest itself in three ways. First, if the amount of adaptation data is limited, then speaker adaptation can overfit to classes seen in the adaptation data and forget about classes that are not present in the adaptation data. Second, the model can also overfit to certain speaker characteristics that are present in the adaptation data but that

are not fully representative of the speaker. Third, in unsupervised speaker adaptation the labels are obtained from erroneous transcripts, which makes the ASR model prone to overfitting to errors made in the first pass decoding. Therefore, model-based adaptation typically needs to be strongly regularised. Overfitting is traditionally prevented either by reducing the number of speaker dependent parameters, or by using appropriate regularisation losses that prevent the adapted model in diverging too far from the original model. We discuss limiting the number of speaker dependent parameters in Section 3.5 and various regularisation techniques in Section 3.6.

3.3 Auxiliary Feature Speaker Adaptation

Both of the aforementioned approaches are typically two-pass methods, because they require some optimisation in order to find speaker dependent parameters. In contrast, *Auxiliary feature* speaker adaptation can be used in one-pass decoding because it uses auxiliary features estimated by some external model to inform the speaker about the target speaker, channel, environment, etc. Many different auxiliary features have been used to inform the acoustic model about target speaker characteristics in the past. For example *i-vectors* (Dehak et al., 2011; Saon et al., 2013), *d-vectors* (Variani et al., 2014), *x-vectors* (Snyder et al., 2018; Rownicka et al., 2019), *speaker codes* (Abdel-Hamid and Jiang, 2013), *bottleneck speaker vectors* (Huang and Sim, 2015), features produced by *summary networks* (Veselý et al., 2016; Delcroix et al., 2018b), *LHUC features* (Xie et al., 2019c) or pooled and transformed activations of another acoustic model (Rownicka et al., 2018, 2019). *r-vectors* (Khokhlov et al., 2019) were used to inform the acoustic model about target room acoustics and *factorised representations* (Fainberg et al., 2017) were used to inform the acoustic model both about speakers and environments independently.

3.4 Estimation of Speaker Dependent Parameters

In the previous sections we introduced speaker adaptation methods according to the traditionally used categorisation into *feature space*, *model-based* and *auxiliary feature* speaker adaptation methods. We think that this categorisation in the context of adaptation of neural network acoustic models might be confusing, because one method could belong to multiple groups. Moreover, it can be shown that all feature-based and auxiliary feature adaptation methods can be posed as model-based adaptation methods.

For example, feature-based methods can be posed as model-based adaptation methods by fusing the feature transformation with the acoustic model and treating the feature transformation as the first hidden layer of the acoustic model. Similarly, auxiliary features, which are typically used for bias adaptation of the first hidden layer (Saon et al., 2013), can be merged with the parameters of the acoustic model. As a result, the main difference between the various methods is how they estimate the speaker dependent parameters. For example, CMVN and formant based VTLN estimates speaker dependent parameters by computing statistics on the adaptation data. cMLLR uses the EM algorithm to estimate linear transformations to maximise the likelihood of the adaptation data and discriminative methods use gradient descent to optimise an objective function on the adaptation data. Finally, auxiliary methods use external models to predict speaker dependent parameters.

We believe that speaker dependent parameter estimation is also an important criterion for distinguishing between speaker adaptation and speaker normalisation. The difference is that speaker adaptation requires transcribed adaptation data to estimate the speaker dependent parameters. In supervised speaker adaptation we use reference transcriptions and in unsupervised speaker adaptation we use transcriptions obtained with a seed speaker-independent model. Speaker normalisation, on the other hand, can estimate speaker dependent parameters only from the acoustic features without any transcriptions.

3.5 Structured Linear Transformations

In the previous section we argued that all speaker adaptation techniques can be framed as model-based speaker adaptation and the only difference is how we estimate the speaker dependent parameters. In this section we will review various ways to incorporate speaker dependent parameters into the neural network acoustic models. For simplicity, we will assume that the speaker dependent parameters can be estimated in any of the aforementioned ways. We will denote speaker dependent parameters with the subscript s, for example A_s, b_s and we will denote the set of all speaker dependent parameters as θ_s. An acoustic model with speaker independent parameters θ and speaker dependent parameters θ_s will be written as $f(x;\theta,\theta_s)$.

The linear hidden networks adapt activations of the i-th layer with parameters $A_s \in \mathbb{R}^{n\times n}$, $b_s \in \mathbb{R}^n$ as

$$h_i' = A_s h_i + b_s. \tag{3.5}$$

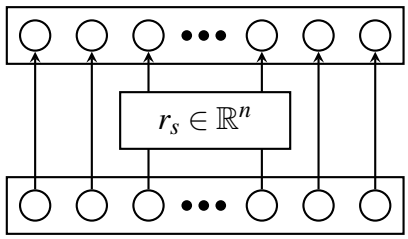

Figure 3.1: A diagram of the Learning Hidden Unit Contributions (LHUC) adaptation.

However, even one transformation matrix A_s can contain too many speaker dependent parameters, which makes adaptation susceptible to overfitting to adaptation data. It also limits its practical usage in real world deployment because of memory requirements. Therefore, in the past, there has been considerable research into how to structure the matrix A_s and the bias b_s to remove the number of speaker dependent parameters.

The first set of approaches restricts the adaptation matrix A_s to be diagonal, we will denote the diagonal elements as $r_s = diag(A_s)$. Thus, the adapted hidden activations become

$$h'_i = r_s \odot h_i + b_s. \tag{3.6}$$

There are several methods that belong to this set of adaptation methods. For example, Learning Hidden Unit Contributions (LHUC) (Swietojanski and Renals, 2014), illustrated in Figure 3.1, that adapts only the parameters r_s.

$$h'_i = r_s \odot h_i. \tag{3.7}$$

Speaker Codes (Abdel-Hamid and Jiang, 2013) essentially allow the adaptation of the speaker dependent biases b_s.

$$h'_i = h_i + b_s. \tag{3.8}$$

Similarly, Wang and Wang (2017) proposed a method that adapts both r_s and b_s as parameters β_s and γ_s of batch normalisation layers. Thus it adapts both the scale and the offset of the hidden layer activations with the mean μ and the standard deviation σ:

$$h' = \gamma_s \frac{h-\mu}{\sigma} + \beta_s. \tag{3.9}$$

There have also been approaches, that further reduce the number of speaker dependent parameters by removing its dependency on the hidden layer width. They achieve this by using speaker independent control networks that predict the speaker-dependent parameters

$$r_s = c_r(z_s, \theta_r), \tag{3.10}$$

$$b_s = c_b(z_s, \theta_b), \tag{3.11}$$

from some lower dimensional speaker dependent representations $z_s \in \mathbb{R}^k$, typically auxiliary features like i-vectors. The control networks $c_*(z_s, \theta_*)$ can be implemented as a single linear transformation or as a multi-layer neural network. For example, Subspace LHUC (Samarakoon and Sim, 2016b) used a control network to predict LHUC parameters r_s from i-vectors z_s. Cui et al. (2017) used auxiliary features to adapt both scale r_s and offset b_s. This style of model-adaptation has the nice property of having a very low memory footprint, because the i-vector dimension (usually 100) is much lower than commonly used dimensions for the hidden layers. Samarakoon and Sim (2016b) reported a 94% memory footprint reduction compared to standard LHUC adaptation.

A similar approach with low-memory footprint adapts the activation functions instead of the scale r_s and offset b_s. Zhang and Woodland (2015) proposed the use of parameterised sigmoid and ReLU activation functions. With the parameterised sigmoid function, hidden activations h_i are computed from hidden pre-activations z_i as

$$h = \eta_s \frac{1}{1 + e^{-\gamma_s z_i + \theta_s}}, \tag{3.12}$$

where η_s, γ_s and θ_s are speaker dependent parameters. $|\eta_s|$ controls the scale of the hidden activations, γ_s controls the slope of the sigmoid function and θ_s controls the midpoint of the sigmoid function. Similarly, parameterised ReLU activations was defined as

$$h = \begin{cases} \alpha_s z & \text{if } z > 0 \\ \beta_s z & \text{if } z \leq 0 \end{cases}, \tag{3.13}$$

where α_s and β_s are speaker dependent parameters that correspond to slopes for positive and negative pre-activations, respectively.

Other approaches factorise the transformation matrix A_s into a product of low-rank matrices to obtain a smaller number of speaker dependent parameters. Zhao et al. (2016) proposed a method called *Low-Rank Plus Diagonal* (LRPD), which reduces the number of speaker dependent parameters by approximating the linear transformation matrix $A_s \in \mathbb{R}^{n \times n}$ as

$$A_s \approx D_s + P_s Q_s, \tag{3.14}$$

where the $D_s \in \mathbb{R}^{n \times n}$, $P_s \in \mathbb{R}^{n \times k}$ and $Q_s \in \mathbb{R}^{k \times n}$ are speaker dependent matrices and D_s is a diagonal matrix. This approximation was motivated by the fact that the adapted hidden activations should not be too different from the unadapted hidden activations

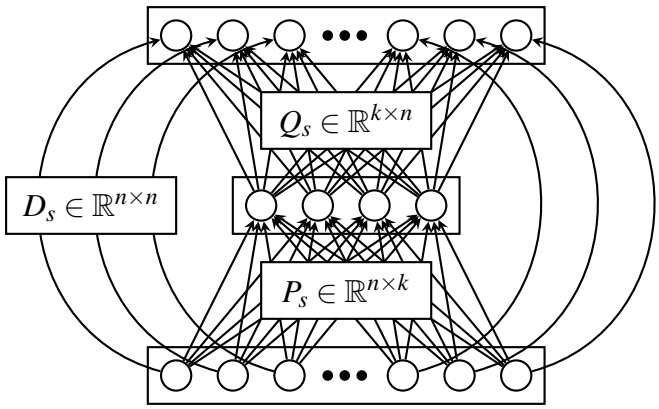

Figure 3.2: A diagram of the Low-Rank Plus Diagonal (LRPD) adaptation method that factorises the adaptation matrix A_s as $D_s + P_s Q_s$.

when we have only a limited amount of adaptation data, therefore the adaptation linear transformation should be close to a diagonal matrix. In fact, for $k = 0$ LRPD reduces to LHUC adaptation. LRPD adaptation can be implemented by inserting two hidden linear layers and a skip connection, as illustrated in Figure 3.2.

Zhao et al. (2017) later presented an extension to LRPD called *Extended LRPD* (eLRPD), which removed the dependency of the number of speaker dependent parameters on the hidden layer size by performing a different approximation of the linear transformation matrix A_s

$$A_s \approx D_s + P T_s Q, \tag{3.15}$$

where matrices $D_s \in \mathbb{R}^{n \times n}$ and $T_s \in \mathbb{R}^{k \times k}$ are speaker dependent and matrices $P \in \mathbb{R}^{n \times k}$ and $Q \in \mathbb{R}^{k \times n}$ are treated as speaker independent. Thus the number of speaker dependent parameters is mostly dependent on k, which can be chosen arbitrarily. Similarly to LRPD, eLRPD can be implemented by inserting three hidden linear layers and a skip connection.

Another set of approaches uses the speaker dependent parameters as mixing coefficients α_s for a set of bases B_i which factorise the transformation matrix. As before, the mixing coefficients can be predicted from lower-dimensional speaker dependent representations. Samarakoon and Sim (2015, 2016a) proposed to use factorised hidden layers (FHL) that would allow both speaker-independent and speaker dependent modelling. With this approach activations of a hidden layer h with an activation function σ are computed as

$$h = \sigma\left((W + \sum_{i=0}^{n} [\alpha_s]_i B_i) x + b_s + b \right). \tag{3.16}$$

Note, that when $\alpha_s = 0$ and $b_s = 0$, the activations correspond to a standard SI model.

In Samarakoon and Sim (2016a), the bases B_i were rank-1 matrices, $B_i = \gamma_i \psi_i^T$, which allows the reparameterisation of Equation 3.16 as:

$$\begin{aligned} h &= \sigma\left((W + \sum_{i=0}^{n} [\alpha_s]_i B_i)x + b_s + b\right) \\ &= \sigma\left((W + \sum_{i=0}^{n} [\alpha_s]_i \gamma_i \psi_i^T)x + b_s + b\right) \\ &= \sigma\left((W + \Gamma D \Psi^T)x + b_s + b\right), \end{aligned} \tag{3.17}$$

where $D = diag(\alpha_s)$. In fact, this approach is very similar to Cluster Adaptive Training of DNN networks (CAT-DNN) (Tan et al., 2016) that uses full rank bases instead of rank-1 bases. Tan et al. (2016) explored three different ways of initialising the bases B_i. They can be initialised randomly, from different checkpoints of a well trained speaker independent model or by combining several condition dependent models (for example channel/gender dependent models). Similarly, speaker dependent mixing coefficients α_s can be initialised randomly, using some prior knowledge (for example channel/gender information), using some automatic data clustering or they can be predicted from auxiliary features as in Samarakoon and Sim (2016a).

Similarly, Delcroix et al. (2018a) proposed to adapt activations of a hidden layer with a Mixture-of-Experts (Jacobs et al., 1991). The adapted hidden unit activations are then

$$h' = \sum_{i=0}^{n} [\alpha_s]_i B_i h. \tag{3.18}$$

Finally, the number of speaker dependent parameters in all the aforementioned linear transformations can be reduced by applying them to bottleneck layers that have much lower dimensionality than the standard hidden layers. These bottleneck layers can be obtained directly by training a neural network with bottleneck-layers or by applying Singular Value Decomposition (SVD) to the hidden layers (Xue et al., 2013, 2014). This is done by applying SVD to the weight matrix $W \in \mathbb{R}^{n \times n}$ which factorises it as:

$$A = U \Sigma V^T \tag{3.19}$$

where the columns of the matrix $U \in \mathbb{R}^{n \times n}$ are called the left singular vectors of A, the columns of the matrix $V \in \mathbb{R}^{n \times n}$ are the right singular vectors of A, and $\Sigma \in \mathbb{R}^{n \times n}$ is a diagonal matrix with singular values of A on the diagonal. By setting the $(n-k)$ smallest singular values of A to 0, we can approximate A as

$$A \approx U' \Sigma' V'^T, \tag{3.20}$$

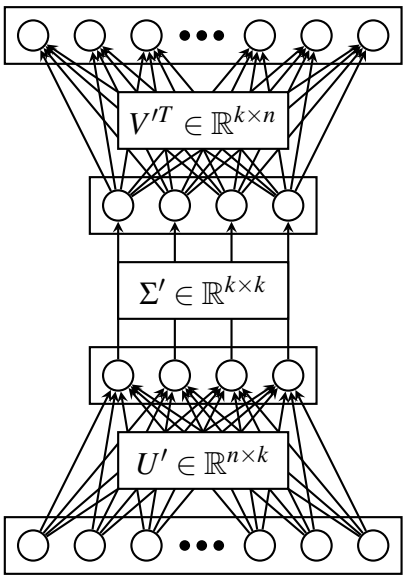

Figure 3.3: An illustration of SVD factorisation of a hidden layer which allows us to significantly reduce the number of speaker dependent parameters.

where $U' \in \mathbb{R}^{n \times k}$ correspond to the first k columns of U and $V'^T \in \mathbb{R}^{k \times n}$ corresponds to the first k rows of V^T. This is illustrated in Figure 3.3. By setting $W' = \Sigma' V'^T$, we approximate A as a product of two low-rank matrices $U' \in \mathbb{R}^{n \times k}$ and $W' \in \mathbb{R}^{k \times n}$

$$A \approx U'W'. \tag{3.21}$$

In Xue et al. (2014) both matrices U' and W' are treated as speaker-independent and the smaller matrix S_s is used for speaker adaptation

$$A \approx U'S_sW'. \tag{3.22}$$

This results in a substantial reduction of speaker dependent parameters, even for LHN adaptation.

In order to get a better idea of how these techniques perform, we ran supervised and unsupervised speaker adaptation experiments with 10s, 30s and 60s of adaptation data. The baseline model was a TDNN model with 6 hidden layers each with 600 units, ReLU activation functions and batch normalisation. The model predicted posterior probabilities for 4208 senones. We trained the acoustic model on a portion of the TED-LIUM dataset (Rousseau et al., 2012, 2014) obtained before the year 2012. We evaluated the model on a combined test set from the IWSLT 2010-2012 challenge (Paul et al., 2010; Federico et al., 2011, 2012). More details about the baseline acoustic model and the dataset can be found in Section 6.1 and Section 5.1, re-

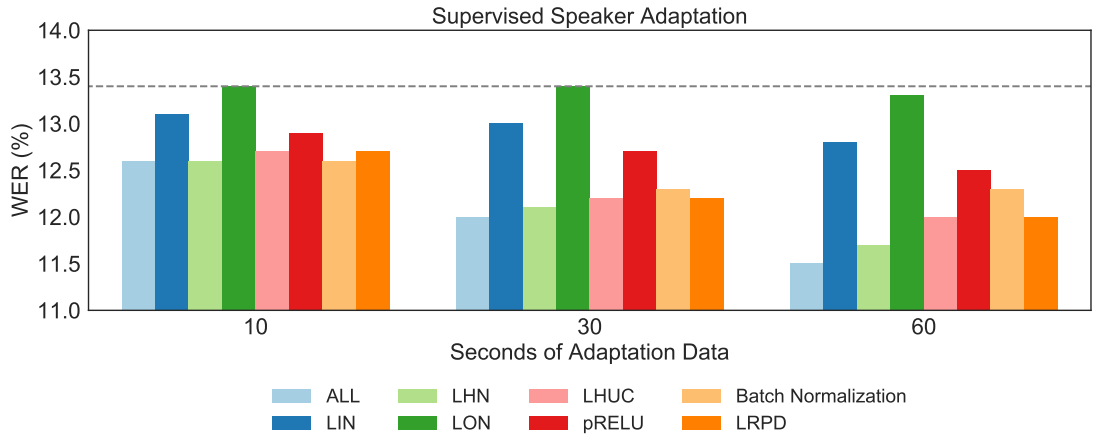

Figure 3.4: WER (%) for a comparison of various adaptation methods in supervised speaker adaptation. The dashed line shows the performance of the unadapted model.

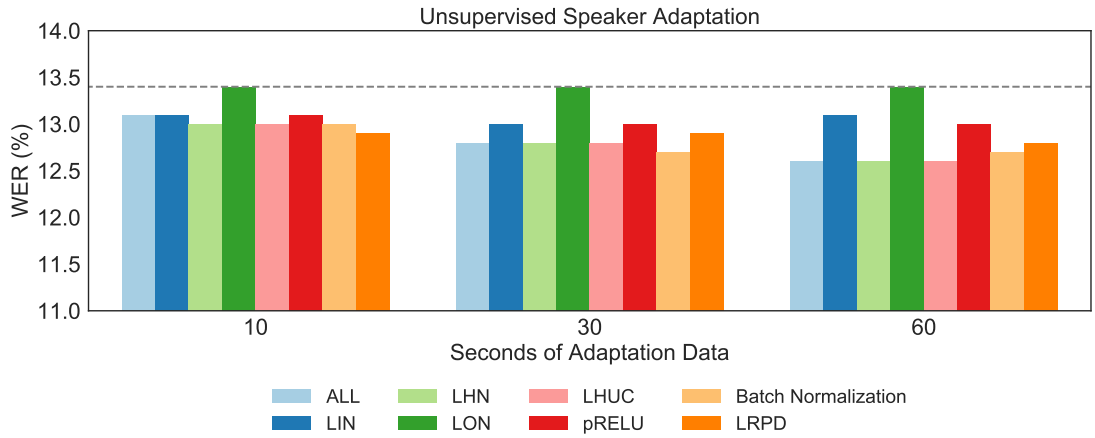

Figure 3.5: WER (%) for a comparison of various adaptation methods in unsupervised speaker adaptation. The dashed line shows the performance of the unadapted model.

spectively. Since all the tested techniques are very sensitive to the choice of hyper-parameters of the adaptation procedure, we used the combined development set from IWSLT 2010 and 2012 to tune the adaptation procedure in order to provide a fair comparison of the techniques. Instead of using a hyper-parameter grid-search, we tuned the speaker adaptation procedure using the meta-learning approach, which will be described in Section 4.2, which can automatically learn an adaptation learning rate for each adapted layer. With this approach we automatically tuned a learning rate for each adapted layer.

In this experiment we compared the following methods: adaptation of the whole model, adaptation of LIN (Neto et al., 1995), LHN (Gemello et al., 2007), LON (Li and Sim, 2010), LHUC (Swietojanski and Renals, 2014), pRELU (Zhang and Woodland, 2015), Batch Normalisation (Wang and Wang, 2017) and LRPD (Zhao et al., 2016).

Method	Number of SD parameters
ALL	5.9M
LIN (Neto et al., 1995)	40.2k
LHN (Gemello et al., 2007)	2.2M
LON (Li and Sim, 2010)	17.7M
LHUC (Swietojanski et al., 2016)	3.6k
Parameterised ReLU (Zhang and Woodland, 2015)	7.2k
Batch Normalisation (γ, β) (Wang and Wang, 2017)	7.2k
LRPD (Zhao et al., 2016) with $k = 10$	75.6k

Table 3.1: Summary of the number of speaker dependent parameters for various techniques.

In Table 3.1 we list the number of speaker dependent parameters that were adapted in each experiment. We adapted all the hidden layers (except for LIN and LON that are adapting only the input or output, respectively).

The results for these experiments are plotted in Figure 3.4 for supervised and Figure 3.5 for unsupervised speaker adaptation. As we can see, there are significant differences between the methods. In supervised speaker adaptation the best results are achieved by adapting the whole network, followed by LHN adaptation, LHUC adaptation and LRPD adaptation. This can be explained by the fact that for supervised adaptation a higher number of speaker dependent parameters is beneficial. However, the meta-learner learned not to perform LON adaptation, because it contains too many parameters that cannot be reliably estimated using the limited amounts of adaptation data. In the unsupervised adaptation experiment the results are more even, which can be explained by the fact the meta-learner learns more conservative learning rates, because it does not want to overfit to errors in the labels.

3.6 Regularisation Methods for Speaker Adaptation

As we mentioned before, model-based speaker adaptation is prone to overfitting to the adaptation data. In the previous section we discussed various ways of limiting the number of speaker dependent parameters which reduces the capacity of the acoustic model and thus prevents overfitting to the adaptation data. In this section we are go-

ing to describe other regularisation approaches, which can be roughly divided into two groups. The first group uses regularisation methods to prevent the model from diverging too far from the original model. The second alters the loss function to obtain better gradients for adaptation.

Preventing the adapted model from diverging too far from the original can be achieved in various ways, depending on how we measure the distance between the original and the adapted model. Liao (2013) proposed to use the L2 regularisation loss of the distance between the original speaker dependent parameters θ_s and the adapted speaker dependent parameters θ'_s

$$\mathcal{L}_{L2} = |\theta_s - \theta'_s|_2^2. \tag{3.23}$$

Yu et al. (2013) proposed to use Kullback-Leibler (KL) divergence to measure the distance between the senone distribution of the adapted model and the senone distribution of the original model

$$\mathcal{L}_{KL} = D_{KL}(f(x;\theta)||f(x;\theta,\theta'_s)). \tag{3.24}$$

If we consider the overall adaptation loss:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{xent} + \lambda\mathcal{L}_{KL}, \tag{3.25}$$

we can show that this loss equals to cross-entropy with the target distribution

$$P(Y\,|\,X) = (1-\lambda)\hat{P}(Y\,|\,X) + \lambda f(x;\theta), \tag{3.26}$$

where $\hat{P}(Y\,|\,X)$ is a distribution corresponding to the provided labels y^{adapt}. A similar approach called *Conservative Training* was presented by Gemello et al. (2007). This approach modified the target distribution as follows

$$p(y_i|x) = \begin{cases} f(x;\theta)_i & \text{if } y_i \in U \\ 1 - \sum_{y_j \in U} f(x;\theta)_j & \text{if } y_i \in S \;\&\; \text{correct}(y_i|x) \\ 0 & \text{if } y_i \in S \;\&\; \neg\text{correct}(y_i|x) \end{cases}, \tag{3.27}$$

where S is a set of labels seen in the adaptation data and U is a set of labels not seen in the adaptation data. This modified target distribution ensures that labels not seen in the adaptation data will not be catastrophically forgotten.

Meng et al. (2019) noted that KL divergence is not a perfect distance metric between distributions because it is asymmetric. Therefore, they proposed to use adversarial learning which guarantees that the local minimum is reached only if the senone

distributions of the speaker independent and the speaker dependent models are identical. They achieve this by adversarially training a discriminator $d(x;\phi)$ whose task is to discriminate between the activations of the i-th hidden layer of a speaker dependent $f(x;\theta,\theta'_s)$ and a speaker independent model $f(x;\theta)$. The regularisation loss of the discriminator is

$$\mathcal{L}_{disc} = -\frac{1}{T}\sum_{t=1}^{T}\left\{\log d(f_i(x_t;\theta);\phi) - \log\left[1 - d(f_i(x_t;\theta,\theta'_s);\phi)\right]\right\}. \tag{3.28}$$

The discriminator is trained in a minimax fashion during adaptation by minimising $\mathcal{L}_{disc}$ with respect to ϕ and maximising $\mathcal{L}_{disc}$ with respect to θ_s. Consequently, the distribution of activations of the i-th hidden layer of the speaker dependent model will be indistinguishable from the distribution of activations of the i-th hidden layer of the speaker independent model, which ought to result in more robust performance of speaker adaptation.

Other approaches try to prevent overfitting by leveraging the uncertainty of the speaker-dependent parameter space. Huang et al. (2015b) proposed Maximum A Posteriori (MAP) adaptation of neural networks by taking an inspiration from MAP adaptation of GMM-HMM models (Gauvain and Lee, 1994). MAP adaptation estimates speaker dependent parameters as a mode of the distribution

$$\theta_s = \arg\max_{\theta_s} P(Y|X,\theta,\theta_s)p(\theta_s), \tag{3.29}$$

where $p(\theta_s)$ is a prior density of the speaker dependent parameters. In order to obtain this prior density, Huang et al. (2015b) employed an empirical Bayes approach and treated each speaker in the training data as a data point. They performed speaker adaptation for each speaker and noticed that the speaker parameters across speakers resemble Gaussians. Therefore they decided to parameterise the prior density $p(\theta_s)$ as

$$p(\theta_s) = \mathcal{N}(\mu,\Sigma), \tag{3.30}$$

where μ is the mean of adapted speaker dependent parameters across different speakers, and Σ is the corresponding diagonal covariance matrix. With this parameterisation the regularisation term of the prior density $p(\theta_s)$ is

$$\mathcal{L}_{MAP} = \frac{1}{2}(\theta_s - \mu)^T\Sigma(\theta_s - \mu), \tag{3.31}$$

which for the prior density $p(\theta_s) = \mathcal{N}(0,I)$ degenerates to the $\mathcal{L}_{L2}$ regularisation loss.

Xie et al. (2019b) proposed a fully Bayesian way of dealing with uncertainty inherent in speaker dependent parameters θ_s. More concretely they proposed a fully

Bayesian way of estimating LHUC parameters (BLHUC). In standard LHUC, speaker dependent parameters θ_s are estimated using maximum likelihood

$$\hat{\theta}_s = \arg\max_{\theta_s} P(Y|X,\theta_s), \tag{3.32}$$

which may deteriorate results when only a small amount of adaptation data is available. Therefore, they decided to model the uncertainty of the speaker dependent parameters $\theta_s \sim p(\theta_s)$, where $p(\theta_s)$ is a prior distribution. They approximate the posterior distribution of the adapted model as:

$$\begin{aligned} P(Y|X,\mathcal{D}^{\text{adapt}}) &= \int P(Y|X,\theta_s)P(\theta_s|\mathcal{D}^{\text{adapt}})d\theta_s \\ &\approx P(Y|Z,\mathbb{E}[\theta_s|\mathcal{D}^{\text{adapt}}]). \end{aligned} \tag{3.33}$$

They proposed to use a distribution $q(\theta_s)$ as a variational approximation of the posterior distribution $P(\theta_s|\mathcal{D}^{\text{adapt}})$ and the following loss to optimise it

$$\begin{aligned} \mathcal{L} &= -\log P(y^{\text{adapt}}|x^{\text{adapt}}) \\ &= -\log \int P(y^{\text{adapt}}|x^{\text{adapt}},\theta,\theta_s)P(\theta_s)d\theta_s \\ &\leq -\int q(\theta_s)\log P(y^{\text{adapt}}|x^{\text{adapt}},\theta,\theta_s)d\theta_s + KL(q||p). \end{aligned} \tag{3.34}$$

For simplicity, they assumed that both $q(\theta_s)$ and $p(\theta_s)$ are normal distributions, such that $q(\theta_s) = \mathcal{N}(\mu_s,\gamma_s)$ and $p(\theta_s) = \mathcal{N}(\mu_p,\gamma_p)$. Which means that the expected value of the speaker dependent parameters in Equation 3.33 is

$$\mathbb{E}[\theta_s|\mathcal{D}^{\text{adapt}}] = \mu_s. \tag{3.35}$$

The KL divergence term can be computed as

$$KL(q||p) = \frac{1}{2}\left\{\frac{(\mu_s-\mu_p)^2+\sigma_s^2}{\sigma_p^2} - \log\frac{\sigma_s^2}{\sigma_p^2} - 1\right\}. \tag{3.36}$$

Finally, the integral in Equation 3.34 can be computed using Monte Carlo approximation with J Monte Carlo samples $\varepsilon \sim \mathcal{N}(0,I)$ as

$$\begin{aligned} &\int q(\theta_s)\log P(y^{\text{adapt}}|x^{\text{adapt}},\theta,\theta_s)d\theta_s \\ &= \int \mathcal{N}(\varepsilon;0,I)\log P(y^{\text{adapt}}|x^{\text{adapt}},\mu_s + exp(\gamma_s)\otimes\varepsilon)d\varepsilon \\ &\approx \frac{1}{J}\sum_{j=1}^{J}\log P(y^{\text{adapt}}|x^{\text{adapt}},\theta_s,\varepsilon_j). \end{aligned} \tag{3.37}$$

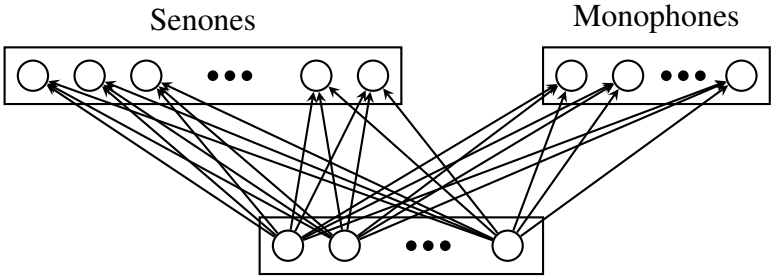

Figure 3.6: Multi-task learning speaker adaptation.

There are several ways to choose the prior density $p(\theta_s)$. Either we can set it to $\mathcal{N}(1, I)$ or we can estimate μ and Σ in the same way as in the MAP adaptation approach. Similarly to MAP adaptation, this will force the adaptation to stay close to the speaker independent model when we perform adaptation with a small amount of adaptation data.

The second group of regularisation approaches proposed to use a lower entropy task such as monophone or senone cluster targets. This has the advantage that the unsupervised targets might be less noisy and also that the targets have higher coverage even with small amounts of adaptation data. Price et al. (2014) proposed to append a new output layer predicting monophone targets on top of the original output layer predicting senones. The layer can be either full rank or sparse – leveraging knowledge of relationships between monophones and senones. Its parameters are trained on the training data with a fixed speaker independent model. Only the mohophone targets are used for the adaptation of the speaker dependent parameters.

Huang et al. (2015a) presented an approach that used the multi-task learning approach (Caruana, 1997) to leverage both senone and monophone/senone clusters targets. It worked by having multiple output layers, each on top of the last hidden layer, that predicted the corresponding targets (Figure 3.6). These additional output layers were also trained after a complete training pass of the speaker independent model with its parameters fixed. Thus, the adaptation loss was a weighted sum of individual losses:

$$\mathcal{L} = \sum_{i=0}^{n} \mathcal{L}_i. \tag{3.38}$$

Swietojanski et al. (2015) combined these two approaches and used multi-task learning for speaker adaptation through a structured output layer, which predicts both monophone targets and senone targets. But unlike the approach by Price et al. (2014), the monophone predictions are used for the prediction of senones, as illustrated in Figure 3.7.

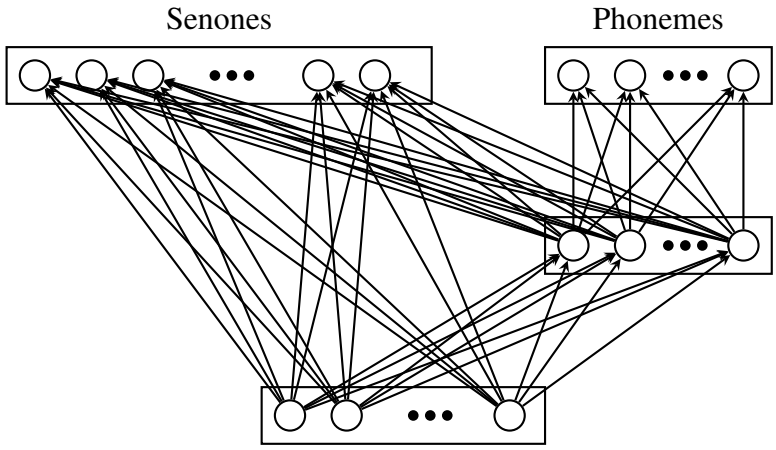

Figure 3.7: Structured output layer by Swietojanski et al. (2015).

3.7 Speaker Adaptive Training

In this chapter we introduced various test-time speaker adaptation methods that reduce the mismatch between training and testing conditions. Speaker adaptive training (SAT) applies these methods also during training of the model in order to allow the acoustic model to focus solely on modelling phonetic variations (Anastasakos et al., 1996). For methods using global statistics or an external model to estimate speaker dependent parameters SAT of DNN models is straightforward. These include VTLN features (Trmal et al., 2010), cMLLR features (Sainath et al., 2013) and auxiliary features (Saon et al., 2013; Miao et al., 2015). However, the situation becomes more complicated when we want to apply SAT with methods using discriminative estimation of speaker dependent parameters such as SAT-LHUC (Swietojanski and Renals, 2016) or speaker codes (Abdel-Hamid and Jiang, 2013).

We will illustrate the training procedure on SAT-LHUC, but the same procedure can be used for speaker codes or any other model-based speaker adaptation method. Recall that in test-time speaker adaptation using LHUC parameters r_s we update the scale of hidden layer activations:

$$h' = r_s \odot h. \tag{3.39}$$

In SAT training using LHUC parameters we maintain a copy of speaker dependent parameters r_s for each speaker $\mathcal{S}$ in the training data

$$\theta_s = \left\{ r_s^i | i \in \{1, \ldots, S\} \right\}. \tag{3.40}$$

Since we know speaker identities for all utterances in the training data, we can jointly

optimise speaker independent and speaker dependent parameters directly during training using the cross-entropy loss

$$\theta, \theta_s = \arg\max_{\theta,\theta_s} - \sum_{x,y,i} \log P(y|x, \theta, r_s^i). \tag{3.41}$$

In order to be able to use this model for a new unseen speaker, we first need to obtain LHUC parameters r_s for the speaker by performing speaker adaptation. To do that we also require a speaker independent model to obtain labels for the adaptation data either by aligning a reference transcript or by a first pass decoding. To remove the need for an additional speaker-independent model Swietojanski and Renals (2016) introduced an artificial speaker into the training data in order to estimate speaker-independent LHUC parameters. This was done by assigning data to the artificial speaker with probability $p \sim Bernoulli(\lambda)$, where the parameter λ controlled how much data was used for the training of speaker dependent and speaker independent LHUC parameters respectively.

Performing SAT like this is complicated because we need to maintain speaker dependent parameters for each speaker in the training data which has a considerable memory footprint. Also, if we have too many speaker dependent parameters we need much more data for each speaker to reliably estimate speaker dependent parameters. And finally, when we use too many speaker dependent transformations the training of the model becomes slower compared to the training of speaker independent models, because applying different speaker dependent transformations is computationally more expensive than applying speaker independent transformations.

3.8 Summary

In this section we described methods for speaker adaptation and speaker adaptive training of neural network based acoustic models. Since these acoustic models are prone to overfitting to the adaptation data due to their large modelling capacity, most of the methods described in this chapter focused on preventing this overfitting issue either by limiting the number of speaker dependent parameters (Section 3.5), or by using appropriate regularisation terms (Section 3.6). However, the main disadvantage of these methods is that they require carefully tuned hyperparameters to obtain the best possible results. Therefore, in the next chapter we review meta-learning approaches and we show how meta-learning can be used to automatically obtain a robust speaker adaptation procedure that is less susceptible to the aforementioned overfitting issues (Section 4.2).

Chapter 4

Learning to Adapt

In the previous chapter we described methods for speaker adaptation of DNN-HMM acoustic models. As we have seen, there has been a lot of effort into making discriminative model-based speaker adaptation robust to overfitting to adaptation data, especially in scenarios in which we have only limited amounts of adaptation data or in unsupervised speaker adaptation when the labels might be erroneous. However, all the methods described in the previous chapter require careful tuning of hyperparameters in order to obtain the best possible results. In this chapter, which is the main theoretical contribution of this thesis, we first describe meta-learning and then we propose using meta-learning to automatically learn robust speaker adaptation procedures for discriminative model-based speaker adaptation. We empirically test the proposed meta-learning approach for speaker adaptation and speaker adaptive training later in Chapters 5 and 6 respectively.

As we described earlier, there are three main challenges for discriminative model-based speaker adaptation. First, we do not want to overfit to senones seen in the adaptation data and forget unseen classes. This is sometimes called catastrophic forgetting. Second, in unsupervised speaker adaptation we do not want to overfit to errors made in the first pass decoding by the seed model. Third, since the amount of adaptation data is limited it is not fully representative of a speaker, therefore the speaker adaptation might overfit to particular speaker characteristics. The main proposed solutions for solving these challenges were to use conservative learning schedules, limiting the number of speaker dependent parameters and to use regularisation methods that restrict the speaker-dependent model from diverging too far from the speaker-independent model. However, we believe that by doing this, especially by limiting the number of speaker-dependent parameters, we inherently restrict the expressivity of

the adaptation procedure which might not lead to the best possible performance of the adapted model. Moreover, even if we decide to apply all these techniques to prevent the aforementioned issues, we still need to tune an adaptation rule that will be used for the adaptation of the speaker-dependent parameters, because an adaptation rule with poor hyperparameters might also lead to inferior adaptation performance. The discriminative adaptation rule,

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{adapt}}), \tag{4.1}$$

updates parameters θ of an acoustic model $f(x;\theta)$ by computing gradients corresponding the loss $\mathcal{L}(\theta, \mathcal{D}^{\text{adapt}})$ computed on the adaptation data $\mathcal{D}^{\text{adapt}}$. We argue that even such a simple adaptation rule contains many hidden hyperparameters that need to be tuned to obtain a robust speaker adaptation procedure:

1. We need to determine which parameters θ should be treated as speaker dependent. This can also be seen as deciding which hidden layers should be adapted or how inserted linear hidden networks should be parameterised (Section 3.5).

2. We need a reliable learning rate schedule α that will not overfit to the adaptation data, but that will maximise the performance of the adapted model on unseen test data.

3. We need to select a suitable loss function $\mathcal{L}(\theta, \mathcal{D}^{\text{adapt}})$, the type of supervision for the computation of the loss and also consider the use of regularisation methods (Section 3.6).

4. In the unsupervised setting we want to avoid training on segments with too many errors from the first pass decoding. Therefore, we might need to use some filtering pipeline that will filter out segments with too low confidence. At its simplest, we need to tune a cut-off confidence threshold for this pipeline.

5. In most scenarios the update rule is applied multiple times, therefore we need to determine how many update steps to make. This also includes ways of employing some form of early stopping.

Tuning all these hyperparameters manually would require considerable effort, but we believe that finding a robust speaker adaptation procedure is worthwhile, because speaker adaptation will be performed for many speakers in many different environments and might have a big impact on the performance of the final model. Therefore,

we decided to explore ways of learning robust speaker adaptation procedures automatically. We call this endeavour *learning to adapt*. It is inspired by meta-learning approaches that are *learning to learn* (Thrun and Pratt, 1998; Hochreiter et al., 2001; Andrychowicz et al., 2016). We explain how meta-learning works and review work done in this field in the next section. Then we show how speaker adaptation can be formulated as a meta-learning task. Subsequently we show that speaker adaptive training is a straightforward extension of this approach. Finally, we discuss different ways of implementing speaker adaptation as a meta-learning task by using different parameterisations of the update rule in Equation 4.1.

4.1 Meta-Learning

In meta-learning, learning is happening on two levels (Younger et al., 2001) – a *learner* is learning a model for specific tasks and a *meta-learner* is learning regularities between tasks in order to better train the *learner* on new tasks. The *learner* is a function $f_{\mathcal{T}}(x) = y$ that maps inputs x to outputs y for a specific task $\mathcal{T} = (\mathcal{D}_{\mathcal{T}}^{\text{train}}, \mathcal{D}_{\mathcal{T}}^{\text{test}})$. Each task $\mathcal{T}$ consists of training examples $\mathcal{D}_{\mathcal{T}}^{\text{train}} = \{(x_j, y_j) | j \in \{1 \dots m\}\}$ for the model $f_{\mathcal{T}}(x)$ and testing examples $\mathcal{D}_{\mathcal{T}}^{\text{test}} = \{(x_j, y_j) | j \in \{1 \dots m\}\}$ which are used to evaluate the model with some loss function $\mathcal{L}_{\mathcal{T}}$

$$\mathcal{L}_{\mathcal{T}}(\theta, \mathcal{D}) = \sum_{(x_j, y_j) \in \mathcal{D}} L(y_j, f_{\mathcal{T}}(x_j; \theta)). \tag{4.2}$$

This loss is a sum of losses for tuples in the data $\mathcal{D}$. For instance we can use the cross-entropy loss

$$L(\hat{y}, y) = -\sum_{c=1}^{C} [\hat{y}]_c \log [y]_c \,, \tag{4.3}$$

where $\hat{y}$ is a gold-truth label represented as a one-hot vector and y is the prediction of the learner. In machine learning we find the mapping from the input space to the output space by parameterising the *learner* with parameters θ

$$f_{\mathcal{T}}(x) \equiv f_{\mathcal{T}}(x; \theta). \tag{4.4}$$

During training of the learner we aim to find the optimal parameters $\hat{\theta}$ that minimise the loss function $L_{\mathcal{T}}$ computed on the training data $D_{\mathcal{T}}^{\text{train}}$

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}_{\mathcal{T}}(\theta, \mathcal{D}_{\mathcal{T}}^{\text{train}}), \tag{4.5}$$

but the goal is to train a *learner* that generalises well on the test data $\mathcal{D}_{\mathcal{T}}^{\text{test}}$. Therefore in practice we hold-out a portion of the training data as a validation data and we use an early stopping criterion estimated on this validation data to prevent the learner from overfitting to the training data.

The *meta-learner* aims to learn a learning algorithm $g(f_{\mathcal{T}_i}, \mathcal{T}_i)$ for the task-specific learner $f_{\mathcal{T}_i}(x;\theta)$ using a distribution of tasks $\mathcal{T}_i \sim p(\mathcal{T})$. In the literature there are two ways of representing the *meta-learner*. In the first one, the meta-learner uses the task specific training data $\mathcal{D}_{\mathcal{T}_i}^{\text{train}}$ to predict task dependent parameters for the learner $\theta_{\mathcal{T}_i}$ (Hochreiter et al., 2001; Andrychowicz et al., 2016; Ravi and Larochelle, 2017; Finn et al., 2017):

$$g_1(f_{\mathcal{T}_i}, \mathcal{D}_{\mathcal{T}_i}^{\text{train}}) = \theta_{\mathcal{T}_i}, \tag{4.6}$$

which can then be used to make predictions about the test data $\mathcal{D}_{\mathcal{T}_i}^{\text{test}}$. In the second way, the meta-learner uses the task specific training data and inputs for the test data $X_{\mathcal{T}_i}^{\text{test}}$ to directly make predictions $Y_{\mathcal{T}_i}^{\text{test}}$ (Vinyals et al., 2016; Santoro et al., 2016; Mishra et al., 2018)

$$g_2(\mathcal{D}_{\mathcal{T}_i}^{\text{train}}, X_{\mathcal{T}_i}^{\text{test}}) = Y_{\mathcal{T}_i}^{\text{test}}. \tag{4.7}$$

It can be shown that these representations are interchangeable for a certain choice of g_1 and g_2

$$f_{\mathcal{T}_i}(X_{\mathcal{T}_i}^{\text{test}}; g_1(f_{\mathcal{T}_i}, \mathcal{D}_{\mathcal{T}_i}^{\text{train}})) = g_2(\mathcal{D}_{\mathcal{T}_i}^{\text{train}}, X_{\mathcal{T}_i}^{\text{test}}). \tag{4.8}$$

This is only true when the meta-learner g_2 does not use $X_{\mathcal{T}_i}^{\text{test}}$ to update its decision boundaries. In the rest of the thesis we will use the first representation g_1, because it better corresponds to the speaker adaptation setting, where we have access only to adaptation data and we cannot leverage unseen test data. For clarity we will denote the *meta-learner* as $g(f_{\mathcal{T}_i}, \mathcal{D}_{\mathcal{T}_i}^{\text{train}})$.

Similar to the *learner*, we can parameterise the *meta-learner* with parameters ϕ:

$$g(f_{\mathcal{T}_i}, \mathcal{D}_{\mathcal{T}_i}^{\text{train}}) \equiv g(f_{\mathcal{T}_i}, \mathcal{D}_{\mathcal{T}_i}^{\text{train}}; \phi). \tag{4.9}$$

We aim to find the parameters $\hat{\phi}$ that minimise the expected loss over a set of tasks $\mathcal{T}_i \sim p(\mathcal{T})$

$$\hat{\phi} = \arg\min_{\phi} \mathop{\mathbb{E}}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(g\left(f_{\mathcal{T}_i}, \mathcal{D}_{\mathcal{T}_i}^{\text{train}}; \phi\right), \mathcal{D}_{\mathcal{T}_i}^{\text{test}}\right). \tag{4.10}$$

In plain words, the *meta-learner* is learning to learn from task specific training data to minimise loss on task specific test data. In practice, the meta-learner is trained on a *meta-training set* $\mathcal{M}^{\text{train}} = \{\mathcal{T}_i\}$ and tested on a *meta-testing set* $\mathcal{M}^{\text{test}} = \{\mathcal{T}_j\}$. Traditionally there is no overlap of tasks between the *meta-training set* and the *meta-testing*

set, which enables us to measure how well the *meta-learner* can transfer knowledge to unseen tasks, and thus how well it is *learning to learn*.

Thrun and Pratt (1998) define the ability of *learning to learn* as follows. Given a task $\mathcal{T}_i$ from a distribution of tasks $\mathcal{T}_i \sim p(\mathcal{T})$, training data $\mathcal{D}_{\mathcal{T}_i}^{\text{train}}$ and a corresponding loss function $\mathcal{L}_{\mathcal{T}_i}$, a learning algorithm is said to be able to *learn to learn* if its performance $\mathcal{L}_{\mathcal{T}_i}$ on each task $\mathcal{T}_i$ is expected to improve with an increasing amount of training data for each task and an increasing number of tasks $\mathcal{T}_i$. Therefore, the meta-learner needs to be able to transfer knowledge between different tasks, for example by learning and using high-level regularities between different tasks. Thrun and Pratt discuss that one way to achieve this is to represent the learner as a composition of a task-dependent learner f_{TD} and a task-independent learner f_{TI}, such that the original learner is for example $f = f_{TD} \circ f_{TI}$ or $f = f_{TI} \circ f_{TD}$.[1] The first composition corresponds to having a shared feature extractor and learning only task specific classifiers and the second composition corresponds to having task-dependent feature extractors and a task-independent classifier. In ASR, the first composition is used for transfer learning and it has been applied for multilingual training of ASR models (Ghoshal et al., 2013; Heigold et al., 2013; Huang et al., 2013) or even LON adaptation (Li and Sim, 2010). The second composition can be compared to feature-based speaker adaptation approaches described in the previous chapter in Section 3.1. Alternatively, we can also use meta-learning to train a task specific learning algorithm.

Learning learning algorithms has a long tradition. Schmidhuber (1992, 1993) presented an approach that allowed neural networks to update their parameters by embedding the learning algorithm directly into the network and allowing both the neural network and the learning algorithm to be trained jointly. Cotter and Conwell (1990); Younger et al. (1999) demonstrated that even neural networks with fixed weights showcase learning abilities. This is achieved by having a recurrent neural network with fixed weights that accumulates gradient descent updates for parameters of some classifier inside its hidden state. This enables evaluating the classifier with different parameters at different steps, and thus it allows on-line learning with fixed weights. Subsequently, Younger et al. (2001); Hochreiter et al. (2001) parameterised the update rule as a Long Short-Term Memory network that learned to update parameters of the learner based on corresponding gradients computed from the previous predictions. The same approach was recently used by Andrychowicz et al. (2016) for the learning of task specific learning rules and by Ravi and Larochelle (2017) for the learning of learning

[1] By function composition we understand $(f \circ g)(x) = f(g(x))$.

rules for few-shot learning. All these models were able to train both the learner and the meta-learner end-to-end, which is the reason why Andrychowicz et al. (2016) call these approaches *learning to learn by gradient descent by gradient descent*. Bengio et al. (1995, 1990); Runarsson and Jonsson (2000) proposed methods that learn simple, biologically plausible parametric learning rules for neural networks using genetic programming or simulated annealing. Therefore, Andrychowicz et al. (2016) calls these methods *learning to learn by gradient descent* ***without*** *gradient descent*.

Learning to learn has also been successfully applied to few-shot learning. The goal of few-shot learning is to rapidly train a classifier for a new task from only a few examples for each target class. For example, our goal might be to train an image classifier that is supposed to predict five different classes but using only ten examples of each class. In order to succeed, the meta-learner needs to learn regularities between various tasks. There has been many different approaches using meta-learning for few-shot learning (Vinyals et al., 2016; Ravi and Larochelle, 2017; Finn et al., 2017; Snell et al., 2017). They usually focus on few-shot image classification on Omniglot (Lake et al., 2011) and MiniImageNet (Russakovsky et al., 2015; Ravi and Larochelle, 2017).

In the rest of this section we describe work done by Andrychowicz et al. (2016); Ravi and Larochelle (2017); Finn et al. (2017) as we used it as a basis for the thesis. Andrychowicz et al. (2016) proposed a neural network based implementation of the meta-learner $g(f, \mathcal{D}^{\text{train}}; \phi)$ (Equation 4.9). With this implementation it is possible to learn task specific update rules for the parameters of the learner $f(x; \theta)$. In order to learn the update rule, the meta-learner needs the gradients of a loss function $\mathcal{L}(\theta, \mathcal{D}^{\text{train}})$ computed on data $\mathcal{D}^{\text{train}}$. We will use a shorter notation for these gradients

$$\nabla_\theta \mathcal{L} = \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{train}}). \tag{4.11}$$

It then passes these gradients $\nabla_\theta \mathcal{L}$ to a multi-layer LSTM network with parameters ϕ, whose output is then used to predict task dependent parameters of the learner $f(x; \theta)$

$$g(f, \mathcal{D}^{\text{train}}; \phi) \equiv \theta - LSTM(\nabla_\theta \mathcal{L}; \phi). \tag{4.12}$$

By using the LSTM network the model can learn a learning rate schedule and even learn to use momentum (Nesterov, 1983), because the network has access to all previous gradients. However, the problem with this approach is that it does not scale to larger networks with at least tens of thousands of parameters, because it would require having a large input to the network and a large hidden state. Thus, the meta-learner would need unfeasibly large weight matrices. Consequently, Andrychowicz

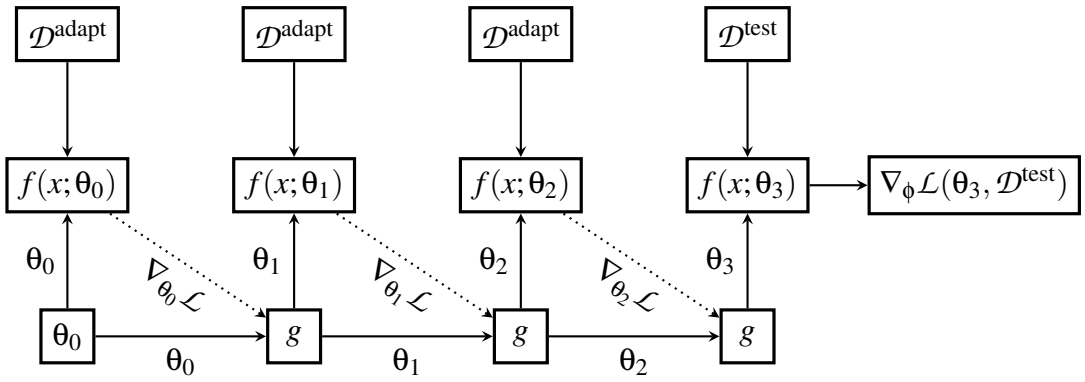

Figure 4.1: A diagram of a computation graph for the coordinate-wise meta-learner performing 3 adaptation steps using the adaptation data $\mathcal{D}^{\text{adapt}}$. Note that θ_i denotes the weights θ at a time-step i. To update the parameters of the meta-learner, we need to follow the solid lines when computing gradients using the first order approximation. In the full version using second order derivatives we also need to follow the dotted lines.

et al. (2016) proposed to learn the update rule for individual parameters, similar to how commonly used update rules such as Adam (Kingma and Ba, 2014) work. Therefore, the meta-learner learns the following rule

$$\left[\theta'\right]_i = [\theta]_i - LSTM([\nabla_\theta \mathcal{L}]_i; \phi), \tag{4.13}$$

where the notation $[\theta]_i$ denotes the i-th element of the vector θ. In order to adapt the whole network, its parameters are presented as a single batch of inputs to the meta-learner. The parameters are jointly updated for several steps. Figure 4.1 illustrates how this process works.

Subsequently, Ravi and Larochelle (2017) proposed an update rule parameterisation inspired by how LSTM networks update their hidden states. They used a two-layer LSTM to predict two values – an input gate i and a reset gate r, that were used in the following update rule:

$$\theta' = r\theta - i\nabla_\theta \mathcal{L}. \tag{4.14}$$

Because the LSTM used the current parameter value θ, the current loss $\mathcal{L}$ and the corresponding gradient $\nabla_\theta \mathcal{L}$, it was able to learn momentum via the input gate i and also to escape local minima by resetting parameters via the reset gate r, when the gradient $\nabla_\theta \mathcal{L}$ is small and the loss $\mathcal{L}$ is high. We describe the coordinate-wise meta-learner proposed by Ravi and Larochelle (2017) in more detail in Section 4.4 and we use it for speaker adaptation experiments in Chapter 5. It is important to note that the training of the meta-learner requires the computation of second order derivatives, because

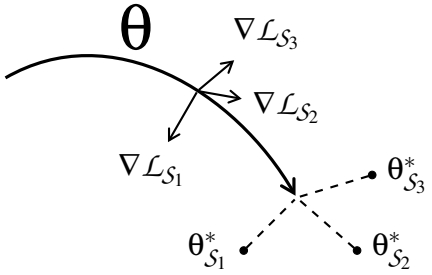

Figure 4.2: Speaker adaptive training using meta-learning steers the training process to find initial parameters θ that are suitable for rapid adaptation using adaptation data corresponding to speakers $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. The solid lines correspond to meta-training and the dashed lines correspond to speaker adaptation.

the updated parameters θ' which are used for the computation of the meta-learner's loss depend on the meta-learners parameters ϕ. See Figure 4.1 for an example of the gradient flow in an unrolled meta-learner. However, both Andrychowicz et al. (2016) and Ravi and Larochelle (2017) argued that using only first order derivatives achieves similar results as when using second order derivatives while being significantly faster to train. Therefore, in our experiments we also only used first order derivatives.

Finn et al. (2017) proposed an alternative approach called Model-Agnostic Meta-Learning (MAML), which aims to find learner parameters that would allow for fast adaptation to new tasks via a few steps of gradient descent in the context of few-shot learning. The motivation is illustrated in Figure 4.2. The process is the same as in Equation 4.10, except that we are not learning parameters of the adaptation function, but we are training the initialisation of the learner θ_0. Thus, the meta-learner parameters are $\phi = \{\theta_0\}$. The meta-learner function $g(f, \mathcal{D}^{\text{train}}_{\mathcal{T}_i}; \phi)$ is parameterised as a simple gradient descent rule

$$g(f, \mathcal{D}^{\text{train}}_{\mathcal{T}_i}; \phi) = \theta_0 - \alpha \nabla_{\theta_0} \mathcal{L}, \tag{4.15}$$

where α is a predefined fixed learning rate. Antoniou et al. (2019) showed that by jointly optimising learner initialisation θ_0 and a learning rate for the update rule α ($\phi = \{\theta_0, \alpha\}$), it is possible to achieve better performance than when keeping the learning rate fixed during training. Finn and Levine (2018) provided a proof showing that a sufficiently deep learner trained with embedded gradient descent can approximate any learning algorithm, such as those using LSTM networks to represent the update rule. Note that MAML was also recently used for low-resource neural machine translation (Gu et al., 2018) and low-resource end-to-end ASR (Hsu et al., 2019). Nichol and Schulman (2018) presented a similar approach called Reptile relying only on first

order derivatives. We describe an implementation of MAML in Section 4.5 and we use it in the speaker adaptive training experiments described in Chapter 6.

4.2 Speaker Adaptation as a Meta-Learning Task

In the previous section we described how meta-learning works. In this section we are going to describe the main contribution of this thesis – we show how speaker adaptation can be formulated as a meta-learning task. First, note that speaker adaptation has many similarities with few-shot learning. Like few-shot learning that uses task specific adaptation data to produce a task specific classifier, speaker adaptation uses the adaptation data to adapt a speaker-independent model to produce a speaker-dependent model. Further, we usually do not train the speaker-dependent model from scratch, because that would require a considerable amount of speaker-dependent training data. However unlike in few-shot learning, ASR models usually model thousands or tens of thousands of classes and we do not have training examples for all classes in the adaptation data. As mentioned before this might lead to catastrophic forgetting of unseen classes. Furthermore, the provided labels might be erroneous in the unsupervised setting. The adaptation algorithm therefore needs to take all this into account when adapting the speaker-independent model, which might already have a reasonable performance.

Speaker adaptation can be interpreted as a meta-learner that adapts the parameters of an acoustic model. Similarly to the meta-learner defined in Equation 4.6 it is a function, denoted as g_{adapt}, that given an acoustic model $f(x;\theta)$ and adaptation data

$$\mathcal{D}^{\text{adapt}} = \left\{ (x_j^{\text{adapt}}, y_j^{\text{adapt}}) | j \in \{1 \dots m\} \right\} \tag{4.16}$$

produces adapted parameters θ':

$$g_{\text{adapt}}(f_\theta, \mathcal{D}^{\text{adapt}}) = \theta' . \tag{4.17}$$

Depending on the scenario, the labels y_i^{adapt}, corresponding to the acoustic input x_i^{adapt}, might be obtained from a reference transcript (supervised adaptation) or obtained from a first pass decode of a speaker independent model (unsupervised adaptation). The performance of an acoustic model on unseen test data

$$\mathcal{D}^{\text{test}} = \left\{ (x_j^{\text{test}}, y_j^{\text{test}}) | j \in \{1 \dots n\} \right\}$$

is measured as the loss

$$\mathcal{L}(\theta, \mathcal{D}^{\text{test}}), \tag{4.18}$$

for example categorical cross-entropy, frame error rate or word error rate (WER). Note that the labels y_i^{test} are always obtained from the reference transcripts since we want to measure the true performance of the model. Similarly, we measure the loss of an adapted acoustic model by:

$$\mathcal{L}(g_{\text{adapt}}(f_\theta, \mathcal{D}^{\text{adapt}}), \mathcal{D}^{\text{test}}). \tag{4.19}$$

To train the adaptation function using the meta-learning approach, we require the function g_{adapt} to be both parametric and differentiable. We therefore add parameters ϕ to the adaptation function:

$$g_{\text{adapt}}(f_\theta, \mathcal{D}^{\text{adapt}}) \equiv g_{\text{adapt}}(f_\theta, \mathcal{D}^{\text{adapt}}; \phi). \tag{4.20}$$

We discuss different ways of parameterising the adaptation function in Section 4.4.

We are now ready to introduce the loss of the meta-learner. Recall that the goal of speaker adaptation is to adapt an acoustic model $f(x;\theta)$ using adaptation data $\mathcal{D}^{\text{adapt}}$ in order to improve performance on test data $\mathcal{D}^{\text{test}}$. The loss of the meta-learner can then be expressed as an expected loss for speakers $\mathcal{S}_i$ sampled from a distribution $\mathcal{S}_i \sim p(\mathcal{S})$:

$$J(\theta, \phi) = \mathop{\mathbb{E}}_{\mathcal{S}_i \sim p(\mathcal{S})} \mathcal{L}(g_{\text{adapt}}(f_\theta, \mathcal{D}_{\mathcal{S}_i}^{\text{adapt}}; \phi), \mathcal{D}_{\mathcal{S}_i}^{\text{test}}). \tag{4.21}$$

In theory, we should use an unlimited amount of unseen test data for the evaluation of the adaptation algorithm. However, this is not practical when training the meta-learner. We approximate it by using n seconds of speech as adaptation data and the following n seconds of speech as unseen data. Further, we use a *meta-training set* $\mathcal{M}^{\text{train}}$ and a *meta-testing set* $\mathcal{M}^{\text{test}}$ for the training and evaluation of the meta-learner, respectively. The meta sets are split such that they contain different speakers, so that we can correctly assess the generalisation of the adaptation function to unseen speakers.

Finally, we use the loss J to optimise the parameters ϕ of the adaptation function using gradient descent:

$$\hat{\phi} = \arg\min_{\phi} J(\theta, \phi)\,. \tag{4.22}$$

The training of the adaptation function then works as follow: in each iteration we sample a batch of S speakers with their corresponding adaptation data $\mathcal{D}_{\mathcal{S}_j}^{\text{adapt}}$ and test data $\mathcal{D}_{\mathcal{S}_j}^{\text{test}}$. We use the adaptation data to adapt the speaker-independent model $f(x;\theta)$ with the adaptation function $g_{\text{adapt}}(f_\theta, \mathcal{D}_{\mathcal{S}_j}^{\text{adapt}}; \phi)$. Subsequently the adapted parameters θ' are used to compute the loss on the test data $\mathcal{L}(\theta'_{\mathcal{S}_j}, \mathcal{D}_{\mathcal{S}_j}^{\text{test}})$. Finally, we sum the loss function across the speakers in the batch and we update the parameters of the adaptation function ϕ with gradients computed by differentiating the loss function. This is

Algorithm 1 Training of $g_{\text{adapt}}(f, \mathcal{D}^{\text{adapt}}; \phi)$

Require: Number of iterations N

Require: Number of speakers per batch S

Require: Trained acoustic model $f(x; \theta)$ with parameters θ

Require: Learning rate α

function TRAIN-ADAPT($f, \mathcal{M}^{\text{train}}$)
    $\phi \leftarrow$ random initialisation of the meta-learner
    for $i \in \{1 \cdots N\}$ **do**
        for $j \in \{1 \cdots S\}$ **do**
            $\mathcal{S}_j \sim p(\mathcal{S})$
            $J_{\mathcal{S}_j} \leftarrow \mathcal{L}(g_{\text{adapt}}(f_\theta, \mathcal{D}^{\text{adapt}}_{\mathcal{S}_j}; \phi), \mathcal{D}^{\text{test}}_{\mathcal{S}_j})$

        $J \leftarrow \sum J_{\mathcal{S}_j}$
        $\phi \leftarrow \phi - \alpha \nabla_\phi J$

return ϕ

repeated until convergence or for a predefined number of iterations. The training of the adaptation procedure is outlined in Algorithm 1.

4.3 Speaker Adaptive Training as a Meta-Learning Task

As we described in Section 3.7, speaker adaptive training is traditionally used to factor out speaker variation in order to enable the canonical acoustic model to fully focus on modelling phonological variations. In model-based speaker adaptive training a copy of the speaker-dependent parameters is maintained and jointly optimised for each speaker during training. However, this does not scale to speaker adaptive training of all parameters. Here, we propose an alternative meta-learning approach in which we embed gradient-based speaker adaptation directly into the acoustic model training. We hypothesise that this should steer the training process into finding parameters that are more amenable to speaker adaptation compared to parameters obtained through standard acoustic model training.

We described how to train a meta-learner for speaker adaptation above. To formulate speaker adaptive training as a meta-learning task, we need to jointly optimise the

Algorithm 2 Speaker Adaptive Training of $f(x;\theta)$ using Meta-Learning

Require: Number of iterations N

Require: Number of speakers per batch S

Require: Learning rate α

function SAT-META($f, \mathcal{M}^{\text{train}}$)
  $\theta \leftarrow$ random initialisation of the acoustic model
  $\phi \leftarrow$ random initialisation of the meta-learner
  for $i \in \{1 \cdots N\}$ **do**
    for $j \in \{1 \cdots S\}$ **do**
      $\mathcal{S}_j \sim p(\mathcal{S})$
      $J_{\mathcal{S}_j} \leftarrow \mathcal{L}(g_{\text{adapt}}(f_\theta, \mathcal{D}^{\text{adapt}}_{\mathcal{S}_j}; \phi), \mathcal{D}^{\text{test}}_{\mathcal{S}_j})$
    $J \leftarrow \sum J_{\mathcal{S}_j}$
    $\theta \leftarrow \theta - \alpha \nabla_\theta J$
    $\phi \leftarrow \phi - \alpha \nabla_\phi J$
return θ, ϕ

parameters of the acoustic model θ and the parameters of the meta-learner ϕ, minimising the loss J (Equation 4.21):

$$\hat{\theta}, \hat{\phi} = \arg\min_{\theta,\phi} J(\theta, \phi). \tag{4.23}$$

Speaker adaptive training using the meta-learning approach is very similar to the training of the adaptation function 4.2. The main difference is that we use the sum of the losses of the adapted models $\mathcal{L}(\theta'_{\mathcal{S}_j}, \mathcal{D}^{\text{test}}_{\mathcal{S}_j})$ to compute gradients with respect to the parameters of the speaker-independent acoustic model $f(x;\theta)$, because our goal is to find speaker-independent parameters that are suitable for fast speaker adaptation. We can keep the parameters of the adaptation function fixed during training or we can train them jointly with the speaker-independent parameter θ. Speaker adaptive training using the meta-learning approach (SAT-META) is outlined in Algorithm 2.

4.4 Implementation of Coordinate-wise Meta-Learner

In the previous two sections we described how speaker adaptation and speaker adaptive training can be formulated as meta-learning tasks. In this section we describe how the

adaptation function $g_{\text{adapt}}(f, \mathcal{D}^{\text{adapt}})$ can be parameterised as a *coordinate-wise meta-learner* (Ravi and Larochelle, 2017) to allow for the joint training of the learner and the meta-learner.

The coordinate-wise meta-learner updates each parameter $[\theta]_i$ individually – each parameter $[\theta]_i$ is presented as a single data sample to the meta-learner. Note, that in practice we batch all the parameters θ and we adapt them jointly. This has two advantages. First, the parameters of the meta-learner, ϕ, are tied for all parameters θ of the acoustic model. Second, the parameters of the meta-learner, ϕ, have much smaller dimensionality because they do not need to work with big inputs. In the following section we will describe how the meta-learner adapts a single parameter $[\theta]_i$ using the adaptation data $\mathcal{D}^{\text{adapt}}$.

The first layer of the coordinate-wise meta-learner is a standard LSTM which at time step t accepts an input $v_t \in \mathbb{R}^{3\times n}$ representing a batch of all the parameters $\theta \in \mathbb{R}^n$, with three columns: the current values of the parameters $\theta_t \in \mathbb{R}^n$, the current loss value L_t repeated n-times, and the corresponding gradients $\nabla_{\theta_t}\mathcal{L}_t \in \mathbb{R}^n$:

$$v_t = (\theta_t, \mathcal{L}_t, \nabla_{\theta_t}\mathcal{L}_t)\,. \tag{4.24}$$

The loss $\mathcal{L}_t$ is computed using adaptation data $\mathcal{D}^{\text{adapt}}$ and the current parameters θ_t:

$$\mathcal{L}_t = \mathcal{L}(f(x;\theta_t), \mathcal{D}^{\text{adapt}}) \tag{4.25}$$

and we initialise $\theta_1 = \theta$. Using this input v_t the first LSTM layer produces a hidden representation

$$h_t = LSTM(v_t). \tag{4.26}$$

The second LSTM layer uses this hidden representation h_t to predict the value of a reset gate r_t with parameters W_R and b_R:

$$r_t = \sigma(W_R \cdot [h_t, r_{t-1}] + b_R), \tag{4.27}$$

and to predict the value of an input gate i_t with parameters W_I and b_I:

$$i_t = \sigma(W_I \cdot [h_t, i_{t-1}] + b_I). \tag{4.28}$$

Both the reset gate r_t and the input gate i_t are used to update the parameter θ_t to θ_{t+1} using the corresponding gradient $\nabla_{\theta_t}\mathcal{L}_t$:

$$\theta_{t+1} = r_t \cdot \theta_t - i_t \cdot \nabla_{\theta_t}\mathcal{L}_t. \tag{4.29}$$

In this update rule the input gate acts as a learning rate scheduler, which can learn to use momentum (Nesterov, 1983) because of the LSTM recurrence, and the reset gate can be used to escape local minima when the loss is high but the gradient is close to zero (Ravi and Larochelle, 2017). Note that we initialised the bias of the input gate b_I to small values (sampled uniformly from $[-5,-4]$) and the bias of the reset gate b_R to large values (sampled uniformly from $[4,5]$) such that the meta-learner starts training with an update rule similar to SGD. Also note that the inputs v_t need to be preprocessed due to a big range of possible input values, which might hurt the learning of the meta-learner. Therefore, we followed Andrychowicz et al. (2016) and preprocessed the losses $\mathcal{L}_t$ with the following method using the suggested setting with $p = 10$:

$$\mathcal{L}_t \rightarrow \begin{cases} (\frac{log(|\mathcal{L}_t|)}{p}, sgn(\mathcal{L}_t)) & \text{if } |\mathcal{L}_t| \geq e^p \\ (-1, e^p \mathcal{L}_t), & \text{otherwise.} \end{cases} \tag{4.30}$$

We similarly preprocessed the gradients $\nabla_\theta \mathcal{L}_t$. We used the coordinate-wise meta-learner as an implementation of $g_{\text{adapt}}(f_\theta, \mathcal{D}^{\text{adapt}}; \phi)$ in the speaker adaptation experiments described in Chapter 5.

Even though presenting parameters θ as a big batch instead of one big input is much more computationally efficient, coordinate-wise meta-learner still requires a lot of memory and computation time to train. The memory requirements of the meta-learner can be estimated as

$$O(nmp), \tag{4.31}$$

where n is a number of adaptation steps, in our case we use 3 adaptation steps, m is the size of LSTM's hidden layer and p is the number of parameters of the acoustic model. Current acoustic models contain millions of parameters. Therefore, training a meta-learner even with a very small hidden layer quickly becomes infeasible, because the model would not fit into GPU memory.

Similarly we can estimate the time complexity of a single training iteration of the meta-learner as

$$O(nF_{\text{AM}} + nB_{\text{AM}} + npU_{\text{AM}} + nB_{\text{adapt}} + mU_{\text{adapt}}), \tag{4.32}$$

where F_{AM} is the cost of the forward pass of the acoustic model, B_{AM} is the cost of the backward pass of the acoustic model, U_{AM} is the cost of updating a single parameter of the acoustic model, B_{adapt} is the cost of backward pass of the meta-learner and U_{adapt}

is the cost of updating the parameters of the meta-learner. Consequently, if we want to speed up the adaptation procedure we can reduce the number of adaptation steps n, limit the number of adapted parameters p or simplify the update rule for updating the parameters of the acoustic model thus reducing U_{AM}. In Section 5.5 we empirically show how limiting the number of adapted parameters p and simplifying the update rule affects the training and inference speed of the meta-learner.

4.5 Implementation of Model-Agnostic Meta-Learner

The main disadvantages of the coordinate-wise meta-learner are the high memory and speed requirements, which prevents it from being used with state-of-the-art models with millions of parameters. As we discussed in the previous section the only way how to reduce the memory and speed footprint of the meta-learner with a predefined number of adaptations steps and a predefined model is to simplify the adaptation update rule learned by the meta-learner. With a simpler update rule with fewer parameters m we reduce the memory footprint and at the same time we speed up training and inference of the meta-learner because the cost of updating a single parameter of the acoustic model U_{AM} becomes much smaller. In order to simplify the update rule we can either use a coordinate-wise meta-learner with fewer units in the hidden layer or we can use a simple update rule

$$g_{\text{adapt}}(f_\theta, \mathcal{D}^{\text{adapt}}; \{\alpha\}) = \theta - \alpha \nabla_\theta \mathcal{L}, \tag{4.33}$$

which has only one learnable parameter α for each adapted layer and is very fast to compute. Learnable parameters α can be learned with Algorithm 1. This simple update rule, which is much more memory- and computationally-efficient than the coordinate-wise meta-learner, was used and evaluated in Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), which we described in Section 4.1. Note that by using this version of the adaptation function we were able to scale the meta-learning techniques to large acoustic models in Chapter 6.

4.6 Learning All Speaker Adaptation Hyperparameters

At the beginning of this chapter we argued that the gradient-based adaptation update rule contains several hyperparameters that need to be tuned if we want to obtain a

robust speaker adaptation procedure that can leverage all the information in the adaptation data and at the same time does not overfit to it. These hyperparameters include: a selection of speaker dependent parameters, a robust learning rate schedule, a good loss function, a filter for selecting relevant adaptation examples; and a tuned number of adaptation steps. The parameterisations described in Section 4.4 and Section 4.5 can automatically learn which layers should be adapted and how they should be adapted by learning a learning rate schedule for each layer. This is achieved by allowing the meta-learner to adapt all the layers of the acoustic model and by having separate parameters for each layer of the acoustic model. Therefore, the meta-learner can learn different adaptation dynamics for each layer. In some cases it can learn that it is better not to adapt certain layers. In this section we outline how the meta-learner could be adjusted to be able to automatically tune the remaining aforementioned hyperparameters.

Combination of Loss Functions

As discussed in Chapter 3, various regularisation terms, such as L2 or KL-divergence, can be used to prevent the adapted model from overfitting to the adaptation data. However, for each of these regularisation terms we need to tune the weight λ. Thus, similar to learning which layers should be adapted, we could provide a set of all possible loss functions

$$\left\{ \mathcal{L}_1 \left(\theta, \mathcal{D}^{\text{adapt}} \right), \ldots, \mathcal{L}_N \left(\theta, \mathcal{D}^{\text{adapt}} \right) \right\} \tag{4.34}$$

and learn a set of corresponding weight terms

$$\{\lambda_1, \ldots, \lambda_N\} \tag{4.35}$$

by providing a compound loss function

$$\hat{\mathcal{L}} \left(\theta, \mathcal{D}^{\text{adapt}} \right) = \sum_{l=1}^{N} \lambda_l \mathcal{L}_l \left(\theta, \mathcal{D}^{\text{adapt}} \right) \tag{4.36}$$

to the meta-learner. However, note that if we use the first-order approximation, we cannot implement the meta-learner like this, because the gradients $\nabla_\theta \hat{\mathcal{L}}$ are treated as constants. And therefore we would not be able to tune the weights λ_l. Instead, we can reparameterise the equation by realising that the gradient of a sum is a sum of gradients

$$\nabla_\theta \mathcal{L} \left(\theta; \mathcal{D}^{\text{adapt}} \right) = \sum_{l=1}^{N} \lambda_l \nabla_\theta \mathcal{L}_l \left(\theta, \mathcal{D}^{\text{adapt}} \right). \tag{4.37}$$

We could also learn a neural network $g(\theta, \mathcal{D}^{\text{adapt}}; \phi)$ that predicts the gradients directly without any need to compute the loss functions:

$$\nabla_\theta \mathcal{L} = g(\theta, \mathcal{D}^{\text{adapt}}; \phi), \tag{4.38}$$

which is similar to the idea of synthetic gradients (Jaderberg et al., 2017). Synthetic gradients are implemented as networks that predict gradients for a certain layer based on its activations and these synthetic gradient networks are trained to be as close to the real gradients as possible. By doing this the network could for example learn a low rank monophone output space from a high rank senone output space as done by Dighe et al. (2017).

Selection of Adaptation Data

Another important part of the adaptation process is the selection of the adaptation data especially in unsupervised speaker adaptation. By training only on frames for which the speaker-independent acoustic model is confident, the speaker dependent model will not be able to learn anything new. On the other hand if we train only on frames for which the speaker-dependent model is not confident the gradient information will be noisy which might hurt the performance of the adapted model.

Therefore, similarly to learning the combination of loss functions we could allow the meta-learner to learn which adaptation examples $(x_i, y_i) \in \mathcal{D}^{\text{adapt}}$ should be used for adaptation. We achieve this by reparameterising the loss computation as

$$\mathcal{L}_l(\theta, \mathcal{D}^{\text{adapt}}) = \sum_{(x_i, y_i) \in \mathcal{D}^{\text{adapt}}} w_i L(y_i, f(x_i; \theta)), \tag{4.39}$$

where the per example weights w_i can be predicted by a trainable neural network whose predictions can be conditioned on the loss value for the particular example but also on the state of the meta-learner, which would allow the adaptation process to learn a form of curriculum learning procedure (Bengio et al., 2009). Such a neural network could, for example, assign a low weight to examples with very high loss, because they are probably too noisy or they might be outliers.

Number of Adaptation Steps

Finally, we need to tune the number of adaptation steps. If we perform only a few steps the speaker dependent model will remain close the speaker independent model

and it will not leverage the potential of the adaptation data. On the other hand with too many adaptation steps the adaptation process becomes computationally expensive and we risk overfitting to the adaptation data. Therefore, it is important to find a good compromise for the number of adaptation steps.

In this thesis rather than tuning this hyperparameter we treat it as a model constraint and we let the meta-learner learn the best strategy, given the allowed number of adaptation steps. An alternative solution would be to allow the meta-learner to learn the number of adaptation steps by using ideas from Graves (2016), which proposed a way of allowing a recurrent neural network to perform an adaptive number of computations for each time step. In our setting this idea would be implemented by reformulating the update rule at time-step t as

$$\theta_{t+1} = \theta_t - p_t \alpha_t \nabla_{\theta_t} \mathcal{L}, \tag{4.40}$$

where p_t is called the *halting probability*. It is computed from halting unit predictions

$$h_t = \sigma(W s_t + b), \tag{4.41}$$

which are predicted from a state of the meta-learner s_t at adaptation step t, using the following equation:

$$p_t = \begin{cases} \mathcal{R} & \text{if } t = \mathcal{N} \\ h_t & \text{otherwise} \end{cases}, \tag{4.42}$$

where $\mathcal{N}$ is the actual performed number of adaptation steps and $\mathcal{R}$ is the halting probability remainder. The actual performed number of adaptation steps $\mathcal{N}$ is computed as

$$\mathcal{N} = \min \left\{ T, \min \left\{ t' : \sum_{t=1}^{t'} h_t \geq 1 - \varepsilon \right\} \right\}, \tag{4.43}$$

with a maximum number of adaptation steps T and a small constant ε that allows the meta-learner to perform only one adaptation step. The halting probability remainder $\mathcal{R}$, which is used for the last adaptation step to ensure that the probability distributions sum up to one, is computed as:

$$\mathcal{R} = 1 - \sum_{t=1}^{\mathcal{N}-1} h_t. \tag{4.44}$$

In order to minimise the number of performed adaptation steps, we can alter the meta-loss to also include a meta-regularisation term with weight τ

$$\tau (\mathcal{R} + \mathcal{N}), \tag{4.45}$$

which ensures that the meta-learner will learn to use as few adaptation steps as possible for each particular speaker adaptation situation. Note that by jointly learning both halting probabilities p_t and the learning rates α_t, we are not enforcing a decreasing learning rate schedule, we are just learning an adaptive number of computation steps. However, the training of a meta-learner with an adaptive number of adaptation steps might be very time-consuming, because during training we always have to run speaker adaptation for the maximum number of adaptation steps T, and it might not yield satisfactory results given the training costs. For more details about adaptive computation time we refer to Graves (2016).

4.7 Summary

In this chapter we introduced meta-learning. In meta-learning learning is happening on two levels, a learner is learning task specific classifiers and a meta-learner is learning how to better train them. This approach is also called learning to learn. As the main contribution of this thesis, we showed that speaker adaptation can also be formulated as a meta-learning task, where the learner is a speaker dependent model and the meta-learner is learning to produce these speaker dependent models by performing speaker adaptation of a speaker independent model. We showed that the speaker adaptation procedure can be implemented as a coordinate-wise meta-learner that replaces the traditional update rule in Equation 4.1. Alternatively it can also be implemented as a simple gradient descent update rule with learned per-layer learning rates. As a result, we can automatically learn robust speaker adaptation procedures with gradient descent as we demonstrate in Chapter 5. Furthermore, we showed that we can also formulate speaker adaptive training as a meta-learning task by embedding the gradient-based speaker adaptation directly into the training of the acoustic model. We compare this meta-learning approach for speaker adaptive training with traditional approaches in Chapter 6. Finally, in Section 4.6 we also sketched out how the meta-learner could be adjusted to be able to automatically learn all hyperparameters for the speaker adaptation procedure.

Chapter 5

Speaker Adaptation Experiments

In the previous chapter we described how speaker adaptation can be formulated as a meta-learning task. In this chapter we report results for speaker adaptation experiments. In particular, we compare the meta-learning approach for speaker adaptation with the adaptation of LHUC parameters and ALL parameters of the acoustic model with a learning rate optimised with a grid search. We also compare the coordinate-wise meta-learner with the meta-learner that learns per layer learning rates.

5.1 TED Talks

All speaker adaptation experiments in this thesis were performed on TED talks. We chose them because they contain clean single speaker recordings with several minutes of speech for each speaker. Thanks to this we believe that TED talks are an ideal task to estimate the effectiveness of various speaker adaptation techniques. We used the TED-LIUM dataset (Rousseau et al., 2014) to train the baseline acoustic models. To comply with the IWSLT evaluation protocol we only used TED talks from the TED-LIUM dataset that were recorded before the end of 2012. Our training set contains 881 speakers and is 134 hours long. We used a combined dev set from IWSLT 2010 and 2012 (Paul et al., 2010; Federico et al., 2012) for speaker adaptation hyperparameter tuning and to train the meta-learner. We evaluated the models on a combined test set of IWSLT 2010, 2011 and 2012 (Paul et al., 2010; Federico et al., 2011, 2012). The combined dev set contains 18 speakers and is 3.2 hours long and the combined test set contains 30 speakers and is 5.3 hours long. The details about the dataset are summarised in Table 5.1.

	hours	number of speakers
train	134	881
dev	3.2	18
test	5.3	30

Table 5.1: Details of the data split used in this thesis. The training data consists of TED talks from the TED-LIUM corpus (Rousseau et al., 2014) that were recorded before the end of 2012. Development data comes from development sets of IWSLT 2010 and 2012 (Paul et al., 2010; Federico et al., 2012) and test data comes from test sets of IWSLT 2010-2012 (Paul et al., 2010; Federico et al., 2011, 2012)

5.2 Baseline Acoustic Models

Training the coordinate-wise meta-learners requires large amounts of memory and is very computationally expensive (Section 4.4). Therefore, in these experiments we evaluated the speaker adaptation methods on the largest acoustic models that allowed us to fit the meta-learner on a single GPU. Note that this results in our models being $4\times$ to $6\times$ smaller than the models trained in corresponding Kaldi recipes (Povey et al., 2011) for the TED-LIUM corpus. We address the speed efficiency of the meta-learning approach in Section 5.5.

Our first baseline acoustic model was a small deep neural network model (called DNN in Table 5.3 and Table 5.4) trained with Kaldi. It consists of 1.5M weights across 6 hidden layers, each with 256 neurons, using sigmoid activation functions and an output layer corresponding to 3792 senones and an input corresponding to 7 acoustic frames. The DNN model used the language model from Bell et al. (2014) for decoding. This language model was a standard Kneser-Ney smoothed 3-gram language model (Kneser and Ney, 1995), which was pruned with a threshold of 10^{-7} to reduce memory requirements.

We also experimented with the adaptation of TDNN models (Peddinti et al., 2015) to show that the meta-learner can work with state-of-the-art architectures. Again we trained a small model with 2.1M weights across 6 hidden layers, with 300 units each and ReLU activation functions and an output layer corresponding to 4208 senones. The input corresponded to 5 acoustic frames projected with an LDA transform (Batlle et al., 1998). Except for the number of units, the architecture is based on the Tedlium 1b

	DNN	TDNN-BN	TDNN-LN
number of layers	6	6	6
hidden layer width	256	300	300
activation function	sigmoid	ReLU	ReLU
normalisation	-	Batch	Layer
number of senones	3792	4208	4208
language model	pruned 3-gram	pruned 4-gram	pruned 4-gram

Table 5.2: Details of the baseline acoustic models.

Kaldi recipe.[1] We evaluated two TDNN models, one using batch normalisation and the other using layer normalisation normalising only variance (TDNN-BN and TDNN-LN in Table 5.3 and Table 5.4). Both normalisation methods were applied after the ReLUs. Note that prior to speaker adaptation we fused the batch normalisation parameters into the subsequent affine transformation. Therefore, we did not have to perform batch normalisation during speaker adaptation. The TDNN models used a pruned 4-gram language model with 2 million n-grams for decoding. This language model was trained with the scripts from the Tedlium Kaldi recipe. Details of the baseline acoustic models are summarised in Table 5.2.

5.3 Speaker Adaptation Setup

In these experiments we explored rapid speaker adaptation using only 10 seconds of data to perform speaker adaptation. For the baseline speaker adaptation experiments we either adapted all layers or only the LHUC layers (denoted ALL and LHUC in Table 5.3 and Table 5.4). LHUC layers were inserted after every hidden layer, either after applying the sigmoid in the DNN model or after applying normalisation in the TDNN models. For both techniques we adapted the acoustic model for 3 epochs using stochastic gradient descent (SGD) with a learning rate of: 0.01 for the DNN model; $2.5 \cdot 10^{-6}$ for the TDNN models (ALL); and 0.7 (LHUC) for both DNN and TDNN models with a batch size corresponding to 256 frames. These learning rates were chosen using grid search. Note that the learning rate 0.7 for adaptation of LHUC layers is close to the learning rate of 0.8 that was used by Swietojanski et al. (2016).

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/tedlium/s5_r2/local/nnet3/tuning/run_tdnn_1b.sh

5.4 Training the Meta-Learner

We used the development sets from IWSLT 2010 and 2012 (Paul et al., 2010; Federico et al., 2012) to train the meta-learner for a pretrained acoustic model. The data for the first 13 speakers was used as the meta-training set and the last 5 speakers as the meta-validation set, as described in Section 4.2. We trained the meta-learner to adapt the acoustic model using n seconds of adaptation data to improve performance on the following n seconds. We implemented the meta-learner as a coordinate-wise meta-learner, as described in Section 4.4, with a single-layer LSTM with 10 hidden units performing 3 full-batch adaptation steps.[2] The meta-learner was trained using Adam (Kingma and Ba, 2014) with a learning rate of 0.001. The goal of the meta-learner was to learn an update rule for speaker adaptation, defined in Equation 4.14, that could replace the SGD update rule used in the baseline adaptation method. During training we monitored the loss on the meta-validation set and we selected the meta-learner that achieved the best meta-validation loss for testing. The meta-learner was implemented and trained with Keras (Chollet et al., 2015) and Tensorflow (Abadi et al., 2016) and the decoding of the adapted models was done with Kaldi (Povey et al., 2011). The source code for the meta-learner is publicly available.[3]

5.5 Results

We experimented with relatively small models in order to be able to train a coordinate-wise meta-learner for them. The DNN model achieves a word error rate (WER) of 20.7% on our test set and both TDNN models achieve a WER of 15.2% which is close to a DNN model used by Bell et al. (2014) which achieves a WER of 14.9% while the TDNN models are 20× smaller. This is probably due to better expressive power of TDNN models, a better language model and also improved training methods for ASR models.

First we performed supervised speaker adaptation using 10 seconds of adaptation data (Table 5.3). When adapting the LHUC parameters of the DNN model we obtained an improvement of 0.6% absolute which is comparable to other previous experiments (Swietojanski and Renals, 2014). Adapting all the weights performed worse as the adaptation schedule may have overfit to the adaptation data, but the meta-learner

[2] Note that this is in contrast with experiments reported in Klejch et al. (2018), where we used only a single full-batch adaptation step.

[3] `https://github.com/ondrejklejch/learning_to_adapt/`

	DNN	TDNN-BN	TDNN-LN
original	20.7	15.2	15.2
LHUC	20.1	14.3	14.6
ALL	20.6	14.5	14.6
META	19.3	14.1	14.1

Table 5.3: WER (%) of the supervised speaker adaptation experiments using 10s of adaptation data.

	DNN	TDNN-BN	TDNN-LN
original	20.7	15.2	15.2
LHUC	20.5	14.6	14.8
ALL	20.6	14.7	14.8
META	19.7	14.5	14.4

Table 5.4: WER (%) of the unsupervised speaker adaptation experiments using 10s of adaptation data.

was able to learn a good adaptation schedule that outperformed adapting LHUC parameters by an additional 0.8% absolute. Similarly, when adapting the TDNN models the meta-learners achieved better performance than when we adapted the LHUC parameters; however the difference was much smaller, only 0.2% and 0.5% absolute. Supervised adaptation of all three models with the meta-learning approach achieved $6-7\%$ relative improvement compared to the baseline.

We then performed unsupervised adaptation using 10 seconds of adaptation data (Table 5.4). When adapting the DNN model, the meta-learner outperformed adapting only LHUC parameters by 0.8% absolute, a relative improvement of 4.8% compared to the baseline. Similarly, unsupervised adaptation of the TDNN models using the meta-learning approach outperformed or matches performance of adapting only LHUC parameters. It achieved 5% relative improvement compared to the baseline.

Subsequently, we wanted to compare performance of the coordinate-wise meta-learner with a meta-learner that learns just per layer learning rates for the adaptation update rule. Also, we wanted to test whether the meta-learning approach can scale to larger amounts of adaptation data. Therefore, we ran supervised and unsupervised speaker adaptation experiments with 10s, 30s and 60s of adaptation data. The results

	DNN		TDNN-BN		TDNN-LN	
	coord	lr per layer	coord	lr per layer	coord	lr per layer
original	20.7		15.2		15.2	
10s	19.3	19.7	14.1	14.1	14.1	14.2
30s	18.7	19.2	13.6	13.7	13.7	13.7
60s	18.2	18.8	13.7	14.0	13.5	13.7

Table 5.5: WER (%) of the supervised speaker adaptation experiments with a coordinate-wise meta-learner and a meta-learner learning a learning rate per layer using 10s, 30s and 60s of adaptation data.

	DNN		TDNN-BN		TDNN-LN	
	coord	lr per layer	coord	lr per layer	coord	lr per layer
original	20.7		15.2		15.2	
10s	19.7	20.2	14.4	14.5	14.5	14.6
30s	19.3	19.6	14.2	14.3	14.4	14.5
60s	19.0	19.5	14.1	14.4	14.3	14.5

Table 5.6: WER (%) of the unsupervised speaker adaptation experiments with a coordinate-wise meta-learner and a meta-learner learning a learning rate per layer using 10s, 30s and 60s of adaptation data.

are summarised in Table 5.5 and Table 5.6. When we look at the results for speaker adaptation of the DNN model, we can see that both meta-learners can scale to more adaptation data. However, the coordinate-wise meta-learner consistently achieves better results compared to learning per layer learning rates. It achieves 6.7%, 9.6% and 12% relative improvements with 10s, 30s and 60s of supervised adaptation data compared to 4.8%, 7.2% and 9.1% with learning per layer learning rates. Similarly, the coordinate-wise meta-learner achieves better results in unsupervised speaker adaptation. We hypothesise that since the coordinate-wise meta-learner has a bigger capacity it can learn more about the adapted model and it can learn better strategies for adaptation. In the case of the TDNN models, both meta-learners can scale to 30s of adaptation data. However they do not benefit from having 60s of adaptation data compared to having only 30s. Additionally, the coordinate-wise meta-learner achieves comparable results with the meta-learner learning only per layer learning rates. This might suggest

that it is much harder to find a good adaptation strategy for a better baseline model with the coordinate-wise meta-learner, therefore the meta-learner learns similar behaviour to just learning per layer learning rates. It could also mean, that our meta-training procedure is not optimal and we might need more meta-training data to learn a better strategy.

Despite the fact that learning only per layer learning rates is not as powerful as using the coordinate-wise meta-learner, learning only per layer learning rates has the benefit that it is much easier to interpret. Note that the learned learning rates depend on how we compute the loss function. In all meta-learning experiments we did full-batch adaptation with the loss computed as

$$\mathcal{L}(\theta, \mathcal{D}^{\text{adapt}}) = 10^{-3} \sum_{(x,y)\in\mathcal{D}^{\text{adapt}}} L\left(y, f\left(x;\theta\right)\right). \tag{5.1}$$

We used this scaled sum of per frame losses, because we wanted to test whether we can use a meta-learner trained with 10s of adaptation data to perform adaptation with 30s or 60s of adaptation data. However, we found that this does not work and it is always better to train the meta-learner with the same amounts of adaptation data as will be used for evaluation. In Table 5.7, Table 5.8 and Table 5.9 we include learned learning rates for supervised and unsupervised speaker adaptation of the DNN, the TDNN-BN and the TDNN-LN models, respectively. In Table 5.7 we can see that the meta-learner learned that it is not good to adapt the last two or three layers in the unsupervised setting. We hypothesise that this is because of the noise in the labels and that the last few layers might be more sensitive to overfitting to those errors.

Interestingly, we can see that the meta-learner learned very small learning rates for the adaptation of the first layer of both TDNN models. This is probably due to the convolutional nature of the TDNN models, as a small change in the first layer has a serious impact on all computations in the TDNN. For a better intuition see how many paths would be affected by changing the input layer in Figure 2.4. More importantly, when we compare the batch normalisation model TDNN-BN and the layer normalisation model TDNN-LN, we see that the meta-learner learned an order of magnitude higher learning rates for TDNN-LN. A possible explanation for this is that by fusing the batch normalisation into subsequent layers, TDNN-BN layers stop being scale invariant and therefore we have to use smaller learning rates. Finally, even though the architectures differ only in the used normalisation, they require very different learning rates, which justifies the usage of meta-learning over hand-crafted learning rules.

	supervised			unsupervised		
	10s	30s	60s	10s	30s	60s
hidden layer 1	0.1822	0.1886	0.2299	0.1999	0.1590	0.0931
hidden layer 2	0.2402	0.1579	0.1104	0.3124	0.1894	0.1583
hidden layer 3	0.2952	0.2836	0.2396	0.5587	0.4613	0.1923
hidden layer 4	0.2940	0.1629	0.0586	0.0974	0.1024	0.1673
hidden layer 5	0.3379	0.2301	0.1769	0	0	0.1262
hidden layer 6	0.3507	0.2545	0.2028	0	0	0
output layer	0.3243	0.2587	0.1549	0	0	0

Table 5.7: Learned learning rates for the adaptation of the **DNN** model.

	supervised			unsupervised		
	10s	30s	60s	10s	30s	60s
hidden layer 1	0.0000	0.0000	0	0.0000	0.0000	0
hidden layer 2	0.0688	0.0456	0.0197	0.0505	0.0652	0.0243
hidden layer 3	0.0072	0.0032	0.0013	0.0041	0.0009	0.0016
hidden layer 4	0.0040	0.0021	0.0010	0	0.0003	0
hidden layer 5	0.0049	0.0028	0.0007	0.0028	0.0026	0
hidden layer 6	0.0072	0.0042	0.0026	0.0034	0.0029	0.0015
output layer	0	0	0	0	0	0

Table 5.8: Learned learning rates for the adaptation of the **TDNN-BN** model.

	supervised			unsupervised		
	10s	30s	60s	10s	30s	60s
hidden layer 1	0.0000	0.0000	0.0000	0.0001	0.0000	0
hidden layer 2	0.1435	0.0397	0.0745	0.1061	0.1317	0.0625
hidden layer 3	0.1062	0.0774	0.0355	0.0538	0.0351	0.0779
hidden layer 4	0.0444	0.0397	0.0073	0.0588	0	0
hidden layer 5	0.1639	0.0665	0.0685	0.0911	0.0416	0
hidden layer 6	0.1091	0.0334	0.0774	0.0462	0.0781	0.0393
output layer	0.0096	0.0115	0.0028	0.0094	0.0035	0.0036

Table 5.9: Learned learning rates for the adaptation of the **TDNN-LN** model.

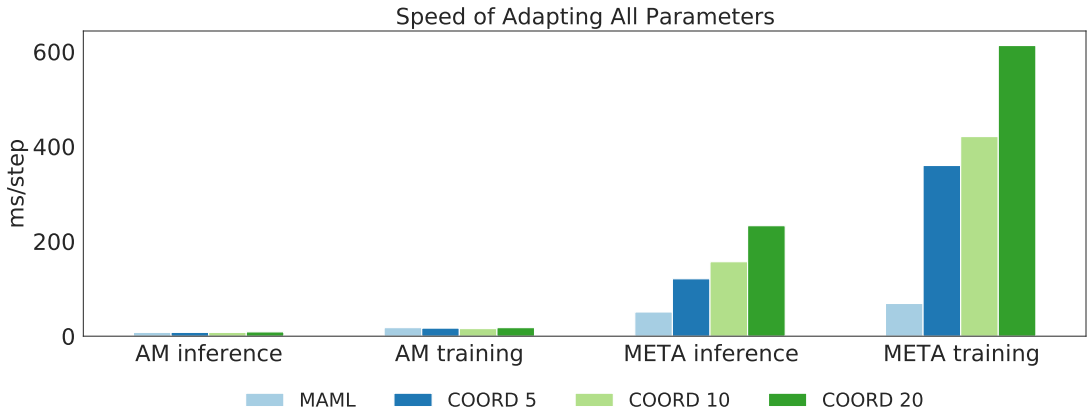

Figure 5.1: Speed benchmark of performing inference and training of a TDNN model with 10s of data. And a benchmark of performing inference and training of meta-learner doing 3 full-batch adaptation steps of all parameters of the acoustic model with 10s of adaptation data and evaluating the adapted model on another 10s of data.

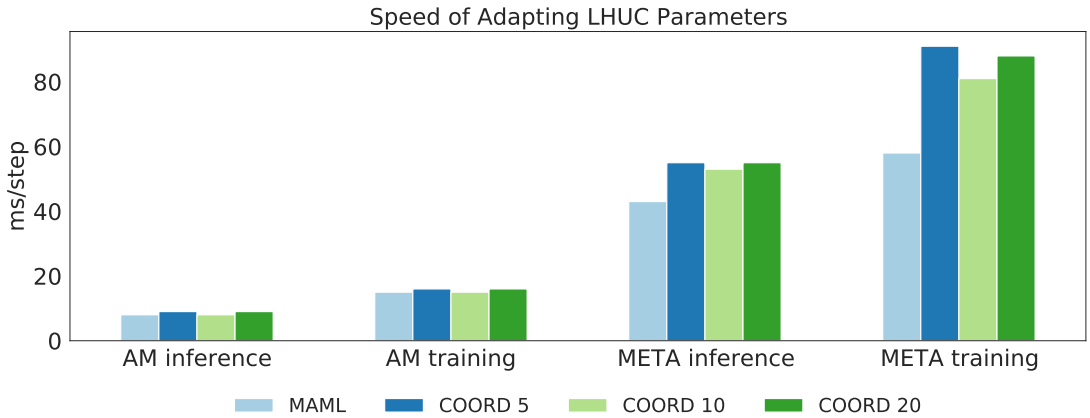

Figure 5.2: Speed benchmark of performing inference and training of a TDNN model with 10s of data. And a benchmark of performing inference and training of meta-learner doing 3 full-batch adaptation steps of LHUC parameters of the acoustic model with 10s of adaptation data and evaluating the adapted model on another 10s of data.

5.6 Benchmarking Meta-Learner Speed

In the previous chapter we claimed that the training of the meta-learner is computationally expensive. In this section we show this empirically by comparing the coordinate-wise meta-learner (Section 4.4) and MAML with a simpler update rule (Section 4.5). For this benchmark we measure how long it takes to:

- perform inference with 10s of data with the TDNN model (AM inference),

- train the acoustic model for one step with 10s of data (AM training),
- perform 3 full-batch adaptation steps with 10s of data with the meta-learner (META inference)
- train the meta-learner with 10s of adaptation and testing data for one speaker (META training).

We conduct the benchmarks on Nvidia Titan X (Pascal) and we report how many milliseconds it takes to perform one step. Figure 5.1 summarises benchmarks for the adaptation of all 2.1M parameters of the acoustic model and Figure 5.2 summarises benchmarks for the adaptation of 1800 LHUC parameters.

When we look at the speed of inference and training of the AM, we see that the training is approximately $2\times$ slower than the inference. This is because during inference we perform only a forward pass, whereas during training we need to do a forward pass, a backward pass and an update of the parameters.

When we adapt all 2.1M parameters of the meta-learner for 3 adaptation steps (Figure 5.1), we see that the time of META inference of MAML corresponds to performing 3 adaptation steps and doing inference on the test data. This is expected because both update rules have identical complexity. However, when we look at the time of META inference of the coordinate-wise meta-learner (COORD) we see that it increases with the number of units in the coordinate-wise meta-learner. This difference becomes even more pronounced when we train the parameters of the meta-learner, because the backward pass through the computation graph (Figure 4.1) takes a considerable time for the complex coordinate-wise meta-learner. This makes the coordinate-wise unusable for speaker adaptive training of state-of-the-art acoustic models on large datasets on a single GPU. Therefore, in Chapter 6 we used MAML for speaker adaptive training.

When we adapt 1800 LHUC parameters (Figure 5.2), we see that MAML is still faster than the coordinate-wise meta-learner but the difference in speeds is significantly smaller. It is interesting to see that the coordinate-wise meta-learner does not become slower with the increasing number of units. This is because we are adapting only 1800 LHUC parameters and the matrix operations with these small sizes are very fast on GPU.

Finally, note that the MAML META-inference benchmarks can also be used as a proxy to estimate the speed of grid search. This is because in grid search we also perform 3 adaptation steps with the simple update rule (Equation 4.1). Therefore, the main difference between training MAML and performing a grid search is that the grid

search does not perform a backward pass to update the meta-learner parameters but it finds the best parameters by evaluating parameters on a parameter grid. A grid search usually uses much more data to evaluate the parameters to obtain a robust performance estimate, which means that for a fixed number of evaluations MAML explores a much larger parameters space. Therefore sometimes a coarse grid search is replaced with Bayesian optimisation (Gelbart et al., 2014) that does not need to evaluate all possible parameters on the grid, but instead it searches only the most promising regions in the parameter space. However, note that while the grid-search approach works for the estimation of the learning rates it would be unfeasible to use it to learn more complex update rules, such as the coordinate-wise meta-learner.

To conclude, as we saw in Figures 5.1 and 5.2 if we want to reduce the memory and compute requirements of meta-learning for speaker adaptation, we either need to adapt fewer weights or we have to use a simpler update rule.

5.7 Summary

In this chapter we presented results for speaker adaptation experiments. We showed that meta-learning approaches, that learn learning rate schedules for adaptation of each adapted layer, can outperform handcrafted learning rates estimated with a coarse grid search. In addition, we compared coordinate-wise meta-learner with a meta-learner that learns per layer learning rates. We found that the coordinate-wise meta-learner can learn better adaptation procedures because of its more expressive power. On the other hand, learning only per layer learning rates allows us to better inspect what is happening during adaptation and it is much more computationally efficient. We showed that the meta-learner can learn not to adapt certain layers and that it learns different strategies for different models and different scenarios.

Chapter 6

Speaker Adaptive Training Experiments

In the previous chapter we experimentally showed that meta-learning can learn a good adaptation schedule that outperforms adaptation schedules with handcrafted learning rates. In this chapter we test whether the performance of test-time speaker adaptation can be further improved by using speaker adaptive training to train the acoustic model. In particular, we compare the traditional speaker adaptive training approach, described in Section 3.7, with our meta-learning approach, described in Section 4.3. In traditional speaker adaptive training we maintain and optimise a copy of speaker dependent parameters for each speaker during training to allow the canonical model to focus solely on modelling phonological variability. We used SAT-LHUC (Swietojanski and Renals, 2016), also described in Section 3.7, as a representative of the traditional methods. Our meta-learning approach embeds the gradient based speaker adaptation into the training of the acoustic model which ought to steer the training process into finding parameters that are suitable for rapid adaptation.

6.1 Baseline Acoustic Model

All models in this chapter used LDA projected 40 dimensional MFCC features without cepstral mean and variance normalisation. The baseline acoustic model was a TDNN, which consisted of 6 hidden layers with 600 neurons each and ReLU activation functions followed by batch normalisation and an LHUC layer. LHUC layers were used only for adaptation, otherwise LHUC parameters were set to 1. The model

predicted posteriors for 4208 senones, and had 5.9M parameters in total.[1] We trained the model for 400 iterations with Adam (Kingma and Ba, 2014). In each iteration we trained on 2000 batches that contained 256 chunks with 8 frames. This roughly corresponds to doing 3 epochs of training on all available training data. We performed early-stopping (Morgan and Bourlard, 1990; Prechelt, 1998) by monitoring the loss on a validation set after each iteration. We used the best performing model for decoding.

6.2 Speaker Adaptation Setup

As a baseline we performed speaker adaptation of LHUC weights and ALL weights of a baseline model. In both methods we used 3 steps of full-batch gradient descent to adapt the weights. In all experiments we used the combined dev set to train a meta-learner to find the per-layer learning rate for 10s, 30s and 60s of adaptation data. We performed both supervised and unsupervised speaker adaptation experiments. Unsupervised labels were obtained using a separately trained baseline model and were used for unsupervised speaker adaptation of all models. During adaptation, we removed frames corresponding to silence from the adaptation data, because silence does not contain any speaker information.

6.3 SAT-LHUC Training Details

The SAT-LHUC model had the same architecture and used the same training schedule as the baseline LHUC model except that during training we used speaker specific LHUC parameters for each speaker in order to enable the model to remove speaker variability. In order to obtain speaker-independent LHUC parameters, we created an artificial speaker and we mapped each utterance to this artificial speaker with probability 0.5 as we discussed in Section 3.7.

6.4 MAML Training Details

We trained a model with the same baseline acoustic model as described in Section 6.1 using the MAML approach. To train the model we used 10s of adaptation data and

[1] Note that this model is 50% smaller than a model used in a corresponding Kaldi recipe that has 850 neurons in each hidden layer. The smaller model achieves WER of 13.4% on the combined test set and the bigger model achieves WER of 13.0%.

the goal was to improve performance on the following 10s of unseen data using three full-batch adaptation steps. We trained different models to adapt LHUC parameters (MAML-LHUC) and to adapt ALL parameters (MAML-ALL). In both cases the meta-learner learned separate learning rates for each adapted layer. We trained the model with a combination of a loss computed with the speaker-independent model and a loss computed with the adapted model. We did this because in our preliminary experiments we found that using only the loss computed with the adapted model lead to worse results. We initialised the model with a baseline model that was trained for 200 iterations and continued training it for another 200 iterations using the MAML objective. We used Adam (Kingma and Ba, 2014) as an optimiser. In each iteration we trained on 1024 batches that contained data for 4 different speakers. This way all the models used the same number of frames for training. At the end we fine-tuned the per-layer learning rates for varying amounts of adaptation data on the dev set.

Since our baseline models use batch normalisation (Ioffe and Szegedy, 2015), we also wanted to use it in the MAML models. In order to explain why using batch normalisation in MAML is complicated let us first briefly recapitulate how batch normalisation works. As seen in Section 2.3, batch normalisation normalises hidden activations h in the following way:

$$h' = \gamma \frac{h - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta, \tag{6.1}$$

where γ and β are the learned scale and offset weights, μ and σ^2 are the mean and variance statistics estimated on the current batch during training (denoted μ_B and σ_B^2) or as running statistics during inference (denoted μ_G and σ_G^2), and ε is a small number preventing division by 0. Previous papers used batch normalisation in MAML with mean and variance statistics computed only on the current batch (Finn et al., 2017) or accumulated different running statistics for each training step, because there was a big shift in distributions of hidden activations between different training steps (Antoniou et al., 2019).

We believe that using statistics computed only on the current batch is not optimal, because it forces the model to perform batch normalisation per utterance during inference. Since each utterance might have a different duration, we would be using inconsistent estimates of the true mean and variance for each utterance – it might be compared to performing cepstral mean and variance normalisation per utterance instead of cepstral mean and variance per speaker. This also hurts performance of speaker adaptation because we are adapting parameters with different statistics rather

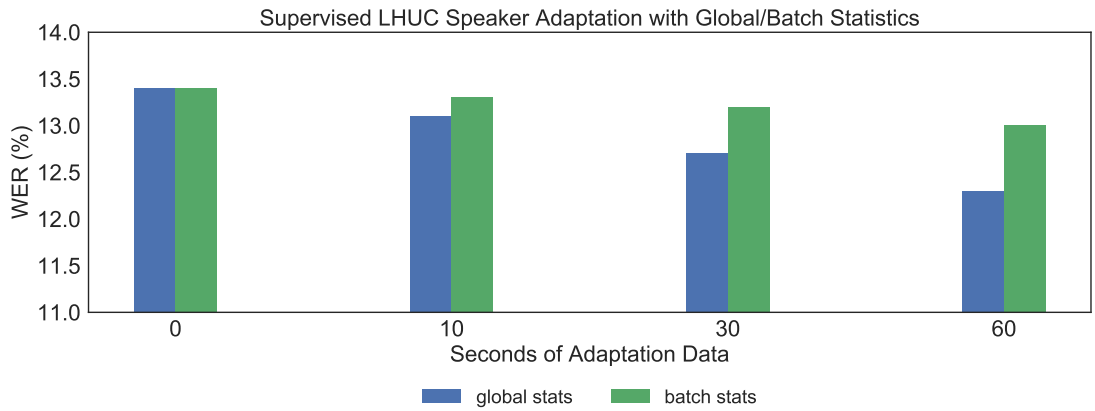

Figure 6.1: WERs (%) of supervised speaker adaptation of LHUC parameters of the baseline model. Batch normalisation statistics are estimated on all the training data (global stats), or on the adaptation data (batch stats).

than statistics that will be used during inference. Therefore we decided to use batch renormalisation (Ioffe, 2017) in MAML models. We described batch renormalisation in more detail in Section 2.3.

6.5 Results

Since our baseline acoustic models use batch normalisation, we first tested how to treat the mean μ and variance statistics σ^2 when adapting the LHUC weights of the baseline model. Our results are shown in Figure 6.1. Traditionally, the batch statistics μ_B, σ_B^2 are used during training, and the running mean μ_G and variance σ_G^2 statistics are used for inference. The results in Figure 6.1 suggest that for speaker adaptation with limited amounts of adaptation data it is better to use global statistics than batch statistics. This is likely because the global statistics are better estimates of the true means and variances – by adapting the LHUC weights we are essentially correcting for errors in the variance estimation. (Note that this was the only experiment where we used a learning rate of 0.7 for the adaptation of LHUC weights, because it was found to work well in Klejch et al. (2018).)

In the second experiment we conducted supervised speaker adaptation of the baseline model, SAT-LHUC, and MAML models (Figure 6.2). In all experiments we used adaptation schedules learned with the meta-learning approach (Equation 4.22), instead of a hand-crafted adaptation schedule. Using the learned adaptation schedule achieves much better results than adaptation using a handcrafted learning rate schedule (Fig-

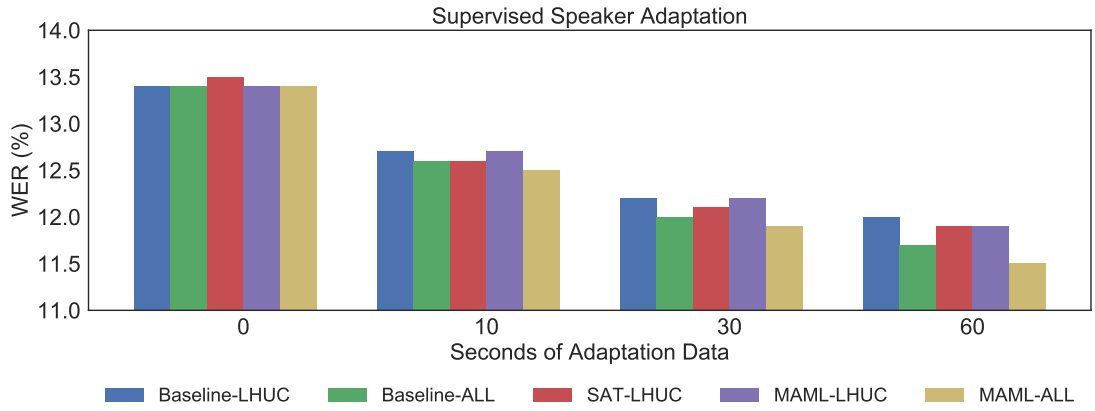

Figure 6.2: WER (%) for supervised speaker adaptation of the baseline, SAT-LHUC, MAML-LHUC, and MAML-ALL models.

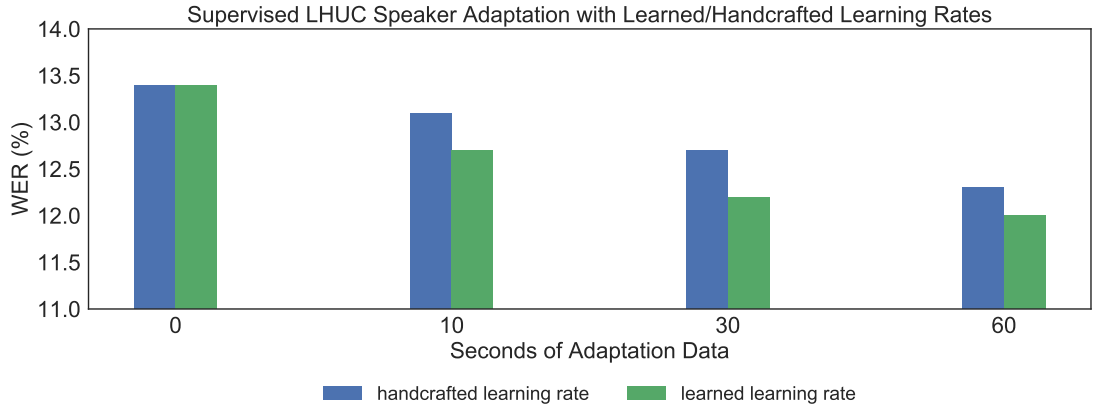

Figure 6.3: WER (%) for supervised speaker adaptation of LHUC parameters of the baseline model with handcrafted and learned learning rates.

ure 6.3). This is because the learned learning rates are three to four times larger than the handcrafted learning rate 0.7, which we found to work well in Klejch et al. (2018). This result has a straightforward explanation: In this chapter we perform full-batch adaptation, whereas in Klejch et al. (2018), adaptation was performed with a batch size of 256 frames. Consequently, adapting using a full batch of 10s (1000 frames) results in performing 4× fewer adaptation steps. Therefore, the meta-learner learned to use approximately 4× larger learning rates than the ones which worked well for a batch size of 256. However, this rule of thumb for scaling the learning rate worked only with 10s of adaptation. This scaling did not work when we tried to adapt the model with 30s or 60s of adaptation data because it resulted in large learning rates of 8.4 and 16.8, respectively, whereas the meta-learner learned learning rates of 1.2 and 1.1.

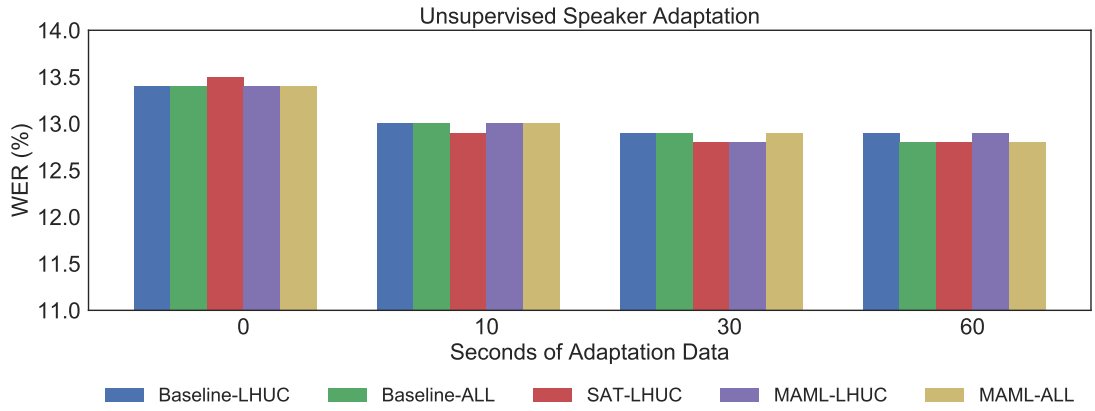

Figure 6.4: WER (%) for unsupervised speaker adaptation of the baseline, SAT-LHUC, MAML-LHUC, and MAML-ALL models.

When adapting all parameters, MAML-ALL typically outperforms the baseline, particularly for 60s of adaptation data. This suggests that it has found an improved schedule using MAML, compared to the hand-crafted schedule for Baseline-ALL. Nevertheless, this highlights the usefulness of using the meta-learning approach for estimating the adaptation schedule compared to an excessive hyperparameter search, which requires selecting appropriate step-sizes and bounds on the hyperparameters. We argue that the learned learning rates for LHUC layers are several orders of magnitude larger than commonly used learning rates. There is therefore a considerable chance that the hyperparameter bounds would not be set optimally to uncover the best solution. We observed similar trends when we performed unsupervised speaker adaptation experiments (Figure 6.4), but obtained much higher WERs than with supervised speaker adaptation.

When we compare the baseline model with SAT-LHUC and MAML-LHUC models we see that there is not a big difference between the adaptation of these models both in supervised and unsupervised scenarios. There are several possible explanations for this observation. First, SAT-LHUC was originally shown to improve speaker adaptation with a feed-forward neural network (Swietojanski and Renals, 2016). This feed-forward neural network was approximately $5\times$ bigger than our TDNN baseline, but it achieved much worse performance before adaptation. It is possible that our TDNN baseline is sufficiently powerful to model both speaker and phonological variability inside the canonical model. It therefore might not benefit from factoring out speaker variability into speaker dependent weights as much as the feed-forward neural network. Second, the feed-forward neural network used in Swietojanski and Renals

(2016) did not employ batch normalisation. It is possible that the batch normalisation may implicitly be removing speaker variation from the data, thus removing the need for speaker adaptation. Third, there are only 881 speakers in the training data and differences between them are probably much smaller than differences between different classes in the few-shot learning scenario (Finn et al., 2017), therefore the model is not forced to factor out speaker variability into speaker dependent weights.

6.6 Summary

In this chapter we compared speaker adaptation of a baseline acoustic model and models trained with speaker adaptive training of LHUC and all parameters. We evaluated the traditional speaker adaptive training using SAT-LHUC and our meta-learning approach for speaker adaptive training which embeds gradient based speaker adaptation directly into training of the acoustic model. We found that neither of these speaker adaptive training methods yield any improvement after test-time unsupervised speaker adaptation compared to the speaker adapted baseline model. We hypothesised that speaker adaptive training might not be helping because the baseline model has enough capacity to model both speaker and phonological variability. Alternatively, batch normalisation might be interfering with speaker adaptive training by removing the speaker variability in the batch that might be important for speaker adaptive training. In the following chapter, we run more experiments to find out why unsupervised speaker adaptation of models training with speaker adaptive training did not work in this chapter.

Chapter 7

Analysis of SAT-LHUC Training

In the previous chapter we reported results for our speaker adaptive training experiments. In those experiments we compared speaker adaptive training using the traditional SAT-LHUC approach and our meta-learning approach. Unfortunately, neither of these speaker adaptive training methods improved the performance of speaker adapted models compared to the adapted baseline model. This is in contrast with results reported in Swietojanski and Renals (2016), in which SAT-LHUC improved the performance of the adapted models. Therefore, we decided to run a series of ablation experiments that would help to determine, why speaker adaptive training did not work in our experiments. In the experiments we tested several factors that might have affected the performance of speaker adaptive training. These factors were neural network architecture, hidden layer width, model size, activation function, training optimiser and the number of training iterations.

7.1 Baseline Acoustic Models

Swietojanski and Renals (2016) originally employed SAT-LHUC with a feed forward neural network with 6 hidden layers with 2000 units and a sigmoid non-linearity; we call this neural network DNN in results. This neural network was trained with SGD with an initial learning rate of 0.08 and used the newbob learning rate schedule (Johnson et al., 2004). In our baseline DNN, we replaced sigmoids with ReLUs, we used batch normalisation; and we trained the model with Adam. Additionally, we also trained a TDNN with 6 hidden layers with 600 units each, ReLU non-linearities and batch normalisation. We trained the network with Adam using an initial learning rate of 0.0015. We gradually decreased the learning rate to a final learning rate of 0.00015

	test 2010	test 2013
DNN (Swietojanski and Renals (2016))	15.2	22.3
DNN (ours)	14.2	18.6
TDNN (ours)	14.0	17.7

Table 7.1: WER (%) comparing various neural network architectures on test sets from IWSLT 2010 and 2013.

using a linear learning rate schedule. Table 7.1 shows results for the baseline models evaluated on the test sets from IWSLT 2010 and 2013 in comparison to the model from Swietojanski and Renals (2016). We can see that both our baseline models achieve comparable WER and they are both significantly better than the DNN from Swietojanski and Renals (2016). Note, that in the rest of this chapter we report results on the combined test set from IWSLT 2010–2012 as we did in all previous experiments in Chapter 5 and Chapter 6.

7.2 Speaker Adaptation Setup

In order to be able to show whether speaker adaptive training affects the performance of speaker adaptation, we conducted unsupervised speaker adaptation experiments with 60s of adaptation data. We chose 60s of adaptation data because we hypothesise, that the effect of speaker adaptive training might not be visible with small amounts of adaptation data. As in Chapter 5, we optimised the adaptation update rule by training a meta-learner with a learning rate for each layer on the combined dev set from IWSLT 2010 and 2012. This ensures that we always use an optimal speaker adaptation procedure for each model.

7.3 Effect of the Normalisation

Since SAT-LHUC has the expressive power to learn to normalise the variance of hidden representations, in the first experiment we tested how SAT-LHUC interacts with different normalisation techniques. Our hypothesis was that if we use batch normalisation, we might be removing the variance of the hidden representations before passing them to the LHUC layer. Depending on how we sample batches during training, for ex-

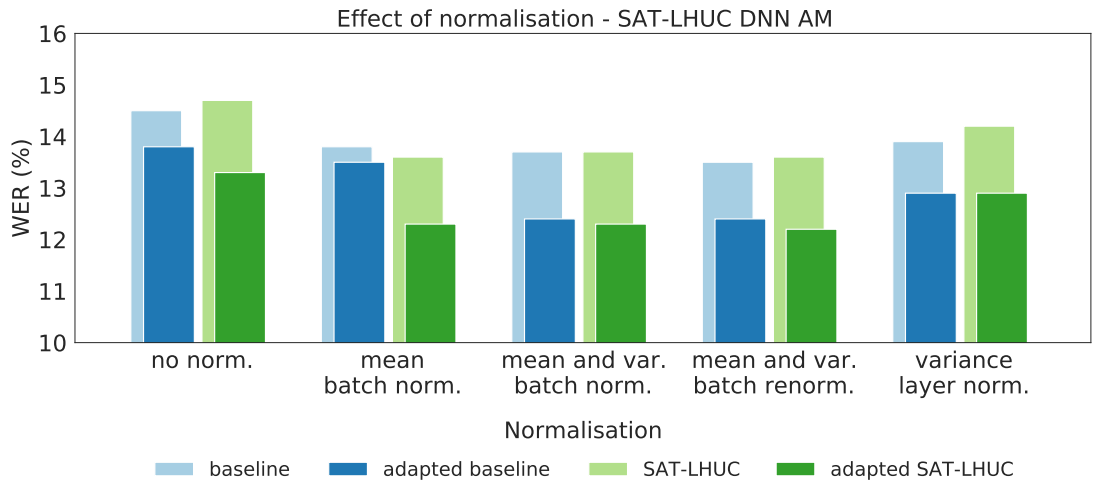

Figure 7.1: WER (%) comparing speaker adaptation of a baseline and a SAT-LHUC **DNN** model with various normalisation functions.

ample if the batch contains data only from a single speaker, the variance of the hidden representations might be highly related to speaker variance. Thus, if we remove the variance with batch normalisation, SAT-LHUC would not be able to learn to remove the speaker variability. To test this hypothesis we trained DNN and TDNN models with no normalisation, batch mean normalisation, standard batch mean and variance normalisation, batch renormalisation; and variance layer normalisation. All these normalisation techniques were described in Section 2.3.

As we can see from the results for the DNN models (Figure 7.1) and the TDNN models (Figure 7.2 and Figure 7.3), SAT-LHUC improves performance of speaker adapted models when we do not normalise the hidden representations or when we perform batch mean normalisation. This confirms our hypothesis that variance normalisation performed by the standard batch normalisation or layer normalisation might be interfering with SAT-LHUC. However, in practice our goal is to find the best performing model, regardless of whether we use speaker adaptive training or not. Therefore, in practice it is better to use batch normalisation or batch renormalisation and adapt those models than to train SAT-LHUC models with no normalisation or batch mean normalisation. Nevertheless, it is worth pointing out that the adapted SAT-LHUC DNN models achieve better results than their SAT-LHUC TDNN counterparts, which is interesting because they achieve comparable performance before adaptation. This observation might be explained by wider hidden layers in the DNN models, which results in a larger number of speaker dependent LHUC parameters discussed below.

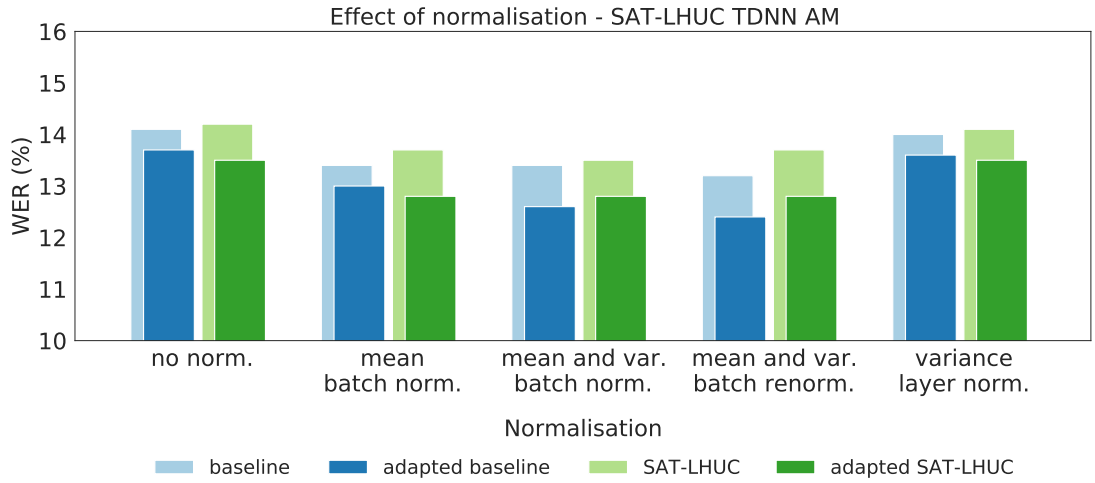

Figure 7.2: WER (%) comparing speaker adaptation of a baseline and a SAT-LHUC **TDNN** model with various normalisation functions.

7.4 Effect of the Hidden Layer Width

In the second set of experiments we experimented with wider hidden layers in the TDNN model to test whether an increased hidden layer width can improve the performance of SAT-LHUC models. We hypothesised that a wider hidden layer width might allow SAT-LHUC models to learn speaker specific units instead of performing only variance normalisation. These speaker specific units can then perform some form of model combination, which is similar to the hypothesis that Dropout implicitly learns an ensemble of smaller models (Srivastava et al., 2014). This might imply that SAT-LHUC is not performing speaker adaptive training in the traditional point-of-view.

In order to keep the model size comparable to the original TDNN model, we decided to use the factorised TDNN model (TDNN-F) (Povey et al., 2018). TDNN-F models factorise each affine transformation matrix $A \in \mathbb{R}^{n \times n}$ into a product of two low-rank matrices $U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times n}$ such that $A = UV$. This factorisation enables an increase of the hidden layer size without increasing the total number of parameters by controlling the bottleneck dimension. In this experiment we trained a TDNN-F model with 2048 units in each hidden layer and a bottleneck dimension of 128. We chose this configuration to ensure a comparable hidden layer width with the DNN models, while having a comparable total number of parameters with the TDNN models. As in the previous experiment, we trained the model with various normalisation techniques to test whether the better performance of SAT-LHUC DNN models can be explained solely by the fact that they have more speaker dependent parameters.

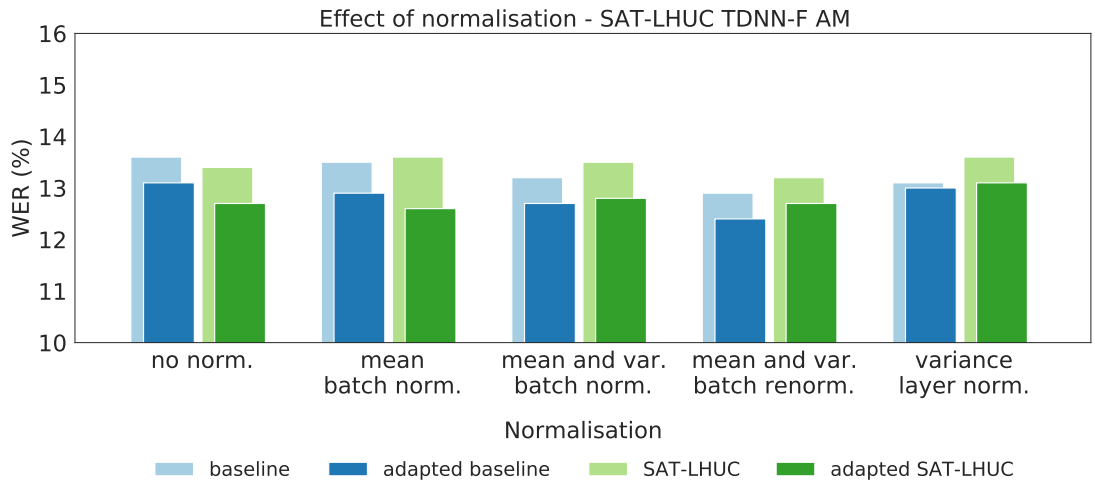

Figure 7.3: WER (%) comparing speaker adaptation of a baseline and a SAT-LHUC **TDNN-F** model with various normalisation functions.

When we look at the results in Figure 7.3, we see a similar trend as we saw in the results for the TDNN models in Figure 7.2. This leads us to the conclusion that the improved performance of the adapted SAT-LHUC DNN models must also be due to the model size, because the DNN models have 30.8 million parameters whereas the TDNN and TDNN-F models have only 6 million parameters. We hypothesise that the bigger capacity of the canonical DNN models is also important to allow SAT-LHUC models to perform model combination instead of variance normalisation.

7.5 Effect of the Model Size

To test the hypothesis that the bigger capacity of the canonical model is important for SAT-LHUC, we ran two sets of experiments. In the first one we increased the hidden layer width of the TDNN models and in the second we increased the bottleneck dimension of the TDNN-F models. In both cases we used batch-renormalisation, because it achieved similar results to batch normalisation in the previous experiments. Also, it should be better suited theoretically for speaker adaptive training, because during training it does not remove batch specific variance, but it removes global variance; therefore the speaker variance might still be present in the hidden representations. As we see from the results in Figure 7.4 and Figure 7.5, increasing the model size improves performance of the models, however, it does not improve performance of the adapted SAT-LHUC models compared to the adapted baselines.

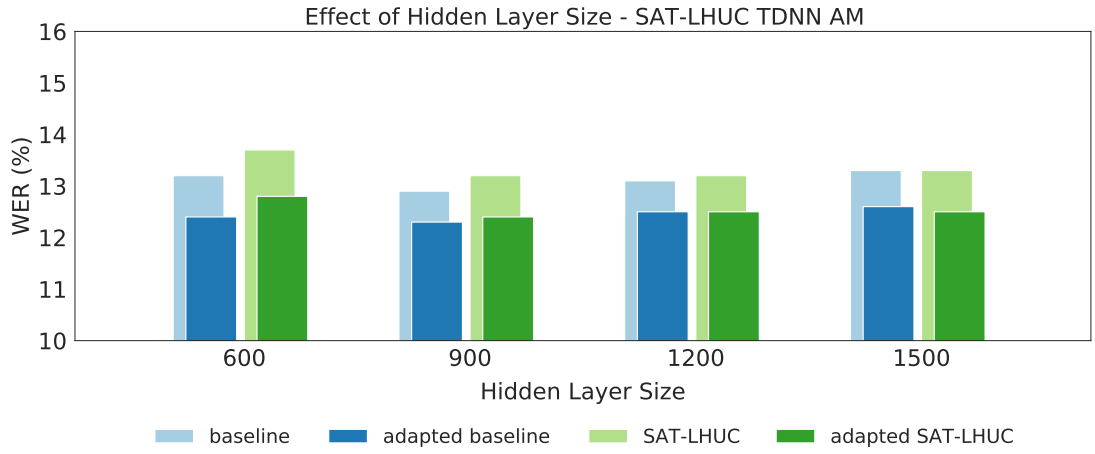

Figure 7.4: WER (%) comparing speaker adaptation of a baseline and a SAT TDNN model with increasing hidden layer widths.

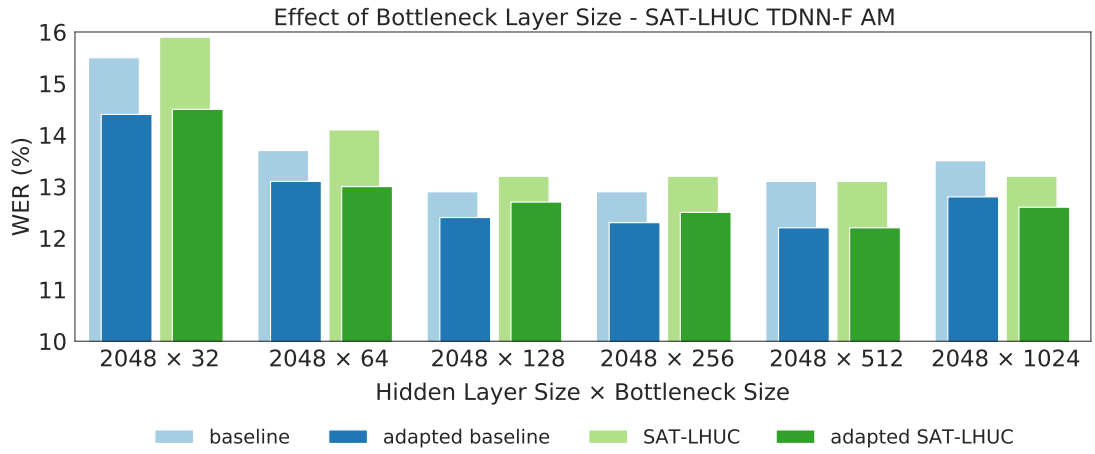

Figure 7.5: WER (%) comparing speaker adaptation of a baseline and a SAT TDNN-F model with increasing bottleneck dimensions.

7.6 Effect of the Activation Function

To see whether the choice of the activation function affects speaker adaptive training, we run experiments evaluating three activations: ReLU (relu), sigmoid (σ) and hyperbolic tangent (tanh). We used a TDNN model with 900 units and batch renormalisation. As we can see from Figure 7.6, in all cases the adapted SAT-LHUC models perform worse than their corresponding baseline models. Therefore, we conclude that the choice of an activation function is not crucial for the performance of speaker adaptation with SAT-LHUC models.

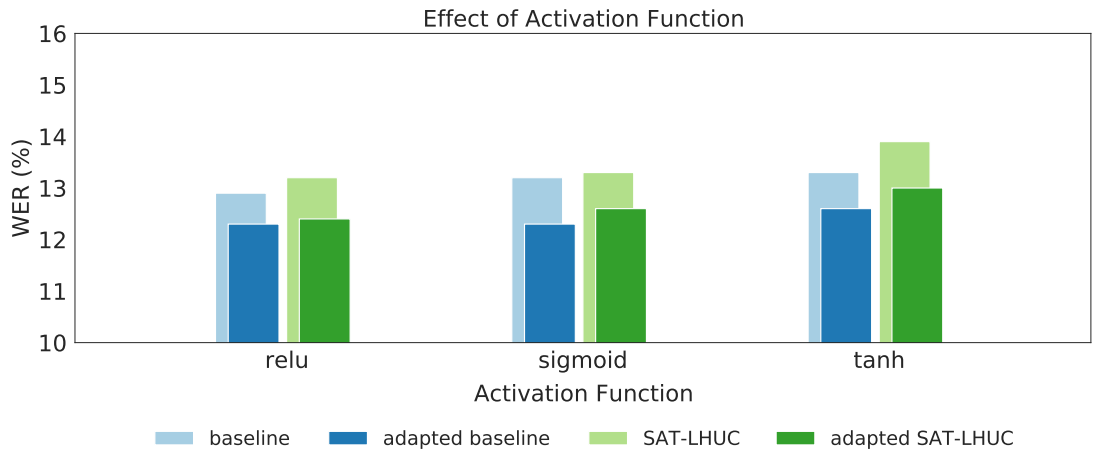

Figure 7.6: WER (%) for comparison of speaker adaptation of SI and SAT TDNN models with 900 units, batch renormalisation and various activation functions.

7.7 Effect of the Training Optimiser

In Chapter 5, we experimentally showed that the best learning rate for the adaptation of LHUC weights with a batch size of 256 frames is 0.7 – 0.8. However, when we train SAT-LHUC models we a used learning rate of 0.0015 with a batch size of 2048 frames. We can easily compute that the per frame learning rate used during training is almost 4000$\times$ smaller than the per frame learning rate used for adaptation. We hypothesise that using larger learning rates for LHUC layers during training might be crucial for an improved performance of SAT-LHUC models.

Moreover, we use Adam (Kingma and Ba, 2014) to train all models, but we use standard gradient descent during adaptation. The problem with Adam might be that it uses an exponential moving average of gradients to update parameters and implementations of Adam in the standard toolkits, Tensorflow (Abadi et al., 2016) and Keras (Chollet et al., 2015), update this exponential moving average for each parameter after every batch regardless of whether the parameter was used in that batch or not. In our case, speaker dependent parameters are used only for speakers present in the batch and only averages corresponding to these weights should be updated. However, because of the incorrect implementation other speaker dependent exponential moving averages are updated with 0. Therefore, the magnitude of the exponential moving averages of gradients for speaker dependent parameters are smaller than they should be, which makes the actual update of speaker dependent parameters smaller compared to the traditional SGD update rule.

Therefore, we trained a SAT-LHUC TDNN model with a hidden layer size of 600

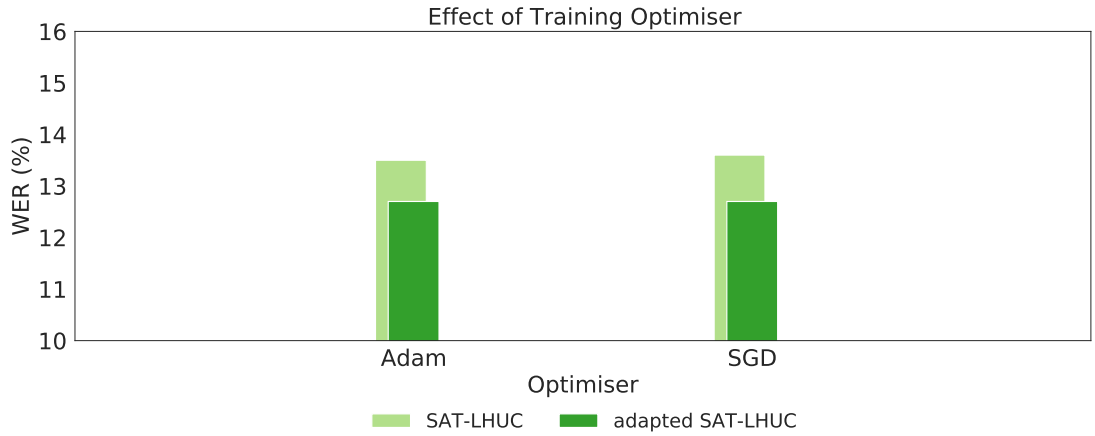

Figure 7.7: WER (%) comparing SAT-LHUC models using Adam with standard settings and SGD with a learning rate of 5.6 for the training of the speaker dependent parameters.

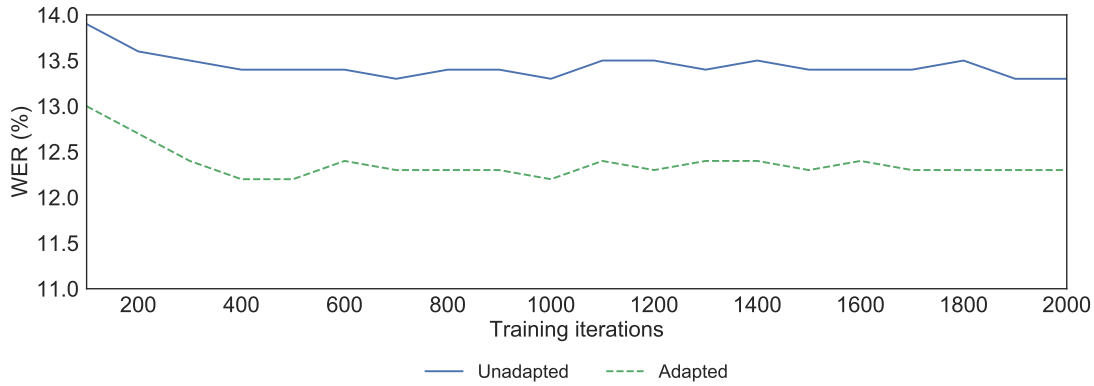

Figure 7.8: WER (%) for unsupervised speaker adaptation of checkpoints of the SAT-LHUC model trained for 2000 iterations.

and batch normalisation. We used SGD with a learning rate of 5.6 to train the speaker dependent layers, which ensures that the per-frame learning rate is consistent with the adaptation setting, and we used Adam for all speaker independent layers. As we can see from the results in Figure 7.7, there is no difference between using Adam and SGD with a high learning rate for the training of the speaker dependent parameters.

7.8 Effect of the Number of Training Iterations

In the last experiment we tried to train the SAT-LHUC TDNN model for more iterations. Our hypothesis was that the model in Swietojanski and Renals (2016) was trained for many more iterations, therefore it had more time to factor out speaker vari-

ance into the LHUC parameters and model only the phonological variances inside the canonical model. Therefore, we trained a SAT-LHUC model for 2000 iterations instead of 400 iterations, which we used in all other experiments. During training we saved checkpoints of the model every 100 iterations. Subsequently, we performed speaker adaptation on every checkpoint of the model. Unfortunately, as we can see from Figure 7.8, training the SAT-LHUC model for more iterations did not have any effect on the performance of the adapted model.

7.9 Conclusions

In this chapter we reported results for experimental ablations, in which we wanted to identify a good strategy for training SAT-LHUC models, especially SAT-LHUC TDNN models. First, we tested the hypothesis that batch normalisation interferes with SAT-LHUC training, because it normalises the variance of the hidden representations. We found that adaptation of SAT-LHUC models works better with normalisation methods that do not normalise variance compared to adaptation of the baseline model. However, we found that in order to train the best performing model, it is better to train a baseline model with batch normalisation or batch renormalisation. We also found that SAT-LHUC DNN models with wider hidden layers benefit much more from speaker adaptation. Therefore, in the second experiment we increased the hidden layer size of TDNN models by using the TDNN-F architecture which factorises the affine transformation matrices into a product of two low rank matrices. This allowed us to have the same hidden layer size as the DNN models, while maintaining the number of parameters of the TDNN model. We found that SAT-LHUC was not affected by increasing the hidden layer size and the adapted SAT-LHUC TDNN and SAT-LHUC TDNN-F models achieved comparable results. Consequently, we decided to also increase the total number of parameters by increasing the hidden layer width of the TDNN model and by increasing the bottleneck dimension of the TDNN-F models. We found that increasing the model size had a positive impact on the overall performance of the models. However, the adapted SAT-LHUC models did not outperform the baseline models. In addition, we found that using different activation functions does not affect the performance of SAT-LHUC training. We tried to change setting of the optimiser to make estimation of LHUC parameters closer to how they are estimated during test-time speaker adaptation. Finally, we tried to train SAT-LHUC models for more iterations to allow the model to factor out parameters dealing with speaker variability into LHUC

parameters. We found that they do not have any positive impact on speaker adaptive training. As we can see from all the results, we explored many hypotheses, none of which fully explained why adaptation of SAT-LHUC TDNN models does not yield any improvements compared to the adaptation of the baseline TDNN models.

We hypothesise that the current performance of adapted SAT-LHUC models might be due to the way we train the speaker independent model. As we explained in Section 3.7, during the training of SAT-LHUC models we create a new artificial speaker, and we map each utterance to this speaker with probability 0.5. Subsequently, we use LHUC parameters for this artificial speaker as LHUC parameters of the speaker-independent model. This way we obtain a competitive speaker independent model. However, we might be limiting the potential improvements obtained by speaker adaptation, because the speaker-independent model is too good and does not need to be adapted to perform well on new speakers. Yin et al. (2020) made a similar observation that if MAML can perform well on all training tasks without any need to adapt to them, its performance on new tasks is much worse, because it did not learn that it needs to adapt to new tasks. Therefore, Yin et al. (2020) proposed to use a meta-regularisation technique that forces MAML to use the task specific adaptation data in order to perform well on new tasks. This observation might explain why other speaker adaptive training methods, especially methods using fMLLR features but also methods using i-vectors, work well with large neural network acoustic models. With these methods it is crucial to correctly estimate the speaker dependent parameters, because with mismatched fMLLR transformations or mismatched i-vectors the performance of the model significantly degrades. Therefore, the adaptation procedure is forced to use the adaptation data in order to work well.

Finally, since we found that batch normalisation interferes with SAT-LHUC it would be interesting to explore speaker adaptive training using only batch normalisation by having only a few speakers in each batch and normalising activations with respect to the speakers. At test-time, speaker normalisation would be performed by updating statistics of the batch normalisation layers as recently proposed by Mana et al. (2019). The main benefit of performing speaker normalisation is that we would not need any labels to improve the performance of the model. However, one issue with this approach is that silence might negatively affect the statistics. To fix that we could use a mixture of batch normalisations (Deecke et al., 2019; Xie et al., 2019a), where one would be used for frames corresponding to silent frames and the other one would correspond to non-silent frames. This resembles cMLLR adaptation when we

estimate only two transforms,s one for silent frames and one for non-silent frames. It would also be interesting to test whether the combination of speaker normalisation and speaker adaptation, which first updates the batch normalisation statistics and then adapts LHUC parameters, would yield any improvements compared to only performing normalisation or adaptation. Note that meta-learning could be used to find the best mixing coefficient for updating the batch statistics by doing a linear combination of global and speaker statistics.

Chapter 8

Lattice Based Unsupervised Speaker Adaptation

In the previous chapters we explored ways of automatically finding a robust speaker adaptation schedule using meta-learning. Meta-learning has the potential to find good hyperparameters for the speaker adaptation procedure such as update rules, learning rates or weight terms for different losses. However, we need to provide the meta-learner with a set of existing loss functions, which can then be used to find the best speaker adaptation schedule.

In supervised adaptation the loss is computed with respect to a label sequence that is provided for the adaptation data. However, for unsupervised adaptation only an unlabelled recording is available. Conventionally, the best path from a first pass decoding is used to estimate the labels for unsupervised adaptation (Woodland, 2001). An important challenge for unsupervised model adaptation of neural networks is that we do not want to overfit to errors made in the first pass decoding. In the past, neural network based acoustic models were prevented from overfitting to those errors by limiting the expressivity of the adaptation procedure. This was done, as discussed in Chapter 3, by drastically reducing the number of speaker dependent parameters (Swietojanski and Renals, 2014; Samarakoon and Sim, 2016b; Zhao et al., 2017) or by imposing strong regularisers that prevent the outputs or weights of the acoustic model from diverging too far from the original model (Li and Bilmes, 2006; Yu et al., 2013). Alternatively, this challenge was tackled by performing adaptation only with suitable data that was obtained by filtering adaptation data by confidence scores produced by an ASR system (Woodland, 2001; Mathias et al., 2005; Liu et al., 2007; Walker et al., 2017; Veselý et al., 2017) or by using an external ASR quality estimation (Falavigna et al., 2017).

In this chapter we explore an alternative solution, in which all the adaptation data is used to adapt the whole neural network acoustic model, but the uncertainty in the decoding is captured through the use of complete lattices for supervision. Since lattices contain compressed and determinised information about the search space for a particular utterance, a suitable loss function can leverage model uncertainties for unsupervised adaptation. Our approach is inspired by recent work on semi-supervised learning using the sequence level lattice-free maximum mutual information (LF-MMI) objective function (Manohar et al., 2018), in which it is shown that using lattices as supervision is beneficial compared to using only best paths in the semi-supervised learning setting. This approach allows us to reliably adapt all weights of neural network models using unsupervised adaptation and a discriminative training criterion, which was problematic in the past.

8.1 Lattice supervision

Discriminative training using criteria such as maximum mutual information (MMI) (Bahl et al., 1986) has been shown to be sensitive to the accuracy of the transcripts (Mathias et al., 2005; Yu et al., 2010). As a replacement for better transcripts, a range of transcript filtering approaches have previously been explored (Mathias et al., 2005; Liu et al., 2007; Walker et al., 2017). In unsupervised or semi-supervised approaches, in which we generate hypothesis transcriptions by decoding with a seed model, we can alternatively use a lattice of supervision. Lattice supervision has previously been used in work on unsupervised adaptation (Padmanabhan et al., 2000) and training (Fraga-Silva et al., 2011) of GMMs, as well as discriminative (Povey, 2005) and semi-supervised training (Manohar et al., 2018) of neural network models.

For instance, lattice supervision can be used with the MMI criterion:

$$\mathcal{F}_{MMI}(\lambda) = \sum_{r=1}^{R} \log \frac{p_\lambda(X_r|\mathbb{M}_r^{num})}{p_\lambda(X_r|\mathbb{M}_r^{den})}, \tag{8.1}$$

where the $\mathbb{M}_r^{num}$ is a numerator lattice containing multiple hypotheses from a first pass decoding and $\mathbb{M}_r^{den}$ is a denominator lattice containing all possible sequences of words. The derivatives of the MMI criterion for a single audio segment r with respect to the output of the acoustic model $y_t(s)$ for state s at time t are:

$$\frac{\partial \mathcal{F}_{MMI}}{\partial y_t(s)} = \gamma_t^{num}(s) - \gamma_t^{den}(s), \tag{8.2}$$

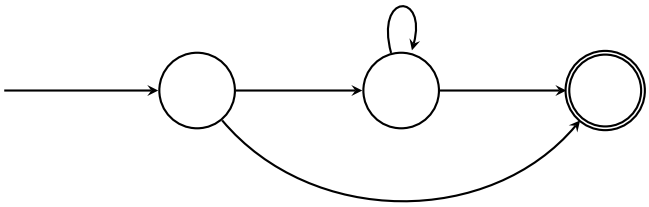

Figure 8.1: A diagram of the HMM topology used in LF-MMI models that allows the processing of a phone in one step. In the LF-MMI framework, HMM transition probabilities are uniform and fixed.

where $\gamma_t^{num}(s)$ is the numerator occupation probability for state s at time t and $\gamma_t^{den}(s)$ is the denominator occupation probability. The occupation probabilities are computed using the Forward-Backward algorithm.

8.2 Lattice-Free MMI

Following Manohar et al. (2018), we explore the use of lattice supervision versus that of only using the best path in the denominator lattice-free version of MMI (LF-MMI) (Povey et al., 2016). LF-MMI was introduced by Povey et al. (2016) as a method to train neural network acoustic models with a sequence discriminative criterion (MMI) without an initial cross-entropy (CE) stage to generate lattices approximating all possible word sequences (e.g. Veselý et al. (2013)). The word-level denominator lattice $\mathbb{M}_r^{den}$ is instead replaced with a phone-level denominator graph encoding all possible sequences given a 3 or 4-gram phone language model. Together with an efficient GPU implementation of the Forward-Backward algorithm, this allows us to compute the denominator occupation probabilities on-the-fly during training. To further reduce complexity, the model outputs at one third of the frame rate. This is enabled by utilising an HMM topology, illustrated in Figure 8.1, that can be traversed in one step instead of three steps as in the standard HMM topology, illustrated in Figure 2.2. This HMM topology was inspired by CTC (Graves et al., 2006). A mixture of regularisation methods are required to reduce overfitting; they include multitask cross entropy loss computed through a separate output layer or L2 regularisation of the activations of the last layer to prevent overconfident predictions. For more details we refer to Povey et al. (2016).

8.3 Baseline Acoustic Models

We conducted test-time model adaptation experiments on two datasets: the TED-LIUM corpus of TED talks (Rousseau et al., 2012, 2014), described in Section 5.1, and a corpus of Somali from the IARPA MATERIAL programme. All models were trained and adapted using the Kaldi toolkit (Povey et al., 2011). The training details are as follows:

- **TED-LIUM** – We trained two TDNN models (Peddinti et al., 2015) with LF-MMI (Povey et al., 2016) following Kaldi TED-LIUM recipe 1f.[1] Both models had the same architecture with 7 hidden layers with 450 units and they used 40 dimensional MFCC features as input. The first model was trained without i-vector features (called *no i-vectors* in the Results section) and the second model was trained with i-vector features (called *i-vectors* in the Results section). We chose to train these two models to show how auxiliary features based adaptation interacts with model-based speaker adaptation. Similarly to previous experiments, all models were trained only on TED talks that were recorded before 2012 in order to conform with the IWSLT (Federico et al., 2012) evaluation guidelines. Speaker adaptation experiments were performed on the combined dev set from IWSLT 2010 and 2012 and test set from IWSLT 2010–2012 (Paul et al., 2010; Federico et al., 2011, 2012). The dev set consisted of 18 speakers with an average speech duration of 10.6 minutes. The test set consisted of 30 speakers with an average speech duration of 10.7 minutes. In contrast to our previous experiments, we used all the speaker data for unsupervised adaptation.

- **Somali** – We carried out experiments on the Somali "surprise language" data released to participants on the IARPA-MATERIAL programme.[2] The training data comprise 499 narrow-band telephone conversations sides, totalling 37 hours of speech. The test data comprises narrowband telephone conversations (NB); and wideband (WB) data from the news and topical broadcast domains that are mismatched to the training material. We trained a TDNN-F model (Povey et al., 2018) using the neural network architecture from Kaldi TED-LIUM recipe 1g.[3] The model had 14 hidden layers with 1024 units. The weight matrices were

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/tedlium/s5_r2/local/chain/tuning/run_tdnn_1f.sh

[2] https://www.iarpa.gov/index.php/research-programs/material

[3] https://github.com/kaldi-asr/kaldi/blob/master/egs/tedlium/s5_r2/local/chain/tuning/run_tdnn_1g.sh

factored into two matrices with a bottleneck dimension of 128. The model used filterbank, pitch and probability of voicing (Ghahremani et al., 2014) features together with multilingual bottleneck features obtained from a neural network that was trained on all Babel languages (Cui et al., 2016; Gales et al., 2017). We used per utterance cepstral mean and variance normalisation, since there were no speaker clusters for the wideband test data.

The model was trained on narrowband data with speed perturbation and evaluated on both narrowband and wideband data. We used data scraped from the web to build a language model for wideband data. We performed speaker adaptation on narrowband data which consisted of 117 speakers with an average speech duration of 4.7 minutes and file-level adaptation on wideband data which consisted of 119 files with an average speech duration of 5 minutes.

8.4 Speaker Adaptation Setup

Here, we were primarily interested in comparing model adaptation methods that use either one best path (called *BP* in the Results section) or a lattice (called *LAT* in the Results section) obtained from the first pass decoding for supervision. We adjusted a recipe for semi-supervised training using LF-MMI criterion (Manohar et al., 2018) to instead perform test-time adaptation. Our main hypothesis was that methods using lattices for supervision are much less likely to overfit to incorrectly transcribed segments in the adaptation data. In the past when only the best path was used for model adaptation, several techniques for data selection were required (Mathias et al., 2005; Liu et al., 2007; Walker et al., 2017). Therefore, we also compared adapting using only utterances with top 25%, 50% or 75% average utterance confidence.

We conducted model adaptation experiments in two regimes: in the first regime, we adapted all the of the acoustic model (called *ALL* in the Results section); in the second regime, we adapted only LHUC parameters inserted after every hidden layer of the acoustic model (called *LHUC* in the Results section). When adapting all the parameters, we adapted the model for three epochs, starting with the learning rate which was used in the last iteration during training. We gradually decreased the learning rate down to one tenth of the initial learning rate. This learning schedule was chosen in order to imitate continued learning of the model. When adapting LHUC parameters, we adapted the model for three epochs with a fixed learning rate of 0.7, which we found to work well in previous experiments with a similar batch size.

	no i-vectors		i-vectors	
	dev	test	dev	test
original	12.3	11.4	11.0	10.2
LHUC-LAT	11.2	10.0	10.7	9.4
LHUC-BP	11.7	10.7	10.8	9.9
ALL-LAT	11.3	9.8	10.8	9.4
ALL-BP	12.0	11.1	11.0	10.1

Table 8.1: WER (%) for speaker adaptation of the TED-LIUM model with and without i-vectors on the combined dev set from IWSLT 2010 and 2012 and the test set from IWLST 2010–2012.

	NB	WB
original	53.7	57.3
LHUC-LAT	53.6	56.7
LHUC-BP	54.1	57.9
ALL-LAT	53.0	56.5
ALL-BP	54.5	58.2

Table 8.2: WER (%) for speaker adaptation of the Somali model on narrow-band (NB) dev data and file-adaptation on wide-band (WB) test data.

8.5 Results

We conducted the first set of experiments on the TED-LIUM dataset. Adaptation of the model without i-vectors using lattices achieves 9% relative improvement when adapting the LHUC parameters and $9-14\%$ relative improvement when adapting all the parameters, whereas improvements when adapting using the best path as supervision were much smaller (Table 8.1). Adaptation of the model using i-vectors (Table 8.1) using lattices as supervision improves performance of a speaker adaptive baseline, however the relative improvement is much smaller, only 2.7% on the dev set and 7% on the test set.

We also evaluated adaptation using lattices as supervision on the Somali data. As can be seen from Table 8.2, Somali data is very challenging – the initial WER of the model is very high on both NB and WB data at 53.7% and 57.3%, respectively. These

	TED-LIUM		Somali	
	dev	test	NB	WB
original	12.3	11.4	53.7	57.3
ALL-LAT 100%	11.3	9.8	53.0	56.5
ALL-LAT 75%	11.3	9.8	53.3	56.2
ALL-LAT 50%	11.5	10.0	53.8	56.5
ALL-LAT 25%	12.0	10.4	56.0	57.0
ALL-BP 100%	12.0	11.1	54.5	58.2
ALL-BP 75%	11.7	10.5	53.8	57.8
ALL-BP 50%	11.6	10.2	53.7	57.1
ALL-BP 25%	12.0	10.4	56.0	57.2

Table 8.3: WER (%) for adaptation of the TED-LIUM model without i-vectors and the Somali model using varying fractions of the adaptation data.

results are similar to other experiments conducted on other IARPA-MATERIAL programme languages with the same TDNN-F neural network architecture (Povey et al., 2018). Here we show that adapting such a model using the best path as supervision does not reduce the WER, because the best path contains too many errors. Nevertheless, adaptation using lattices as supervision gives $0.7 - 0.8\%$ absolute improvements. Even though the relative improvement is small, it is interesting to see that using lattices as supervision allows us to improve performance at all. We believe that adapting to entire files is sub-optimal, because the speaker variance in the wide-band data might be too high. Therefore, we plan to perform per utterance adaptation experiments in the future.

One common way to prevent adaptation to erroneous first pass transcripts is to filter the adaptation data by confidences (Veselý et al., 2017), for example by the average utterance confidence. This filtering can be done by using a hard threshold, or by using only the fraction of utterances with the highest confidences. Either way one extra hyperparameter that needs to be tuned is introduced. In Table 8.3 we compare adaptation using lattices as supervision with adaptation using only best paths on various fractions of the adaptation data when adapting all parameters. We experiment with the TED-LIUM model without i-vectors, and the Somali model. As can be seen from the table, filtering utterances improves results when using best path supervision. The biggest improvement can be achieved when using only 50% of the adaptation data. Even then

the TED-LIUM model does not obtain similar performance as when adapted using lattices for supervision. Furthermore, adaptation of the Somali model using best path supervision only barely matches the performance of the not adapted baseline. This is probably due to the fact that the WER of the initial Somali model is high and that the lattice provides much more information than a combination of best path supervision and corresponding confidences. We also performed the same filtering experiment with lattices as supervision. We found that using a threshold of 75% – 100% achieves the best results. Overall, adaptation using lattice supervision does not benefit from filtering utterances as much as adaptation using best path supervision.

8.6 Summary

In this chapter we compared unsupervised model adaptation using a lattice with the best path obtained from the first pass decoding as supervision using the LF-MMI objective function. We found that using the lattice as supervision outperforms using the best path, even when we filter out utterances with too many possible errors using confidence-based data selection. This is because the lattice from the first pass decoding encodes much more information about confidences and possible phone confusions than the best path. Furthermore, we showed that when using lattices as supervision it is possible to adapt a model whose initial WER is higher than 50%, for which adapting with best path supervision often produced worse WERs than the baseline acoustic model.

Chapter 9

Conclusions

The topic of this thesis is model-based speaker adaptation of acoustic models in ASR, in particular automatic tuning of the hyperparameters of the adaptation procedure. We have argued that even the simplest gradient descent-based adaptation procedure contains many hyperparameters that need to be tuned in order to make the speaker adaptation robust. Without correctly tuned hyperparameters the speaker adaptation procedure is susceptible to overfitting to the adaptation data. For example, if we have only a small amount of adaptation data, the acoustic model could overfit to the senones seen in the adaptation data and forget about the unseen senones, it could also overfit to certain speaker characteristics seen in the adaptation data that might not be fully representative of the speaker; or if we perform unsupervised speaker adaptation, the model could overfit to errors in the labels. In the past these issues were usually prevented by limiting the number of speaker dependent parameters or the choice of regularisers that enforce the adapted model to not diverge too far from the original model. These hyperparameters had to be manually designed and tuned, which required considerable effort. In this thesis we proposed to formulate speaker adaptation as a meta-learning task, which allows us to learn speaker adaptation hyperparameters using gradient descent.

We built upon work by Andrychowicz et al. (2016) and Ravi and Larochelle (2017) that trained a neural network acting as a task specific optimiser which was able to replace and outperform traditional optimisers used in deep learning on a given task. This approach is generally called *meta-learning* because learning is happening on two levels: a *learner* is learning a task classifier and a *meta-learner* is learning how to train task specific classifiers. Here, we showed that speaker adaptation can be also formulated as a meta-learning task. A learner, the acoustic model, is adapting to speaker specific characteristics and a meta-learner, the speaker adaptation procedure, is learn-

ing the best way to adapt the acoustic model for a particular speaker. As a result, this formulation enables automatic tuning of the hyperparameters used in gradient descent-based speaker adaptation using gradient based optimisation. Our experimental results suggest that the meta-learning approach can find a speaker adaptation procedure that outperforms speaker adaptation using LHUC or all parameters with a learning rate tuned by a hyperparameter grid-search.

Subsequently, we showed that the meta-learning approach for speaker adaptation can be extended to support speaker adaptive training. Traditionally, speaker adaptive training uses speaker dependent parameters to remove speaker variability from the data such that the canonical speaker independent model can focus solely on modelling phonological variances. In the case of the model-based speaker adaptive training methods, such as SAT-LHUC, a copy of speaker dependent parameters is maintained and optimised for each speaker during the training of the model. Because of that the traditional speaker adaptive training methods have a big memory impact and thus they do not allow training of the whole acoustic model in speaker adaptive fashion. Moreover, we rarely have enough data for each speaker to estimate all parameters reliably. In contrast to previous approaches, we embed gradient based speaker adaptation directly into the training of the acoustic model. We hypothesised that this should result in the acoustic model learning parameters that are more amenable to test-time adaptation, because the optimisation process using the meta-learning objective should steer parameters of the acoustic model to regions that allow rapid adaptation. However, in our experiments we found that speaker adaptive training using SAT-LHUC or MAML did not improve performance of speaker adaptation of a strong state-of-the-art baseline TDNN model.

Consequently, we ran a series of experimental ablations that would help us determine why SAT-LHUC does not improve results. First, we tested whether the performance might be affected by batch normalisation, which is a common component of state-of-the-art models. Therefore, we trained both DNN and TDNN models with different normalisation techniques and we found that the adapted SAT-LHUC models achieved better results than their baseline counterparts when the normalisation technique does not normalise the variance of the hidden representations. However, if we want to train the best model possible it is better to use standard batch normalisation. In this experiment we noticed that the adapted SAT-LHUC DNN models that have much wider hidden layers than TDNN models generally achieve better performance after adaptation. We hypothesised that the difference in performance might be caused by the increased number of speaker dependent parameters, which might allow the SAT-

LHUC DNN model to perform some form of implicit model combination. Thus in the second experiment we tried to increase the hidden layer width of the TDNN models by using TDNN-F models. However, this change did not affect the performance of the adapted SAT-LHUC TDNN-F models compared to the adapted SAT-LHUC DNN models. Based on this observation we hypothesised that the model size might also be important for SAT-LHUC training. In the following experiments we increased the model size of TDNN models by making their hidden layers wider and we made the TDNN-F models bigger by changing the size of the bottleneck layers. Increasing the model size had positive effect on the performance of the models, however there was no difference between the adapted baseline and the SAT-LHUC models. We also tested various activation functions and we tried to change the optimiser, but we did not observe any differences. As a result a way of speaker adaptive training still remains an open question.

Finally, in the last chapter we compared unsupervised model adaptation using a lattice with the best path obtained from the first pass decoding as supervision. Our experiments showed that using the lattice as supervision achieves better results than using the best path, even when confidence-based data selection is used to remove transcripts with many possible errors. This is due to the fact that the lattice from the first pass decoding contains much more information, such as confidence and phonetic confusions, than the best path. We found that the use of a lattice as supervision is particularly important when adapting all parameters, when over-fitting to incorrect first-pass transcriptions is a particular problem. Moreover, we showed that when using lattices as supervision it is possible to adapt a model whose initial WER is higher than 50%, for which adapting with best path supervision often produced worse WERs than the speaker-independent baseline.

9.1 Future work

In the previous chapters we described how meta-learning approaches can be used for speaker adaptation and speaker adaptive training. Here we describe ideas which could lead to further improvements in the performance of ASR systems. Moreover, they could also improve our understanding of speaker adaptation and speaker adaptive training.

Improving Meta-Learners for Speaker Adaptation

The parameterisation of the meta-learner that we used throughout this thesis enables us to find a good learning rate schedule for each adapted layer given a predefined number of full-batch adaptation steps. To do that the meta-learner uses different parameters for each layer, which allows it to learn different strategies for each layer. As a result, it can also learn which layers are suitable for adaptation. As we described in Section 4.6, it is theoretically possible to train the meta-learner to automatically optimise other hyperparameters such as the linear combination of various loss functions, predicting per example importance as a form of filtering of adaptation examples with low confidence or learning an adaptive number of adaptation steps. We think that it would be interesting to test whether this more expressive parameterisation of the meta-learner would lead to an improved performance of speaker adaptation, especially in the unsupervised speaker adaptation scenario. Additionally, it would be good to train a meta-learner that can use lattice supervision with LF-MMI. This is because models trained with LF-MMI achieve significantly better performance than the models trained with cross-entropy and unsupervised speaker adaptation with lattice supervision outperforms using the best path.

The big issue with the coordinate-wise meta-learner is that it is very computationally and memory expensive. In order to scale this method to larger models we would need GPUs with much larger memory. Alternatively, we could adapt only a subset of the model. One obvious way would be to not adapt the output layer, which consists of around 50% of all parameters in the small models or 30% of all parameters of the biggest models tested in this thesis. We believe that not adapting the output layer should not restrict the power of speaker adaptation. This is based on our finding in Table 5.8, which shows that meta-learners sometimes learn not to adapt the output layer. Therefore, not adapting the last hidden layer might be seen as a good inductive bias for meta-learner training. Moreover, Hoffer et al. (2018) showed that it is possible to train a neural network from scratch with a fixed, randomly initialised output layer without any loss of performance. However, for big models meta-learning would still be too memory-inefficient. Therefore, it would be interesting to train a meta-learner that would perform adaptation in a low dimensional space that might better capture different speaker characteristics. In some way this can be seen as a form of factorised hidden layers (Samarakoon and Sim, 2016a) or context adaptive networks (Delcroix et al., 2018a). Finally, once we can scale meta-learning models to larger models it

would be interesting to evaluate the meta-learning approaches for speaker adaptation of sequence-to-sequence models and also to scale these approaches to larger adaptation tasks such as dialect adaptation or domain adaptation.

Better Understanding of Speaker Adaptive Training

As we saw in Chapter 5 and Chapter 7, model-based speaker adaptive training, namely speaker adaptive training of LHUC parameters, does not improve the performance of adapted TDNN models. In Chapter 7 we evaluated various neural network architectures in order to understand why SAT-LHUC is not working. We hypothesised that SAT-LHUC might not be performing variance normalisation but some form of model combination. Thus, it would be interesting to test this hypothesis with speaker dependent model pruning which we describe in the next section. We believe that speaker dependent model pruning might prove or disprove the model combination hypothesis by showing that we obtain significantly different sub-networks for different speakers.

Another possible hypothesis that might explain why speaker adaptive training does not yield any improvements compared to the standard training is that the state-of-the-art speaker independent models have sufficient capacity to model all speaker variability and can benefit only a little from speaker adaptation. A similar thing was recently observed by Yin et al. (2020), who showed that MAML training might converge to an optimum which can memorise all tasks and therefore does not need to be adapted to a new task. In order to overcome this, Yin et al. (2020) proposed to use meta-regularisation that forces the model to perform a few adaptation steps, because otherwise the model achieves inferior performance. In a sense this is similar to what we see with the other SAT methods, for example when we use fMLLR input features or i-vectors, that do not work or work significantly worse with mismatched input features. Therefore, we would like to try to use this meta-regularisation method for model-based speaker adaptive training.

Speaker Dependent Model Pruning

So far all the work on speaker adaptation focused on improving the accuracy of the adapted models in the test conditions. However, accuracy is not the only metric that is important for successful deployment of ASR models. Another important metric is speed, which might be improved by reducing the size of the acoustic model. Therefore, with a better understanding of speaker adaptive training it might be possible to explore

joint speaker adaptation and speaker dependent model pruning. Model pruning and model compression were already explored for ASR models (Mantena and Sim, 2016; Liu et al., 2014), however they were never used in a speaker dependent fashion.

Our hypothesis is based on the fact that the speaker independent model has to model both speaker variability and phonological variance. Therefore, when we use the model only for a target speaker, the model might be performing redundant computations trying to remove inter-speaker variability. One way of enforcing smaller model size would be to enforce sparse LHUC representations with L0 regularisation (Louizos et al., 2018). In practice however enforcing sparse layers can be quite challenging. Another way to achieve LHUC sparsity would be to use pruning methods like Optimal Brain Damage (LeCun et al., 1990) and Optimal Brain Surgeon (Hassibi et al., 1993). However, it is important to note that smaller models do not necessarily imply faster inference, because smaller models can become less confident, which results in slower decoding. On the other hand, when we compare models trained with cross-entropy and models trained with LF-MMI, we can see that it is possible to train much smaller models that are significantly faster in decoding.

Bibliography

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

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 *ICASSP*.

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 *ICASSP*.

Anastasakos, T., McDonough, J., Schwartz, R., and Makhoul, J. (1996). A compact model for speaker-adaptive training. In *ICSLP*.

Andreou, A. (1994). Experiments in vocal tract normalization. In *Proc. the CAIP Workshop: Frontiers in Speech Recognition II, 1994*.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *NIPS*.

Antoniou, A., Edwards, H., and Storkey, A. (2019). How to train your MAML. In *ICLR*.

Arora, S., Li, Z., and Lyu, K. (2019). Theoretical analysis of auto rate-tuning by batch normalization. In *ICLR*.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bacchiani, M. and Roark, B. (2003). Unsupervised language model adaptation. In *ICASSP*.

Bahl, L., Brown, P., De Souza, P., and Mercer, R. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *ICASSP*.

Batlle, E., Nadeu, C., and Fonollosa, J. A. (1998). Feature decorrelation methods in speech recognition. a comparative study. In *ICSLP*.

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. *The annals of mathematical statistics*, 41(1):164–171.

Bell, P., Swietojanski, P., Driesen, J., Sinclair, M., McInnes, F., and Renals, S. (2014). The UEDIN ASR systems for the IWSLT 2014 evaluation. In *Proc. IWSLT*.

Bengio, S., Bengio, Y., and Cloutier, J. (1995). On the search for new learning rules for ANNs. *Neural Processing Letters*, 2(4):26–30.

Bengio, Y., Bengio, S., and Cloutier, J. (1990). *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche .

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bourlard, H. and Morgan, N. (1993). Continuous speech recognition by connectionist statistical methods. *IEEE Transactions on Neural Networks*, 4(6):893–909.

Bourlard, H. A. and Morgan, N. (1994). *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*.

Chollet, F. et al. (2015). Keras. `https://github.com/keras-team/keras`.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In *NIPS*.

Cotter, N. E. and Conwell, P. R. (1990). Fixed-weight networks can learn. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 553–559. IEEE.

Cui, J., Kingsbury, B., Ramabhadran, B., Sethy, A., Audkhasi, K., Cui, X., Kislal, E., Mangu, L., Nussbaum-Thom, M., Picheny, M., Tüske, Z., Golik, P., Schlüter, R., Ney, H., Gales, M. J. F., Knill, K. M., Ragni, A., Wang, H., and Woodland, P. C. (2016). Multilingual representations for low resource speech recognition and keyword search. In *IEEE ICASSP*.

Cui, X., Goel, V., and Saon, G. (2017). Embedding-based speaker adaptive training of deep neural networks. In *Interspeech*.

Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Audio, Speech, and Language Processing*, 28(4):357–366.

Deecke, L., Murray, I., and Bilen, H. (2019). Mode normalization. In *ICLR*.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Delcroix, M., Kinoshita, K., Ogawa, A., Huemmer, C., and Nakatani, T. (2018a). Context adaptive neural network based acoustic models for rapid adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(5):895–908.

Delcroix, M., Watanabe, S., Ogawa, A., Karita, S., and Nakatani, T. (2018b). Auxiliary feature based adaptation of end-to-end ASR systems. In *Interspeech*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

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 *ICASSP*.

Dighe, P., Asaei, A., and Bourlard, H. (2017). Low-rank and sparse soft targets to learn better DNN acoustic models. In *ICASSP*.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. Wiley.

Eide, E. and Gish, H. (1996). A parametric approach to vocal tract length normalization. In *Interspeech*.

Fainberg, J., Klejch, O., Loweimi, E., Bell, P., and Renals, S. (2019a). Acoustic model adaptation from raw waveforms with SincNet. In *ASRU*.

Fainberg, J., Klejch, O., Renals, S., and Bell, P. (2019b). Lattice-based lightly-supervised acoustic model training. In *Interspeech*.

Fainberg, J., Renals, S., and Bell, P. (2017). Factorised representations for neural network adaptation to diverse acoustic environments. In *Interspeech*.

Falavigna, D., Matassoni, M., Jalalvand, S., Negri, M., and Turchi, M. (2017). DNN adaptation by automatic quality estimation of ASR hypotheses. *Computer Speech & Language*, 46:585–604.

Federico, M., Bentivogli, L., Paul, M., and Stücker, S. (2011). Overview of the IWSLT 2011 evaluation campaign. In *IWSLT*.

Federico, M., Cettolo, M., Bentivogli, L., Paul, M., and Stüker, S. (2012). Overview of the IWSLT 2012 evaluation campaign. In *IWSLT*.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Finn, C. and Levine, S. (2018). Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR*.

Fraga-Silva, T., Gauvain, J.-L., and Lamel, L. (2011). Lattice-based unsupervised acoustic model training. In *ICASSP*.

Gales, M. (1998). Maximum likelihood linear transformations for HMM-based speech recognition. *Computer speech & language*, 12(2):75–98.

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

Gales, M. J. F., Knill, K. M., and Ragni, A. (2017). Low-resource speech recognition and keyword-spotting. In *SPECOM*.

Gangireddy, S. R., Swietojanski, P., Bell, P., and Renals, S. (2016). Unsupervised adaptation of recurrent neural network language models. In *Interspeech*.

Gauvain, J.-L. and Lee, C.-H. (1994). Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Audio, Speech, and Language Processing*, 2(2):291–298.

Gelbart, M. A., Snoek, J., and Adams, R. P. (2014). Bayesian optimization with unknown constraints. *arXiv preprint arXiv:1403.5607*.

Gemello, R., Mana, F., Scanzio, S., Laface, P., and De Mori, R. (2007). Linear hidden transformations for adaptation of hybrid ANN/HMM models. *Speech Communication*, 49(10-11):827–835.

Ghahremani, P., BabaAli, B., Povey, D., Riedhammer, K., Trmal, J., and Khudanpur, S. (2014). A pitch extraction algorithm tuned for automatic speech recognition. In *ICASSP*.

Ghoshal, A., Swietojanski, P., and Renals, S. (2013). Multilingual training of deep neural networks. In *ICASSP*.

Gibson, M. and Hain, T. (2006). Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition. In *ICSLP*.

Golik, P., Tüske, Z., Schlüter, R., and Ney, H. (2015). Convolutional neural networks for acoustic modeling of raw time signal in lvcsr. In *Interspeech*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.

Graves, A. (2012). Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Graves, A. (2016). Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*.

Gretter, R. and Riccardi, G. (2001). On-line learning of language models with word error probability distributions. In *ICASSP*.

Gu, J., Wang, Y., Chen, Y., Cho, K., and Li, V. O. K. (2018). Meta-learning for low-resource neural machine translation. In *ACL*.

Hassibi, B., Stork, D. G., and Wolff, G. J. (1993). Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*. IEEE.

Heigold, G., Vanhoucke, V., Senior, A., Nguyen, P., Ranzato, M., Devin, M., and Dean, J. (2013). Multilingual acoustic models using distributed deep neural networks. In *ICASSP*.

Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29.

Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001). Learning to learn using gradient descent. In *ICANN*.

Hoffer, E., Hubara, I., and Soudry, D. (2018). Fix your classifier: the marginal value of training the last weight layer. In *ICLR*.

Hsu, J.-Y., Chen, Y.-J., and Lee, H.-y. (2019). Meta learning for end-to-end low-resource speech recognition. *arXiv preprint arXiv:1910.12094*.

Huang, H. and Sim, K. C. (2015). An investigation of augmenting speaker representations to improve speaker normalisation for DNN-based speech recognition. In *ICASSP*.

Huang, J.-T., Li, J., Yu, D., Deng, L., and Gong, Y. (2013). Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *ICASSP*.

Huang, Y. and Gong, Y. (2015). Regularized sequence-level deep neural network model adaptation. In *Interspeech*.

Huang, Z., Li, J., Siniscalchi, S. M., Chen, I.-F., Wu, J., and Lee, C.-H. (2015a). Rapid adaptation for deep neural networks through multi-task learning. In *Interspeech*.

Huang, Z., Siniscalchi, S. M., Chen, I.-F., Wu, J., and Lee, C.-H. (2015b). Maximum a posteriori adaptation of network parameters in deep models. *arXiv preprint arXiv:1503.02108*.

Hwang, M.-Y. and Huang, X. (1992). Subphonetic modeling with Markov states-senone. In *ICASSP*.

Ioffe, S. (2017). Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NeurIPS*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. (2017). Decoupled neural interfaces using synthetic gradients. In *ICML*.

Johnson, D., Ellis, D., Oei, C., Wooters, C., Faerber, P., Morgan, N., and Asanovic, K. (2004). ICSI QuickNet software package. `http://www1.icsi.berkeley.edu/Speech/qn.html`.

Juang, B.-H., Levinson, S., and Sondhi, M. (1986). Maximum likelihood estimation for multivariate mixture observations of Markov chains. *IEEE Transactions on Information Theory*, 32(2):307–309.

Jurafsky, D. and Martin, J. H. (2019). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 3st edition.

Kaiser, J., Horvat, B., and Kacic, Z. (2000). A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Sixth International Conference on Spoken Language Processing*.

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. *Interspeech*.

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

Klejch, O., Bell, P., and Renals, S. (2016). Punctuated transcription of multi-genre broadcasts using acoustic and lexical approaches. In *SLT*.

Klejch, O., Bell, P., and Renals, S. (2017). Sequence-to-sequence models for punctuated transcription combining lexical and acoustic features. In *ICASSP*.

Klejch, O., Fainberg, J., and Bell, P. (2018). Learning to adapt: a meta-learning approach for speaker adaptation. In *Interspeech*.

Klejch, O., Fainberg, J., Bell, P., and Renals, S. (2019a). Lattice-based unsupervised test-time adaptation of neural network acoustic models. *arXiv preprint arXiv:1906.11521*.

Klejch, O., Fainberg, J., Bell, P., and Renals, S. (2019b). Speaker adaptive training using model agnostic meta-learning. In *ASRU*.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *ICASSP*.

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. (2011). One shot learning of simple visual concepts. In *CogSci*.

Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3(1):23–43.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *NIPS*.

Lee, L. and Rose, R. C. (1996). Speaker normalization using efficient frequency warping procedures. In *ICASSP*.

Li, B. and Sim, K. C. (2010). Comparison of discriminative input and output transformations for speaker adaptation in the hybrid nn/hmm systems. In *Interspeech*.

Li, X. and Bilmes, J. (2006). Regularized adaptation of discriminative classifiers. In *ICASSP*.

Liao, H. (2013). Speaker adaptation of context dependent deep neural networks. In *ICASSP*.

Liepins, R., Germann, U., Barzdins, G., Birch, A., Renals, S., Weber, S., van der Kreeft, P., Bourlard, H., Prieto, J., Klejch, O., et al. (2017). The SUMMA platform prototype. In *Software Demonstrations ACL*.

Liu, C., Zhang, Z., and Wang, D. (2014). Pruning deep neural networks by optimal brain damage. In *Interspeech*.

Liu, S.-H., Chu, F.-H., Lin, S.-H., and Chen, B. (2007). Investigating data selection for minimum phone error training of acoustic models. In *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE.

Louizos, C., Welling, M., and Kingma, D. (2018). Learning sparse neural networks through L0 regularization. In *ICLR*.

Mana, F., Weninger, F., Gemello, R., and Zhan, P. (2019). Online batch normalization adaptation for automatic speech recognition. In *ASRU*.

Manohar, V., Hadian, H., Povey, D., and Khudanpur, S. (2018). Semi-supervised training of acoustic models using lattice-free MMI. In *ICASSP*.

Mantena, G. and Sim, K. C. (2016). Entropy-based pruning of hidden units to reduce DNN parameters. In *SLT*.

Mathias, L., Yegnanarayanan, G., and Fritsch, J. (2005). Discriminative training of acoustic models applied to domains with unreliable transcripts [speech recognition applications]. In *ICASSP*.

Meng, Z., Li, J., and Gong, Y. (2019). Adversarial speaker adaptation. In *ICASSP*.

Miao, Y., Zhang, H., and Metze, F. (2015). Speaker adaptive training of deep neural network acoustic models using i-vectors. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(11):1938–1949.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2018). A simple neural attentive meta-learner. In *ICLR*.

Morgan, N. and Bourlard, H. (1990). Continuous speech recognition using multilayer perceptrons with hidden Markov models. In *ICASSP*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *ICML*.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate of $(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376.

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. *Eurospeech*.

Nichol, A. and Schulman, J. (2018). Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2.

Padmanabhan, M., Saon, G., and Zweig, G. (2000). Lattice-based unsupervised mllr for speaker adaptation. In *ASR2000-automatic speech recognition: challenges for the New Millenium ISCA Tutorial and Research Workshop (ITRW)*.

Palaz, D., Doss, M. M., and Collobert, R. (2015). Convolutional neural networks-based continuous speech recognition using raw speech signal. In *ICASSP*.

Paul, M., Federico, M., and Stücker, S. (2010). Overview of the IWSLT 2010 evaluation campaign. In *IWSLT*.

Peddinti, V., Povey, D., and Khudanpur, S. (2015). A time delay neural network architecture for efficient modeling of long temporal contexts. In *Interspeech*.

Povey, D. (2005). *Discriminative training for large vocabulary speech recognition*. PhD thesis, University of Cambridge.

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. *Interspeech*.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlíček, P., Qian, Y., Schwarz, P., Silovský, J., Stemmer, G., and Veselý, K. (2011). The Kaldi speech recognition toolkit. In *ASRU*.

Povey, D. and Kingsbury, B. (2007). Evaluation of proposed modifications to mpe for large scale discriminative training. In *ICASSP*.

Povey, D., Peddinti, V., Galvez, D., Ghahrmani, P., Manohar, V., Na, X., Wang, Y., and Khudanpur, S. (2016). Purely sequence-trained neural networks for ASR based on lattice-free MMI. *Interspeech*.

Povey, D., Zhang, X., and Khudanpur, S. (2014). Parallel training of deep neural networks with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767.

Price, R., Iso, K.-i., and Shinoda, K. (2014). Speaker adaptation of deep neural networks using a hierarchy of output layers. In *SLT*.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *ICLR*.

Roth, J., Chaudhuri, S., Klejch, O., Marvin, R., Gallagher, A., et al. (2019). AVA-ActiveSpeaker: An audio-visual dataset for active speaker detection. *arXiv preprint arXiv:1901.01342*.

Rousseau, A., Deléglise, P., and Estève, Y. (2012). TED-LIUM: an automatic speech recognition dedicated corpus. In *LREC*.

Rousseau, A., Deléglise, P., and Estève, Y. (2014). Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *LREC*.

Rownicka, J., Bell, P., and Renals, S. (2018). Analyzing deep CNN-based utterance embeddings for acoustic model adaptation. In *SLT*.

Rownicka, J., Bell, P., and Renals, S. (2019). Embeddings for DNN speaker adaptive training. *ASRU*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Runarsson, T. P. and Jonsson, M. T. (2000). Evolution and design of distributed learning rules. In *2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (Cat. No. 00*, pages 59–63. IEEE.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Sainath, T. N., Kingsbury, B., Soltau, H., and Ramabhadran, B. (2013). Optimization techniques to improve training speed of deep neural networks for large speech tasks. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(11):2267–2276.

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 *Interspeech*.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*.

Samarakoon, L. and Sim, K. C. (2015). Learning factorized feature transforms for speaker normalization. In *ASRU*.

Samarakoon, L. and Sim, K. C. (2016a). Factorized hidden layer adaptation for deep neural network based acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2241–2250.

Samarakoon, L. and Sim, K. C. (2016b). Subspace LHUC for fast adaptation of deep neural network acoustic models. In *Interspeech*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *ICML*.

Saon, G., Soltau, H., Nahamoo, D., and Picheny, M. (2013). Speaker adaptation of neural network acoustic models using i-vectors. In *ASRU*.

Schmidhuber, J. (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139.

Schmidhuber, J. (1993). A neural network that embeds its own meta-levels. In *IEEE International Conference on Neural Networks*, pages 407–412. IEEE.

Seide, F., Li, G., and Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *NIPS*.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., and Khudanpur, S. (2018). X-vectors: Robust DNN embeddings for speaker recognition. In *ICASSP*.

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

Stolcke, A. (2001). Error modeling and unsupervised language modeling. In *NIST LVCSR Workshop*.

Swietojanski, P., Bell, P., and Renals, S. (2015). Structured output layer with auxiliary targets for context-dependent acoustic modelling. In *Interspeech*.

Swietojanski, P., Li, J., and Renals, S. (2016). Learning hidden unit contributions for unsupervised acoustic model adaptation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14:1450–1463.

Swietojanski, P. and Renals, S. (2014). Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *SLT*.

Swietojanski, P. and Renals, S. (2016). SAT-LHUC: Speaker adaptive training for learning hidden unit contributions. In *ICASSP*.

Tan, T., Qian, Y., and Yu, K. (2016). Cluster adaptive training for deep neural network based acoustic model. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 24(3):459–468.

Thrun, S. and Pratt, L. (1998). Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer.

Trmal, J., Zelinka, J., and Müller, L. (2010). On speaker adaptive training of artificial neural networks. In *Interspeech*.

Tsunoo, E., Klejch, O., Bell, P., and Renals, S. (2017). Hierarchical recurrent neural network for story segmentation using fusion of lexical and acoustic features. In *ASRU*.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Valtchev, V., Odell, J. J., Woodland, P. C., and Young, S. J. (1997). Mmie training of large vocabulary recognition systems. *Speech Communication*, 22(4):303–314.

Variani, E., Lei, X., McDermott, E., and Lopez-Moreno, I. (2014). Text dependent speaker verification using deep neural networks. In *ICASSP*.

Veselý, K., Burget, L., and Černocký, J. (2017). Semi-supervised DNN training with word selection for ASR. In *Interspeech*.

Veselý, K., Ghoshal, A., Burget, L., and Povey, D. (2013). Sequence-discriminative training of deep neural networks. In *Interspeech*.

Veselý, K., Watanabe, S., Žmolíková, K., Karafiát, M., Burget, L., and Černocký, J. H. (2016). Sequence summarizing neural network for speaker adaptation. In *ICASSP*.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. In *NIPS*.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.

Wakita, H. (1977). Normalization of vowels by vocal-tract length and its application to vowel identification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(2):183–192.

Walker, S., Pedersen, M., Orife, I., and Flaks, J. (2017). Semi-supervised model training for unbounded conversational speech recognition. *arXiv preprint arXiv:1705.09724*.

Wang, Z. Q. and Wang, D. (2017). Unsupervised speaker adaptation of batch normalized acoustic models for robust ASR. In *ICASSP*.

Woodland, P. C. (2001). Speaker adaptation for continuous density HMMs: A review. In *ISCA Workshop on Adaptation Methods for Speech Recognition*.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A., and Le, Q. V. (2019a). Adversarial examples improve image recognition. *arXiv preprint arXiv:1911.09665*.

Xie, X., Liu, X., Lee, T., Hu, S., and Wang, L. (2019b). BLHUC: Bayesian learning of hidden unit contributions for deep neural network speaker adaptation. In *ICASSP*.

Xie, X., Liu, X., Lee, T., and Wang, L. (2019c). Fast DNN acoustic model speaker adaptation by learning hidden unit contribution features. *Interspeech*.

Xue, J., Li, J., and Gong, Y. (2013). Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, pages 2365–2369.

Xue, J., Li, J., Yu, D., Seltzer, M., and Gong, Y. (2014). Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In *ICASSP*.

Yin, M., Tucker, G., Zhou, M., Levine, S., and Finn, C. (2020). Meta-learning without memorization. In *ICLR*.

Young, S. J., Odell, J. J., and Woodland, P. C. (1994). Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the workshop on Human Language Technology*, pages 307–312. Association for Computational Linguistics.

Younger, A. S., Conwell, P. R., and Cotter, N. E. (1999). Fixed-weight on-line learning. *IEEE Transactions on Neural Networks*, 10(2):272–283.

Younger, A. S., Hochreiter, S., and Conwell, P. R. (2001). Meta-learning with backpropagation. In *IJCNN*.

Yu, D. and Deng, L. (2016). *Automatic speech recognition.* Springer.

Yu, D., Yao, K., Su, H., Li, G., and Seide, F. (2013). KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *ICASSP*.

Yu, K., Gales, M., Wang, L., and Woodland, P. C. (2010). Unsupervised training and directed manual transcription for LVCSR. *Speech Communication*, 52(7-8):652–663.

Zhang, C. and Woodland, P. C. (2015). Parameterised sigmoid and ReLU hidden activation functions for DNN acoustic modelling. In *Interspeech*.

Zhao, Y., Li, J., and Gong, Y. (2016). Low-rank plus diagonal adaptation for deep neural networks. In *ICASSP*.

Zhao, Y., Li, J., Kumar, K., and Gong, Y. (2017). Extended low-rank plus diagonal adaptation for deep and recurrent neural networks. In *ICASSP*.