Master's Thesis

Speaker Adaptation For Deep Neural Network Based Automatic Speech Recognition Systems

SONY

Author:
Anis Chihi
Institute for Data Processing
Technische Universität München

Supervising Professor:
Prof. Dr. M. Kleinsteuber
Institute for Data Processing
Technische Universität München

Supervisor:
M.Sc Alexander Sagel
Institute for Data Processing
Technische Universität München

Advisor:
Dr. Thomas Kemp
Sony Europe Ltd.
EuTEC Speech and Sound Group

August 26, 2016

Master's thesis

Speaker Adaptation For Deep Neural Network Based Automatic Speech Recognition Systems

Anis Chihi

August 26, 2016

Institute for Data Processing
Technische Universität München

Anis Chihi. *Speaker Adaptation For Deep Neural Network Based Automatic Speech Recognition Systems.* Master's thesis, Technische Universität München, Munich, Germany, 2016.

Supervised by Prof. Dr. M. Kleinsteuber and M.Sc A.Sagel ; submitted on August 26, 2016 to the Department of Electrical Engineering and Information Technology of the Technische Universität München.

Institute for Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de`.

Acknowledgment

I would like to first thank my Advisor Thomas Kemp for his constant support and wise advice throughout my stay at the Sony Technology Center of Stuttgart. His brilliant expertise and scientific guidance made this project possible.
I am also grateful to Professor Dr. Martin Kleinsteuber who has accepted to be my supervising professor and without whom this thesis would not have been possible. I would like also to thank him for all the great courses (*Information retrieval in high dimensional data, Non-convex-optimization for analyzing big data & Hauptseminar Datenverarbeitung*) which gave me the taste for machine learning.
I am also thankful to Alexander Sagel for accepting to be my academic supervisor and for helping me throughout the last stages of this work.
And last but not least, I would like to thank all my friends, family, fellow students and colleagues at Sony for their unconditional support during this six intense months. They made this adventure pleasant as well as a great experience.

Stuttgart, August 2016

Abstract

In the recent years, the use of deep neural networks for acoustic modeling of automatic speech recognition (ASR) systems outperformed the formerly Gaussian mixture model based systems. Still, the training-testing mismatch between the speech data used for training the models and the test conditions remains a hurdle towards achieving a better recognition performance. Within this context, speaker adaptation techniques have been proposed as a possible solution to overcome this issue. Nevertheless, adapting deep neural networks remains a challenging task, since DNNs can suffer from catastrophic forgetting through adaptation. In this work, we investigate several techniques proposed by Liao [1] and Yu et al. [2] which consist respectively in training portions of the DNN and using Kullback-Leibler divergence regularized adaptation. We conduct these adaptation experiments on two different ASR systems over different sets of hyperparameters and adaptation data sizes for supervised and unsupervised adaptation settings. We show in the scope of this work how do these methods compare to each other in terms of gain in recognition accuracy and that these techniques can provide up to 6.9% absolute error reduction against a very strong state of the art ASR system.

Contents

1 Introduction

1.1 Automatic Speech Recognition and the Adaptation Issue

Automatic speech recognition (ASR) has been a very active research area in the past decades since it represents one important bridge to forward the human-machine communication. Nevertheless speech was never in the past the most obvious approach for human machine interaction. This is partly due to the technology constraints in this research area and its inability to provide efficient and reliable systems for most real usage conditions. Furthermore, alternative communication modalities like mouse and keyboards outperformed significantly speech recognition systems in many situations and real life scenarios.

In the recent years however, we observed a growth of interest towards automatic speech recognition systems. The rush toward this trend can be explained by the progress made in certain key areas such as the actual computational power availability. This made the training of more powerful and complex models possible, and hence increased the performance of the ASR systems. The access to much more data due to the advance of the Internet played also an important role in this performance enhancement, since training models on big data collected from real life scenarios can make systems more robust. Last but not least is the increasing popularity of mobile wearable devices, in-vehicule infotainment systems and intelligent living room devices that require more convenient interaction modalities than classic keyboard and mouse. Speech represents in this case a more comfortable mean for human-machine interaction, since it is a skill that the majority of people already have.

The development of deep learning techniques using deep neural networks (DNNs) is equally important to the performance enhancement of ASR systems. Hence, using these techniques in acoustic modeling have been shown to outperform state-of-the art Gaussian Mixture Models (GMMs) used for that purpose [3]. A combination of this set of techniques led to more than 1/3 error rate reduction over conventional GMM based frameworks [4].

Even with the gain in robustness achieved through the introduction of deep learning techniques in the ASR framework, the performance can vary from one speaker to another. This variability in performance is mainly due to the difference between the training and the testing conditions. Speaking styles, accents and speech production anatomy vary between different speakers and cause the ASR system to perform differently over various speakers. To overcome this problem, speaker adaptation techniques have been developed to improve the systems accuracy, and fill the performance gap caused by the variant

nature of speech. Several adaptation techniques were developed for GMM based ASR systems, but since GMMs are generative models, these adaptation techniques cannot be directly applied to DNN based ASR systems.

1.2 Scope of the Thesis

Deep learning usage in acoustic modeling started making its impact in 2010 [5]. Since then, DNN adaptation methods are being developed to enhance the performance of the deployed ASR systems. Two challenges might be faced in this quest. The first one consists in the nature of this task, since neural networks suffer when adapted from a phenomenon called catastrophic forgetting [6], which consists in neutral network being unable to adapt without at some point losing the information already stored within them. The second one is the high dimensionality of the neural network hyperparameters that could be potentially considered in this task. The effectiveness of hyperparameter values used in neural network adaptation can be highly dependent from the considered system (DNN architecture, used training data ...). This makes the values proposed in literatures dealing with this subject almost impossible to directly be taken over. In the scope of this thesis our aim is to implement several DNN adaptation techniques proposed by Hank Liao in [1] and the method based on Kullback-Leibler divergence regularization proposed by Yu et al. [2] and evaluate their respective performances against a baseline. For this purpose, we will try to investigate different hyperparameters used in the adaptation framework and try to examine their impact in different adaptation scenarios. We aim in the end of this thesis to test these methods on a state of the art ASR system developed within the SONY European Research Center, and potentially come up with a recommendation on how this ASR system might be adapted.

1.3 Thesis Organization

The organization of the Thesis is the following. Chapter 2 presents the basic architecture of the ASR systems with a focus on the conventional acoustic models based on GMMs and hidden Markov models (HMMs). Chapter 3 gives an overview on the DNNs background and how they are used for acoustic modeling. Chapter 4 gives a general overview over adaptation techniques and focuses on the methods considered in the scope of this thesis. We describe the experiments we conducted over two ASR systems with different architectures in chapters 5 and 6 and then conclude this thesis in chapter 7.

2 Automatic Speech Recognition Systems

2.1 Basic Framework of ASR Systems

A typical architecture of an ASR system can be seen in the figure 2.1 below. The four basic components that compose the ASR framework are the following: Signal processing and feature extraction, acoustic model, language model and hypothesis search [4].

The Signal processing and feature extraction part includes commonly dividing the speech signal into a sequence of frames. Each frame can then be analyzed in an independent way and be represented by a single feature vector [7]. These frames are then converted from time to frequency domain and feature vectors are extracted to be used as input for the acoustic model. Typically Mel-frequency cepstral (MFCC) coefficients are used as feature vectors [8].

The first component in recognizing speech is the acoustic model (AM). It establishes a statistical representation for the feature vectors extracted from the speech signal. Creating an AM consists basically in compiling audio recordings and their transcriptions into a statistical representations of the words sounds [9]. Hidden Markov Models (HMMs) were used to comprise with the time variable nature of speech, and Gaussian Mixture Models (GMMs) were used to model the observation probability distributions of the speech features over the different states of the HMMs [4], in other words it determines how well a HMM state fits a speech feature that itself represents an acoustic input [3]. In the 1990s most state of the art ASR systems used GMM-HMM acoustic models, until the recent years when systems such as deep neural networks (DNNs) became computationally feasible and performed better in terms of statistical modeling [4]. AMs will be dealt with in more detail in section 2.2.

The second important component that takes part in elaborating the hypothesis search is the language model (LM). The LM estimates the probabilities of the hypothesized sequence of words [4]. A very good example is "It's fun to recognize speech" and "It's

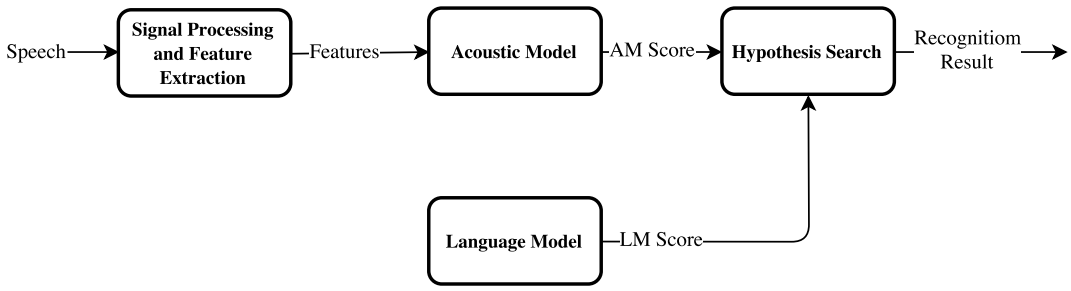


Figure 2.1: Basic architecture of ASR systems [4]

fun to wreck a nice beach". Both utterances sound about the same. In this case the LM helps recognize the most probable sequence of words independently of its acoustic properties, which in that case would probably be the first sequence.

Both AM and LM components will deliver a score as for the word sequence hypothesis. The word sequence with highest score will eventually be outputted as the recognition result [4]. This process is also called decoding and can be performed using weighted finite state transducers (WFST) [10].

In this work, we will concentrate our discussion on the theory behind the acoustic models since it is the part where we will intervene in the context of the adaptation task. Speech signal preprocessing, language modeling and hypothesis search fall out the scope of this thesis and will not be discussed any further.

2.2 Conventional Acoustic Models

In this Chapter we will have an overview over the conventional acoustic models used for speech recognition tasks before the emergence of DNN based acoustic models. Since DNN-HMM models are generated partly over former GMM-HMM systems, we find it important to get a basic understanding of the acoustic modeling process. Because of the complexity of this subject and since a full overview of the GMM-HMM based acoustic models is beyond the scope of this work, we will instead refer to previous articles for a full review of the theory behind large vocabulary continuous speech recognition Systems (LVCSR) [11], [12], [13].

2.2.1 Hidden Markov Models

The Hidden Markov Models (HMMs) represent one of the corner stones of the current state-of-the-art ASR systems. As described by Gales and Young [11], in the context of using HMMs, each input speech signal is converted framewise to a sequence of feature vectors $X_{1:T} = x_1, ..., x_T$ during the signal preprocessing step. Together, these feature vectors represent the acoustic sequence associated to a certain utterance $W_{1:L} = w_1, ..., w_L$, where w_i is the $i-th$ word of the utterance. Now the goal of the recognition is to find the most likely word sequence W given the acoustic features X. In other words we need to find:

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(W|X) \tag{2.1}$$

Using the Bayes rule, we can rewrite the previous equation as follows:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \frac{P(W)p(X|W)}{p(X)} \tag{2.2}$$

Where in this case P denotes a probability and p denotes a probability density function. We can leave $p(X)$ since it is irrelevant for the optimization. This makes our problem equivalent to finding:

$$\hat{W} = \underset{W}{\operatorname{argmax}} P(W)p(X|W) \tag{2.3}$$

The prior probability $P(W)$ and the likelihood $p(X|W)$ are determined respectively by the language model and the acoustic model. Each of the words w_i is composed of basic units of sound called *phones*. For example the word "hand" has four phones /h /ae /n /d. About 40 of these phones a required to model words in English language. Each word w is composed of a concatenation of K_w phones defined by a pronunciation dictionary. This sequence of base phones can be written as the following: $q_{1:K_w}^{(w)} = q_1, ..., q_{K_w}$.

Since every sequence of words can be pronounced in a different way, we compute the likelihood $p(X|W)$ over multiple pronunciations:

$$p(X|W) = \sum_{Q} p(X|Q)P(Q|W) \tag{2.4}$$

with Q being a particular sequence of pronunciations valid for W. $P(Q|W)$ which represents the probability of a particular sequence of pronunciations given a word sequence can be written as the following:

$$P(Q|W) = \prod_{i=1}^{L} P(q^{(w_i)}|w_i) \tag{2.5}$$

with $q^{(w_i)}$ representing a particular pronunciation of the word w_i. Now each of the base phones that compose the language can be modeled by a continuous density HMM [11]. An HMM can be described as a probabilistic finite state automaton composed of a set of states where the state sequence is hidden [14]. A probability a_{ij} rules the transition likelihood between two states s_i and s_j and each state of the model has an output observation distribution $b_j(x_t)$, which returns the probability of generating an acoustic feature vector x in state s_j at time t. Such a HMM based phone model is shown in figure 2.2.

The intuition behind using HMMs for phone modeling is the following: since the phone being said is still unknown, it is not possible to observe the state of sequence corresponding to it. Instead, we observe the acoustic feature vectors (audio input) and through the transitions probabilities a_{ij} and the state observation distributions $b_j(x_t)$ of the model, we could find out the most probable HMM state sequence corresponding to the observed sequence of acoustic feature vectors.

Using HMMs for modeling speech requires making two principal assumptions [11]:

1. The next state is only dependent on the current state

2. The current output observation is independent of the previous output observations

These two assumptions make the developed acoustic model mathematically and computationally controllable. Although these assumptions seem to result in an unrealistic acoustic model that does not represent well the complexity of speech, HMM based acoustic models showed to be successful in both speech recognition and speech synthesis applications [14].

Details about HMM parameter estimation via EM algorithm and state decoding using Viterbi algorithm will not be invistigated further. However more detailed insight about these algorithms can be found in [4].

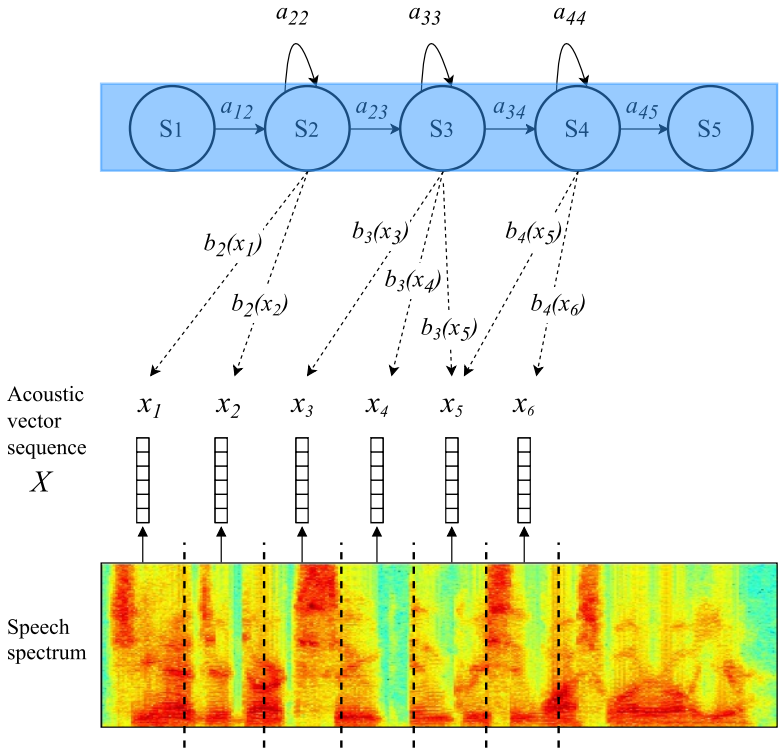


Figure 2.2: HMM representing how a phone is modeled. Acoustic features X are extracted from the speech spectrum at the bottom. a_{ij} represent transition probabilities and $b_j(x_t)$ represent the observation distributions. The hidden Markov states are highlighted in blue.

2.2.2 Gaussian Mixture Models

Before the rise of DNN usage for speech recognition tasks, Gaussian Mixture Models were most commonly used to model the output distributions of the HMM states $b_j(x_t)$ mentioned above. Let $d \in \mathbb{N}$ be the dimension of the acoustic feature vector x. The likelihood of this vector to be outputted by the state s_j can be expressed as the following [4]:

$$b_j(x) = \sum_{m=1}^{M} \frac{c_{j,m}}{(2\pi)^{d/2} |\sum_{j,m}|^{1/2}} exp\left[-\frac{1}{2}(x-\mu_{j,m})^T \Sigma_{j,m}^{-1}(x-\mu_{j,m})\right] \tag{2.6}$$

This likelihood can be expressed in a compacter way :

$$b_j(x) = \sum_{m=1}^{M} c_{j,m} \mathcal{N}(x; \mu_{j,m}; \Sigma_{j,m}) \tag{2.7}$$

Where $c_{j,m}$, $\mu_{j,m} \in \mathbb{R}^d$ and $\Sigma_{j,m} \in \mathbb{R}^{d \times d}$ are respectively the mixture components weights, the Gaussian mean vectors and the Gaussian covariance matrices of the observation distribution at the state s_j. M represents the number of mixture components, which

need to satisfy:

$$\sum_{m=1}^{M} c_{j,m} = 1, \quad c_{j,m} \geq 0 \tag{2.8}$$

GMMs have many advantages that made them suitable to be used for modeling the acoustic feature vectors distributions of HMM states, mainly their ability to model probability distributions to a high level of accuracy by using enough mixture components [3]. This ability of modeling these complex distributions made GMMs a successful tool for acoustic modeling in ASR systems for many years. But despite their advantages, these models started to be outperformed by DNN based acoustic models around year 2010-2011. Over the last few years, thanks to the development of available computational power and the advancement of machine learning algorithms, deep neural networks showed to be capable to enhance the performance of ASR systems due to their larger representational capacity [15].

In the next chapter we will introduce DNNs in more details. We will depict their architecture and mathematical background by mainly focusing on the DNNs used in acoustic modeling.We will then investigate further how the HMM-DNN ASR systems we use in our experiments are engineered.

3 Deep Neural Networks in ASR systems

Deep neural network (DNN) based acoustic models showed colossal improvements in large vocabulary continuous speech recognition tasks in the last years. It appears though that the theory behind these DNN based systems is not new. Techniques used in acoustic modeling through DNNs are similar to approaches that have been investigated earlier [16]. The main reason of the gain in performance through these models consists in the improvement of computational power and the increase of training corpus size [15], which permitted to unleash the potential of DNNs in acoustic modeling. In this chapter we give an overview over the DNNs architecture and investigate how they are used for acoustic modeling in the ASR system framework.

3.1 Deep Neural Networks

3.1.1 General Framework

A deep neural network is a multilayer perceptron (MLP) with usually multiple hidden layers. It represents a deep learning model that can be used for classification tasks [17]. Say a classifier $y = f^*(x)$ maps an input x to a class y. A DNN tries to define a mapping $y = f(x;\theta)$ and learn the parameters θ that give the best function f^* approximation. Now deep neural networks typically compose many different such functions together, and can be represented by a directed graph which describes this composition. In this case each function is represented by a *layer*. The figure 3.1 shows a typical architecture of such a network with four hidden layers, one input layer and one output layer.

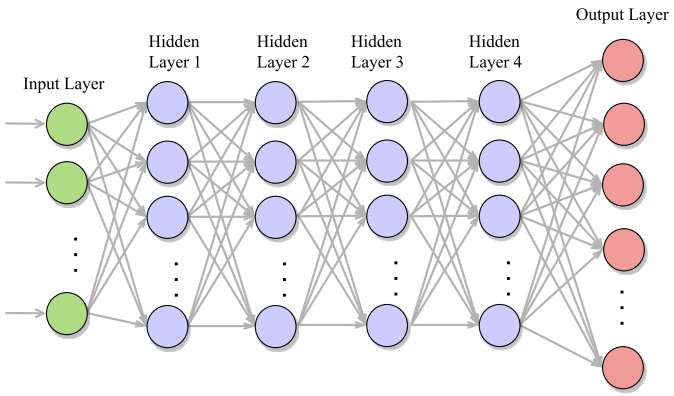


Figure 3.1: Example of DNN architecture with one input layer, four hidden layers and one output layer

The DNN typically accepts a vector x as input and outputs a vector y. For notation, we will denote the input layer as 0 and the output layer as L following the notation used in [4]. In the first L layers the information propagation follows this rule:

$$v^l = f(z^l) = f(W^l v^l + b^l), \quad 0 < l < L \tag{3.1}$$

where $z^l \in \mathbb{R}^{N_l}$ is the excitation vector in the l^{th} layer, $v^l \in \mathbb{R}^{N_l}$ the activation vector, $W^l \in \mathbb{R}^{N_l \times N_{l-1}}$ the weight matrix containing the optimization parameters, $b^l \in \mathbb{R}^{N_l}$ the bias vector and $N_l \in \mathbb{N}$ the number of neurons in the l^{th} layer. In this case $v^0 \in \mathbb{R}^{N_0}$ is the input feature vector x and $v^L \in \mathbb{R}^{N_L}$ represents the output vector. The function $f(.)$ with the following mapping $\mathbb{R}^{N_l} \rightarrow \mathbb{R}^{N_l}$ is called activation function. the sigmoid function is mostly used by practitioners [4]:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

The output layer v^L is chosen depending on the task the DNN is accomplishing. For multi-class classification, which is the case in ASR systems (will be explained in detail in the next section), we use a softmax function. In this case each output neuron $i \in \{1, ..., C\}$ represents a class, with $C = N_L$ is the number of classes. Each output neuron y_i represents the likelihood $p(i|x)$ that the input vector x belongs to the class i. In order to consider the output vector v^L as a valid probability distribution, it needs to satisfy the following:

$$\sum_{i=1}^{C} v_i^L = 1, \quad v_i^L \geq 0 \tag{3.3}$$

This requirement is satisfied by the softmax function where:

$$v_i^L = p(i|x) = softmax_i(z^L) = \frac{e^{z_i^L}}{\Sigma_{j=1}^{C} e^{z_i^L}} \tag{3.4}$$

Now given an input vector x, we can compute the corresponding output vector y by computing the activation vectors over the layers. The information flows through the network and propagates to the hidden units corresponding to each layer to finally produce the output v^L. That is what we call feed forward computation [4].

3.1.2 Parameter Estimation

In order to compute the output values corresponding to the fed input, we must estimate the model parameters $\{W, b\} = \{W^l, b^l | 0 < l \leq L\}$ which are still unknown. These parameters can be estimated through a training set $\mathbb{T} = \{x^m, y^m\}$ for $0 \leq m < M$ where M is the total number of training samples, x^m the m^{th} observation input vector and y^m the target output vector corresponding to x^m. The procedure consisting in estimating the DNN parameters is called training and it is specified by two components: The training criterion and the learning algorithm.

The Training Criterion

The training criterion also known as cost function is a very important design aspect of the neural network. The intuition behind it is to try to find optimal parameters $\{W, b\} = \{W^l, b^l | 0 < l \leq L\}$ that would reduce the gap between our estimated outputs v^L and the desired outputs y. One popular cost function described by Yu [4] is usually used as a training criteria for classification tasks, mainly the cross-entropy (CE) criterion:

$$J_{CE}(W, b) = -\frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{C} y_i \log(v_i^L) \tag{3.5}$$

The Learning Algorithm

Now that the training criterion is set, the DNN can be trained using the error back-propagation algorithm [18], which is derived from the gradient computation chain rule. Using the first order gradient information, we can define the update rule of the model parameters $\{W, b\}$ as the following:

$$W_{t+1}^l = W_t^l - \epsilon \Delta W_t^l \tag{3.6}$$

$$b_{t+1}^l = b_t^l - \epsilon \Delta b_t^l \tag{3.7}$$

where l represents the layer number to which the weight matrix W and the bias vector b belong, t the update step number and ϵ the learning rate. The update functions can written as follows:

$$\Delta W_t^l = \frac{1}{M_{batch}} \sum_{m=1}^{M_{batch}} \nabla_{W_t^l} J_{CE}(W, b) \tag{3.8}$$

$$\Delta b_t^l = \frac{1}{M_{batch}} \sum_{m=1}^{M_{batch}} \nabla_{b_t^l} J_{CE}(W, b) \tag{3.9}$$

where M_{batch} is the training batch size and $\nabla_a J_{CE}$ is the gradient of J_{CE} regarding a. For large training sets, it is typical to use a batch size M_{batch} that is smaller than the training set size M. This training method is called "minibatch" training.

3.2 Deep Neural Networks in ASR systems

In the framework of ASR systems, DNNs are used in acoustic modeling. Basically the acoustic system approximates a distribution $p(x|y = s_i)$, where x represents an acoustic feature vector and y a senone id $s_i, i \in \{1, ..., K\}$. In state of the art ASR systems, HMMs do not model monophones anymore like described in section 2.2.1, but they model senones: clustered context dependent sub-phonetic states [15]. Modeling context dependent phone states allows to take advantage of the context hidden information and showed an enhancement in ASR system performance in comparison with simple phone-modeling [4].

The HMM uses then a the deep neural network in order to approximate $p(x|y = s_i)$ instead of using a GMM. This approach is called the *HMM hybrid approach* [15]. In this approach, a DNN cannot directly model the distribution $p(x|y)$, but is instead trained to model $p(y|x)$. This represents the estimation of the senone class y given an acoustic input x, like shown in the figure 3.2. In order to obtain the distribution $p(x|y)$, the Bayes rule is used:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{3.10}$$

where $p(y)$ represents the prior probability of senone s in the training set, and $p(x)$ the prior probability of observing a particular acoustic feature. The last term can be considered as constant and is independent of the senone id s [2].

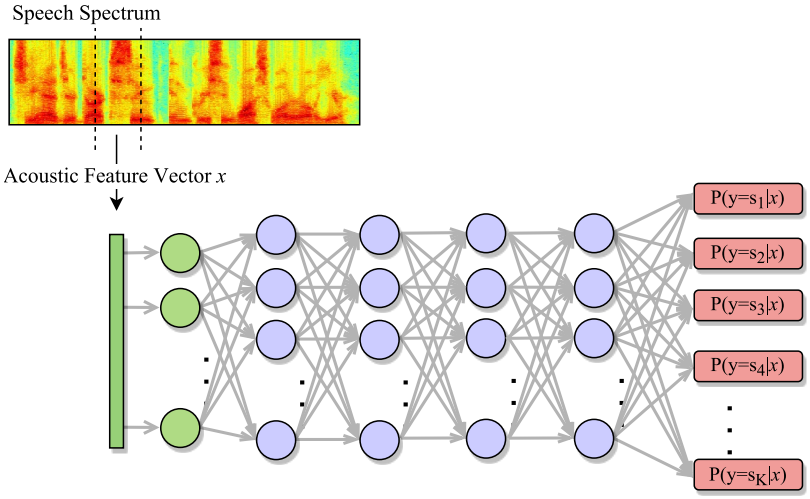


Figure 3.2: Representation of how a DNN acts as an acoustic model in an ASR system. The DNN takes the acoustic feature vector x corresponding to a speech frame as input and outputs the senone likelihood given x, $p(y = s_i|x)$

Now that the HMM-DNN hybrid system has been defined, and the use of the senone posterior distribution $p(y|x)$ has been depicted, we will describe how such a DNN is built. Describing this process should give the reader a better insight on how these acoustic models are constructed, which might be useful for understanding the adaptation techniques related to these models we will discuss later. As described by Maas et al. [15], the process of building a DNN acoustic model can be split into four steps:

3.2.1 Label Setting

Like for every neural network model, we need a set of training data $\{x^m, y^m\}$ where x^m represents the $m-th$ input (acoustic feature vector), y^m the labels corresponding to the m^{th} input and $m \in \{1, ..., M\}$ with M the total number of training data. Now each vector x corresponds to a frame of about 25 ms of speech that is preprocessed like described in

section 2.1. the label y corresponds to a senone (clustered context dependent triphones) label which is assigned to the acoustic feature vector x.

Since our training data consists typically in hundred of hours of speech, this assignment process can't be done manually. Instead, a baseline GMM-HMM system is used for that purpose. This is called forced alignment and can be understood as a time alignment of the utterance transcriptions with the corresponding audio data [19]. Through this procedure a sequence of senone labels $s_i^1, ..., s_i^n$ is generated for each frame sequence $x^1, ..., x^n$ that composes an utterance of the training data. Here n represents the number of frames in a training utterance and $i \in \{1, ..., K\}$ with K number of output units or senones. This procedure is shown in figure 3.3.

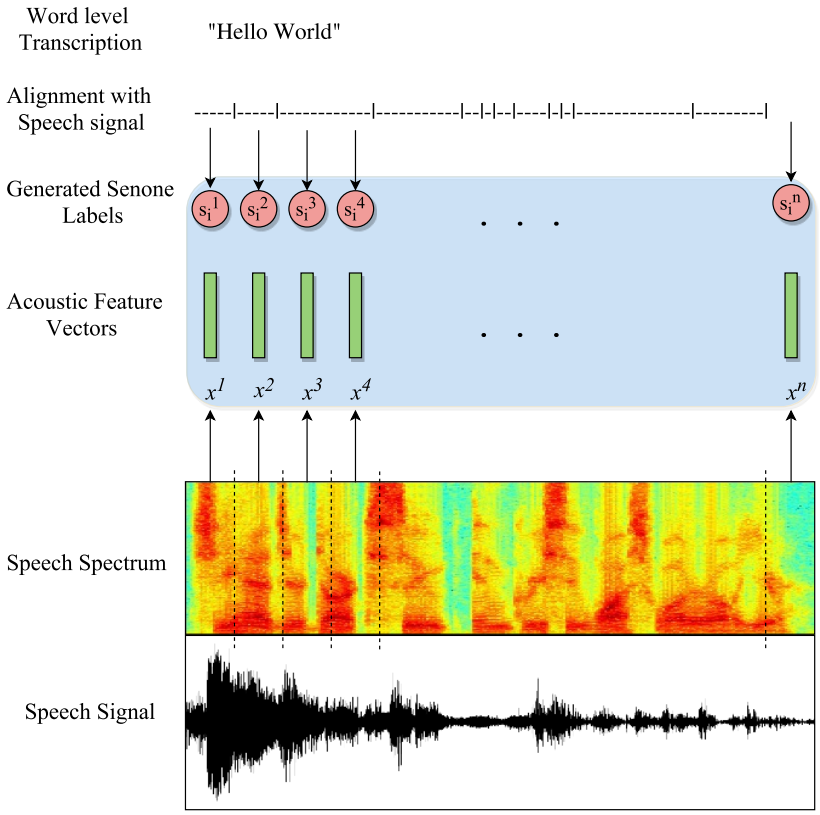


Figure 3.3: Forced alignment of ground-truth transcriptions for generating senone labels (red) for each acoustic feature vector (green) of the treated utterance. The blue square represents the obtained labeled data used for training the DNN acoustic model

3.2.2 Choosing the DNN Architecture

The architecture of the used neural network can be seen as a decisive criterion for building a powerful acoustic model. Modern DNNs used in ASR systems have more than one hidden layer, hence the prefix *deep*. For modern DNNs depth represents an important

feature leading to recognition gains. Experimental results from a series of studies [20, 21, 22] showed that two other key components affect the acoustic model performance which are: using neighboring frames to construct the acoustic input feature vectors in order to capture the dynamic property of speech and using senones as output.

3.2.3 Setting the DNN Loss Function

Now that we have a training set of speech utterances with frame level senone labels, we need to choose a training criterion to train the acoustic model. The cross entropy loss function discussed previously in section 3.1.2 is considered as a default choice for training DNNs in this case since we are dealing with a classification task. The DNN parameters are then trained to maximize the negative cross entropy loss function below [2]:

$$J_{NCE}(W, b) = \frac{1}{M} \sum_{t=1}^{M} \sum_{y=s_1}^{y=s_K} \tilde{p}(y|x^t) \log p(y|x^t) \tag{3.11}$$

where M is the number of samples in the training set, and $\tilde{p}(y|x^t) = \delta(y = s^t)$ is the target probability we get from the forced alignment. δ is in this context the *Kronecker-Delta* and s^t is the senone label corresponding to the t^{th} training sample x^t.

3.2.4 Choosing the Optimization Algorithm

Neural network parameter optimization lead mostly to a non-convex optimization problem. Basically, it is not possible to find a global minimum nor to estimate how good is the reached local minimum compared to the potential best solution. One of the standard approaches of optimizing the DNN parameters is using the stochastic gradient descent (SGD). Updating the DNN parameters is performed through the back propagation (BP) algorithm discussed in section 3.1.2.

4 Speaker Adaptation Techniques for DNN based ASR systems

Automatic speech recognition systems are trained on large sets of data, including a large amount of speakers. Still, speaking styles, accents and speech production anatomy may vary among the end-user speakers, which would probably lead to a difference in performance of the developed ASR system over different speakers. Adaptation of the acoustic models have the goal to compensate this performance gap.

In the past years, many adaptation techniques specific to DNN based acoustic models have been investigated. In this chapter we introduce the concept of adaptation in speech recognition tasks and describe the problem from a mathematical point of view. We later discuss the potential drawbacks from which these techniques could potentially suffer and then describe the proposed adaptation methods we investigated in the scope of this thesis.

4.1 The Adaptation Problem

Typically in machine learning frameworks like DNNs, we assume that the training and the test data follow a similar probability distribution. However, in reality this assumption is barely satisfied. This is also the case of DNN based acoustic models. Indeed, the DNN training is set to optimize the average performance over the speech data used for training. That means, if our system targets a specific speaker or a specific speech environment we will still have what was called by Yu [4] a training-testing mismatch.

In order to get a clearer perception of this mismatch problem in the context of speech recognition systems, imagine we train an acoustic model using a US-English speech corpus and then test it on both US-English and British-English speakers. Common sense implies that the system would perform better for US speakers than for British speakers, even though the language is the same for both parts. Yan and Vaseghi [23] proved this reasoning to be right by performing this experiment in both ways, and they showed that mismatch in accents between the training data and test data can lead to more than 100% increase in error rates. This showcase proves that speaker variation can deteriorate the recognition accuracy and presents an obstacle to speech recognition systems.

To solve this training-testing mismatch problem, adaptation techniques come into play. Many methods has been investigated and developed for GMM based acoustic models. But since DNNs are discriminative models, unlike GMMs which are generative models, they require other adaptation techniques. These techniques can be classified into two categories: The ones that consist in adapting the testing condition to fit the model and the others that try to adapt the model to fit the testing conditions [4]. The latter category, which is investigated in the scope of this thesis, consists mainly in retraining

a subset or the entire speaker-independent (SI) DNN parameters with data from the testing environment [24] in order to enhance recognition performance. In the case of speaker adaptation, The SI DNN is retrained with data specific to a certain speaker, with the aim to enhance recognition performance for that specific speaker. This adaptation approach will be described in detail in section 4.2.

4.2 Retraining DNNs For Adaptation

Like described before, the speaker adaptation techniques we will investigate in this thesis belong to the category that consists of adapting an already existing acoustic model to fit the testing conditions well. In other words, we will retrain the speaker-independent DNN of the ASR system with additional data from a certain speaker, to enhance the performance of the ASR system over this speaker. The idea behind adaptation is to learn new inputs, without loosing the valuable information already existing in the acoustic model.

In a real life scenario, this process can happen as follows: After the ASR application is deployed, as a phone intelligent assistant or infotainment system... it starts collecting data from each user under a real usage scenario. This collected data is then used to retrain the acoustic model in order to adapt it to the concerned user. The End-user will then benefit from a custom-made application that performs for him/her better than the non-customized one.

Like shown in figure 4.1, Adaptation can take place following two scenarios: Supervised or unsupervised.

- **Supervised**: The transcription of the adaptation data is available for the ASR system. In a real life scenario, the system would ask the user to read a certain sentence, which transcription is known by the system itself.
- **Unsupervised**: The transcription of the adaptation data is not available. In this case the transcription is obtained by decoding the data using the already implemented speaker-independent model. It is also called pseudo-transcription. In a real life scenario, the system would interact with the user, and use the data and the transcriptions generated by itself for adaptation. Pseudo-transcriptions will inevitably contain errors which will reduce the adaptation effect. This will limit the improvement achievable with unsupervised adaptation.

4.2.1 Mathematical Background

Say we are in the scenario where we want to develop an ASR application for controlling a TV in a leaving room. We assume that only one speaker is using it. The best way to develop this ASR system is to train it on data that is collected from this certain speaker in the leaving room environment. This would potentially solve the problem of training-testing mismatch mentioned earlier. In reality, this scenario is infeasible, since

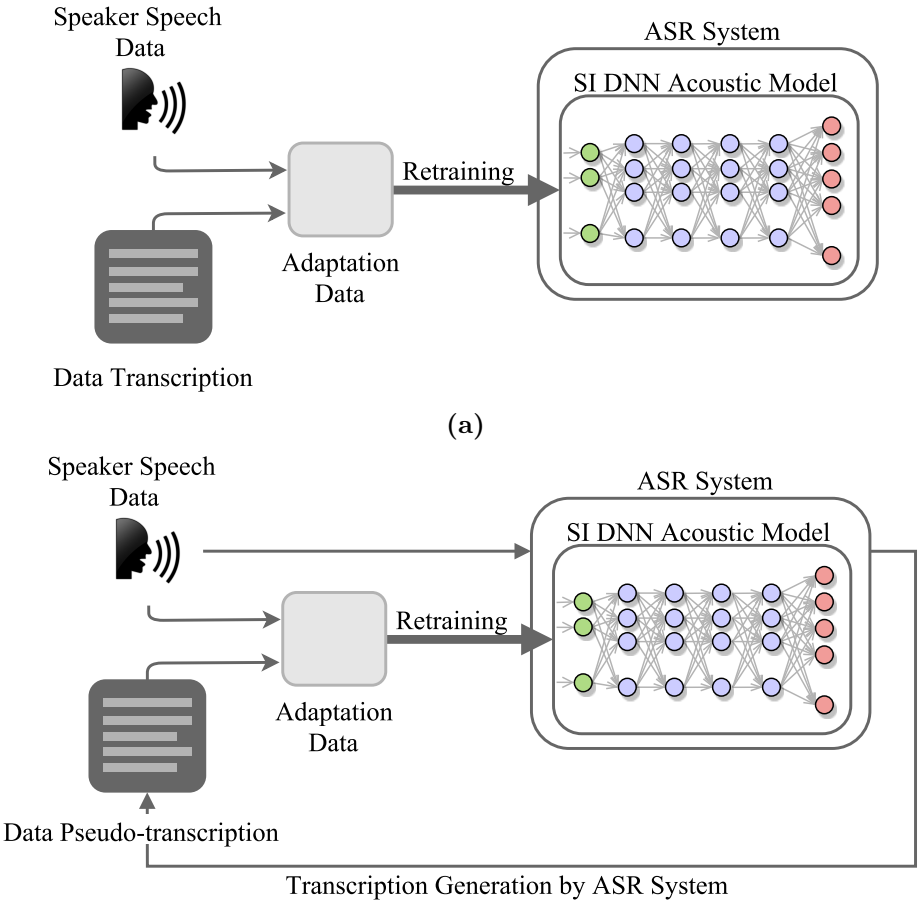


Figure 4.1: Two possible adaptation scenarios: Supervised(a) where the transcription of the adaptation data is available vs. unsupervised(b) where the transcriptions are generated by the ASR system itself

we would need the user to provide a very large amount of data (a few hours) to train the application before it is even operational.

In reality, the developed ASR system will be trained on a certain amount of speech data D_{train} from a speech corpus. Now adaptation relies on the assumption that the training data (Speech corpus) $D_{train} = \{(x^t, y^t)|(x^t, y^t) \sim \mathcal{P}_{train}\}$ with $t \in \{1, ..., |D_{train}|\}$ and the test data (Speaker in leaving room environment) $D_{test} = \{(x^t, y^t)|(x^t, y^t) \sim \mathcal{P}_{test}\}$ with $t \in \{1, ..., |D_{test}|\}$ have different probability distributions. In other words $\mathcal{P}_{train}$ and $\mathcal{P}_{test}$ are different [25]. Therefore even if we dispose of a large amount of training data, $|D_{train}| \to \infty$, it will still not be possible to obtain an estimate of the distribution $\mathcal{P}_{test}$.

What can be done is collecting a small amount of the test data during the usage of the ASR system, which we call the adaptation data $D_{adapt} = \{(x^t, y^t)|(x^t, y^t) \sim \mathcal{P}_{test}\}$ with $t \in \{1, ..., |D_{adapt}|\}$ and $|D_{adapt}| << |D_{train}|$. Since the collected adaptation data

size $|D_{adapt}|$ stays relatively small, using it for training an acoustic model from scratch would lead to one of these scenarios: overfitting or high bias [26].

An obvious approach that has been investigated in recent years for adapting DNN based acoustic models is adjusting the DNN parameters with the collected adaptation data D_{adapt} starting from the speaker independent model (model trained on D_{train}) [2]. In other words this approach consists in adapting the DNN parameters $\{W_{SI}, b_{SI}\}$ of the SI model by optimizing the loss function J_{NCE} mentioned in section 3.2.3 over the adaptation data $D_{adapt} = \{(x^t, y^t)|(x^t, y^t)$ [4]. Unfortunately, this approach may destroy the previously learned information, and lead to a phenomena investigated by Albesano et al. [27] called catastrophic forgetting.

4.2.2 The Catastrophic Forgetting Problem

The catastrophic forgetting problem was investigated by [28] in the late 80's as a problem that affects both biological and artificial learning systems. Indeed, if a learning system is trained on a certain task and then on a second one, it may "forget" how to execute the first task [6]. In the case of machine learning systems, this problem may occur when a network trained on a large set of data, has to learn new patterns, or needs to be a adapted to a certain environment. The catastrophic forgetting can be particularly high if the new data (adaptation data) does not represent the original training data sufficiently.

Like in the case of the speaker adaptation scenario, this effect should be taken into account when the amount of adaptation data is limited and does not include enough samples for all the output classes [27]. In fact, since DNNs use discriminative training, they penalize the units that do not have observations for certain output classes by assigning them a zero target value. This causes the DNNs to forget their capability to classify the concerned classes.

To prevent this problem from happening, Albesano et al. [27] proposed the concept of conservative training (CT) as a possible solution. CT represents a popular technique, used also in the context of speech recognition for acoustic models adaptation. It can take the form of regularized adaptation, where a regularization term is added to the adaptation criterion like described in [25]. It can also be achieved by adapting only selected weights, or by using small learning rates and early stopping for adaptation [4]. The speaker adaptation techniques we investigated in the scope of this thesis can be classified under this category. We describe these techniques in detail in section 4.3.

4.3 Investigated Speaker Adaptation Techniques

4.3.1 Training Input or Output Layers Only

This adaptation technique belongs to one of the techniques proposed by Liao [1]. This method consists in updating only the input or the output layers of the DNN when adapting. The intuition behind it is to try not to loose the information contained in the inner layers and capture the adaptation information within the layers in the extremities.

4.3.2 Kullback-Leibler Divergence Regularized Adaptation

Like stated earlier, catastrophic forgetting can be avoided by using a form of regularized adaptation. One of the techniques that belongs to this category of adaptation methods is the Kullback-Leibler Divergence(KLD) regularized adaptation proposed by Yu et al. [2]. The idea behind this approach is that the senone distribution which will be estimated by the new adapted model, should not deviate much from the distribution estimated by the SI model. The straight forward method of adapting the SI DNN parameters by optimizing the loss function J_{NCE} over the adaptation data $D_{adapt} = \{(x^t, y^t)|(x^t, y^t)$ with $t \in \{1, ..., N\}$ looks like the following:

$$J_{NCE}(W, b) = \frac{1}{N} \sum_{t=1}^{N} \sum_{y=s_1}^{y=s_K} \tilde{p}(y|x^t) \log p(y|x^t) \tag{4.1}$$

where N is the number of samples in the adaptation set, and $\tilde{p}(y|x^t) = \delta(y = s^t)$ is the target probability we get from the forced alignment. δ is in this context the *Kronecker-Delta* and s^t is the senone label corresponding to the t^{th} adaptation sample x^t. Using KLD regularization technique results in the following optimization criterion:

$$\hat{J}_{NCE}(W, b) = (1 - \rho) J_{NCE} + \rho \frac{1}{N} \sum_{t=1}^{N} \sum_{y=s_1}^{y=s_K} p^{SI}(y|x^t) \log p(y|x^t) \tag{4.2}$$

where p^{SI} is the probability estimated from the speaker independent model, which is computed through a forward pass over the SI model. ρ represents in this context the weight used for regularization. The equation 4.2 can be rewritten as following:

$$\begin{aligned} \hat{J}_{NCE}(W, b) &= \frac{1}{N} \sum_{t=1}^{N} \sum_{y=s_1}^{y=s_K} [(1 - \rho)\tilde{p}(y|x^t) + \rho p^{SI}(y|x^t)] \log p(y|x^t) \\ &= \frac{1}{N} \sum_{t=1}^{N} \sum_{y=s_1}^{y=s_K} \hat{p}(y|x^t) \log p(y|x^t) \end{aligned} \tag{4.3}$$

where $\hat{p}(y|x^t) = (1 - \rho)\tilde{p}(y|x^t) + \rho p^{SI}(y|x^t)$. So using KLD regularization equals to changing the target probability distribution used in the original training criterion 4.1 from $\tilde{p}(y|x^t)$ to $\hat{p}(y|x^t)$. This new target probability distribution represents a linear interpolation of the distribution estimated from the alignment $\tilde{p}(y|x^t)$ and the distribution estimated from the SI model $p^{SI}(y|x^t)$. This method aims to constraint the output probabilities and can be directly applied by using the Back-propagation algorithm. The only things to be changed are the target probabilities used for the adaptation data. The regularization coefficient ρ can be adjusted and as the equation 4.3 shows, the larger it is, the more we trust the SI model, and the smaller it is the more we rely on the adaptation data.

5 Experiments on ASR System 1

Training acoustic models of speech recognition systems in the scope of adaptation experiments can be very time consuming, particularly when dealing with Large Vocabulary Continuous Speech Recognition systems. This is partly due to the very large architecture of DNNs used for acoustic modeling and the training time resulting from it. In order to be able to investigate the proposed adaptation techniques in a reasonable time that does not exceed the time allowed for this work, we decided to first conduct our experiments on an ASR with a slightly smaller DNN architecture than those used in state of the art systems. We refer to this system as "ASR System 1". This enables us to get an overview of how these methods perform in a "relatively" short time, an decide then what could be potentially applied to the state of the art system.

5.1 The Kaldi Toolkit

To conduct the experiments, we used the Kaldi toolkit. Kaldi is an open source software used mainly in the framework of speech recognition research with the aim to develop ASR systems [29]. The advantage of Kaldi is that it provides recipes for building state of the art ASR systems [30] from widely available databases of speech corpora provided by the Linguistic Data Consortium (LDC). Kaldi is written in C++ and can be compiled on a Unix-like system. Figure 5.1 gives an overview of the architecture of the used toolkit. For GMM-HMM based speech recognition systems, Kaldi provides several adaptation

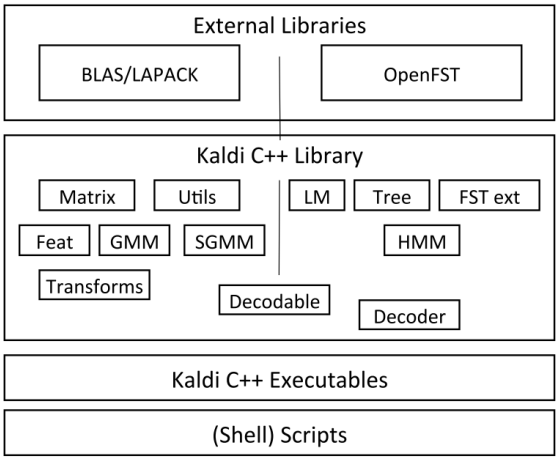


Figure 5.1: An overview of Kaldi components. Lower modules depend on the higher ones. Shell scripts are used to access the implemented functions [29]

techniques. However, the techniques we want to investigate for DNN-HMM frameworks are not supported. Therefore the first task before evaluating these methods was to implement them. The implementation of the adaptation framework was a challenging task, but since we focus in the evaluation of the adaptation techniques in this work, we will not further discuss the implementation matters.

5.2 System Architecture

Our initial experiments were conducted on the system we call "ASR system 1". As speech data for training the system, we used the Switchboard corpus. Switchboard is a telephone speech corpus which was collected by Texas Instruments in the 90's under DARPA (Defense Advanced Research Projects Agency) sponsorship. It represents about 2500 telephone conversations over a total of 500 Speakers from the United States resulting in about 250 hours of speech data and nearly 3 million words [31].

The acoustic model was trained on about 110 hours of the Switchboard speech corpus following the recipe proposed by Kaldi [30]. The language model was also trained on the Switchboard corpus transcriptions. An amount of data of this corpus was excluded from training the acoustic and language models in order to be used for adaptation. A more detailed description of this data is to be found in section 5.3.

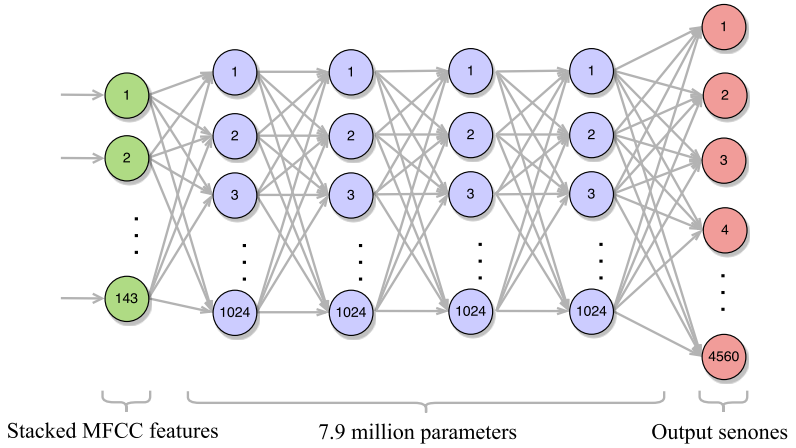


Figure 5.2: Architecture of the DNN used for acoustic modeling in "ASR system 1"

The resulting baseline GMM-HMM acoustic model contained 4560 tied triphone states and 140K Gaussians. The features used in the DNN framework were 13-dimensional Mel-filter-bank cepstral coefficients with a context window size of 11, resulting in 143 dimensional input vectors (13×11). The DNN was composed of 4 hidden layers containing each 1024 neurons, and an output layer of 4560 neurons, each corresponding to a specific senone. The figure 5.2 depicts the architecture of the used DNN. The developed context-dependent DNN-HMM system will be used as baseline for the adaptation tasks discussed within this chapter. We will refer to it as baseline or speaker independent (SI) system.

The DNN was trained using forced alignments of the training data generated from the former GMM-HMM model. As training criterion, frame level cross-entropy described in section 3.2.3 was used. Mini-batch gradient descent was used, with a batch size of 256 frames. Back-propagation algorithm was used to update the weights. Graphics Processing Units (GPUs) were used to accelerate the training of the DNN.

5.3 The Test Speakers Set

The adaptation experiments were conducted on 6 different speakers, which audio data and transcriptions were extracted from the Switchboard corpus. These speakers were excluded from the acoustic model training set of the SI model and from the data used to generate the trigram language model we use for these experiments. Excluding the test speakers from the data exploited to train the SI model permits to simulate a real usage adaptation scenario. For each of these speakers, we extracted respectively 300 utterances (sentences). The first 200 utterances were used as **adaptation set**. We split them in 5, 10, 50 and 200 utterances in order to investigate the effect of adaptation data size over recognition accuracy. The last 100 utterances were used as **evaluation set**. The total number words in the evaluation set is 8790. Figure 5.3 shows a clearer overview over the used test speakers set. The goal of the experiments are to improve the overall recognition performance (Average WER) over the evaluation set through speaker adaptation. The performance of the SI system over the respective evaluation sets of the test speakers is shown in the table 5.1.

Speaker id	SI system WER
Spk1	37,77 %
Spk2	27,86 %
Spk3	25,62 %
Spk4	26,22 %
Spk5	28,98 %
Spk6	24,71 %
Average WER	28,28 %

Table 5.1: Word error rate (WER) over the 6 evaluation sets of the respective test speakers, and average WER using the speaker independent system introduced in section 5.2.

To evaluate the accuracy of a speech recognition system, we use the Word Error Rate (WER) metric. The WER can be understood as the proportion of word errors over the processed sequence of words and can be calculated as follows [32]:

$$WER = \frac{S + D + I}{N} \tag{5.1}$$

where S, D, I and N are respectively the number of substitutions, deletions, insertions and the total number of words in the reference.

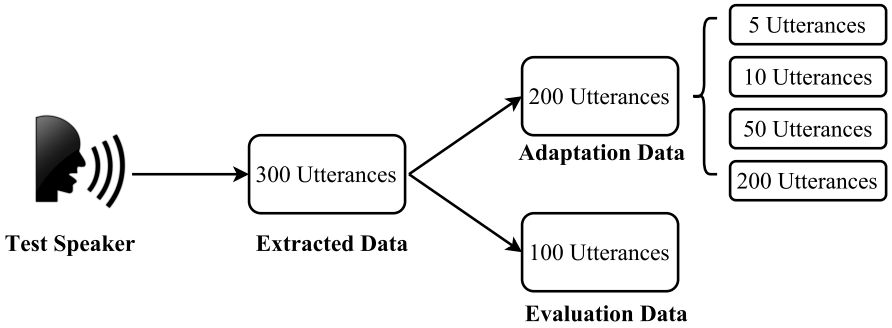


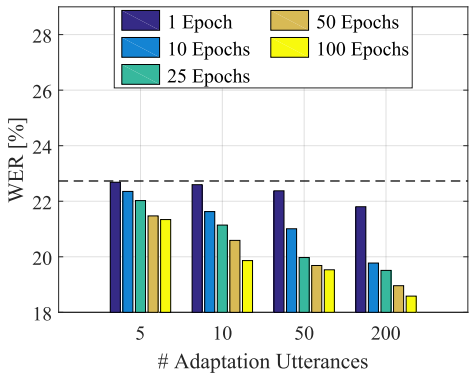
Figure 5.3: Overview of the test data extracted from each speaker and used for adaptation and evaluation

5.4 Choosing an appropriate learning rate

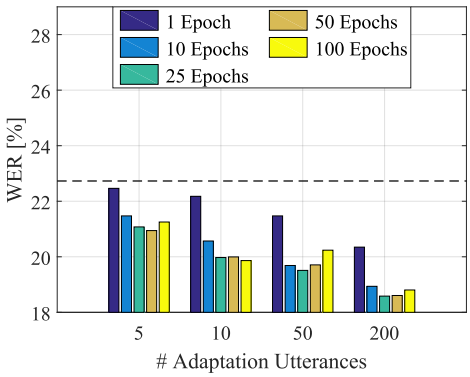
Now when retraining the SI DNN with adaptation data, there are two hyperparameters that may vary: The number of adaptation epochs and the learning rate. In this context, one epoch consists in a full training cycle over the used adaptation data. Following what has been done by Liao [1], we decided to set a constant learning rate and vary the number of adaptation epochs. Setting the learning rate value might be a quite delicate task. A too large value could rapidly lead to catastrophic forgetting resulting in overshooting a possible enhancement in recognition performance over adaptation epochs. A too low value might take too many epochs until the system reaches a local minimum. And since adaptation experiments on DNNs can be very time consuming, it would not be reasonable to choose a too small learning rate.

To determine which learning rate we would use in the scope of our experiments, we decided to run supervised adaptation over a development speakers set of 3 speakers extracted from the Switchboard corpus and excluded from training the acoustic and language models of the baseline DNN. The data extracted from each of the development speakers had the same structure as the data of the test speakers, namely 200 utterances for adaptation and 100 utterances for evaluation per speaker. We used 5,10,50 and 200 utterances to adapt and evaluated the recognition accuracy over the 100 evaluation utterances by extracting the corresponding word error rates. We adapted with 1, 10, 25, 50 and 100 epochs. These experiments were run 4 times using different learning rate values, mainly $10^{-5}, 5 \times 10^{-5}, 10^{-4}$ and 5×10^{-4}. The figure 5.4 shows the recognition results when using different learning rates for adaptation. The recognition results (WER) represent the mean performance over the evaluation data of the 3 speakers after being adapted with their respective adaptation data. The dashed line represents the mean WER of the SI system over the evaluation utterances of the 3 development speakers.

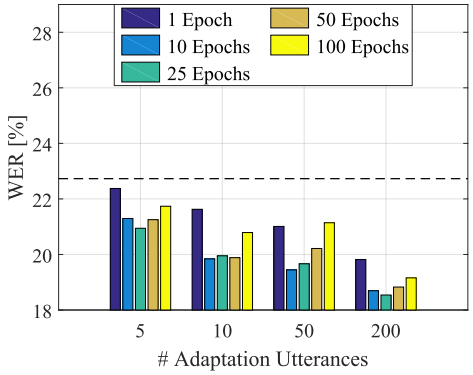
Now the interesting information we would like to extract from figure 5.4 is the trend of the WER over adaptation utterances for different learning rates. Figure 5.4d shows clearly that starting from 50 epochs of adaptation, the overall performance gets worse than the baseline performance when using 5,10 and 50 utterances. This is in our case not desirable and could be potentially due to catastrophic forgetting in some of the speaker



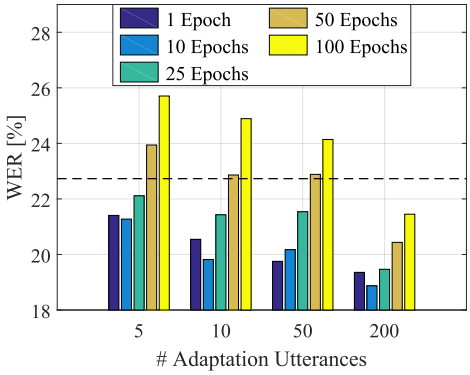
(a) Learning rate = 10^{-5}



(b) Learning rate = 5×10^{-5}



(c) Learning rate = 10^{-4}



(d) Learning rate = 5×10^{-4}

Figure 5.4: Supervised adaptation results over development set with different learning rates [(a) (b) (c) (d)] . In each experiment, different adaptation epochs (1, 10, 25, 50 and 100) and different adaptation data sizes (5, 10, 50 and 200 utterances) were used. The dashed line represents the mean SI system performance over the evaluation data

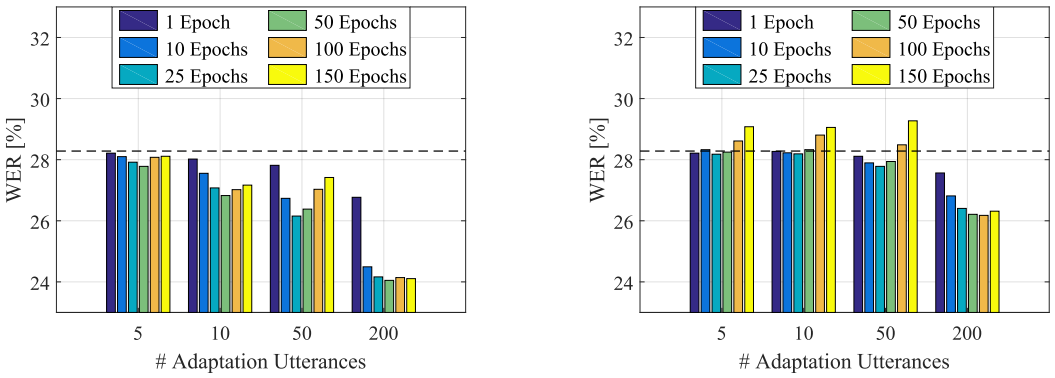
adapted networks caused by the large learning rate (5×10^{-4}). Figure 5.4a shows on the contrary a decreasing trend in WER over adaptation epochs. But WERs could potentially still decrease after 100 epochs. That is why the used learning rate seemed to be too small (10^{-5}) to reach a potential minimum over the chosen range of adaptation epochs (1 to 100). This leaves us now with both learning rates 5×10^{-5} and 10^{-4}. Figures 5.4b and 5.4c show a bowl shaped course of the WER over adaptation epochs for almost all adaptation data sizes. A minimum value in performance is to be seen in the chosen range of adaptation epochs.

Since the bowl shaped trend of WERs over adaptation epochs is well perceived for all adaptation data sizes in figure 5.4c, the learning rate value 10^{-4} seemed to us to be reasonable to be used within the next experiments.

5.5 Supervised Adaptation Results

5.5.1 Training Input or Output Layers Only

These adaptation techniques were proposed by Liao [1]. The intuition behind these methods consists in trying to adapt the model to the speaker, by capturing its specific features in the layers on the extremities, without updating the core weights in the inner layers. We conducted these experiments on the test speakers, using 5, 10, 50 and 200 utterances per speaker, over 1, 10, 25, 50, 100 and 150 adaptation epochs. We used the learning rate determined in section 5.4 (10^{-4}). The results are in figure 5.5.



(a) Adapting only input layer

(b) Adapting only output layer

Figure 5.5: Supervised adaptation results when training input or output layers only

The results show clearly that training the input layer only like shown in figure 5.5a outperforms training only output layers (figure 5.5b). For a small number of utterances (5 and 10), The first method seems to be more efficient with an absolute WER enhancement up to respectively 0.5% and 1.5%, whereas the second method does not register any meaningful WER enhancement, but on the contrary leads potentially to worse recognition results than the baseline system. For a higher number of utterances, training the input layer records an absolute enhancement of recognition up to respectively 2.1% and 4.2%. Training output layers however scores only an absolute enhancement of 0.5% and 2.1 % respectively.

5.5.2 KLD regularized adaptation

This adaptation technique was proposed by Yu et al. [2]. We run the experiments on the test speakers, using 5, 10, 50 and 200 utterances per speaker, over 1, 10, 25, 50, 100 and 150 adaptation epochs. We considered different regularization terms (a.k.a interpolation weight) ρ, mainly 0, 0.125, 0.25, 0.5, 0.75 and 1. If $\rho = 1$, then we basically trust completely the SI model, and ignore all information coming from the adaptation data. If $\rho = 0$, that means that we trust completely the adaptation data, which translates in

retraining the DNN without any regularization term. Results of the experiments can be seen in figure 5.6.

As it is shown in figure 5.6f, using an interpolation term $\rho = 1$ delivers the same results as the baseline itself, since we retrain in that case the network with the network's own outputs. That means that the network does not learn anything new and the recognition results stay the same over adaptation epochs. What is also interesting to notice is that the only time when recognition results after adaptation happen to be worse than the baseline system is when no regularization is used ($\rho = 0$) like shown in figure 5.6a.

Figures 5.6b to 5.6e show that increasing the regularization term value seems to smoothen the course of the WER curve over adaptation epochs. The higher the regularization term is, the slower the WER curve over adaptation epochs moves toward a potential minimum. Nevertheless, using regularization when adapting does not seem to always necessarily deliver better recognition results, but seems to hold back the catastrophic forgetting phenomena which potentially leads to worse recognition results than the baseline system.

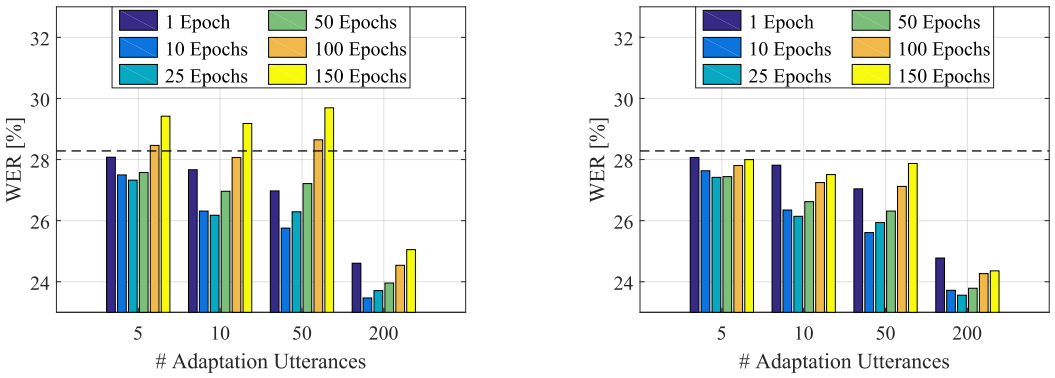
Figure 5.7 shows the effect of different regularization coefficients over different adaptation epochs. Figures 5.7a and 5.7b show the recognition results after respectively 1 and 10 epochs of adaptation. What can be seen is that no matter the adaptation size, adapting without regularization mostly gives a better performance than with regularized adaptation. Figures 5.7d, 5.7e and 5.7f show that starting from 50 Epochs of adaptation, using $\rho = 0.25$ or $\rho = 0.5$ as regularization coefficient seems to give the best performance in recognition no matter what the adaptation size is.

Now, these results suggest two different facts concerning using KLD regularization: The first one is that the effect of catastrophic forgetting that may occur after a certain amount of adaptation epochs (like in fig.5.6a), can be hold back by using the KLD regularized adaptation. In fact figure 5.6 shows that the larger ρ is, the more epochs can be used for adaptation without risking to get an adapted model performing worse than the baseline model (mainly due to catastrophic forgetting). The second fact is that using KLD regularization ($\rho \neq 0$) does not necessarily lead to a better performance than adapting without regularization ($\rho = 0$). In fact, in the early steps of adaptation, in our case 1 and 10 epochs, regularized adaptation performs worse than the non-regularized one like shown in figures 5.7a and 5.7b.

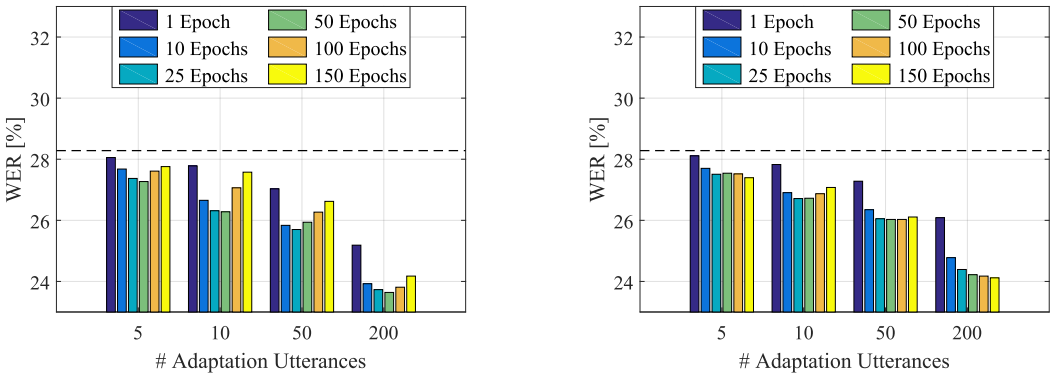
5.5.3 Overall Overview

In this section, we would like to have an overall overview over the different techniques we used for supervised adaptation and see how they compare to each other in performance. For that purpose, we draw the box plot in figure 5.8 that shows the performance of each of the techniques discussed above over the adaptation epochs window (1 to 150) and using the learning rate we fixed in section 5.4. Typically before adapting, the window of adaptation epochs is unknown. So, through this graph we would like to know how stable is each adaptation technique over the window of adaptation epochs we chose (1 to 150).

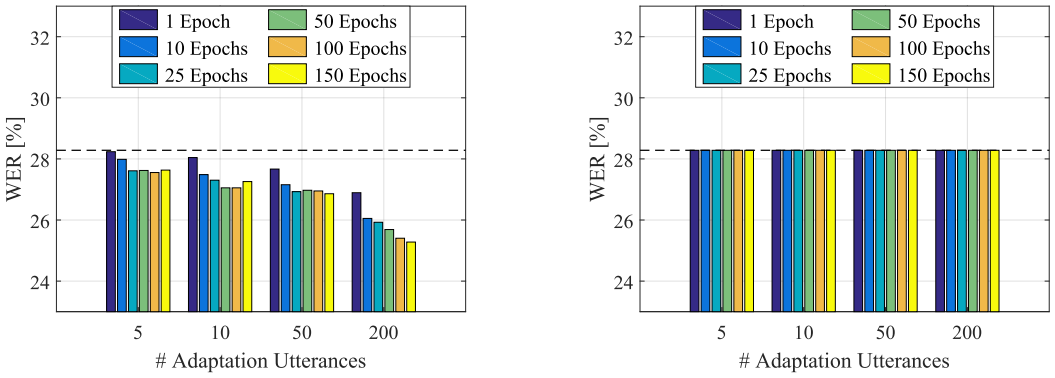
The figure 5.8 shows that when not using regularization ($\rho = 0$), the WER over adaptation epochs varies the most, leading sometimes to worse results than the baseline.



(a) $\rho = 0$ **(b)** $\rho = 0.125$

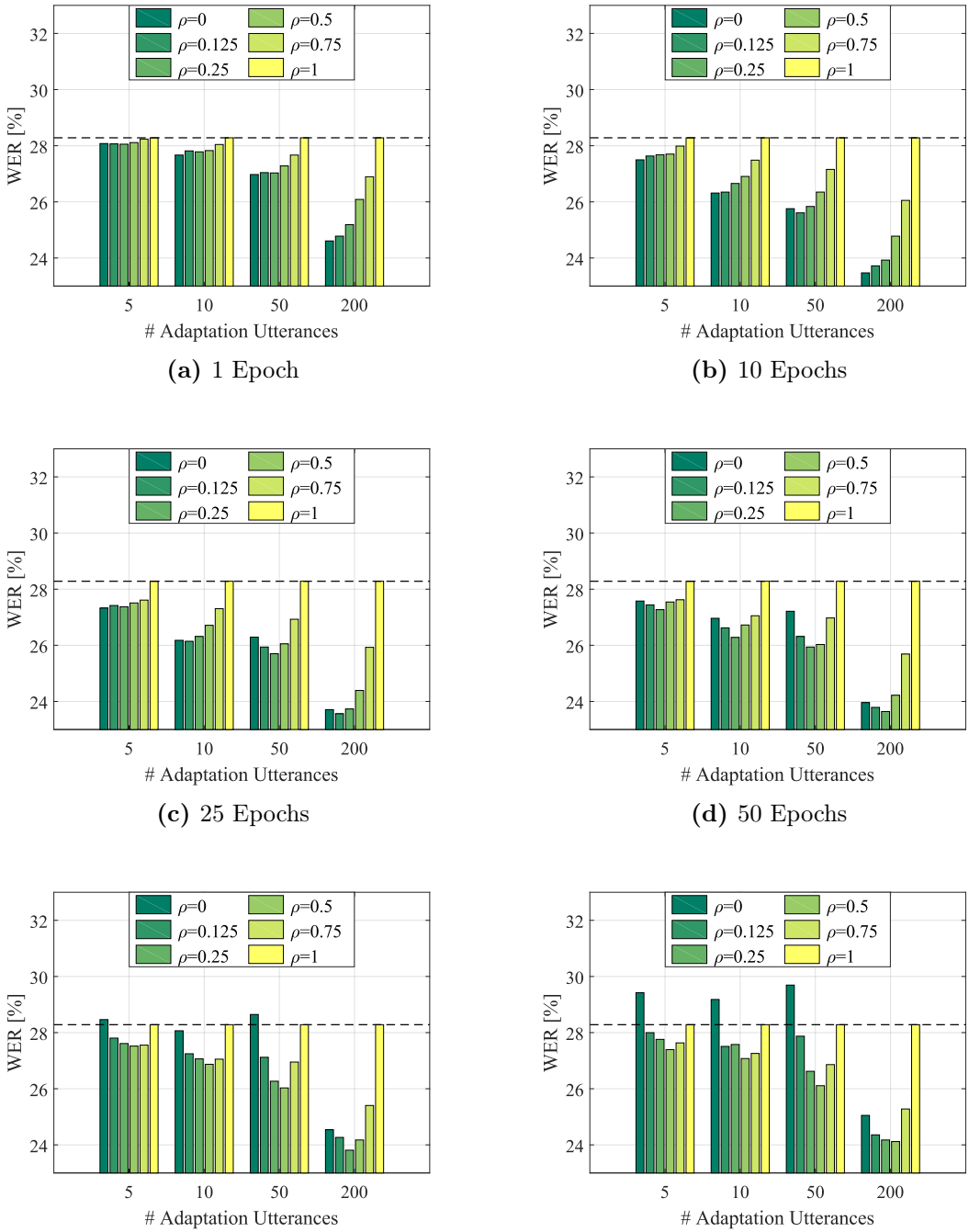


(c) $\rho = 0.25$ **(d)** $\rho = 0.5$



(e) $\rho = 0.75$ **(f)** $\rho = 1$

Figure 5.6: Supervised adaptation results using different regularization coefficients ρ [(a) (b) (c) (d) (e) (f)]. The dashed line represents the baseline WER using the SI model (28.28%)



(a) 1 Epoch **(b)** 10 Epochs

(c) 25 Epochs **(d)** 50 Epochs

(e) 100 Epochs **(f)** 150 Epochs

Figure 5.7: Supervised adaptation results using different regularization coefficients over different adaptation epochs [(a) (b) (c) (d) (e) (f)]. The dashed line represents the baseline WER using the SI model (28.28%)

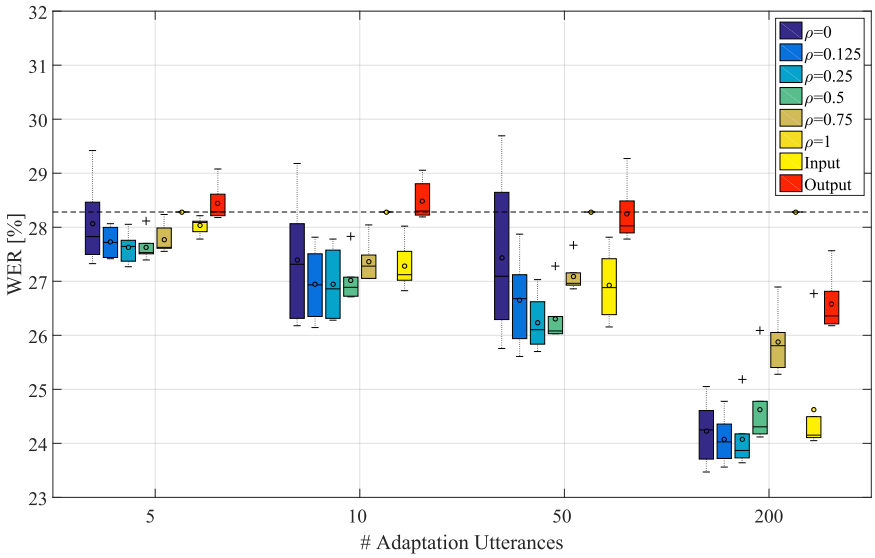


Figure 5.8: Overview of the performance of the investigated supervised adaptation methods for different adaptation data sizes over the adaptation epochs window from 1 to 150.

Using several KLD regularization coefficients or training only parts of the network might reduce this variance like shown in this figure 5.8. When adapting with a data size of 5, 10, and 50 utterances, it seems that using an interpolation weight of $\rho = 0.5$ is a good choice, since it reduces the WER compared to the baseline but at the same time has a small variance over the adaptation epochs, which reduces the risk of performing worst than the baseline system. That said, using $\rho = 0.5$ does not necessarily lead to the best possible WER after adaptation. For an adaptation data size of 200 utterances, figure 5.8 show that a smaller regularization coefficient ρ might be better to use (0.125 or 0.25) for better adaptation results.

Training only output layer seems to be the worse approach of all since it delivers the worse results for all adaptation data sizes. Training the input layer on the contrary shows a better performance. Even though it doesn't deliver better WER values than with KLD regularized adaptation for several interpolation weight values ρ, it delivers comparable performance for large adaptation data sizes (200 utterances). Training the input layer is also the method that updates the smaller number of weights when adapting, which can be advantageous in terms of time and memory for the scenario of supervised adaptation.

Also interesting is to notice how the adaptation data size matters for the adapted model performance. Indeed, no matter what adaptation method is used, the adapted models seem in our case to outperform the baseline model when the adaptation data size is large enough (200 utterances).

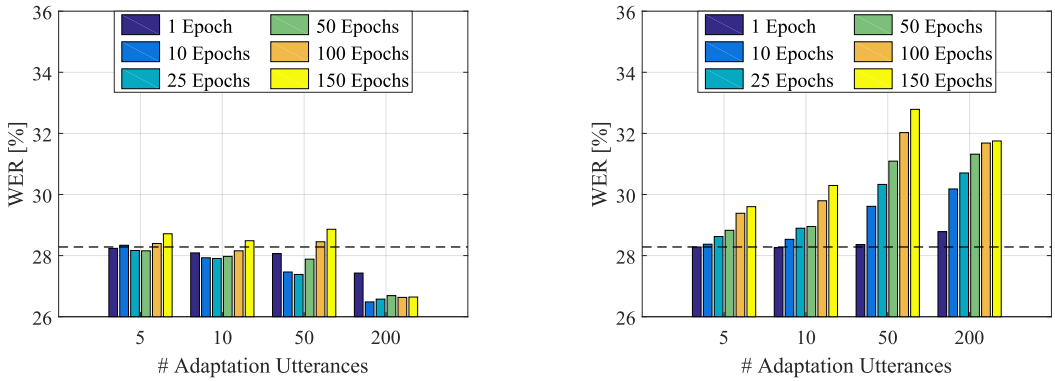
5.6 Unsupervised Adaptation Results

5.6.1 Training Input or Output Layers Only

For these experiments, we used the same test speakers used for supervised adaptation. We conducted these experiments by using 5, 10, 50 and 200 utterances per speaker as adaptation set, over 1, 10, 25, 50, 100 and 150 adaptation epochs. We used the learning rate we fixed in section 5.4 (10^{-4}). The results are shown in figure 5.9.

Adapting only the input layer seems to enhance recognition performance starting for the adaptation data sizes 10, 50 and 200. For small adaptation data size (5 utterances), no enhancement is to be seen. Still the recognition performance after unsupervised adaptation is worse than the performance obtained with supervised adaptation, since pseudo transcriptions used as labels for adaptation data may be partly wrong.

Adapting the output layer leads on the contrary to a worse performance than the baseline. In fact, no improvement in performance was noticed over the adaptation for all sizes of adaptation data. Instead WER kept rising over the epochs.



(a) Adapting only the input layer

(b) Adapting the output layer

Figure 5.9: Unsupervised adaptation results when training input or output layers only

5.6.2 KLD regularized adaptation

In order to investigate this approach, we run the experiments on the test speakers, using 5, 10, 50 and 200 utterances per speaker, over 1, 10, 25, 50, 100 and 150 adaptation epochs. We considered different regularization terms (a.k.a interpolation weight) $\rho = 0$, mainly 0, 0.125, 0.25, 0.5, 0.75 and 1. As figure 5.10 shows, unsupervised adaptation delivers worse results than the supervised one, which is predictable since pseudo-transcriptions used to generate the adaptation labels contain inevitably errors. Without using regularization (ρ), an improvement in recognition is barely to be seen in the first adaptation epochs (max 0.6%). The WER value then rises over the adaptation epochs, leading to worst recognition results than the baseline model, symptom of catastrophic forgetting.

Increasing the regularization coefficient value ρ seems to result in "flattening" the WER course over adaptation epochs.

Figure 5.10e shows that using $\rho = 0.75$ seems capable to contain at most the effect of catastrophic forgetting, but doesn't seem to bring much gain in performance. Also interesting to notice in figure 5.10 is that the size of adaptation data seems to have less effect than when using supervised adaptation, since the gains in performance provoked by 10 and 200 utterances are very similar and using only 10 utterances shows better overall results in adaptation than using 50 utterances.

Now the figure 5.11 shows that in the early stages of adaptation 1 to 10 epochs, using a regularization coefficient $0.125 \leq \rho \leq 0.75$ could potentially help get some improvement over the non-regularized adaptation. But these improvements shown in figures 5.11a and 5.11b are quite insignificant. In the late stages of adaptation however, using KLD regularization seems to be a must. Adapting with $\rho = 0$ results in worst recognition accuracy due to catastrophic forgetting. In this case using regularization could prevent that from happening like shown in figures 5.11e and 5.11f.

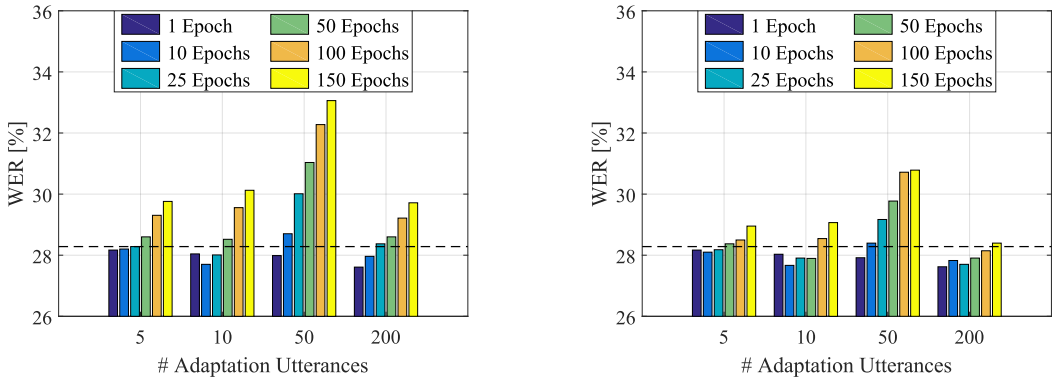
5.6.3 Overall Overview

Now let's take an overall overview over the unsupervised adaptation techniques we investigated earlier. The figure 5.12 shows the performance of each of the techniques discussed above over the adaptation epochs window (1 to 150) and using the learning rate we fixed in section 5.4. The first thing to notice is that most of the techniques investigated do not deliver a better performance, and if so, the gain in WER is relatively insignificant. For small adaptation utterances (5 and 10), it seems that a high interpolation coefficient $\rho = 0.75$ for adaptation can bring a tiny gain in WER (less than 1%). The most significant gain in performance happens when using 200 utterances for adaptation and only training the input layer. This yields to 1.6% enhancement in recognition compared to the baseline system.

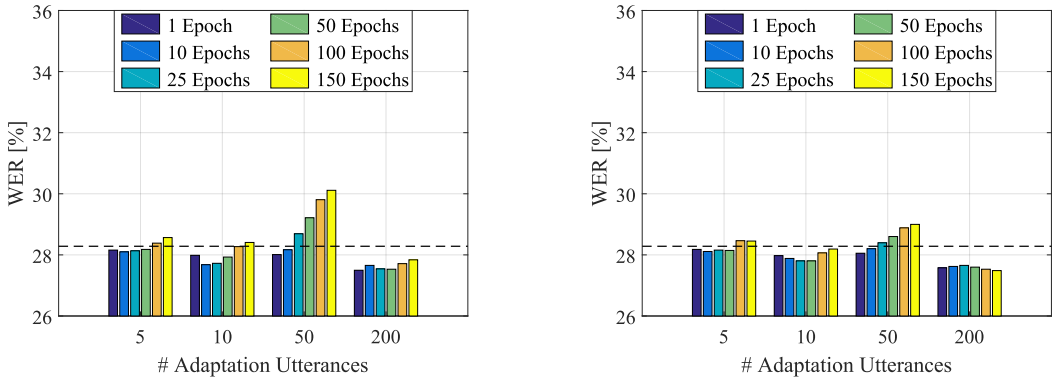
Since unsupervised adaptation uses partly wrong pseudo-transcriptions with the adaptation audio data, the gain in performance through it stays limited. In our experiments, the average WER over the speaker data is 28.28%, which implies that at least one over four phonemes is not recognized by the baseline system. The limited performance of the baseline system could have an effect on the limited WER gain after adaptation.

5.6.4 Test-Only Setup for Unsupervised Adaptation

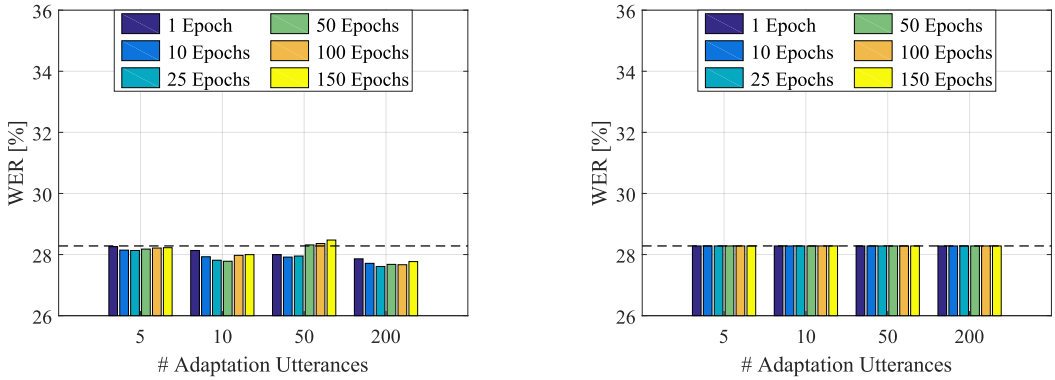
The above investigated techniques are based on an adaptive training setup. This mainly means that for each speaker we would like to adapt to, we have to gather adaptation data (in a supervised or unsupervised way), adapt the baseline model, and store the adapted model parameters. A way to avoid this would be to adapt on the fly in a test-only adaptation setup. This works as following: The speaker interacts with the ASR system, the system decodes the speech data and generates pseudo-transcriptions with a WER value = wer_1. The ASR system is adapted with these pseudo transcriptions and the speech data aiming to generate new transcriptions with a WER value = wer_2 so



(a) $\rho = 0$ **(b)** $\rho = 0.125$

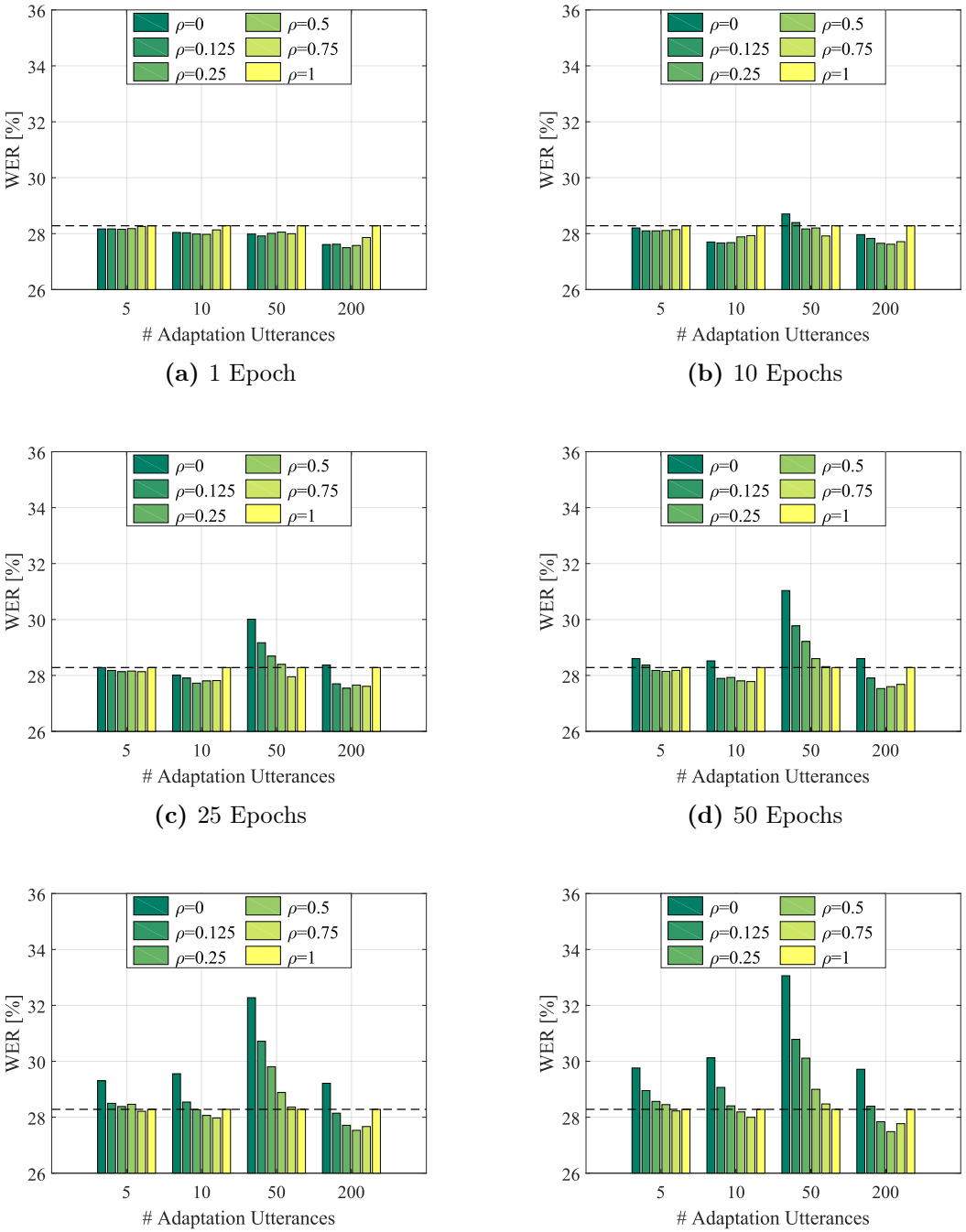


(c) $\rho = 0.25$ **(d)** $\rho = 0.5$



(e) $\rho = 0.75$ **(f)** $\rho = 1$

Figure 5.10: Unsupervised adaptation results using different regularization coefficients ρ [(a) (b) (c) (d) (e) (f)]. The dashed line represents the baseline WER using the SI model (28.28%)



(a) 1 Epoch

(b) 10 Epochs

(c) 25 Epochs

(d) 50 Epochs

(e) 100 Epochs

(f) 150 Epochs

Figure 5.11: Unsupervised adaptation results using different regularization coefficients over different adaptation epochs [(a) (b) (c) (d) (e) (f)]. The dashed line represents the baseline WER using the SI model (28.28%)

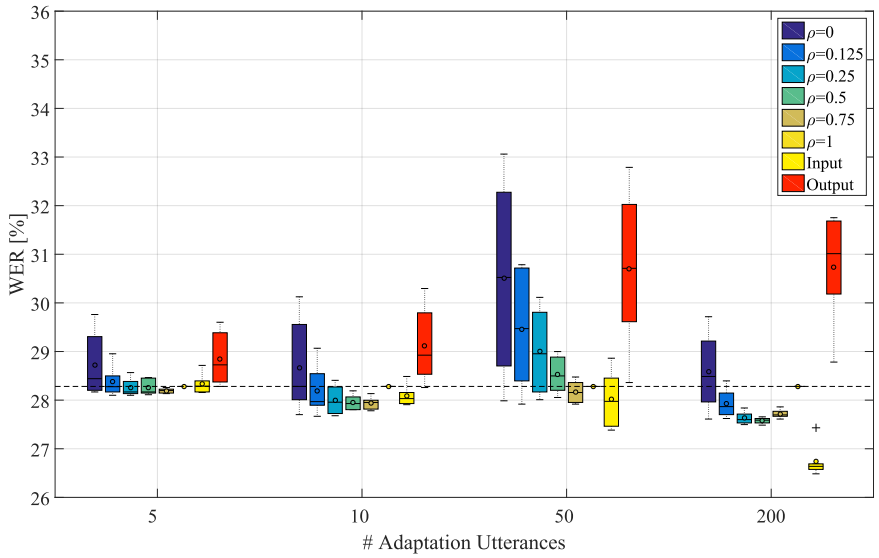
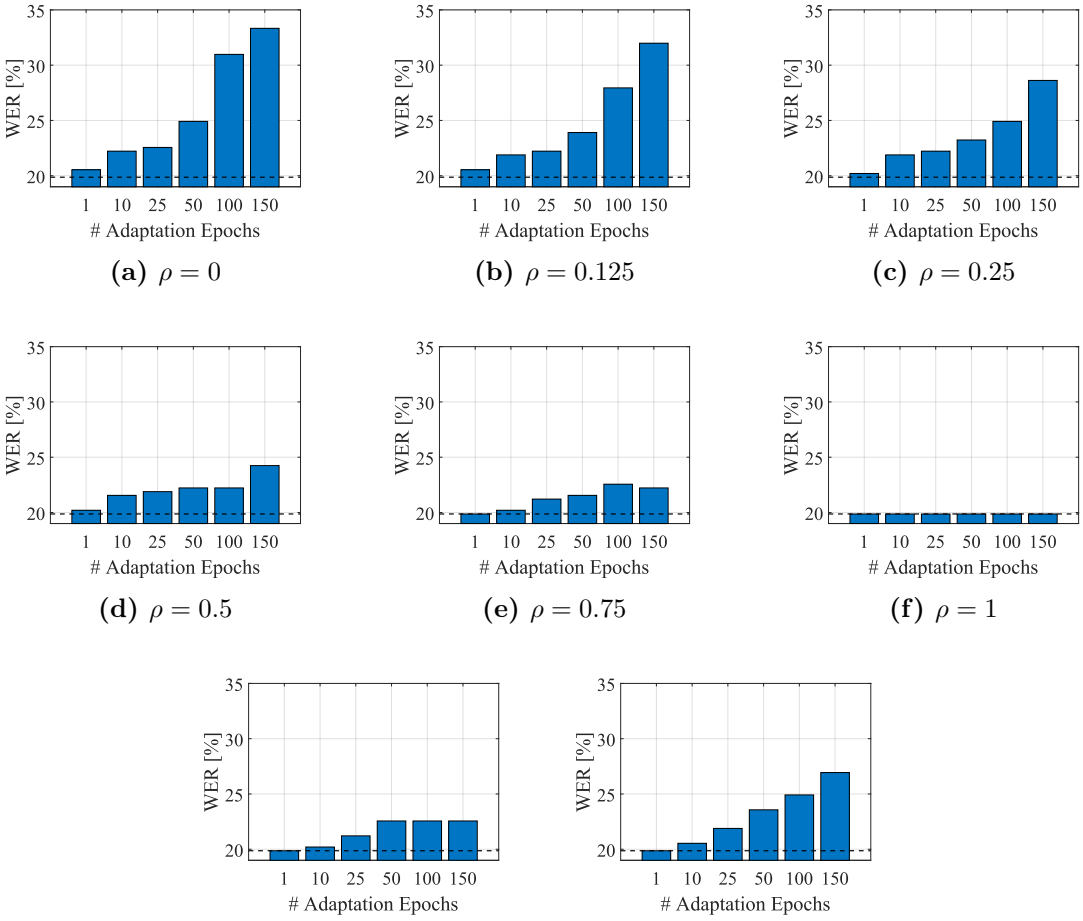


Figure 5.12: Overview of the performance of the investigated unsupervised adaptation methods for different adaptation data sizes over the adaptation epochs window from 1 to 150.

that $wer_2 > wer_1$. This method corresponds to unsupervised adaptation where, after the adaptation is finished, we evaluate how good the adapted system performs on the adaptation data itself.

In a real usage conditions, this adaptation method can only be conducted using a small amount of speech data for adaptation. For that, we will consider only the results of adaptation using 5 utterances. The calculated WER is the mean over the 6 adaptation subsets of the 6 test speakers. An average of about 50 words was used per speaker for adaptation and evaluation. The mean WER over the adaptation data using the baseline system is 19.86 %.

Figure 5.13 shows that all the adaptation techniques used in a test-only setup for unsupervised adaptation with a small data size (5 utterances) did not succeed in enhancing recognition performance. This can be due to the limited performance of the baseline system and its effect in generating the adaptation data labels. Also the small amount of adaptation data possibly does not include enough samples for all the output classes resulting in catastrophic forgetting, which in this case could not be contained by using regularization techniques.



(a) $\rho = 0$ **(b)** $\rho = 0.125$ **(c)** $\rho = 0.25$

(d) $\rho = 0.5$ **(e)** $\rho = 0.75$ **(f)** $\rho = 1$

(g) Training input layer **(h)** Training output layer

Figure 5.13: Recognition results over the unsupervised adaptation data (5 utterances) when using different adaptation techniques in a test only setup

6 Experiments on State of The Art ASR System

After investigating the adaptation techniques on "ASR System 1", we will test some of the methods considered earlier on a state of the art ASR systems. Such a system implies a larger DNN for acoustic modeling and hence a system with a better recognition performance. Details of the system and the conducted experiments will be discussed in this chapter.

6.1 System Architecture

The baseline system we used for our experiments is a state of the art speech recogniton system. The acoustic model was trained using the full audio data of the Switchboard corpus. The trigram language model was trained on 3M words of the Switchboard transcripts and was interpolated with another trigram language model trained on Fischer corpus transcripts (11M words).

The resulting baseline GMM-HMM acoustic model contained 8831 tied triphone states and about 200K Gaussians. The features used in the DNN framework were 25 dimensional filter bank features up to the second derivative, with a context window of size 11 resulting in a 825 dimensional input vector ($25 \times 3 \times 11$). The DNN was composed of 6 hidden layers containing 2048 neurons each and an output layer of 8831 neurons, each corresponding to a specific senone. The figure 6.1 depicts the architecture of the used DNN. We will use this developed context dependent DNN-HMM system as a baseline for the adaptation tasks discussed within this chapter.

Table 6.1 shows the word error rates of our baseline system compared to two other state of the art systems with comparable DNN architectures, developed by Maas et al. [15] and Veselỳ et al. [30] on Kaldi. The systems were tested on the 2000 HUB5 evaluation set, often used in literature to determine the performance of ASR systems. the HUB5 set (H5) contains 40 telephone conversations, 20 of them are from unreleased Switchboard conversation calls (SWBD), and 20 telephone conversations from the Callhome corpus (CH).

6.2 The Test Speakers Set

Since we used the entire Switchboard corpus for training our baseline system, we couldn't use speakers from that same corpus for adaptation. Therefore, we used adaptation data from a different corpus called Librispeech, a speech corpus based on public domain audio

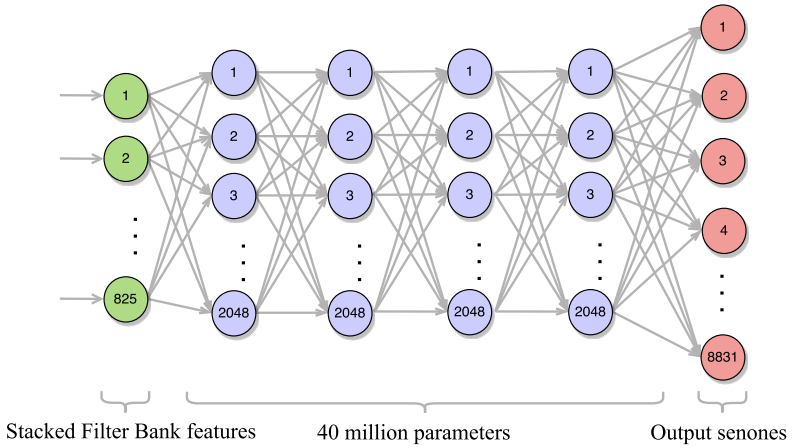


Figure 6.1: Architecture of the DNN used for acoustic modeling in the state of the art ASR system

ASR System	SWBD WER	CH WER	H5 WER
Our System	13,9 %	24,5 %	19,2 %
System [15]	15,1 %	27,1 %	21,2 %
System [30]	12,6 %	24,1%	18.4 %

Table 6.1: Performance comparison between the ASR system we use for the experiments and other state of the art systems proposed in [15] and [30]

books [33], in order to conduct our experiments. We extracted 8 different speakers for adaptation. For each of these speakers we extracted respectively 100 Utterances (about 25 minutes of speech data). The first 70 utterances were used as **adaptation set**. We split them in 5, 10, 25, 50 and 70 utterances in order to investigate the effect of adaptation data size over recognition accuracy. The last 30 utterances were used as **evaluation set**. The total number words in the evaluation set is 8290. Figure 6.2 shows a clearer overview over the used test speakers.

The goal of the experiments are to improve the overall recognition performance (Average WER) over the evaluation set through speaker adaptation. The performance of the SI system over the respective evaluation sets of the test speakers is shown in the table 6.2.

6.3 Choosing an appropriate learning rate

In order to choose the learning rate, we followed the same method described in section 5.4. We run supervised adaptation over a development speakers set of 4 speakers extracted from the Librispeech corpus. The data of the speakers had a similar structure as the data extracted for the test speakers, namely 70 utterances for adaptation and 30 for evaluation. We used 5, 10, 25, 50, 70 utterances to adapt and evaluated the recognition accuracy over the 30 utterances evaluation sets. We adapted over 1, 10, 25, 50 and 100

Speaker id	SI system WER
Spk1	21,27 %
Spk2	32,33 %
Spk3	27,56 %
Spk4	15,17 %
Spk5	20,21 %
Spk6	32,71 %
Spk7	21,6 %
Spk8	23,46 %
Average WER	24,28 %

Table 6.2: Word error rate (WER) over the 8 evaluation sets of the respective test speakers, and average WER using the speaker independent system introduced in section 6.1.

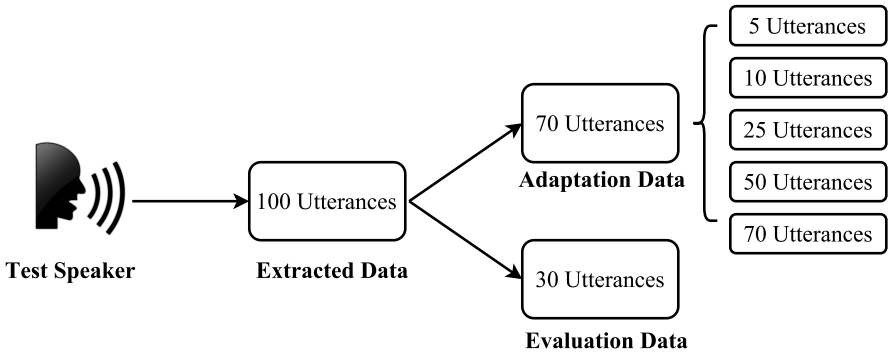
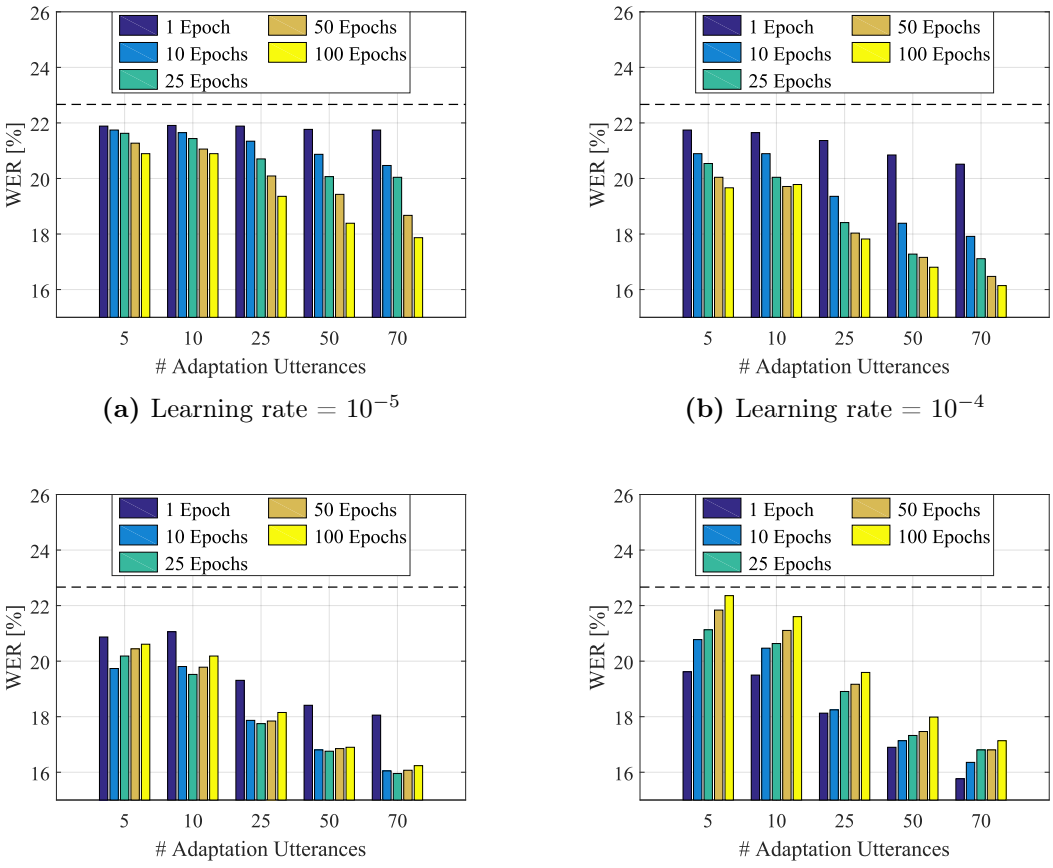


Figure 6.2: Overview of the test data extracted from each speaker and used for adaptation and evaluation

adaptation epochs. These experiments were run 4 times using 4 different learning rate values, mainly: $10^{-5}, 10^{-4}, 10^{-3}$ and 10^{-2}. The figure 6.3 shows how the recognition accuracy evolves over the adaptation epochs. Figures 6.3a and 6.3b show that using the learning rates 10^{-5} and 10^{-4} for adaptation enhances the recognition accuracy for different adaptation data sizes. However, it seems there is still a potential for a better accuracy at the hundredth epoch of adaptation in both cases, meaning that the used learning rate is possibly too small to observe a clear WER minimum in our adaptation range of 100 epochs.

Using a learning rate of 10^{-2} like shown in figure 6.3d shows an incremental progression in the word error rate over the the adaptation epochs. Even though the WER values show an enhancement of the recognition over the several adaptation epochs, this increasing trend doesn't enable us to observe a possible minimum in WER over our adaptation window. Figure 6.3c shows instead a bowl shaped trend of the WER over the adaptation window we chose. This permits us to observe the whole spectrum of how the WER value of the adapted models progresses over the adaptation epochs, reaches a potential



(a) Learning rate = 10^{-5} **(b)** Learning rate = 10^{-4}

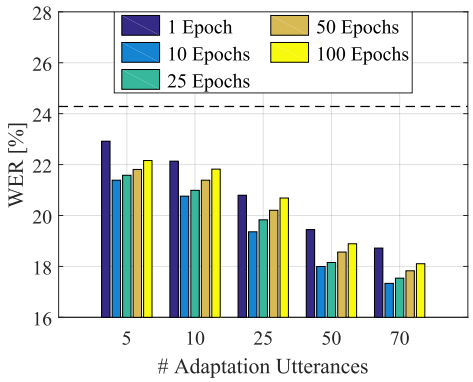
(c) Learning rate = 10^{-3} **(d)** Learning rate = 10^{-2}

Figure 6.3: Supervised adaptation results over development speakers with different learning rates [(a) (b) (c) (d)] . In each experiment different adaptation epochs (1,10,25,50 and 100) and different adaptation utterances (5,10,50 and 200) were used. The dashed line represents the mean SI system performance over the evaluation data

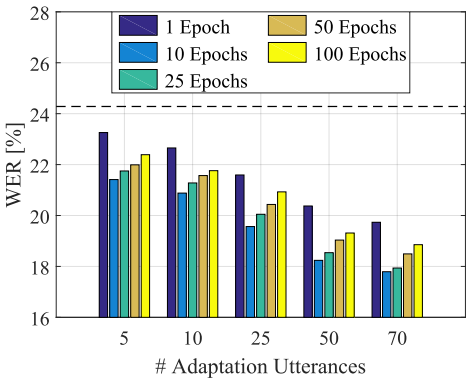
minimum and then starts raising up again. This is why the learning rate 10^{-3} seemed to us to be a reasonable choice to use in the adaptation experiments we conduct later.

6.4 Supervised Adaptation Results

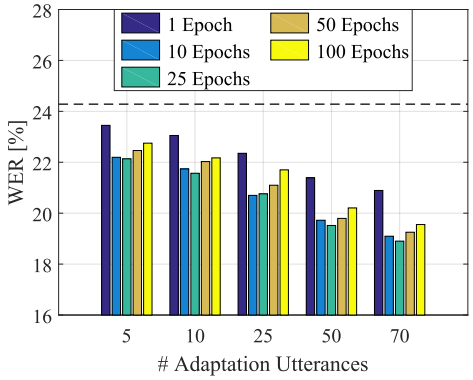
Since training only the output layer showed poor results for the first ASR system, we decided not to take it in consideration in adapting the state of the art system. The two techniques we decided to investigate are the following: training input layer and regularized adaptation with $\rho = 0$, $\rho = 0.25$, $\rho = 0.5$ and $\rho = 0.75$. We conducted these experiments on the test speakers, using 5, 10 ,25, 50 and 70 utterances per speaker for adaptation, over 1, 10, 25, 50 and 100 adaptation epochs. We used the learning rate we determined in section 6.3 (10^{-3}).



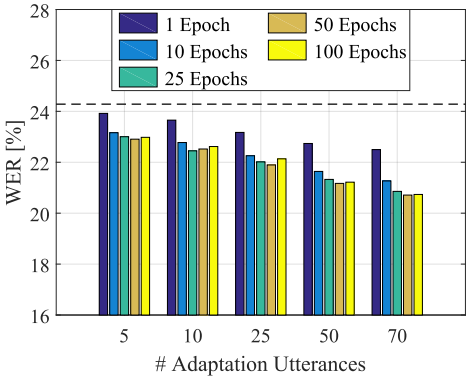
(a) $\rho = 0$



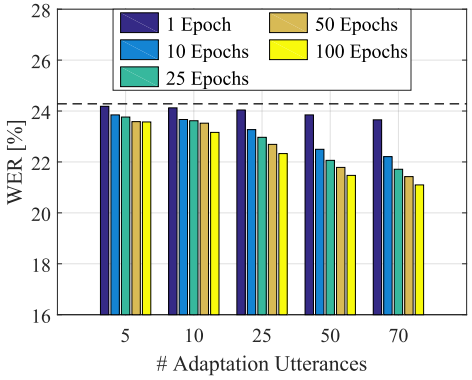
(b) $\rho = 0.25$



(c) $\rho = 0.5$

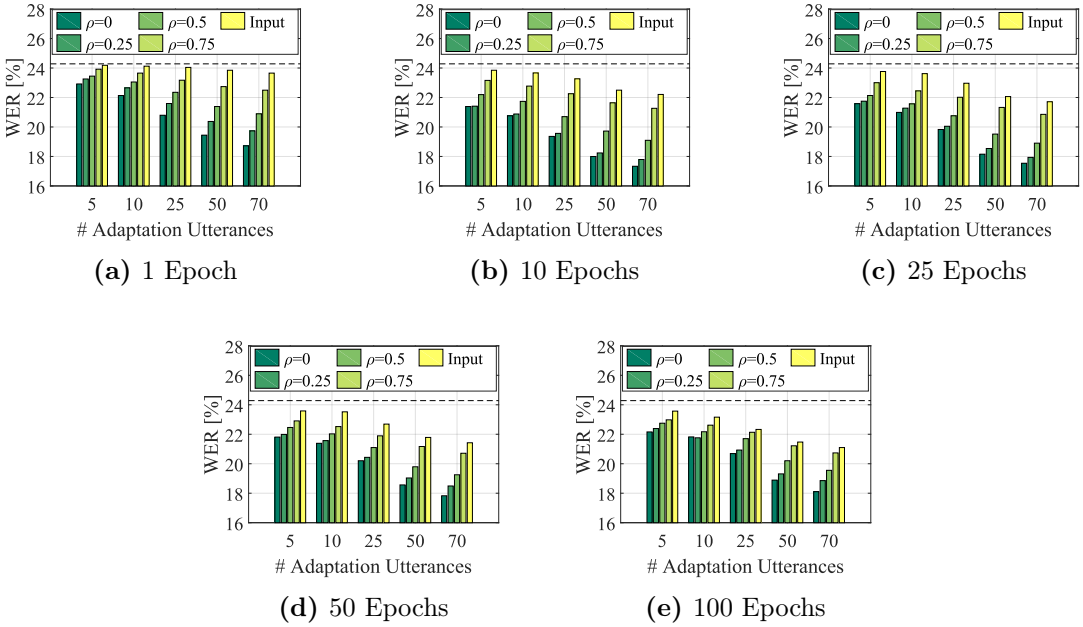


(d) $\rho = 0.75$



(e) Training input layer

Figure 6.4: Supervised adaptation results using different adaptation methods. The dashed line represents the baseline WER using the SI model (24.28%)



(a) 1 Epoch **(b)** 10 Epochs **(c)** 25 Epochs

(d) 50 Epochs **(e)** 100 Epochs

Figure 6.5: Supervised adaptation results using different adaptation techniques over different adaptation epochs. The dashed line represents the baseline WER using the SI model (24.28%)

Figure 6.4 shows the results obtained for each of these methods. The first thing to notice is that for all adaptation techniques, over the whole adaptation window and for different data sizes, there is always an improvement in performance against the baseline system. Adapting without regularization ($\rho = 0$, fig.6.4a) shows that for all adaptation data sizes, the best performance was reached after 10 epochs of adaptation, before the WER starts rising up again. Using only 5 utterances yielded to a maximal improvement of 2.9% in recognition. The use of 70 utterances for adaptation could achieve an enhancement of 6.9%. It is easy to notice the importance in data size for the improvement of the gain brought by adaptation.

When the regularization coefficient ρ is set to 0.25, we record the best performance in adaptation at 10 epochs, where the gain reaches 2.8% for 5 utterances and 6.5% for 70 utterances like shown in figure 6.4b. When the regularization coefficient ρ is set to 0.5 (figure 6.4c) the best performance in WER after adaptation can be seen at about 25 epochs of adaptation before the WER starts rising again. Using only 5 utterances for adaptation yielded in the best case scenario to 2.1% of improvement against the baseline. Using 70 utterances for adaptation could bring about 5.4% in improvement, way less than the results obtained when no regularization was used ($\rho = 0$). For a higher regularization coefficient ($\rho = 0.75$), we reach the best gain in performance later in our adaptation window, namely at 50 epochs. Figure 6.4d shows that the best improvement reached 1.3% with 5 utterances and a maximum of 3.5% with 70 utterances of adaptation.

When training only the input layer for adaptation we notice that the best performance in WER after adaptation can be seen at the last epoch of adaptation (100). In con-

trary to the regularized adaptation observed earlier, the WER decreases over adaptation epochs without rising again. Using 5 utterances for adaptation yielded to about 0.6% of improvement. Using a larger adaptation data size, mainly 70 utterances, produced 3.2% of improvement. The gain brought by this adaptation technique remains very limited compared to the use of regularized adaptation and is slower in terms of needed adaptation epochs.

The results of the five experiments conducted above showed that adapting the network without regularization ($\rho = 0$) yielded to the best improvement in performance of the state of the art ASR system. Using a regularization coefficient couldn't in this case outperform the non-regularized version of adaptation. Also using regularization didn't have the same effect as for "ASR system 1", since like shown in figure 6.5 non-regularized adaptation showed to perform better over all adaptation epochs. This could be partly due to the chosen learning rate value, which was probably small enough for this network, resulting in better adaptation results without the use of regularization even in the last epochs of our adaptation window. Training the input showed the worst results over the whole adaptation window.

6.5 Unsupervised Adaptation Results

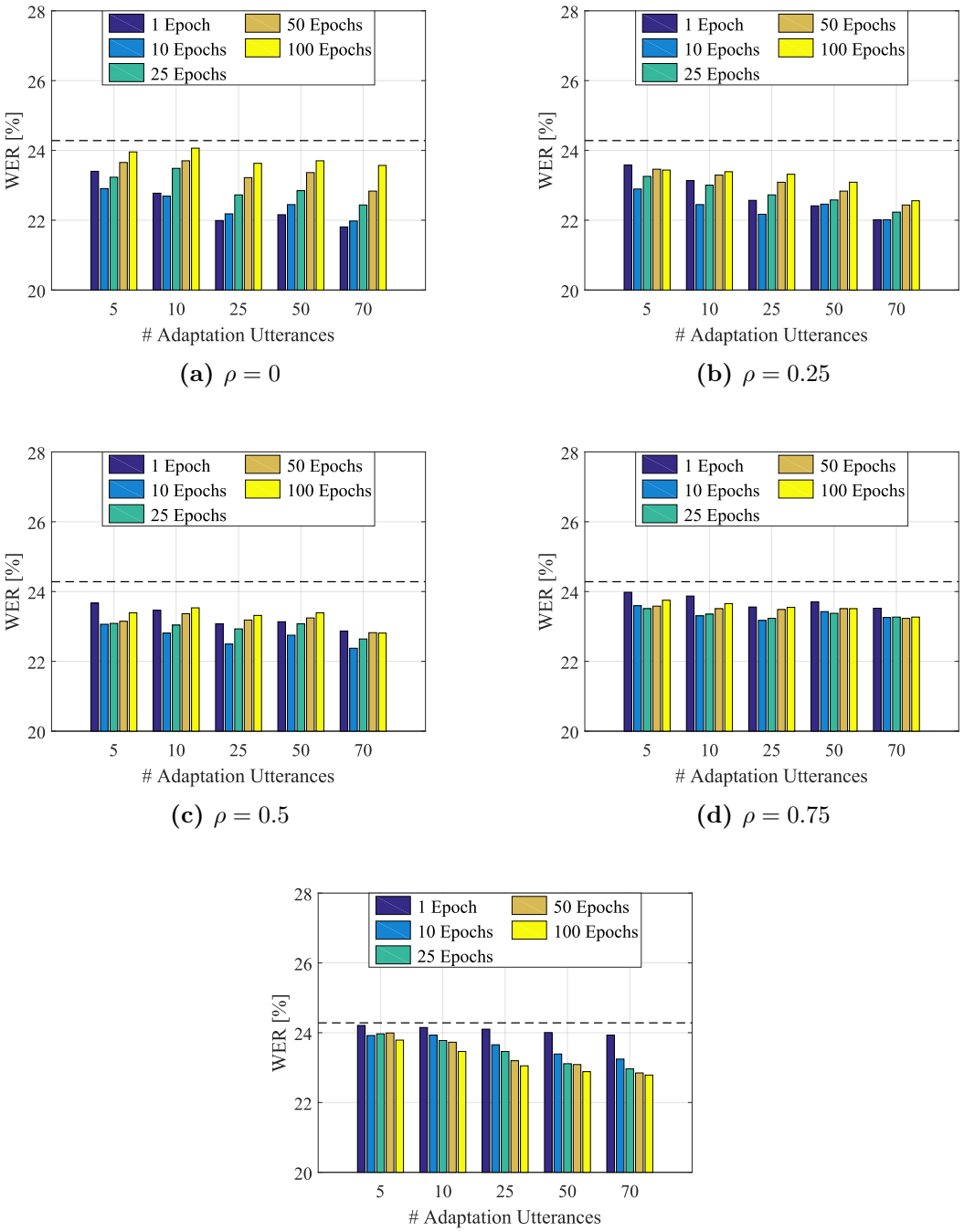
For unsupervised adaptation, we investigated the same techniques as for supervised adaptation, mainly: training input layer and regularized adaptation with $\rho = 0$, $\rho = 0.25$, $\rho = 0.5$, $\rho = 0.75$. We conducted these experiments on the test speakers, using 5, 10 ,25, 50 and 70 utterances per speaker for adaptation, over 1, 10, 25, 50 and 100 adaptation epochs. We used the learning rate we fixed in section 6.3 (10^{-3}).

Figure 6.6 shows the results of the conducted experiments. The first thing to notice is that through the whole adaptation epochs window and for different adaptation data sizes, the obtained WER was always better than the baseline. This was not the case when adapting "ASR system 1", and one reason could be the better overall recognition performance of this new baseline system.

Now as figure 6.6a shows, WER values over adaptation epochs have for most adaptation data sizes an incremental progression. That suggests that for more adaptation epochs, the recognition results would be worse than the baseline itself. The best performance reached for 5 utterances of adaptation data is 1.4% and for 70 utterances it reached 2.5 %. The incremental behavior of the WER seems to be contained by the use of regularization.

When using the regularization term $\rho = 0.25$, best gain in recognition with 5 utterances is 1.4% and 2.3% for 70 utterances. When $\rho = 0.5$, the gained performance in recognition goes down in comparison to the two previous experiments. In fact, like shown in figure 6.6c, we have an improvement of respectively 1.2% and 1.9% with 5 and 70 utterances. Finally when a high regularization term is used ($\rho = 0.75$), the potential improvement over adaptation goes further down, since we record a best case improvement of respectively 0.7% and 1% for a 5 and 70 utterances of adaptation data.

When training the input layer only (figure6.6e), the WER decreases over adaptation epochs. The best WER values achieved by this technique remain higher than what can be



(a) $\rho = 0$

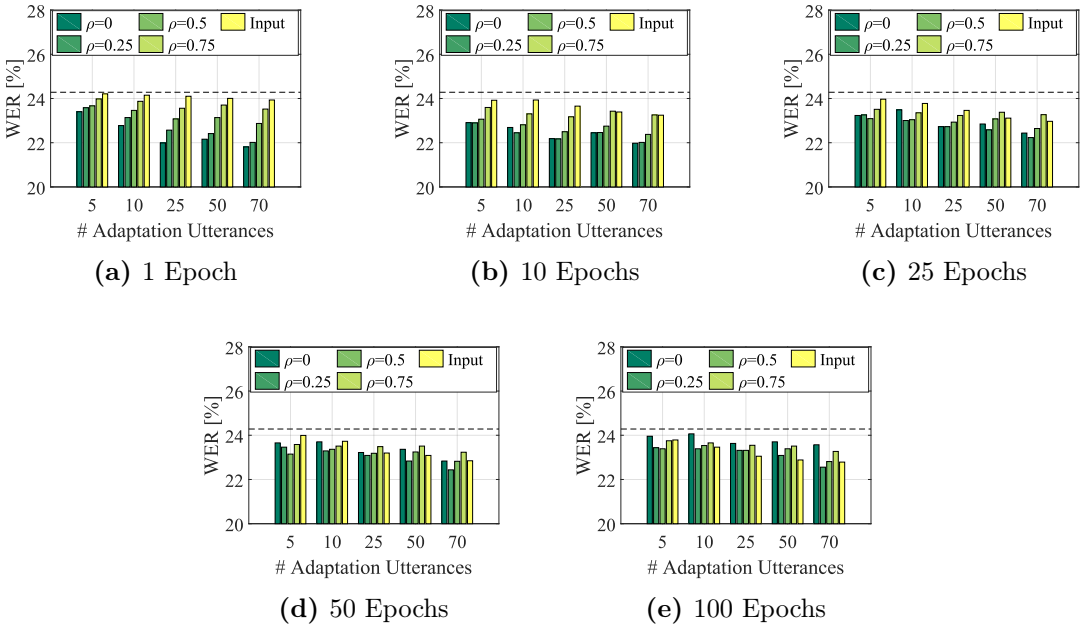
(b) $\rho = 0.25$

(c) $\rho = 0.5$

(d) $\rho = 0.75$

(e) Training input layer

Figure 6.6: Unsupervised adaptation results using different adaptation methods. The dashed line represents the baseline WER using the SI model (24.28%)



(a) 1 Epoch **(b)** 10 Epochs **(c)** 25 Epochs

(d) 50 Epochs **(e)** 100 Epochs

Figure 6.7: Unsupervised adaptation results using different adaptation techniques over different adaptation epochs. The dashed line represents the baseline WER using the SI model (24.28%)

achieved by regularized adaptation (0.5% and 1.5% for respectively 5 and 70 utterances). Though, WER values still maintain a decremental progression over adaptation epochs. This suggests that this technique could be the "safest" one to use to avoid generating a less efficient model than the baseline after adaptation. It gives a higher margin in terms of adaptation epochs. However it cannot reach the WER enhancement achieved in most of the previous experiments.

The Figure 6.6 shows WER progress over different adaptation epochs for the adaptation techniques we investigated. It shows that in the early adaptation epochs, not using regularization ($\rho = 0$) gives better results than other techniques. However in the last epochs of adaptation, using regularization or training only the input layer leads to achieving better results. This underlines the fact that using a non-zero regularization term could give a higher margin in terms of adaptation epochs without risking to deteriorate the baseline system, but do not necessarily deliver the best enhancement you can get from adaptation. The same applies for the training the input layer, where even if the gain recognition is small, we do not see any sign of the WER values wondering up towards the baseline value.

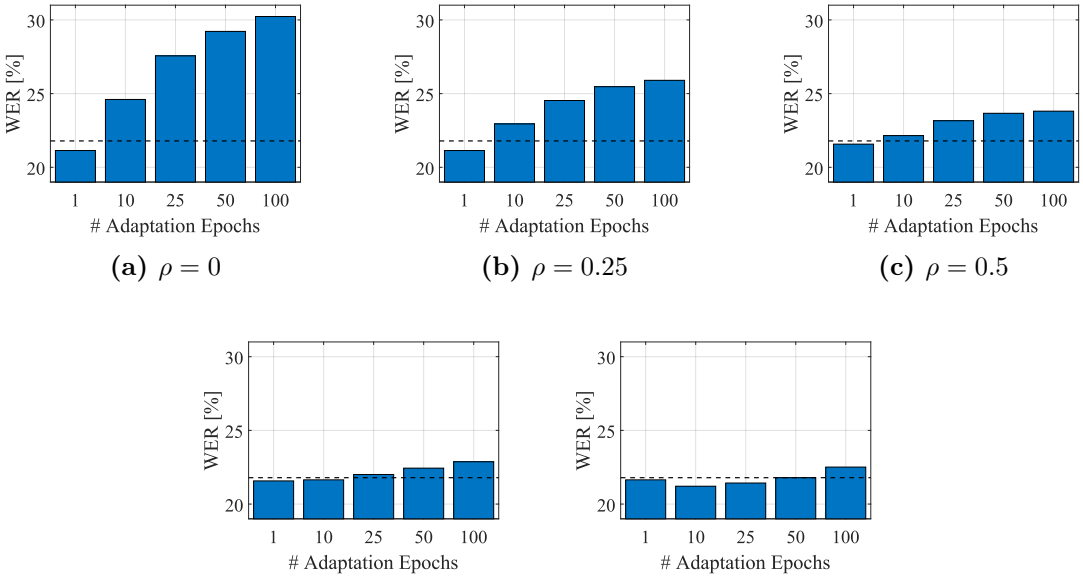
Figure 6.7 shows the improvements in WER for the different adaptation epochs. In the early stage of adaptation, namely in the 1^{st} and 10^{th} adaptation epochs, adaptation without regularization ($\rho = 0$) achieves the best WER values. However, starting from the 25^{th} adaptation epoch, the improvements achieved by using regularization tend to be better. This underlines the fact that regularized adaptation might help avoiding a potential catastrophic forgetting that might happen when the model is over-trained. But,

since regularization constraints the weights, it limits the model from reaching the best possible improvement like when it is the case of $\rho = 0$.

The enhancement in performance achieved through unsupervised adaptation is like expected worse than the enhancement achieved by the supervised one. It is also important to notice that the size of adaptation data has less influence on the WER improvement (figure 6.6) like it is the case for supervised adaptation (figure 6.4).

6.5.1 Test-Only Setup for Unsupervised Adaptation

Like explained in section 5.6.4, a test-only setup for unsupervised adaptation can be conducted in real usage scenario if the adaptation data size is small. That is why in this case we will consider this kind of adaptation for only 5 utterances as adaptation data. The calculated WER is the mean over the 8 adaptation subsets of the 8 test speakers. An average of about 110 words was used per speaker for adaptation and evaluation. The mean WER over the adaptation data using the baseline system is 21.78%.



(a) $\rho = 0$ **(b)** $\rho = 0.25$ **(c)** $\rho = 0.5$

(d) $\rho = 0.75$ **(e)** Training input layer

Figure 6.8: Recognition results over the unsupervised adaptation data (5 utterances) when using different adaptation techniques in a test only setup

Figure 6.8 shows the results of the conducted experiments using the several selected adaptation techniques. For the regularized adaptation with $\rho = 0$ (figure 6.8a), an improvement in recognition (0.6%) is to be seen after the first epoch of adaptation before the WER values start raising over the baseline WER.

When the regularization term is set to the values 0.25, 0.5 and 0.75, we see that the WER progression over the adaptation epochs is flattened, but no meaningful gain in

accuracy is to be noticed. Training only the input layer also shows improvement in recognition over the 1, 10 and 25 epochs that reaches at most 0.5%.

The potential performance gained in a test only setup is very limited. The 0,6 % improvement over 110 words is very small and the risk of performing worse than the baseline system are high. We conclude that the gain in a test-only setup using these methods is limited and doesn't give promising results.

7 Conclusion and Outlook

7.1 Conclusion

In the last six years, speech recognition systems started to gain some interest as a mean for human-machine interaction. This is mainly due to the recent availability of high computational capabilities and larger sets of speech data that allowed to introduce a new framework for acoustic modeling: deep neural networks. The use of DNNs in acoustic modeling showed a huge gain in recognition performance against the conventional Gaussian mixture models used before. But, even though DNNs improved recognition performance, we are still far from the 100 % recognition accuracy.

One of the reasons of the limits of speech recognition systems in a real usage scenario is the training-testing mismatch. This means that the assumption that the collected speech data used for training models follows the same distribution of the testing conditions is barely satisfied. This issue is not proper only to speech recognition tasks and can be encountered in other machine learning systems. In ASR systems, this problem translates in the mismatch between the speech data collected for training the models (mostly in a controlled environment) and the test data that can be encountered in a real usage scenario. Different accents, velocities in speaking or speaking environment (noisy or reverberant environment ...) can affect the recognition results and lead to a limited performance.

One of the proposed approaches to solve this issue is speaker adaptation. Several adaptation techniques of DNN based acoustic models have been proposed in order to reduce the performance gap caused by the training-testing mismatch issue. Adaptation can be achieved either by adapting the testing input (speech of certain user) to better fit the acoustic model or by adapting the acoustic model to better fit the testing conditions. In this work, we investigated methods from the latter category proposed by Liao [1] and Yu et al. [2] in order to explore their capabilities in improving the performance of a state of the art large vocabulary continuous speech recognition system developed by SONY. Now, adapting DNN based acoustic models can be quite complicated for several reasons. First, adapting neural networks may lead to catastrophic forgetting. In fact retraining a network with adaptation data from testing conditions may result in losing the former valuable information stored inside it. That can occur if the adaptation data is small and does not include enough samples from the output classes, leading potentially to a model that delivers a worse performance than the non-adapted one. The second issue with adaptation resides in the fact that high number of parameters have to be considered when retraining a DNN. In fact most of the literature dealing with adaptation, and especially the ones investigated in here, would use their own ASR systems trained on their own data with different architectures, and would adapt by using different learning

rates and adaptation epochs with very little justification of the values used during these tests. This makes the results obtained in these experiments difficult to verify and makes it almost impossible to retrieve a recipe-like approach of how to best adapt an acoustic model.

In this work, our main goal was to investigate these techniques and find out how are they are capable to improve the performance of a potentially deployable ASR system developed by SONY. We conducted different experiments by varying values of different parameters (adaptation data size, adaptation epochs, regularization coefficients...) for supervised and unsupervised scenarios and came to this conclusion:

Retraining the entire DNN without any KLD regularization brought mostly the best possible enhancement in performance. However this method might be risky if we ignore or don't fix a certain number of adaptation epochs beforehand, since it can lead to catastrophic forgetting. KLD regularization could potentially help avoiding this problem, but as the results for the state of the art network showed, it seems to have a very limited effect. What we are sure from is that KLD regularization doesn't necessarily score better results than adaptation without regularization. This mostly depends on the working point we consider during the experiments. Retraining only the input layers seems to be a more stable choice for adaptation. Even if it didn't lead to the best improvement, it showed to work for both supervised and unsupervised scenarios without any sign of deteriorating the recognition accuracy compared to the baseline non-adapted system. Adapting only the input layers has also the advantage of consuming less memory (about 4 % of the parameters) if the values relative to the speaker has to be stored.

For supervised adaptation, the used data size had a bigger impact over the improvement that can be achieved through adaptation. That was not the case for unsupervised adaptation, since the data size showed less impact over the improvement. This suggests the possibility of economizing in the adaptation data size in an unsupervised adaptation scenario. The test-only setup for the unsupervised adaptation scenario showed barely improvements in recognition accuracy and leaded mostly in deteriorating the performance of the baseline system, which suggests that this method shouldn't be considered in a future adaptation scenario.

Apart from the difference in performance between the several methods considered in this work, we showed that model based adaptation could bring much gain in performance even for an already very strong speaker independent ASR system (up to 6.9% and 2.5% respectively for supervised and unsupervised adaptation).

7.2 Outlook

These results show how adaptation can be quite delicate and how the potential improvements in recognition accuracy can be strongly dependent from several parameters like number of adaptation epochs, learning rate values and adaptation data size. Adapting with small learning rates and number of epochs with early stopping can be a good approach to find the "best" hyperparameters to use for a certain system. However it is quite difficult to extrapolate a recipe-like approach that can be applied without any sort

of monitoring. One strategy that could be investigated in future work would be setting for example a *time budget* for adaptation that combines the data size and the number of adaptation epochs (exp: *time budget=nb. adaptation frames* $\times$ *nb. adaptation epochs*). Since the data size is probably unknown in beforehand in a real usage scenario and the number of adaptation epochs costs time, *time budget* can be a more meaningful parameter to consider in an adaptation scenario. It would also reduce the number of parameters considered in these experiments and possibly lead to more usable results. Since the unsupervised adaptation form is strongly preferred to the supervised form, one could in future work consider the promising adaptation method proposed by [34]. It introduces a method called LHUC (Learning hidden unit contributions) which is a model based speaker adaptation technique that learns speaker specific hidden unit contributions from a given set of non-labeled adaptation data.

Bibliography

1. Hank Liao. Speaker adaptation of context dependent deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7947–7951. IEEE, 2013.

2. Dong Yu, Kaisheng Yao, Hang Su, Gang Li, Frank Seide, Online Services Division, and Microsoft Corporation. Kl-Divergence Regularized Deep Neural Network Adaptation for. pages 7893–7897, 2013.

3. G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012.

4. Dong Yu. *Automatic speech recognition : a deep learning approach*. Springer, London, 2015. ISBN 1447157788.

5. Li Deng and Dong Yu. Deep learning: Methods and applications. *Foundations and Trends Signal Processing*, 7(3-4):197–387, June 2014.

6. I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *ArXiv e-prints*, December 2013.

7. Noelia Alcaraz Meseguer. *Speech analysis for automatic speech recognition*. PhD thesis, Norges teknisk-naturvitenskapelige universitet, 2009.

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

9. Acoustic modeling - microsoft research, June 2016. URL `http://research.microsoft.com/en-us/projects/acoustic-modeling/`.

10. Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

11. Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2008.

12. Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book. *Cambridge university engineering department*, 3:175, 2002.

13. Geoffrey Zweig and Michael Picheny. Advances in Large Vocabulary Continuous Speech Recognition. *Advances in Computers*, 60(C):249–291, 2004.

14. Alexander Clark, Chris Fox, and Shalom Lappin. *The handbook of computational linguistics and natural language processing*. John Wiley & Sons, 2013.

15. Andrew L. Maas, Awni Y. Hannun, Christopher T. Lengerich, Peng Qi, Daniel Jurafsky, and Andrew Y. Ng. Building dnn acoustic models for large vocabulary speech recognition. *CoRR*, abs/1406.7806, 2014.

16. Herve A. Bourlard and Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach (The Springer International Series in Engineering and Computer Science)*. Springer, 1994 edition, 10 1993.

17. Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016. URL `http://www.deeplearningbook.org`.

18. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

19. Ben Gold, Nelson Morgan, and Dan Ellis. *Speech and audio signal processing: processing and perception of speech and music*. John Wiley & Sons, 2011.

20. George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

21. George E Dahl, Dong Yu, Li Deng, and Alex Acero. Large vocabulary continuous speech recognition with context-dependent dbn-hmms. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4688–4691. IEEE, 2011.

22. Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*, pages 437–440, 2011.

23. Q. Yan and S. Vaseghi. A comparative analysis of uk and us english accents in recognition and synthesis. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–413–I–416, May 2002.

24. Jan Trmal, Jan Zelinka, and Luděk Müller. *Adaptation of a Feedforward Artificial Neural Network Using a Linear Transform*, pages 423–430. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

25. Xiao Li and Jeff Bilmes. Regularized adaptation of discriminative classifiers. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE, 2006.

26. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer, 2nd ed. 2009. corr. 7th printing 2013 edition, 4 2011.

27. Dario Albesano, Roberto Gemello, Pietro Laface, Franco Mana, and Stefano Scanzio. Adaptation of artificial neural networks avoiding catastrophic forgetting. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1554–1561. IEEE, 2006.

28. Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.

29. Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi Speech Recognition Toolkit. *Proc. ASRU*, 2011.

30. Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey. Sequence-discriminative training of deep neural networks. In *INTERSPEECH*, pages 2345–2349, 2013.

31. J. J. Godfrey, E. C. Holliman, and J. McDaniel. Switchboard: telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520 vol.1, Mar 1992.

32. Andrew Cameron Morris, Viktoria Maier, and Phil D Green. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *INTERSPEECH*, 2004.

33. Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

34. P. Swietojanski and S. Renals. Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 171–176, Dec 2014.