University of Szeged

Ph.D thesis

Training Methods for Deep Neural Network-Based Acoustic Models in Speech Recognition

Author:
Tamás Grósz

Supervisor:
Dr. László Tóth

PhD School in Computer Science
MTA-SZTE Research Group on Artificial Intelligence
Faculty of Science and Informatics
University of Szeged

Szeged, 2018

Preface

Nowadays, speech recognition technology is built on Deep Neural Networks. These networks represents the latest direction of machine learning. They are based on the theory of artificial neural networks, which have been used for decades. However, unlike traditional Neural Networks, all deep networks contain many processing layers, which allow the hierarchical processing of the input data. While the concept of deep networks is not totally new, their efficient training required several new achievements. These new networks managed to completely replace the Gaussian Mixture Models in the state-of-the-art speech recognition systems.

In this study, I decided to focus on Deep Neural Network-based recognition systems. First, I compared the performance of several new training algorithms with each other, in order to determine the best one for later use. Then, I turned my attention to the algorithms that the new speech recognition systems have inherited from the previous Gaussian Mixture Model-based approaches, as the algorithms might not be optimal for Deep Neural Networks. I proposed new algorithms for obtaining the initial alignment of the frame-level state labels and the creation of context-dependent states, and found that they are better suited for the new acoustic models. Lastly, I also experimented with a data re-sampling method to improve the accuracy of the models.

The first of my acknowledgements goes to my supervisor, László Tóth, for his help, guidance and support throughout my PhD studies. Secondly, I would like to thank Gábor Gosztolya for his constant support and constructive suggestions, which were essential for the accomplishment of the work presented in this thesis.

This study was supported by the 'New National Excellence Programme' of the Ministry of Human Capacities (ÚNKP-16-3 and ÚNKP-17-3). I am grateful for this support, which acted as a great incentive for my research work and the submission of this thesis.

Last but not least, I would like to thank David P. Curley for scrutinising and correcting this dissertation from a linguistic point of view.

Tamás Grósz, February 2018.

Abbreviations

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CD	Context-dependent
CE	Cross Entropy
CI	Context-independent
CPU	Central Processing Unit
CTC	Connectionist Temporal Classification
DNN	Deep Neural Network
DRN	Deep Rectifier Network
FBANK	mel-Filter Bank
FMLLR	Feature space Maximum Likelihood Linear Regression
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
MFCC	Mel-Frequency Cepstral Coefficient
MMI	Maximum Mutual Information
WER	Word Error Rate

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Automatic Speech Recognition (ASR) is a key topic of speech technology, where the goal is to transcribe an audio recording (an *utterance*) in an automatic way. For decades the traditional ASR systems used Hidden Markov Models (HMM) with Gaussian Mixture Models (GMM) and, until very recently, these HMM/GMM models represented the state-of-the-art technology in ASR. Nowadays, with the advent of Deep Neural Networks (DNN) the original HMM/GMM models have been replaced by the new HMM/DNN hybrids.

DNNs are a new type of Artificial Neural Networks, which differ in one important aspect from the previous ones, namely that they have more than one hidden layer (usually three or more). This definition might seem a little vague and the question arises of whether this means that we only need to add a few new hidden layers to an ANN to upgrade it to a DNN. The simple answer of course is no, as with the addition of extra hidden layers we come up against several problems that make it hard to train the nets. So besides adding new hidden layers, we need other modifications like changing the activation function of the neurons or the learning algorithm itself.

The new HMM/DNN hybrids are now routinely used in state-of-the-art ASR systems, but they inherited many of the algorithms from their predecessors (the standard HMM/GMM systems). However, the optimality of these algorithms is not guaranteed with the new models. The main focus of this dissertation is to modify some of these earlier methods in speech recognition so that they better suit the new DNN-based acoustic models. Our main goal is to create new solutions that allow the training of HMM/DNN acoustic models without relying on GMMs during the training process.

To achieve the GMM-free training of a HMM/DNN hybrid, we have to solve two key problems, namely the initial alignment of the frame-level state labels and the creation of context-dependent (CD) states. We solved the first problem by modifying a standard sequence discriminative training method and showed that with the modified algorithm it is possible to train randomly initialised DNNs without the frame-level alignment

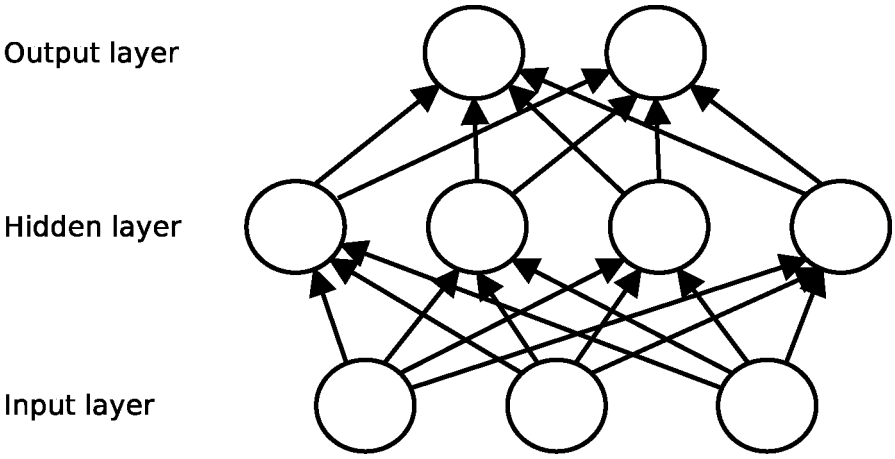

Figure 1.1: An example ANN structure with one hidden layer.

of the context independent (CI) labels. For the creation of CD states, we proposed a solution which applies a Kullback-Leibler divergence-based decision criterion during state clustering. Quite recently, several articles have been published about GMM-free systems, so we also compared the performance of our methods with some of these new approaches and found that our algorithms are quite competitive. Furthermore, we also addressed a special problem of the CD states, namely that of the imbalanced class distribution. We showed that a very simple re-sampling method with the adjustment of the priors can significantly improve the accuracy of DNN-based acoustic models.

1.1 Artificial Neural Networks

Now, we will give a brief description of Artificial Neural Networks (ANN) [2]. The concept of ANNs was inspired by biological neural networks, and the basic building block of these networks is the artificial neural model called the *perceptron*. In an ANN, these neurons form layers so that the neurons in one layer are connected to neurons from other layers (the connection is directed), and each connection has a weight which represents the strength of the given connection. The layers can be grouped into three categories. The input stimuli are passed to the network through the input layer, and the response of the ANN is observable in the output layer. The hidden layers are responsible for extracting different features (hidden representations), and this is where the actual processing is done. Figure 1.1 depicts a simple ANN structure.

The neurons are very simple processing units. They receive the activations of other neurons (x) through the incoming connections, then they calculate the weighted sum of these values using the weights (w). After the sum has been calculated, bias value (b) of the neuron is also added. Then, the activation function (f) is applied to determine

the output (o) of a given neuron. Formally,

$$o(x) = f\left(\sum_{i=1}^{M} w_i x_i + b\right), \tag{1.1}$$

where M is the number of inputs for the given neuron.

One of the most widely spread activation functions for hidden neurons is the sigmoid function,

$$Sigmoid(x) = \frac{1}{1+e^{-x}}. \tag{1.2}$$

This function is still in use mostly in shallow networks, as in deep structures it is plagued by an effect called the vanishing gradient effect. The output neurons of an ANN classifier use a special activation function called the softmax function. It is defined by the relation

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}, \tag{1.3}$$

where N is the number of output neurons. By applying this type of activation, we can interpret the output of the network as a posterior probability vector, as the values fulfil all the requirements, since they are guaranteed to be non-negative and add up to one.

The last thing we need to address here is the training algorithm of the ANN. The backpropagation algorithm offers a simple solution to this [2]. The first thing it requires is an error function, which determines the error by comparing the output produced by the ANN and the expected output. For classification tasks, we minimise the cross-entropy (CE) cost function

$$CE(p, y) = -\sum_{i=1}^{N} y_i log(p_i), \tag{1.4}$$

where y is the one-hot expected output vector and the p_i values are the activations of the output neurons. Using CE we can easily calculate the error of each output neuron, then all we need to do is to propagate this error back to the hidden neurons. Once each hidden neuron has an error value, the gradients of the weights and biases can be calculated. After the gradient computations, the parameters are updated in an attempt to minimise the error function. For more details on how the backpropagation algorithm works, see [2].

1.2 Automatic Speech Recognition

Automatic Speech Recognition or Speech-to-Text systems seek to transcribe the audio input automatically, where the transcription is usually a sequence of words or in some

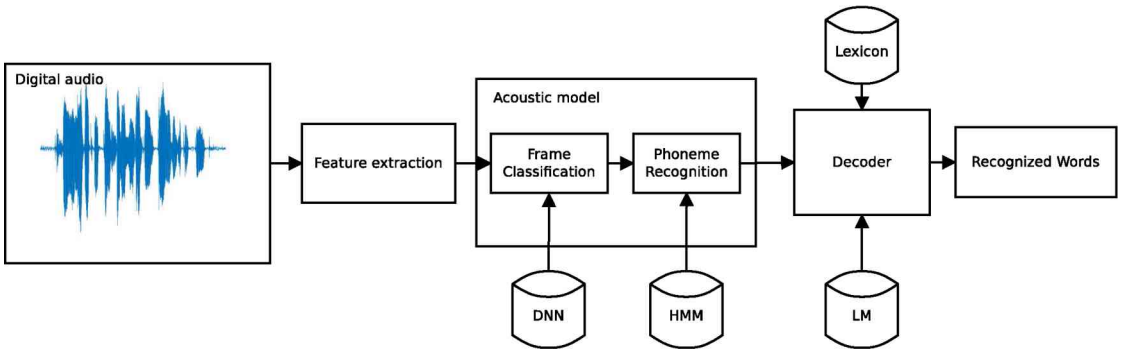

Figure 1.2: The standard workflow of a HMM/DNN-based ASR system.

cases a sequence of phonemes. This transformation is not an easy task; it takes humans years to learn it, and even then we still make mistakes (we mishear words) [3].

A standard ASR system consists of three main components, as shown in Figure 1.2; namely the feature extractor, the acoustic model and the language model. In the following sections we will explain the functions of these parts, but before that we need to address an important question. It is how we can determine which methods are better and which are worse. Many ASR solutions have been proposed over the years so we need an evaluation metric to compare the results of these methods. Perhaps the most straightforward way of comparing different ASR solutions is to evaluate them on the same test data and calculate their accuracy values. To calculate the accuracy, we need to compare the transcriptions produced by the system with the original reference transcriptions. For this, first the optimal alignment is found using a dynamic programming method, and then the number of substitution (S), deletion (D) and insertion errors (I) can be calculated. The accuracy metric is then defined as

$$Accuracy = \frac{N - D - S - I}{N}, \tag{1.5}$$

where N is the total number of words or phonemes in the reference transcriptions. With this metric, we can easily compare the performance of different systems.

1.3 Feature Extraction for Speech Processing

Next, we will focus on the most popular methods that are used to transform the raw speech waveform into a sequence of parameter vectors. The feature extraction step is an essential part of the speech recognition pipeline, but we should mention that very recently a new alternative has appeared. Some new networks attempt to use the raw audio input without any transformation [4]; however the recognition accuracies of these approaches are still far from those of the best systems built on standard feature extraction methods.

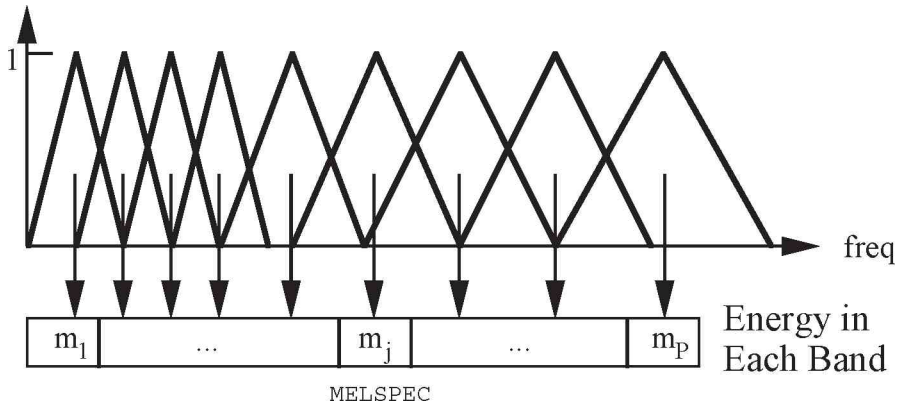

Figure 1.3: Illustration of the triangular filters on the mel-scale (image from the HTK-Book [1])

1.3.1 Filterbanks

It is well known that the human ear resolves frequencies non-linearly across the audio spectrum, and empirical evidence also suggests that designing a front-end to operate in a similar non-linear manner ought to improve recognition performance. Filterbank analysis (FBank) offers a straight-forward route for obtaining the desired non-linear frequency resolution [1]. As can be seen in Figure 1.3, the filters used by the FBank analysis have a triangular form and they are equally spaced along the mel-scale, which is defined by

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700}). \tag{1.6}$$

The FBank extractor first transforms a window of speech data using a Fourier transform and the magnitude is calculated. The magnitude coefficients are then binned by correlating them with each triangular filter. These triangular filters are spread over the whole frequency range from zero up to the Nyquist frequency, and binning means that each Fast Fourier Transformation (FFT) magnitude coefficient is multiplied by the corresponding filter gain. After the accumulation of the results, each bin holds a weighted sum representing the spectral magnitude in that filterbank channel. For the last step, we take the logarithm of the bins values to get the final FBank features.

The main problem with FBank features is that they are highly correlated, so if we want to use it as the input of a HMM/GMM based recogniser, we will need to apply a cepstral transformation first.

1.3.2 Mel-Frequency Cepstral Coefficients

For decades the most favoured feature type in speech recognition was the Mel-Frequency Cepstral Coefficients (MFCCs) [1]. These are calculated from the FBank amplitudes

$\{m_j\}$ using the Discrete Cosine Transform (DCT)

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j-0.5)\right), \tag{1.7}$$

where N is the number of filterbank channels. By taking just the first few basis vectors we get a good, data-independent approximation of the principal dimensions. Doing this, the features becomes decorrelated, hence they can be used by a GMM.

To further augment the FBank or MFCC feature sets it is common practice to concatenate the energy of the speech window with the features. The energy in a frame is computed as the log of the signal energy; that is, for speech samples $\{s_n\}$

$$E = log \sum_{n=1}^{T} s_n^2, \tag{1.8}$$

where T is the number of samples in a given frame.

1.3.3 Feature-space Maximum Likelihood Linear Regression

Nowadays, speaker adapted systems achieve the best results on many speech recognition tasks. Speaker adaptive training (SAT) could and should be used if the training corpus contains a sufficient amount of speech from multiple speakers. The main idea of SAT is to transform every utterance in the training corpus before we train the acoustic model, the goal of this transformation being to reduce the interspeaker differences, while keeping the intraspeaker variations. Feature-space Maximum Likelihood Linear Regression (fMLLR), also known as constrained MLLR, is a widely used speaker adaptation technique, but it can only be used if a HMM/GMM system trained on MFCCs is available. Next, we shall give a brief description of how fMLLR works based on the article of Povey and Saon [5]. The fMLLR method applies a very simple affine transformation in the form of

$$\hat{x^t} = W^s \phi^t, \tag{1.9}$$

where $\phi^t = [{}^{x^t}_{1}]$ is the extended input feature vector at time t and $W^s = [A^s, b^s]$ is the transformation matrix of speaker s. The name "constrained" comes from the fact that only one transformation matrix (A) is used instead of using separate transformations for the means and the covariances. To find the best W^s transformations, first we define the auxiliary function as the sum of $log|det(A^s)|$ and the likelihood of $\hat{x^t}$. The part of the auxiliary function that changes with the current transform W^s (excluding the determinant) can be written as

$$-0.5 \sum_{m=1}^{M} c^{sm} E\Big(\sum_{i=1}^{d} \frac{(\mu_i^{(m)} - w_i^T \phi^t)^2}{\sigma_i^{2(m)}} \Big)^{sm}, \tag{1.10}$$

where c^{sm} is a constant, $E(.)^{sm}$ calculates the average value for speaker s and Gaussian m, $\mu^{(m)}$ and $\sigma^{2^{(m)}}$ are the means and covariances of Gaussian m, respectively.

To simplify the auxiliary function, let us define the linear and quadratic terms in w_i (the ith row of W^s) as k_i and G_i:

$$k_i = \sum_{m=1}^{M} \frac{c^{sm}\mu_i^{(m)} E(\phi)^{sm}}{\sigma_i^{2^{(m)}}}, \tag{1.11}$$

$$G_i = \sum_{m=1}^{M} \frac{c^{sm} E(\phi\phi^T)^{sm}}{\sigma_i^{2^{(m)}}}. \tag{1.12}$$

Then the auxiliary function can be expressed as

$$log(|det(A)|) - \sum_{i=1}^{d}(w_i^T k_i - 0.5 w_i^T G_i w_i). \tag{1.13}$$

Lastly, the W^s matrices are estimated by using a row-by-row method that maximises Equation 1.13. For more details on fMLLR, see [6].

To summarise, fMLLR offers an easy way to perform SAT, but it has a big drawback as speaker annotations are needed for each utterance before adaptation.

1.3.4 Using the Δ and $\Delta\Delta$ features

Empirical studies demonstrate that the performance of a speech recognition system can be greatly enhanced by adding time derivatives to the basic static parameters. One possible explanation of why this helps is that by doing so we basically extend the input window of the acoustic model, thus we allow it to use information from a wider time window. The delta coefficients can be computed using the following regression formula [1]

$$\Delta_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2}, \tag{1.14}$$

where Δ_t is a delta coefficient at time t computed in terms of the corresponding static coefficients from $c_{t-\Theta}$ to $c_{t+\Theta}$. If we apply the same formula to the delta coefficients, we get the acceleration coefficients ($\Delta\Delta$).

1.4 Acoustic modelling with HMMs

After the feature extraction step, we can train the acoustic model component of the recogniser. This task is not an easy one as the acoustic model has to learn the relationship between the input features and the words uttered. Traditional systems first split

the transcription at the word-level into a phoneme sequence, and these phonemes can be defined as speech segments that possess distinct physical or perceptual properties. The phonemes are the basic building blocks in speech recognition.

Now we need a method that can be trained to produce the correct phoneme sequence given the input features. Hidden Markov Models (HMMs) are the most popular choice for this task. A HMM is characterised by the following:

- S is the finite set of the states, which are also called hidden states (as they are not directly observed). In speech recognition, each state corresponds to a phoneme or part of the phoneme. We shall denote the individual states by $S = s_1, s_2, \ldots, s_N$, and the state at time t by q_t
- $O = o_1, o_2, \ldots, o_T$ is the observation sequence, and $V = v_1, v_2, \ldots, v_M$ is the set of the individual symbols, which can be omitted.
- The actual state could be determined using the state transition probability values $A = a_{ij}$, where
$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i). \tag{1.15}$$
- The output of the HMM is controlled by the observation probability distribution $B = \{b_i(k)\}$, where
$$b_i(k) = P(o_t = v_k | q_t = s_i). \tag{1.16}$$
- Lastly the model needs an initial state distribution, which stores the probability values of $P(q_1 = s_i)$.

As can be seen, HMMs make two key assumptions. First, they assume that the current observation (o_t) is only dependent on the actual state (q_t). The problem with this is that we expect the model to correctly guess q_t using only one input frame, which is usually a 25 ms-long MFCC or Fbank vector, which represents only a fraction of the average phoneme duration ($\tilde{}150$ ms). A possible solution to overcome this problem is to extend the input with a few neighbouring frames, thus allowing the model to make decisions using a more appropriate time window. The second assumption that HMMs make is that s_t only depends on s_{t-1} and it is conditionally independent of the other preceding states. One could compensate for this by further extending the input window, but empirically a better solution is to use the Δ and $\Delta\Delta$ features, as this allows the system to guess the previous and successive values efficiently.

As we mentioned before, the hidden states usually correspond to a phoneme, but the problem of co-articulation complicates this. The most common way of dealing with co-articulation is to use a tri-state model. In a tri-state model each phoneme is split into three parts; namely the beginning, the middle and the ending part. Figure 1.4

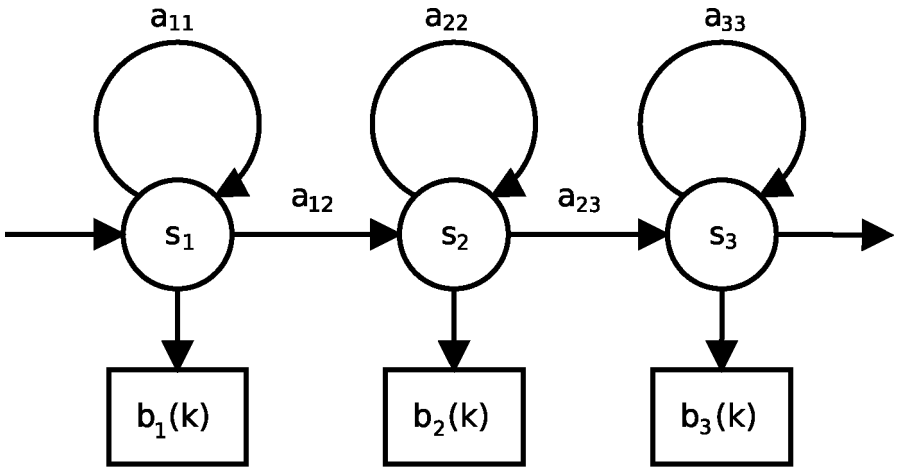

Figure 1.4: The tri-state HMM phoneme model.

shows the tri-state phoneme model. The idea behind this is simple. As co-articulation does not affect the middle part of the phone, by separating the problematic parts we hope to make the learning task easier. The tri-state model has an additional benefit that it enforces a minimal duration (3 frames at least) on each phoneme as the decoder has to go through all three states. Using context dependent (CD) labels is another option. In this case, we use three phonemes to label the actual observation, namely the preceding, the actual and the succeeding phonemes. Unfortunately the large amount of possible observations makes it impossible to train the acoustic model, so we need to cluster these CD states before training. We will describe state clustering in more detail later on in Chapter 4.

We should mention that Equation 1.16 describes a HMM with discrete observations, but speech is represented by a continuous signal. Of course, HMMs can omit continuous observations with the use of Gaussian Mixture Models (GMMs), the task of these GMMs being to provide a continuous estimation for $b_i(k)$. These models are called HMM/GMM and they were the standard technique in ASR for decades, until the appearance of DNNs. In the past few years the GMM part has been replaced by DNN, resulting in the new HMM/DNN hybrid model, which will be explained later.

Now we can focus on the three basic tasks that can be handled with an HMM, namely the evaluation, decoding and learning problems. Evaluation means that we wish to find the probability of an observation given the HMM parameters, an example of this in speech recognition being the task of isolated word recognition. The second problem (decoding) is essential in continuous speech recognition, which seeks to find the most likely sequence of hidden states (phonemes) given the HMM and and observation sequence. The Viterbi algorithm is a dynamic programming method that offers a simple solution to the decoding problem [7].

The Baum-Welch algorithm [7] is the standard method used to train a HMM. It

applies the forward-backward algorithm (described in Chapter 3) to find the maximum likelihood estimates of the parameters given a set of observed feature vectors and the known sequence of hidden states.

1.4.1 HMM/DNN model

After having outlined the HMM part of the hybrid model, we will now focus on the DNN part. It is clear that the acoustic model requires a component that can estimate the $b_i(k)$ values (the observation probability values). For decades GMMs were routinely used for this task, but we should add that shallow ANNs were also used by scientists but they failed to significantly outperform GMMs. This situation has changed with the appearance of DNNs, which have a superior performance in general compared to that of the GMMs.

The success of the DNNs could be credited to several factors. These are:

- DNNs are able to estimate the posterior probabilities of HMM states using any kind of input, even highly correlated ones, unlike GMMs.
- DNNs can be trained efficiently using a large amount of training data. Actually, it is an essential aspect of deep learning to use a lot of data.
- DNNs generalise better than GMMs. The explanation for this is the fact that the output of a DNN is sensitive to a lot of weights in the network, hence it can learn far more complex relationships between the inputs and the labels.

Of course, we should mention that it is also harder to train a DNN than it is to fit a GMM. DNNs have many meta-parameters that need to be fine-tuned and usually they have many more parameters than GMMs do. Fortunately, nowadays one can train DNNs on special hardware called the Graphical Processing Unit (GPU) to speed up the training procedure and quickly tune the meta-parameters.

When DNNs are trained as acoustic models, they attempt to estimate the probability values of the hidden states using observations as input. The ith output of the network at time t can be written as

$$dnn_{i,t} = P(q_i|o_t). \tag{1.17}$$

The problem is that the HMM requires the estimates of $P(o_t|q_i)$, so using the Bayes rule the outputs of the DNNs must be reformulated as

$$dnn_{i,t} = P(q_i|o_t) = \frac{P(o_t|q_i)P(q_i)}{P(o_t)}. \tag{1.18}$$

After reordering, we get

$$P(o_t|q_i) = P(o_t)\frac{P(q_i|o_t)}{P(q_i)} = P(o_t)\frac{dnn_{i,t}}{P(q_i)}, \tag{1.19}$$

where $P(q_i)$ is the prior probability of state i. Since the decoding process aims to find the most probable sequence of states and $P(o_t)$ is constant for each state, it can be ignored.

1.5 Language Model

The purpose of a language model (LM) is to assign probabilities to word sequences. In speech recognition it is used to convert the output of the acoustic model into a word sequence, and to achieve this it requires a lexicon (also called the pronunciation dictionary) that contains the pronunciations for all recognizable words. In our study, we utilised the simplest model called an N-gram. N-grams use the Markov assumption, meaning that the probability of a word depends only on the previous N word. Thus the N-gram model approximates the probability of the next word in a sentence as

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{1+n-N}^{n-1}). \tag{1.20}$$

Using a 2-gram (also called a bigram model), the probability of the whole sequence can be calculated by using the chain rule

$$P(w_1^n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1}). \tag{1.21}$$

Next, we need a way to estimate the N-gram probability values. Luckily, we can calculate the N-gram probabilities using a very simple method called Maximum Likelihood Estimation (MLE) [7]. The conditional probabilities can be calculated by getting the number of occurrences from a corpus and normalising them to a value between 0 and 1. In the case of a bigram model this means that we need to count all appearances of the words x and y, when they are in the correct order and divide this by the number of all word pairs, which start with x. Formally,

$$P(y|x) = \frac{C(xy)}{\sum_w C(xw)}. \tag{1.22}$$

Typically, however, the N-gram model probability values are not calculated directly from the frequency counts, as it gives 0 when confronted with any N-gram that is not present in the training corpus. To overcome this problem, some form of smoothing is necessary, diverting a portion of the total probability mass to unseen N-grams. Various

methods are used, ranging from the simple add-one smoothing (assign a count of 1 to unseen n-grams) to more sophisticated methods, like the Good-Turing discounting or the back-off models. Here we used the Katz back-off model [7], which simply reduces N if the N-gram was not seen enough times in the training data. The new estimate for $P_{Katz}(w_i|w_{i-N+1}\dots w_{i-1})$ is

$$\begin{cases} d_{w_{i-N+1}\dots w_i}\frac{C(w_{i-N+1}\dots w_{i-1}w_i)}{C(w_{i-N+1}\dots w_{i-1})}, & if C(w_{i-N+1}\dots w_i) > k \\ \alpha_{w_{i-N+1}\dots w_i-1}P_{Katz}(w_i|w_{i-N+2}\dots w_{i-1}) & otherwise, \end{cases} \tag{1.23}$$

where α is the back-off weight, k is a threshold and d is a scaling factor, which typically has a value of the amount of discounting found by the Good-Turing estimation.

1.6 The Szeged Broadcast News Corpus

The Szeged Hungarian Broadcast News Corpus was the dataset used in all chapter. It was recorded and transcribed at the Research Group on Artificial Intelligence, belonging to the Hungarian Academy of Sciences and the University of Szeged Institute of Informatics [8]. The corpus contains 115 news broadcasts which were recorded from 8 different television channels. These recordings were cut into short utterances, and the resulting segments were placed into one of the following categories:

- Clean speech: utterances in this category contain well-articulated, mostly planned speech, and have a minimal level of background noise. Most recordings in this category were originally filmed in a studio, and were spoken by professional newscasters.

- Noisy speech: speech in this category is still mostly planned, but it has a higher level of background noise. Recordings in this category are typically taken from on-site reporters speaking in a noisy environment.

The database contains approximately 28 hours of recordings, from which 22 hours were selected for the training set, 2 hours for the development set and 4 hours for the test set. The clean part of the corpus was also partitioned, 44 newscasts (altogether approximately 5.5 hours) were used for training, 9 newscasts (altogether approximately 1 hour) were used for development and validation, while the remaining 17 newscasts (altogether approximately 2 hours) were used for testing purposes. Both partitionings of the recordings were carried out in such a way that each set contained recordings from all television stations. All the recordings were orthographically typed, and the corresponding phonetic transcripts were created with a simple phonetic transcriber. The phonetic labels of the database consist of 52 categories. In this thesis, we conducted both phoneme and word recognition experiments on this corpus.

1.7 Summary by Chapters

The thesis is organised as follows:

- In **Chapter 2**, we compare the performance of four algorithms used to train DNNs. The first two algorithms are two-phase methods as they apply a pre-training step before fine-tuning the DNN, the first one being the original algorithm proposed by Hinton et al. [9]. The second algorithm created by Seide et al. [10] is called discriminative pre-training. As for the third option, we choose the Deep Rectifier Network (DRN), which differs greatly from the above two in the sense that here it is not the training algorithm that is modified, but the activation of the hidden neurons. The fourth training algorithm is a regularization method called Dropout[11], which simply turns off neurons temporarily during training. In our experiments, we first compared the results got by using these methods on the English TIMIT database and on a Hungarian audiobook corpus [12], but the main goal in this case is to obtain results for a large vocabulary Hungarian recognition task. For this purpose, we trained a recognition system on the 28-hour speech corpus of the Szeged Hungarian Broadcast News [8].

- In **Chapter 3**, we compare two sequence training approaches, namely the Connectionist Temporal Classification (CTC) and the Maximum Mutual Information (MMI) method. Our aim here is to find a purely DNN-based solution that could be used to train randomly initialised DNNs without force-aligned labels. Although CTC was originally proposed for the training of recurrent neural networks, here we show that it can also be employed to train rectifier networks as well. We will also show that with our modifications, MMI is also suitable for this task.

- In **Chapter 4**, we focus on creating a GMM-free HMM/DNN system. For this we have to solve two problems, namely the initial alignment of the frame-level state labels and the creation of context-dependent states. To create the initial alignments, we rely on the MMI-based method described in Chapter 3 and compare it with another solution, which iteratively trains and realigns the DNN. Recently, some new methods have been published which offer a way to create CD states using only HMM/DNNs. We will compare the performance of three of these new approaches with that of our own solution.

- In **Chapter 5**, we explore a possible way of handling the imbalance in the CD state distribution. This imbalance in the class distribution poses a significant

challenge to DNNs. A straightforward solution is to re-sample the training data, either by upsampling the rarer classes or by dowsampling the more common classes. Here, we experiment with the so-called probabilistic sampling method that applies downsampling and upsampling at the same time. We also propose a new method to re-estimate the class priors, to remedy the mismatch between the training and the test data distributions introduced by re-sampling. Our experimental results indicate that by applying the probabilistic re-sampling algorithm during the training and properly setting the priors, we can markedly improve the accuracy of CD DNNs.

- In **Chapter 6**, we provide a brief summary of the contributions outlined in the thesis and discuss possible directions for future research.

Chapter 2

A Comparison of Deep Neural Network Training Methods for Large Vocabulary Continuous Speech Recognition

In the past few years there has been a renewed interest in applying neural networks to speech recognition, thanks to the invention of Deep Neural Networks. As we already remarked in Chapter 1, DNNs differ from conventional ones in that they consist of several hidden layers. The application of a deep structure can provide significant improvements in speech recognition results compared to previously used techniques [13]. However, modifying the network architecture also requires modifications to the training algorithm, because the conventional backpropagation algorithm encounters difficulties when training many-layered feedforward networks [14]. As a solution, Hinton et al. presented a pre-training algorithm that works in an unsupervised fashion [9]. After this pre-training step, the backpropagation algorithm can find a much better local optimum of the parameters. The first applications of Deep Networks for speech recognition were performed on the TIMIT database [15], which is much smaller than the corpora routinely used for the training of industrial-scale speech recognizers. Hence, since their invention, a lot of effort has been devoted to scaling up DNNs so that they could be trained using much larger datasets and large vocabulary tasks [10, 16, 17]. The main problem is that Hinton's pre-training algorithm is quite intensive computation-wise, even when implemented on graphic processors. Several solutions have been proposed to alleviate or circumvent the computational burden of pre-training, but the search for the optimal training technique is still going on.

2.1 Introduction

In this chapter, we compare four different technologies used for the training of DNNs. The first one is the original pre-training algorithm of Hinton et al. [9]. It treats the network as a deep belief network built out of restricted Bolztmann machines, and optimizes an energy-based target function using the contrastive divergence (CD) algorithm. After pre-training, the network has to be fine-tuned using some conventional training method like backpropagation.

The second algorithm is called "discriminative pre-training" by Seide et al. [10]. This method constructs a deep network by adding one layer at a time, and trains these sub-networks after the addition of each layer. Both the pre-training of the partial nets and the final training of the full network can be performed using backpropagation, so no special training algorithm is required.

As for the third method, it is different from the two above in the sense that in this case it is not the training algorithm that is modified, but the neurons themselves. Namely, the usual sigmoid activation function is replaced with the rectifier function $\max(0, x)$. These kinds of neural units were proposed by Glorot et al., and were successfully applied to image recognition and NLP tasks [14]. Rectified linear units were also found to improve restricted Boltzmann machines [18]. It has been shown recently that a deep rectifier network can attain the same phone recognition performance as that for the pre-trained nets of Mohamed et al. [15], but without the need for any pre-training [19].

The fourth method called Dropout was introduced a few years ago by Hinton et al. [11]. Unlike the previous methods, it is a regularisation technique, meaning that it is just a refinement of the training algorithm. The term "dropout" refers to the fact that during training, neurons in the network are randomly dropped out. Dropping neurons from the network can be achieved by simply zeroing out the activation of the chosen units. The main advantage of this method is that it helps neural networks to generalise better, thus it improves their performance especially in case of noisy input. Another advantage of Dropout is the fact that it can be combined with any training algorithms, since it is a regularisation method, so we will use it not just with standard sigmoid networks but with rectifier ones too.

In our experiments, we first compared the performance of the four methods on the English TIMIT database and on a Hungarian audiobook corpus [12], but the main goal of this study is to obtain results for a large vocabulary Hungarian recognition task. For this purpose, we trained a recognition system on a 28-hour speech corpus of Hungarian Broadcast News, presented in Section 1.6.

The recogniser is a hybrid HMM/DNN system [20] that estimates the state-level

posterior probability values from the neural net, while the decoder is the HDecode program, which is a part of the HTK package [1]. As Hungarian is an agglutinative language, our system runs with a relatively large dictionary of almost five hundred thousand word forms.

2.2 Training Methods for Deep Neural Networks

In the last few years there has been a renewed interest in applying neural networks, especially Deep Neural Networks, to various tasks. To properly train these multi-layered feedforward networks, the training algorithm requires modifications as the conventional backpropagation algorithm encounters difficulties ("vanishing gradient" and "explaining away" effects).

In this case the "vanishing gradient" effect means that the error might converge to zero as it gets propagated back through the hidden layers [21]. The reason for this is simple: each weight in the neural network receives an update proportional to the gradient of the error function with respect to the current weight during each iteration of training. The most commonly used activation functions like the sigmoid function have gradients in the range (0, 1), and backpropagation computes gradients using the chain rule. This means that if we multiply these small values to compute gradients of a "deeper" layer in a neural network, the gradient (error signal) decreases exponentially. This could cause some hidden layers, in particular those that are closest to the input layer, to have gradients close to zero and as a consequence the whole network may fail to learn.

At the same time, in fully connected deep networks, the "explaining away" effects make inference extremely difficult in practice [9]. Explaining away is a well-known phenomenon in Bayesian networks which has a V shaped structure (two input and one output node). For DNNs the assumption that hidden neurons are independent becomes invalid as they could become anticorrelated. For example, if an output node can be activated by two equally rare and independent events (hidden neurons) with an even smaller chance of occurring simultaneously, then the activation of one of the hidden nodes negates (explains away) the occurrence of the other in such a way that a negative correlation is obtained between the two neurons. This makes the training of DNNs using the standard backpropagation difficult as it could converge to a suboptimal model. Several solutions have been proposed to overcome these problems, and here we compare four of them empirically.

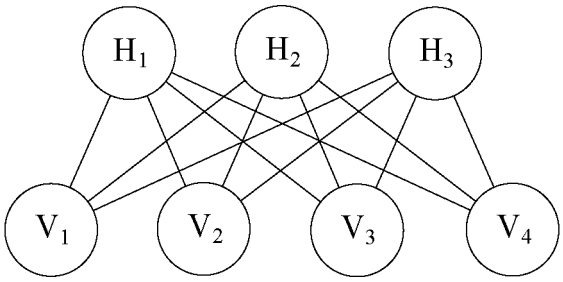

Figure 2.1: RBM with 3 hidden and 4 visible neurons.

2.2.1 DBN Pre-Training

This efficient unsupervised algorithm, first described in [9], can be used to learn the connection weights of a Deep Belief Network (DBN) consisting of several layers of Restricted Boltzmann Machines (RBMs). The RBMs are a variant of Boltzmann Machines, with the restriction that their neurons must form a bipartite graph. They have an input layer (called the visible layer), representing the features of the given task, and a hidden layer which has to learn some representation of the input. The restriction compared to the simple Boltzmann Machines is that each connection in an RBM must be between a visible unit and a hidden unit, thus forming a bipartite graph. Figure 2.1 provides a graphical depiction of an RBM. These RBMs can be trained using the Contrastive Divergence (CD) algorithm proposed by Hinton et al. in [9]. The main idea behind CD is that the RBM assigns the following energy value to each configuration of visible and hidden state vectors, denoted by v and h, respectively:

$$E(v,h;\Theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}v_i h_j - \sum_{i=1}^{V} a_i v_i - \sum_{j=1}^{H} b_j h_j \tag{2.1}$$

where the weights of the connection between a visible and hidden neuron are stored in the matrix w, while b_i and a_i are the hidden and visible biases respectively. A key element of the CD algorithm is Gibbs sampling, which is a Markov chain Monte Carlo algorithm. For RBMs one can sample the visible and hidden units using block Gibbs sampling, as the layers are conditionally independent. A sampling step is performed as follows:

$$h^n = f(Wv^{n-1} + b) \tag{2.2}$$

$$v^n = f(W'h^{n-1} + a) \tag{2.3}$$

In theory, each learning epoch would require Gibbs sampling to be repeated until full convergence is achieved. It is clear that in practice one cannot run the sampling chain up to convergence, as it would be computationally expensive and extremely time-consuming. As a solution, Hinton proposed the one-step contrastive divergence (CD-1)

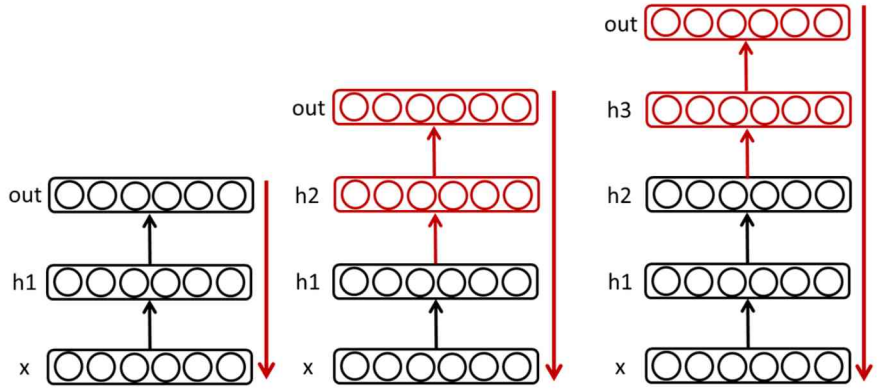

Figure 2.2: The DPT training process is shown, the new parts of the network being shown in red; after a new layer is added we train the whole network not just the newly added parts.

update rule for the visible-hidden weights:

$$\Delta w_{ij} \propto \langle v_i^0 h_j^0 \rangle - \langle v_i^1 h_j^1 \rangle. \tag{2.4}$$

Note that CD-1 does not wait for the Markov chain to converge, it runs a Gibbs sampler initialised on the data for one full step.

Although RBMs with the energy function of Equation (2.1) are applicable for binary data, in speech recognition the acoustic input is typically represented by real-valued feature vectors. For real-valued input vectors, the Gaussian-Bernoulli restricted Boltzmann machine (GRBM) can be used, and it requires making only a minor modification of Equation (2.1). The GRBM energy function is given by:

$$E(v, h|\Theta) = \sum_{i=1}^{V} \frac{(v_i - a_i)^2}{2} - \sum_{i=1}^{V} \sum_{j=1}^{H} w_{ij} v_i h_j - \sum_{j=1}^{H} b_j h_j \tag{2.5}$$

Hinton et al. showed that the weights resulting from the unsupervised pre-training algorithm can be used to initialise the weights of a deep, but otherwise standard, feed-forward neural network. After this initialisation step, a softmax output layer needs to be added to the network, then we simply use the backpropagation algorithm to fine-tune the network weights with respect to a supervised criterion.

2.2.2 Discriminative Pre-Training

'Discriminative pre-training' (DPT) was proposed in [10] as an alternative to DBN pre-training. It is a simple algorithm where first we train a network with one hidden layer to full convergence using backpropagation; then we replace the softmax layer by another randomly initialized hidden layer and a new softmax layer on top, and we train

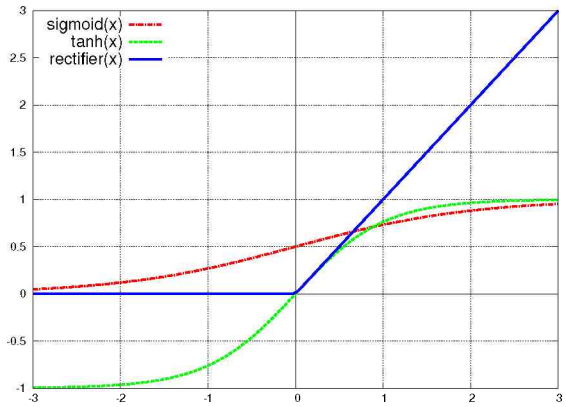

Figure 2.3: The rectifier activation function and other commonly used activation functions.

the network again; this process is repeated until we reach the desired number of hidden layers. Figure 2.2 illustrates the general procedure.

This training method is very similar to the greedy layer-wise training algorithm of Bengio et al. [22], but differs in that Bengio only updates the newly added hidden layers and the output layer. Seide et al. found that this method gives the best results if one performs only a few iterations of backpropagation in the pre-training phase (instead of training to full convergence) with an unusually large learn rate. In their article, they concluded that this simple training strategy performs just as well as the much more complicated DBN pre-training method described above [10].

2.2.3 Deep Rectifier Networks

Rectified neural units were recently applied with success in standard neural networks, and they were also found to improve the performance of DNNs on tasks like image recognition and speech recognition [14, 19]. These rectified neurons apply the rectifier function ($max(0, x)$) as the activation function instead of the sigmoid or hyperbolic tangent activation. The main advantage of Deep Rectifier Networks (DRNs) is that thanks to their properties, they can be trained with the standard backpropagation algorithm, without any time-consuming pre-training. As Figure 2.3 shows, the rectifier function is one-sided, hence it does not enforce a sign symmetry or antisymmetry. Here, we will examine the two key properties of this one-sided function, namely its hard saturation at 0 and its linear behaviour for positive input.

The hard saturation for negative input means that only a subset of neurons will be active in each hidden layer. For example, when we initialize the weights uniformly, around half of the hidden units output are real zeros. This allows rectified neurons to achieve truly sparse representations of the data. In theory, this hard saturation at 0 could harm optimization by blocking gradient back-propagation. Fortunately,

experimental results do not support this opinion, suggesting that hard zeros can actually help supervised training. These results show that the hard non-linearities do no harm as long as the gradient can propagate along some path [14].

For a given input, the computation is linear on the subset of active neurons. Once the active neurons have been selected, the output is a linear combination of their input. This is why we can treat the model as an exponential number of linear models that share parameters. Based on this linearity, there is no vanishing gradient effect [14], and the gradient can propagate via the active neurons. Another advantage of this linear behaviour is the smaller computational cost: there is no need to compute the exponential function during the activation, and the sparsity of neuron activity can also be exploited. A disadvantage of the linearity property is the "exploding gradient" effect, when the gradients can grow without limit. To prevent this, one could apply a regularisation technique called weight normalisation [23]. Weight normalisation attempts to keep the L1- or L2-norm of the weight matrices the same as it was after initialization by scaling the weights during training. What makes this possible is that for a given input the subset of active neurons behaves linearly, so a scaling of the weights is equivalent to a scaling of the activations.

2.2.4 Dropout

Dropout differs from the previous methods in that it is a regularisation technique. The name refers to the fact that this method drops out neurons during training. In practice, the neuron dropout can be performed by applying a random binary mask. By dropping a neuron out and temporarily removing it from the network, along with all its incoming and outgoing connections, we basically create a different model for each training example. The goal of dropout is to prevent overfitting by combining exponentially many different neural network architectures efficiently. The dropout mask could be generated randomly with a λ parameter, and this controls the percentage of the dropped neurons.

During validation or testing, it is not feasible to average the predictions from exponentially many models. However, a very simple approximate averaging method works well in practice. The idea is to use the original neural net structure without dropout, but the weights of this network need to be a scaled-down version of the trained weights. The scaling-down could be carried out in the following way: if a neuron drops out with probability p during training, then the weights of that unit are multiplied by $1-p$ before testing. The main advantage of this regularisation technique besides its simplicity is that it can be readily used with other training algorithms and it can provide significant improvements.

2.3 Experimental Setup

Here, we report the results of applying the ANN-based recognisers on three databases. The first one is the classic TIMIT database of English sentences, while the second is a corpus of a Hungarian audiobook. The third database is called Szeged Broadcast News. On TIMIT quite a lot of phone recognition results are available, so it is good for comparative purposes. However, TIMIT is quite small and usually only phone-level results are reported on it. The training set consisted of the standard 3696 'si' and 'sx' sentences, while testing was performed on the core test set (192 sentences). A random 10% of the training set was held out for validation purposes, and this block of data will be referred to as the 'development set'. The scores reported are phone recognition error rates using a phone bigram language model.

As the second database we chose an audiobook for which the original novel is so old that its text is no longer copyrighted. Our choice fell on the short story collection by Gyula Krúdy entitled 'Sinbad's Voyages', presented by the actor Sándor Gáspár. The total duration of the corpus was 212 minutes and it was was carefully annotated, and the differences between the original text and the sound material were corrected [12]. Each file in the corpus was segmented further into roughly two-minute long parts, and for training and test purposes the recordings were divided into two parts. From the ten short stories, eight were used for training (186 minutes) and two for testing (26 minutes). As the training data was limited we only performed phoneme level recognition with a phone bigram. One could say that this task is very different from a real-life recognition task, as there was minimal noise and the training and testing set are not speaker independent, actually the entire database is spoken by one person. Due to these facts one could say that this task is speech recognition under optimal conditions, and the results could provide an empirical glass ceiling for other tasks.

Lastly, in our tests on the Szeged Broadcast News corpus we sought to measure the large vocabulary recognition performance of the methods applied. The language model was created from texts taken from the Origo news portal (www.origo.hu), from a corpus of about 50 million words. Hungarian is an agglutinative language with a lot of word forms, hence we limited the size of the recognition dictionary to 486982 words by keeping only those words that occurred at least twice in the corpus. The pronunciations of these words were obtained from the 'Hungarian Pronunciation Dictionary' [24]. Based on the Origo corpus, a trigram language model was built using the language modelling tools of HTK [1].

As for the acoustic features, we applied the standard 39 MFCC coefficients, extracted from 25 ms frames with 10 ms frame skips. We used MFCC coefficients (including the energy), along with the corresponding Δ and $\Delta\Delta$ values. In each case,

the neural network was trained on 15 neighbouring frames, so the number of inputs to the acoustic model was 585.

Neural networks require a frame-level labeling of the training data. For this purpose, we first trained a standard hidden Markov model (HMM) speech recogniser, again using the HTK toolkit. For the TIMIT dataset, monophone 3-state models were created, which resulted in 183 states. For the broadcast news dataset, triphone models were constructed, consisting of 2348 tied triphone states in total. The HMM states were then aligned to the training data using forced alignment. These labels served as training targets for the neural nets.

For the recognition process, we applied the decoders of the HTK package. We used HVite for the phone recognition experiments on TIMIT and the Hungarian audiobook, while the HDecode routine was applied for the large vocabulary recognition tests on the broadcast news task. In both cases, the acoustic modeling module of HTK required a slight modification so that it could use the posterior probability values produced by the neural nets. For the TIMIT and the audiobook dataset, the language model weight and the insertion penalty factor were set to 1.0 and 0.0, respectively. With the broadcast news dataset, these meta-parameters were tuned on the development set. Lastly, for a fairness of comparison, the pruning beam width was set to the same value for each network.

2.3.1 Training Parameters for the Neural Networks

As is standard in machine learning, all hyperparameters of the training methods were fine-tuned on the development set. In the case of the DBN-based pre-training method, we applied stochastic gradient descent (i.e. backpropagation) training with a mini-batch size of 128. For Gaussian-binary RBMs, we ran 50 epochs with a fixed learning rate of 0.002, while for binary-binary RBMs we used 30 epochs with a learning rate of 0.02. Then, to fine-tune the pre-trained nets, again backpropagation was applied with the same mini-batch size as that used for pre-training. The initial learn rate was set to 0.01, and it was halved after each epoch when the error on the development set increased.

During both the pre-training and fine-tuning phases, the learning was accelerated by using a momentum of 0.9. Momentum is a well-known regularisation technique for accelerating gradient descent [25], which accumulates a velocity vector of gradient updates across previous iterations. The momentum update rule in our implementation is given by:

$$v_{t+1} = m * \Delta W_{t+1} + (1 - m)v_t \tag{2.6}$$

$$W_{t+1} = W_t + \epsilon v_{t+1}, \tag{2.7}$$

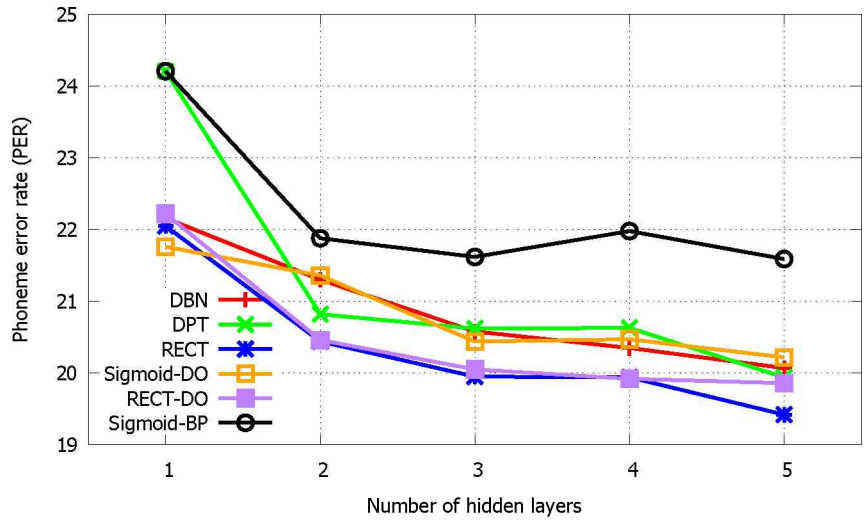

Figure 2.4: Phone error rates on the TIMIT dev set as a function of the number of hidden layers.

where v is the matrix in which the momentum of the gradient is stored, m is the momentum coefficient parameter and ϵ is the learning rate.

Turning to the discriminative pre-training method, the initial learn rate was set to 0.01, and it was halved after each epoch when the error on the development set increased. The learn rate was restored to its initial value of 0.01 after the addition of each layer. Furthermore, we found that using 5 epochs of backpropagation after the introduction of each layer gave the best results. For both the pre-training and fine-tuning phases, we used a batch size of 128 and momentum of 0.8 (except for the first epoch). The initial learn rate for the fine-tuning of the full network was again set to 0.01.

The training of deep rectifier nets did not require any pre-training at all. The training of the network was performed using backpropagation with an initial learn rate of 0.001 and a batch size of 128. The dropout method was applied with standard sigmoid networks and with rectified ones as well. We did not combine dropout with the pre-training methods since their training already required a lot of time and dropout would have increased it even further. In the case of the sigmoid network 20% of the neurons were dropped randomly, while the rectifier networks required only 10% dropout to achieve the best results.

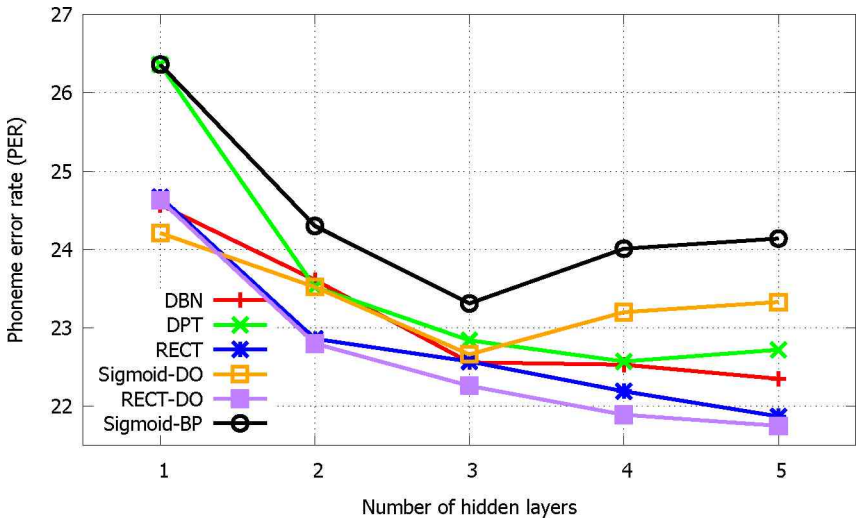

Figure 2.5: Phone error rates on the TIMIT core test set as a function of the number of hidden layers.

2.4 Results

2.4.1 TIMIT

Figures 2.4 and 2.5 show the phone recognition error rates obtained on the TIMIT dev and core test set, respectively, with a varying number of hidden layers, each hidden layer containing 2048 neurons. As can be seen, the deep learning methods performed very similarly, up to 4 hidden layers and in the case of five hidden layers, the rectifier nets performed slightly better than the others. Each deep learning method outperformed the standard backpropagation method (BP) once the network had at least two hidden layers. The best results (21.87% and 21.75%) were obtained with rectifier networks, which had five hidden layers. The dropout regularisation improved the deep sigmoid networks significantly, but it gave only a minor improvement in the case of the rectifier networks. Besides the fact that dropout failed to improve the results significantly, the new hyperparameter, namely the dropout rate noticeably increased the time needed to tune the hyperparameters. Using similar features, training labels and network sizes, Mohamed at al. reported a 22.3% error rate with DBN pre-training [15], while Tóth reported a 21.8% figure with rectifier nets [19]. As our scores fall in the same range, the results also demonstrate the soundness of our methodology.

2.4.2 Hungarian Audiobook

Figure 2.6 shows the results for the Audiobook corpus. The standard method achieves its best performance with three hidden layers, only slightly better than the one achieved

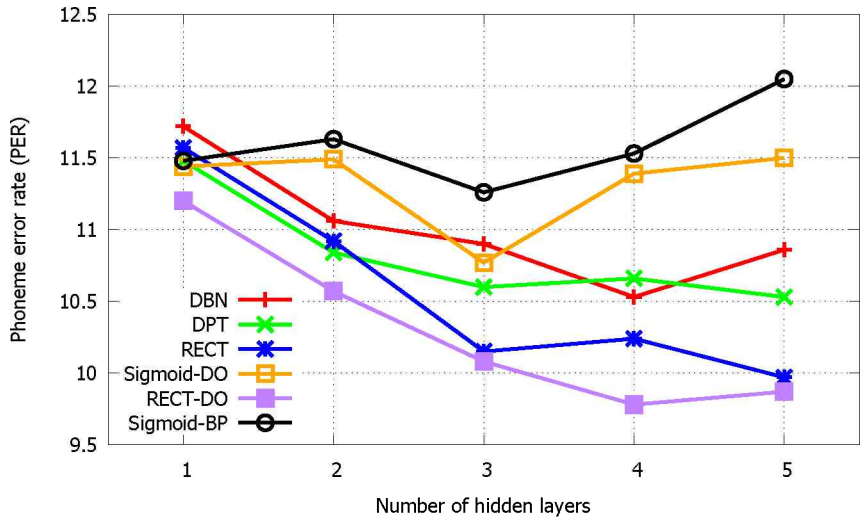

Figure 2.6: Phone error rates on the Audiobook test set as a function of the number of hidden layers.

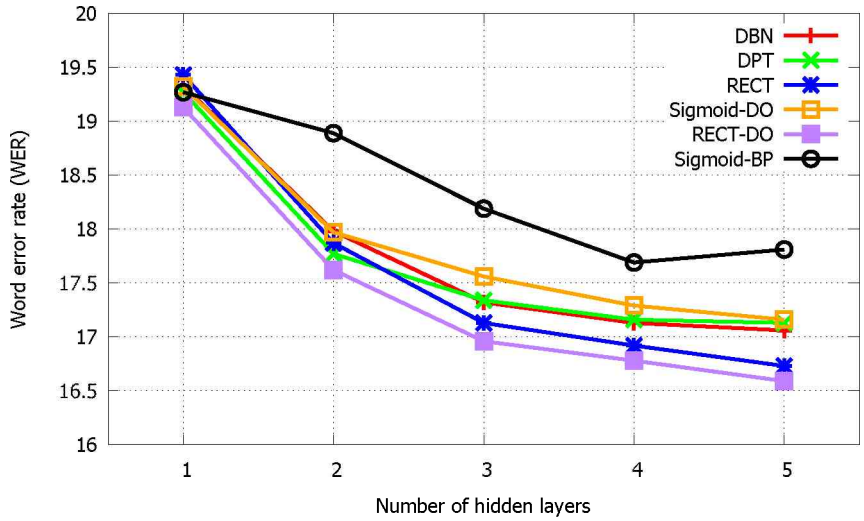

Figure 2.7: Word error rates for the broadcast news corpus as a function of the number of hidden layers.

with only one hidden layer. It is also interesting that using more than three hidden layers leads to an increasing PER. The Dropout regularisation increased the accuracy of the sigmoid networks, but they still followed the same trend. The two pre-training methods performed in a quite similar way and just like on the TIMIT dataset the rectifier networks proved to be the best models performance-wise.

2.4.3 Hungarian Broadcast News

Figure 2.7 shows the word error rates got for the large vocabulary broadcast news recognition task. Similar to the TIMIT tests, 2048 neurons were used for each hidden layer, with a varying number of hidden layers. The trends of the recognition results are quite similar to those for the TIMIT database. The error rates seem to saturate at 4-5 hidden layers, and the curves for the methods run parallel and have only slightly different values. The lowest error rate is attained with the five-layer rectifier network, both on the development set and the test set.

Although their recognition accuracy scores are quite similar, there is another factor we need to consider, namely the training times. These methods differ significantly in the training times required,and Table 2.1 shows the training times we measured using a NVIDIA GTX-560 TI graphics card. These values tell us how long it took to train one DNN, after we found the optimal hyperparameters. We should add that in the case of the DBN pre-training method and the dropout method, we spent far more time on properly tuning their hyperparameter values than in the case of the others. Evidently, the DBN pre-training algorithm also has the largest computational requirements. This algorithm has no clearly defined stopping criterion, and various authors run it with a widely differing number of iterations. The iteration count we applied here (50 for Gaussian RBMs and 30 for binary RBMs) is an average value, and follows the experiments carried out by Seide et al. [10]. Mohamed applies many more iterations [15], while Jaitly et al. use far fewer iterations [16]. Discriminative pre-training and dropout regularisation are also much faster than the DBN-based method, but they are still slower than the training of rectifier nets.

Training method	Pre-training time	Fine-tuning training time
Sigmoid + BP	0 hours	4.5 hours
Sigmoid + Dropout	0 hours	5.5 hours
DBN pre-training	1 hours	4 hours
Discr. pre-training	2.5 hours	3 hours
Rectifier network	0 hours	4 hours
Rectifier network + Dropout	0 hours	4.5 hours

Table 2.1: The training times required by the various methods for one network with five hidden layer.

Lastly, although the main goal here was to compare the four deep neural network algorithms, let us now compare the large vocabulary recognition scores with those of a conventional HMM. The same HMM model that was used to generate the training labels attained a word error rate of 20.07% (with maximum likelihood training), while

the best DNN system achieved a WER of 16.59%, meaning that by replacing the GMM with a rectifier network we got a 17% relative error rate reduction. Tuning the parameters so that the two systems had a similar real-time factor was also out of the question, as the hybrid model was implemented on a GPU, while the HMM used a normal CPU.

2.5 Summary

It is perhaps no exaggeration to say that deep neural nets have led to a breakthrough is speech recognition. However, they are computationally intensive, and the quest for the optimal network architecture and training method is still continuing. In this chapter I presented and compared two training methods, a new type of activation function and a regularisation technique for DNNs, and evaluated them on two smaller phoneme recognition tasks and on a Hungarian large vocabulary recognition task. To the best of my knowledge, I was the first to apply HMM/DNN systems to Hungarian speech recognition. These deep learning algorithms yielded pretty similar recognition performances on a medium-sized corpus, yet rectifier networks produced better results and their training was considerably faster. Based on these facts, in my later experiments deep rectifier networks became my preferred choice.

In this chapter, the author regards the following as his main contributions:

- Performing an experimental comparison of four deep learning methods.

- First results for Hungarian speech recognition using HMM/DNN hybrids.

And the results presented in this chapter were published in [26].

Chapter 3

Sequence Training Methods for Deep Rectifier Neural Networks

In our pursuit of a strictly DNN-based ASR solution, we first turned our attention to the task of flat start training. Most of the current DNN technologies require frame-aligned labels, which are usually created by first training an HMM/GMM. Obviously, it would be far more efficient to just use DNN-based recognisers without the need to create an HMM/GMM to do the same task. Although flat start training via iteratively realigning and retraining the DNN using a frame-level error function is viable, it is quite cumbersome. In this chapter, we compare two sequence training approaches, namely the Connectionist Temporal Classification (CTC) and the Maximum Mutual Information (MMI) method. Our aim here is to find a purely DNN based flat start solution, which could be used to train randomly initialised DNNs without using force-aligned labels.

The first method (CTC) that we examined was originally proposed for the training of recurrent neural networks, but here we will show that it can also be used to train more conventional feed-forward networks as well. As our second choice, from the wide variety of sequence discriminative training methods we opted for MMI training [27]. While this is routinely applied only in the final phase of model training, here we will show that with proper modifications it is also suitable for obtaining the alignments of context-independent models.

In the experimental part, we evaluate the two methods on several phone recognition tasks. For each database we tested, we found that the sequence training methods give better results that those obtained with force-aligned training labels produced by an HMM/GMM system. These results suggests that flat start training is possible without the use of GMMs.

3.1 Problem description and literature overview

For three decades now, Hidden Markov Models (HMMs) have been the dominant technology in speech recognition. Their success is due to the fact that they handle local (frame-level) likelihood estimation and combine these local estimates to get a joint global (utterance-level) score, in a unified mathematical framework. Recently, however, it was shown that DNN-based solutions can significantly outperform standard HMMs [28]. As described in Chapter 1, this new technology replaces the Gaussian mixtures of the HMM by a DNN, while the utterance-level decoding is still performed by the HMM. The DNN component of these hybrid models is usually trained only at the frame level. That is, we generate frame-level training targets for the network, and during training we optimise some frame-level training criteria. However this frame-by-frame training has several drawbacks. Firstly, we have to have frame-level labels to be able to start the training. For very old and small databases (like the TIMIT dataset used here), a manual phonetic segmentation is available. However, for more recent corpora which might be hundreds of hours long, manual segmentation is clearly out of the question. Hence, the usual solution for obtaining frame-level labels is to train a standard HMM/GMM system, and then use it in forced alignment mode. This means that, based on the current technology, the training of a DNN-based recogniser should always be preceded by the training of a standard HMM model. This clearly makes the creation of a DNN-based system much more tedious and time-consuming, and although quite recently there have been some attempts at having the standalone training of DNN systems, these technologies are still far from complete [29].

Secondly, besides the cost of creating forced aligned labels, the frame-level training of a neural network has a deeper, more theoretical limitation. During this training, we minimise the frame-level error cost, such as the frame-level cross-entropy (CE) between the network output and the training targets. These training targets are hard-labeled, which means that we expect the network to give an output of 1 for the correct class and 0 for the remaining ones. This is not necessarily optimal regarding the decoding process, which combines the frame-level scores. A more sophisticated method that derives "soft" training targets from the sentence-level scores can be expected to result in a better performance.

Graves et al. proposed a method that provides a solution to both the above-mentioned problems, and called it the Connectionist Temporal Classification (CTC) method for Recurrent Neural Networks (RNNs) [30]. This method requires just the transcription of the utterance, without any further label alignment information. Nevertheless, their architecture differs fundamentally from the standard HMM/ANN model: owing to the use of recurrent neural network classifiers, they apply the training method

called *backpropagation through time* [31], making the training process much more time-consuming and quite complex. The number of model parameters is also quite high. Furthermore, as frames have to be processed in a strictly increasing order, decoding is much harder to parallelise. When using bidirectional recurrent networks (which are required to achieve the best performance with this approach [32]), we have to wait for the end of the utterance before we can start evaluating the network, making real-time speech processing impossible. Lastly, instead of using a standard language model like a phoneme n-gram, they use a special technique called a prediction network, which is also based on an RNN. Thus, their approach is quite involved and quite different from the usual HMM/ANN model.

In this chapter we show that the CTC training scheme is not an inseparable part of the RNN-based architecture, and with a few small modifications it can also be applied to the training of HMM/ANN models. Here, we use it to train standard feed-forward deep neural nets on a phone recognition task over three different databases.

Altough CTC is a viable option for flat start training [33], it has a serious drawback, namely that it cannot be used to generate accurate forced alignments of the phone labels. To overcome this problem, we also experimented with the MMI training method. Within the framework of HMM/GMM systems, several sequence-discriminative training methods have been developed, and these have now been adapted to HMM/DNN hybrids as well [27, 34]. However, most authors apply sequence-discriminative criteria only in the final phase of training, for the refinement of the DNN model. That is, the first step is always CE-based training, either to initialise the DNN (e.g. [35, 36, 37]) or just to provide frame-level state labels (e.g. [27, 34, 38, 39, 40]). In contrast with the previous authors, here we propose a training procedure that applies sequence-discriminative training in the flat start training phase. This requires several small modifications compared to the standard usage of sequence-discriminative training, which will be elaborated on later.

3.2 Flat start training of HMM/GMMs

The flat start training first initialises a so-called flat model, which does this by estimating a uniform GMM from all the training data and then applies it for all initial distributions. This also implies that during the first cycle of training, each training utterance will be uniformly segmented.

After the flat initialisation, the Baum-Welch algorithm [7] could be used to train the HMM/GMM acoustic model, which makes use of the forward-backward algorithm described later in Section 3.3.1. The basic assumption of the initialisation is that a sufficient number of the phone models align or overlap with the actual position of

that phone, so that during the second and subsequent iterations, the models align as intended.

3.3 Connectionist Temporal Classification

Following the research work of Graves et al. [30], first we will outline the Connectionist Temporal Classification training scheme. Similar to standard frame-level backpropagation training, it is an iterative method, where we sweep through the whole audio training data set several times. A special feature of this training method is that we process one whole utterance at a time instead of using just fixed-sized batches of it; furthermore, we only need the correct transcription of the utterance, and time-aligned labels are not required.

The CTC training method is built on the dynamic search method called forward-backward search [41], which is a standard part of HMM training. The forward-backward algorithm not only gives the optimal path, but at the same time we also get the probability of going through the given phoneme of the transcription for all the frames of the utterance. Using the forward-backward algorithm, we can calculate a probability distribution over the possible phonemes, for each frame; then these values can be used as target values when training the acoustic classifier.

3.3.1 The Forward-Backward Algorithm

Let us begin with the formal description of the forward-backward algorithm. First, let us take the utterance with length T, and let its correct transcription be $z = z_1 z_2 \dots z_n$. We will also use the output vectors y^t of the neural network trained in the previous iteration. In the first iteration, due to the random initial DNN weights, these will be practically random values. The forward variable ($\alpha(t,u)$) can be defined as the summed probability of outputting the u-long prefix of z up to the time index $t \leq T$. The initial conditions state that the correct sequence starts with the first label in z:

$$\alpha(1,u) = \begin{cases} y^1_{z_1} & \text{if } u = 1, \\ 0 & \text{if } u \geq 2. \end{cases} \tag{3.1}$$

Thereafter the forward variables at time t can be calculated recursively from those at time $t-1$; and we can remain in state z_{u-1}, or move on to the next one (z_u). Thus,

$$\alpha(t,u) = \begin{cases} y^t_{z_u}\alpha(t-1,u) & \text{if } u = 1, \\ y^t_{z_u}\big(\alpha(t-1,u) + \alpha(t-1,u-1)\big) & \text{otherwise.} \end{cases} \tag{3.2}$$

In the backward phase we calculate the backward variables $\beta(u,t)$, which represent the probability of producing the suffix of z having length $n-u$ starting from the frame

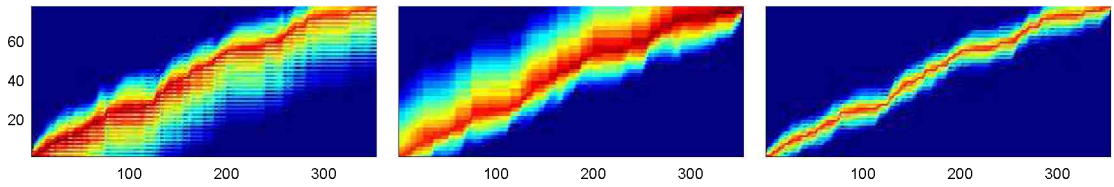

Figure 3.1: The α (left), β (middle) and $\alpha\beta$ (right) values for a given utterance. The horizontal axis corresponds to the frames of the utterance, while the vertical axis represents the phonemes.

$t+1$. The backward variables can be calculated recursively using the following rules:

$$\beta(T,u) = \begin{cases} 1 & \text{if } u = n, \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

and for each $t < T$

$$\beta(t,u) = \begin{cases} y^t_{z_u}\beta(t+1,u) & \text{if } u = n, \\ y^t_{z_u}\big(\beta(t+1,u) + \beta(t+1,u+1)\big) & \text{otherwise.} \end{cases} \tag{3.4}$$

Figure 3.1 depicts the forward variables, the backward variables and their product for a short utterance.

3.3.2 Using the $\alpha\beta$ values for ANN training

The $\alpha(t,u)\beta(t,u)$ product values express the overall probability of two factors, summed along *all paths*: the first is that we recognise the correct sequence of phonemes, and the second is that at frame t the system omits the uth phoneme of z. For neural network training, however, we would need a distribution over the phoneme set for frame t. It is not hard to see that such a distribution over the phonemes of z can be obtained by normalising the $\alpha(t,u)\beta(t,u)$ products so that they sum up to one (by which step we eliminate the probability of recognising the correct sequence of phonemes). Then, to normalise this distribution to one over the *whole set of phonemes*, we need to sum up the scores belonging to the multiple occurrences of the same phonemes in z. That is, the regression targets for any frame t and phoneme ph can be defined by the formula

$$\frac{\sum\limits_{i:z_i=ph} \alpha(t,i)\beta(t,i)}{\sum\limits_{i=1}^{n} \alpha(t,i)\beta(t,i)}. \tag{3.5}$$

We can use these values as training targets instead of the standard binary zero-or-one targets with any gradient-based non-linear optimisation algorithm. In our experiments, we applied the backpropagation algorithm to train the networks.

3.3.3 Garbage Label

Although the above training method may work well for the original phoneme set, Graves et al. introduced a new label (which we will denote by $\mathcal{X}$), by which the neural network may choose not to omit any phoneme. This label can be inserted between any two phonemes, but of course it can also be skipped. They called this label "blank", but we consider the term "garbage" more reasonable as frames belonging to this class are thrown away during decoding.

To interpret the role of this label, let us consider a standard tri-state model. This divides each phone into three parts. The middle state corresponds to the steady-state part of the given phone, whereas the beginning and end states represent those parts of the phone that are affected by coarticulation with the preceding and the subsequent phones, respectively. By introducing the label $\mathcal{X}$, we allow the system to concentrate on the recognition of the cleanly pronounced middle part of a phone, and it can map the coarticulated parts to the symbol $\mathcal{X}$. Therefore, we find it more logical to use the term *garbage label* instead of *blank*, as the latter would suggest that the label $\mathcal{X}$ covers silences, but in fact this label more likely corresponds to the coarticulated parts of phones.

Formally, introducing this label means that instead of the phoneme sequence z we will use the sequence $z' = \mathcal{X}z_1\mathcal{X}z_2\mathcal{X}\ldots\mathcal{X}z_n\mathcal{X}$. The forward-backward algorithm also has to be modified slightly. Namely, the initial α values are set to

$$\alpha(1,u) = \begin{cases} y^1_{z'_1} & \text{if } u = 1 \text{ or } u = 2, \\ 0 & \text{if } u \geq 3, \end{cases} \tag{3.6}$$

while for the latter labels we allow skipping the $\mathcal{X}$ states:

$$\alpha(t,u) = \begin{cases} y^t_{z'_u}\,\alpha(t-1,u) & \text{if } u = 1, \\ y^t_{z'_u}\big(\alpha(t-1,u) + \alpha(t-1,u-1)\big) & \text{if } z'_u = \mathcal{X}, \\ y^t_{z'_u}\big(\alpha(t-1,u) + \alpha(t-1,u-1) + \alpha(t-1,u-2)\big) & \text{otherwise.} \end{cases} \tag{3.7}$$

The calculation of the $\beta(t,u)$ values is performed in a similar way.

CTC is a process with a positive feedback: phonemes with generally high y values will have higher $\alpha\beta$ values, resulting in higher target target values, and during the iterations they typically tend to suppress all other phonemes. When using the label $\mathcal{X}$, usually this label dominates during training, and the outputs of a trained CTC network tend to form a series of spikes, which are separated by the garbage class. Figure 3.2 shows the outputs of a trained network. As can be seen, the garbage label permits the classifier to choose that a given frame does not belong to any of the original phoneme set, similarly to the anti-phoneme model of segment-based speech recognition [42]. It

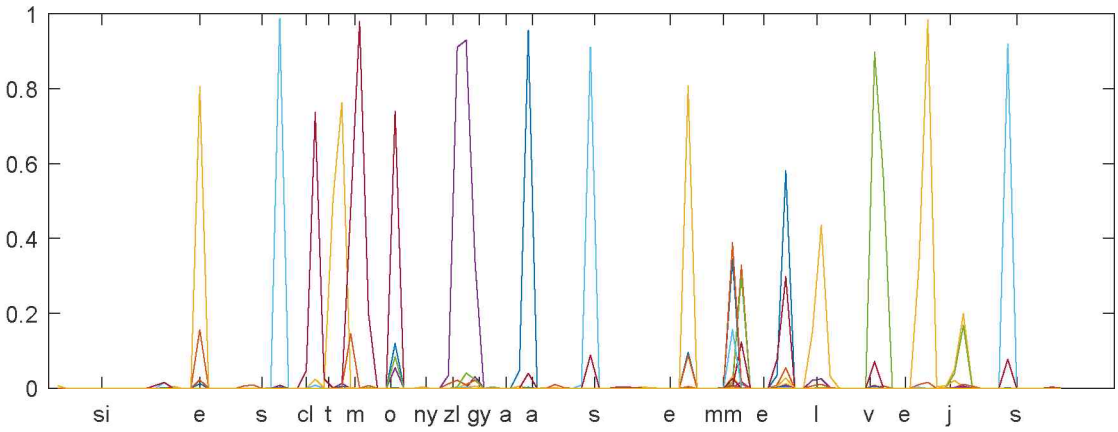

Figure 3.2: Non-garbage outputs generated by a CTC network for the utterance "Ezt mondja a Semmelweis". The network predicts the sequence of phones as a series of spikes, which are separated by the garbage class.

is also possible to use the garbage label with a tri-state model: then $\mathcal{X}$ is inserted between every state of all the phonemes, while still being optional.

3.3.4 Decoding and generating forced alignments

When a predictor RNN is used for decoding, it is obvious that we cannot perform a standard Viterbi beam search; this is why Graves et al. had to modify the decoding algorithm as well. However, when we switch to a HMM/DNN that has a feed-forward DNN architecture, this constraint vanishes and we can apply any kind of standard decoding method.

The only reason why we need to alter the decoding part is that we need to remove the garbage label from the resulting phoneme sequence. Luckily, in other respects the use of the garbage label does not affect the strictly-interpreted decoding part. This label of course has to be ignored during search when we apply a language model like a phoneme n-gram. In our tests, we used our own implementation of the Viterbi algorithm [41].

Generating time-aligned labels for each frame is important for later steps like the state clustering phase. Strictly speaking, it is impossible to generate accurate alignments as most of the frames will be labelled as garbage, and the presence of the phones are represented only by spikes. In [33] the authors tackle this problem by just using the frames that correspond to the spikes, however this means that a large portion of the data will be ignored during state clustering.

3.4 Sequence-Discriminative Training Using MMI

Several sequence-discriminative training criteria have been developed for the traditional HMM/GMMs [43] – and adapted to HMM/DNNs [27, 34, 39, 44] – from which the maximum mutual information (MMI) criterion is the oldest and simplest. The MMI function measures the mutual information between the distribution of the observation and the phoneme sequence. Denoting the sequence of all observations by $O_u = o_{u1}, \ldots, o_{uT_u}$, and the label-sequence for utterance u by W_u, the MMI criterion can be defined by the formula

$$F_{MMI} = \sum_u \log \frac{p(O_u|S_u)^\alpha p(W_u)}{\sum_W p(O_u|S)^\alpha p(W)}, \tag{3.8}$$

where $S_u = s_{u1}, \ldots, s_{uT_u}$ is the sequence of states corresponding to W_u, and α is the acoustic scaling factor. The sum in the denominator is taken over all phoneme sequences in the decoded speech lattice for u. Differentiating Equation (3.8) with respect to the log-likelihood $\log p(o_{ut}|r)$ for state r at time t, we get

$$\begin{aligned} \frac{\partial F_{MMI}}{\partial \log p(o_{ut}|r)} &= \alpha\delta_{r;s_{ut}} - \frac{\alpha \sum_{W:s_t=r} p(O_u|S)^\alpha p(W)}{\sum_W p(O_u|S)^\alpha p(W)} \\ &= \alpha\big(\delta_{r;s_{ut}} - \gamma_{ut}^{DEN}(r)\big), \end{aligned} \tag{3.9}$$

where $\gamma_{ut}^{DEN}(r)$ is the posterior probability of being in state r at time t, computed over the denominator lattices for utterance u using the forward-backward algorithm, and $\delta_{r;s_{ut}}$ is the Kronecker delta function (the one-hot frame-level phonetic target vector).

3.5 Performing MMI training without frame alignments

Sequence training criteria like the MMI error function are now widely used for DNN training. However, almost all authors initialise their networks using CE training, and apply the sequence-discriminative criterion only in the final phase of the training procedure, to fine-tune their models [34, 39]. This makes it necessary to use some method (like HMM/GMM or iterative CE training) to provide frame-level state targets. In contrast with these authors, here we propose to apply MMI training in the flat start phase. In order to be able to perform flat start of randomly initialised DNNs using MMI training, we made some slight changes in the standard process, which we will describe next.

Firstly, we use the numerator occupancies $\gamma_{ut}^{NUM}(r)$ in Eq. (3.9) instead of the $\delta_{r;s_{ut}}$ values. This way we can work with smoother targets instead of the crude binary

ones usually employed during DNN training. Another advantage of eliminating the $\delta_{r;s_{ut}}$ values is that it allows us to skip the preceding (usually GMM-based) label alignment step, responsible for generating the frame-level training targets. We applied the forward-backward algorithm to obtain the $\gamma_{ut}^{NUM}(r)$ values, this solution has been mentioned in some studies (e.g. [34, 44]); but we only found Zhou et al. [35] actually doing this. However, they pre-trained their DNN with the CE criterion first, while we apply MMI training from the beginning, starting with randomly initialised weights.

The second difference is that sequence training is conventionally applied only to refine a fully trained system. Therefore, the MMI training criterion is calculated with CD phone models and a word-level language model. This makes the decoding process slow, and hence the numerator and denominator lattices are calculated only once, before starting MMI training. In contrast to this, we execute sequence DNN training using only phone-level transcripts and CI phone models. This allows very fast decoding, so we can recalculate the lattices after each sentence. This difference is crucial for the fast convergence of our procedure. For converting the orthographic transcripts to phone sequences, one can follow the strategy of HTK. That is, in the very first step we get the phonetic transcripts from the dictionary, with no silences between the words. Pronunciation alternatives and the optional short pause at word endings can be added later on, when realignment can be performed with a sufficiently well-trained model [1].

A further difference is that we use no state priors or language model, which makes the α scaling factor in Eq. (3.9) unnecessary as well. Next, to reduce the computational requirements of the algorithm, we estimated $\gamma_{ut}^{DEN}(r)$ using just the most probable decoded path instead of summing over all possible paths in the lattice (denoted by $\hat{\gamma}_{ut}^{DEN}(r)$).

With these modifications, the gradient with respect to the output activations (a_{ut}) of the DNN is found using

$$\begin{aligned}\frac{\partial F_{MMI}}{\partial a_{ut}(s)} &= \sum_r \frac{\partial F_{MMI}}{\partial \log p(o_{ut}|r)} \frac{\partial \log p(o_{ut}|r)}{\partial a_{ut}(s)} \\ &= \gamma_{ut}^{NUM}(s) - \hat{\gamma}_{ut}^{DEN}(s),\end{aligned} \tag{3.10}$$

which can be applied directly for DNN training. A standard technique in DNN training is to separate a hold-out set from the training data (see [45]). If the error increases on this hold-out set after a training iteration, then the DNN weights are restored from a backup and the training continues with a smaller learning rate. This strategy can be readily adapted to sequence DNN training [27], and we found it to be essential for the stability of our flat start MMI training method.

In summary, the modifications that we propose in order to make MMI training suitable for DNN flat start are:

1. Frame-level phonetic targets ($\gamma_{ut}^{NUM}(r)$) are determined by a forward-backward search.

2. We employ only phoneme-level transcripts and CI phoneme states.

3. We do not apply state priors or language model.

4. We estimate $\gamma_{ut}^{DEN}(r)$ by just using the most probable decoded path ($\hat{\gamma}_{ut}^{DEN}(r)$).

5. We measure training error on a hold-out set; when the error increases after a training iteration, we restore the weights and decrease the learning rate.

Note that steps (1) to (4) seek to simplify the procedure both to speed it up and to make it more robust. Step (2) also helps us to perform sequence-discriminative DNN training before CD state tying, which is essential for applying it in flat start. Step (5), however, is applied in our general DNN training process, but we found it crucial to avoid the "runaway silence model" issue [46], which is a common side-effect that haunts sequence-discriminative DNN training. The "runaway silence model" is caused by the poor lattice quality, meaning that the number of silence frames after decoding increases as the training epoch increases leading to a high number of deletion errors. Our solution simply monitors the performance of the network, and once the decoding result deteriorates the weights of the DNN are reverted to their previous values and the training is continued with a lower learning rate.

3.6 Experiments and Comparison

3.6.1 Databases

We tested the CTC and MMI training methods on three different databases. The first was the well-known TIMIT set [47], which is frequently used for evaluating the phoneme recognition accuracy of a new method. Although it is a small dataset by today's standards, a lot of experimental results have been published for it; also, due to its relatively small size, it is ideal for experimentation purposes. We used the standard (core) test set, and withheld a small part of the training set for development purposes. The standard phoneme set consists of 61 phonemes, which is frequently reduced to a set of 39 labels when evaluating the models; we experimented with training on these 61 phonemes and also on the restricted set of 39 phonemes.

The second database was a Hungarian audiobook, the same one described in the previous chapter. Lastly, for the third database, the clean part of the Szeged Hungarian Broadcast News Corpus was chosen.

3.6.2 Experimental Setup

In our experiments, we compared only the phoneme error rates of the algorithms, since our aim was to develop a method capable of performing flat start training. Although it is standard practice to use a phoneme bigram, we chose to focus only on the acoustic models and we did not utilize any language model. The reason for this is that the following steps in the ASR training process rely heavily on the quality of the initial acoustic models. Furthermore, due to the introduction of the garbage symbol in the phoneme set in the case of CTC, including a phoneme n-gram in the dynamic search method seems overly complicated.

As the frame-level classifier we utilised Deep Rectifier Neural Networks (DRN) [14, 19], which have been shown to achieve state-of-the-art performance on TIMIT [48]. DRN differ from traditional deep neural networks in that they use rectified linear units in the hidden layers; these units differ from standard neurons only in their activation function, where they apply the rectifier function ($\max(0, x)$) instead of the sigmoid or hyperbolic tangent activation. Due to the better behaviour of this activation function, we can build deep networks with many hidden layers without the need for complicated pre-training methods, just by applying standard backpropagation training. Nevertheless, to keep the weights from growing without limit, we have to use some kind of regularisation technique; here, we applied the method called L2 normalisation. Our DRN consisted of 5 hidden layers, with 1000 rectifier neurons in each layer. The initial learn rate was set to 0.2 and held fixed while the error on the development set kept decreasing. Afterwards, if the error rate did not decrease for a given iteration, the learn rate was subsequently halved. The learning was accelerated by using a momentum value of 0.9. We used the well-known MFCC+Δ+$\Delta\Delta$ feature set as acoustic features. In each case, the neural network was trained on 15 neighbouring frames.

3.6.3 Results

First we evaluated the CTC and MMI training methods on the TIMIT database, the results of which can be seen in Table 3.1. In this data set a manual segmentation is also available, so we decided to use the results obtained by training using the manually given boundaries as baseline. As a further comparison, the training was repeated in the usual way, where the training labels are obtained using forced alignment. We found that the results obtained using the hand-labeled set of labels were noticeably worse

Database	Method		Dev. set	Test set
TIMIT	Monostate (39)	CTC + DRN	**26.69%**	**28.60%**
		MMI + DRN	27.70%	30.94%
		Hand-labeled	27.26%	29.35%
		Forced Alignment	27.10%	28.92%
	Monostate (61)	CTC + DRN	**26.07%**	**27.34%**
		MMI + DRN	25.16%	27.89%
		Hand-labeled	26.42%	27.94%
		Forced Alignment	25.92%	27.55%
	Tristate (183)	CTC + DRN	23.20%	24.41%
		MMI + DRN	**20.32%**	**22.76%**
		Hand-labeled	22.75%	24.7%
		Forced Alignment	22.78%	24.48%

Table 3.1: The phoneme error rates got for the different DRN training methods.

than those we got when we used forced-aligned labels got by a HMM/GMM. This reflects the fact that the manually placed phone boundaries are suboptimal compared to the case, where the algorithm is allowed to re-align the boundaries according to its needs. On all three databases, the results obtained using tri-state models were always better than those got with monostate ones.

Furthermore, the CTC+DRN training model consistently outperformed the other two non-sequence-based training schemes (although sometimes only slightly), when evaluated on the test set. On the development set usually the standard training strategies were better than the CTC method, which can probably be attributed to overfitting.

The MMI+DRN networks performed poorly with monostate labels, mostly because of the runaway silence problem. By using tristate labels we enforce each phone to have a minimal length, hence it is harder for the silence to suppress the other phones. As can be seen in Table 3.1 and Table 3.2 the tri-phone MMI+DRN models significantly outperformed all other methods.

Comparing the training times of the MMI and CTC we noticed that they were slightly slower than the baseline cases: calculating the α and β values increased the execution times only by a very small amount, but it took a few more iterations to make the weights converge. On TIMIT, CTC used all training vectors 24-25 times and MMI performed 21-24 training iterations, whereas the baseline required only 18-19. This is probably due to the fact that the sequence-based methods strongly rely on the acoustic classifier trained in the previous iteration, so it takes a few iterations before

Database	Method		Dev. set	Test set
Audiobook	Monostate (52)	CTC + DRN	17.85%	16.55%
		MMI + DRN	**16.95%**	**16.12%**
		Forced Alignment	17.76%	16.98%
	Tristate (156)	CTC + DRN	12.58%	11.67%
		MMI + DRN	**10.08%**	**9.67%**
		Forced Alignment	12.53%	11.96%
Broadcast news	Monostate (52)	CTC + DRN	25.96%	**25.58%**
		MMI + DRN	35.66%	65.26%
		Forced Alignment	**25.82%**	25.64%
	Tristate (156)	CTC + DRN	21.62%	21.23%
		MMI + DRN	**20.74%**	**20.42%**
		Forced Alignment	22.13%	21.74%

Table 3.2: The phoneme error rates got for the two different DRN training methods.

the training starts to converge. We think these values are not high, especially as Graves et al. reported much higher values for CTC (frequently over 100 iteration) [32].

Another interesting point is that besides the similar accuracy scores, standard non-sequence-based method leads to a relatively high number of phoneme insertions, while when performing sequence training it is common to have a lot of deletion errors. The reason is that the correct phonemes are often suppressed by $\mathcal{X}$s (in the case of the CTC) or by the silence model (in the case of the MMI). During decoding, the $\mathcal{X}$ labels are deleted from the output before the accuracy score is calculated. This behaviour, fortunately, does not affect the overall quality of the result.

3.7 Summary

In this chapter, I adapted two sequence learning method to a standard HMM/DNN architecture. The CTC method was originally developed for RNNs, but here I showed that it can be used with DRNs as well. The CTC method relies on the blank or garbage label, which makes the decoding process problematic. Furthermore, the DRNs trained with CTC cannot be used to generate proper forced alignment of the labels.

The other method described here (MMI) is widely used in DNN training, but it usually requires some initialisation before training. To circumvent this constraint, I proposed several modifications to the original method, to make it suitable for flat start training.

Compared to standard zero-or-one frame-level backpropagation DNN training, I found that networks trained with these sequence learning method always produced higher accuracy scores than the baseline ones. From the results it was also clear that MMI with tristate labels works best, so it is well suited for flat start training.

In this chapter, the author regards the following as his main contributions:

- The use of CTC with feedforward DRNs;
- Modifications to MMI training algorithm, to make it applicable to flat start training;

The methods and results presented in this chapter were published in [49] and [50].

Chapter 4

A GMM Free Training Method for Deep Neural Networks

Today, deep neural network based speech recognizers have completely replaced Gaussian mixture-based systems as the state-of-the-art. While some of the modeling techniques developed for the GMM-based framework may directly be applied to the HMM/DNN systems, others may be inappropriate. One such example is the creation of context-dependent tied states, for which an efficient decision tree state tying method exists. The tied states used to train DNNs are usually obtained using the same tying algorithm, even though it is based on likelihoods of Gaussians, hence it is more appropriate for HMM/GMMs. Recently, however, several refinements have been published which seek to adapt the state tying algorithm to the HMM/DNN hybrid architecture. Unfortunately, these studies reported results on different (and sometimes very small) datasets, which does not allow their direct comparison.

Some of the new state tying methods change only the input of the clustering algorithm, while the whole state tying algorithm remained intact. These methods feed the output or the activations of the neurons in the last hidden layer to the clustering method and use the same standard Gaussian-based decision tree clustering method. Other studies proposed novel decision criteria as well for the standard state tying method, which better suit the new input.

In this chapter, we present a new state tying criterion, and evaluate it by comparing its performance to three other methods on the same LVCSR tasks, under the same circumstances. We found that, besides changing the input of the context-dependent state tying algorithm, it is worth adjusting the tying criterion as well. The methods which utilised a decision criterion designed specifically for neural networks consistently, and markedly outperformed those which employed the standard Gaussian-based algorithm.

4.1 Problem description and literature overview

While deep neural network-based speech recognizers have recently replaced Gaussian mixture-based systems as the state-of-the-art in ASR, the training process of HMM/DNN hybrids still relies on the HMM/GMM framework. Recently, however, attempts have been made to remove GMMs from the training process of deep neural network-based hidden Markov models (HMM/DNN). For the GMM-free training of a HMM/DNN hybrid we have to solve two problems, namely the initial alignment of the frame-level state labels and the creation of context-dependent states. Conventionally, we start the training of a HMM/DNN by constructing a HMM/GMM system, which is then applied to get an alignment for the frame-level state labels. These labels are then used as the training targets for the DNN. The second task that requires GMMs is the state-tying algorithm utilised for the construction of context-dependent (CD) phone models. We propose a GMM-free solution for state clustering here [51], and we will combine it with DNN-based methods, which can generate an initial state alignment.

The most convenient way of training the DNN component of a HMM/DNN hybrid is to apply a frame-level error criterion, which is usually the cross-entropy (CE) function. This solution, however, requires frame-aligned training labels, while the training dataset contains just orthographic transcripts in most cases. Of course, one could train a HMM/GMM system to get aligned labels, but this is clearly a waste of resources.

The procedure for training HMM/GMM systems without alignment information is commonly known as 'flat start training' [1]. This consists of initialising all phone models with the same parameters, which would result in a uniform alignment of phone boundaries in the first iteration of Baum-Welch training. It is possible to construct a flat start-like training procedure for CE-trained DNNs as well, by iteratively training and realigning the DNN. For example, Senior et al. randomly initialised their neural network [52], while Zhang et al. trained their first model on equal-sized segments for each state [53]. As these solutions have a slow convergence rate, they require a lot of training-realignment loops.

Although training the DNN at the frame level is straightforward, it is clearly not optimal, as the recognition is performed and evaluated at the sentence level. Within the framework of HMM/GMM systems, several sequence-discriminative training methods have been developed, and these have now been adapted to HMM/DNN hybrids as well [27, 34, 49]. However, most authors apply sequence-discriminative criteria only in the final phase of training, for the refinement of the DNN model. That is, the first step is always CE-based training, either to initialize the DNN (e.g. [35, 36, 37]) or just to provide frame-level state labels (e.g. [27, 34, 38, 39, 40]).

The CTC approach has recently become very popular for training DNNs without

an initial time alignment being available [32]. Rao et al. proposed a flat start training procedure which is built on CTC [33]. However as we explained earlier, CTC has several drawbacks compared to MMI. First, it introduces blank labels, which require special care in the later steps (e.g. CD state-tying) of the training process. Nevertheless, the CTC algorithm is not a sequence-discriminative training method, so for the best performance it has to be combined with techniques like sMBR training [32, 33]. After performing flat-start training with CTC, however, it is not clear how to revert to the standard phoneme set for CD state-tying or standard CE CD DNN training. Therefore integrating the CTC flat start method into the standard DNN training framework does not appear to be a suitable option. Still, MMI with proper modifications is suitable for flat start training a DNN.

While hybrid models applied only context-independent (CI) phone models for a long time [20], there is now common agreement that HMM/DNN systems also greatly benefit from using context-dependent tied states [54, 17]. Thus, it is necessary to find an approach for efficiently creating context-dependent tied states for systems built on DNNs. Currently, the dominant solution is the decision tree-based state tying method of Young et al. [55]. This technique fits Gaussians on the distribution of the states, and uses the likelihood gain to govern a decision tree-based state-splitting process. Thanks to the Gaussian assumption and the decision tree representation, this approach is computationally very efficient. However, as we have already mentioned, it may be inappropriate to just impose the common HMM/GMM-based techniques on the HMM/DNN training procedure, and this may hold for this state-tying algorithm as well.

GMM-based methods assume that the Gaussian components have diagonal covariance matrices, and hence require decorrelated features like cepstral coefficients (MFCCs). However, HMM/DNN hybrids tend to work better on more primitive features like mel filter bank energies (Fbank) [15]. Since conventional HMM/GMM systems cannot be efficiently trained on these features, the usual approach is to build a HMM/GMM system on a standard feature set like MFCCs, create the tied-state inventory and alignment, and then throw away the feature set and the whole model. This process, besides wasting resources, also implies that state tying is done on an mismatched feature set. Furthermore, intuitively, the state clustering algorithm should split those states where the splitting would be beneficial for the respective classifier. Since the objective functions during GMM and DNN training are different, measuring how a Gaussian fits a given class may be unrelated to the difficulty of modeling that class by a DNN. This suggests that if we perform the CD state-tying by following the standard approach, we do it on a mismatched feature set and using a mismatched similarity metric.

Not so long ago, a number of articles were published on CD state tying for HMM/DNNs. The issue of the "inappropriate feature set" can be handled by performing the state clustering process on the output of a DNN instead of the raw features. This idea was investigated in a couple of studies (e.g. [56, 53, 52, 57]). In those studies, however, only the input of the clustering algorithm was modified, while the whole state-tying algorithm remained intact. Other studies proposed novel decision criteria for the standard state-tying method, which better suit neural networks. Here, we propose the use of the Kullback-Leibler divergence-based decision criterion, originally developed for KL-HMMs by Imseng et al. [58]. Zhu et al. [59] constructed a criterion that relied on entropy. Lastly, Wang et al. [60] trained a special network that optimised for Deep Canonical Correlation Analysis, and clustered the output of this network via k-means clustering.

All these studies experienced a drop in the word error rate (WER) compared to the baseline that uses the standard Gaussian likelihood-based state-tying method with the MFCC vectors. Yet, none of these studies compared their results with other neural network-based state-tying approaches, which makes these methods quite hard to compare. Furthermore, the datasets used differed to a huge extent as well: we used a Hungarian database, Zhu et al. used a German one, while Wang et al. used the quite small TIMIT corpus, where only phoneme error rates can be reported. In this chapter we compare four such approaches on the same LVCSR task, where the same context-independent neural network will provide the input vectors for the state clustering. Note that, since we obtain the frame-level CI labels by purely DNN-based flat-start methods, the CI models have no inherent GMM dependency. Therefore those state-tying methods that have a decision criterion designed for DNNs are completely GMM-free.

In the experimental part we compare the two flat start methods, namely the one that applies MMI with the CE-based iterative retraining realignment procedure of Zhang et al. [53]. We found that our method is not only faster, but it achieves lower word error rates as well. Furthermore, we combine the flat start training with various DNN-based state clustering methods to eliminate all dependencies from a HMM/GMM system, making the whole training procedure of context-dependent HMM/DNNs GMM-free.

4.2 Flat Start

The first step of training a speech recognition system is to get time-aligned labels for the transcription. Traditionally this is achieved by using the Baum Welch algorithm to train a HMM/GMM, as described in the previous chapter. Here, we compare two approaches that seek to eliminate GMMs from this process. As the baseline method, we apply a simple solution that iterates the loop of CE DNN training and realignment.

Afterwards, we compare it with an approach that creates time-aligned transcriptions for the training data by training a DNN with a sequence training criterion. From the wide variety of sequence training methods, we opted for MMI training [27]. Applying sequence training to flat start requires some slight modifications, which we will now discuss.

4.2.1 Iterative Flat Start

For comparison we will also test what is perhaps the most straightforward solution for flat start DNN training, namely just using the CE training criterion and iterating DNN training and realignment. Here, we used the following algorithm that was based on the description by Zhang et al. [53]:

1. Train a DNN using sound files uniformly segmented into phones.
2. Use the current DNN to realign the labels.
3. Train a randomly initialised DNN using the new alignments.
4. Repeat steps 2–3 several times.

The final DNN was utilised to create time-aligned labels for the training set.

The main advantage of this method is that it requires only an implementation of CE training for the DNN, and the realignment step can also be readily performed by using standard ASR toolkits. The drawback is that the procedure of retraining and realignment tends to be rather time-consuming, which was also confirmed by our experiments.

4.2.2 Sequence Training Based Flat Start

As an alternative to the iterative method we also used the MMI-based method described in Chapter 3). For the MMI training we commenced with a randomly initialised CI DNN and with the modifications of the original method the networks were successfully trained. As a further refinement we also tried to improve the segmentation by training another CI DNN with CE training, where the training labels were obtained by forced aligning the labels using the MMI trained DNN.

4.3 State Clustering

Nowadays, state-of-the-art ASR systems are trained with CD labels. Usually these labels consist of three phonemes, namely the actual phone, the one before and the one

after. With the triphone labels, one can achieve better recognition accuracies than with CI labels, simply because the acoustic model can handle co-articulation better in this way. Still, switching to triphone acoustic models creates a serious problem, namely the data insufficiency problem. This problem is caused by the fact that the number of possible triphones is quite large and we have only a limited amount of training data. For instance, there are almost 14,000 triphones in the training data of the Szeged Broadcast News corpus. Furthermore, the data is usually unevenly spread, meaning that some of the triphones occur only once or twice, while other labels are quite common. Additionally, it is possible for some triphones to appear only in the test data, and understandably we cannot expect the acoustic model to learn to recognise these.

Traditionally, the state-tying algorithm is used to handle these problems, which is basically a clustering method that aims to cluster the triphones derived from the same monophone. This way, we can use a shared label for the triphones which are similar and have the same central phoneme, thus solving the data sparsity problem.

The decision tree-based state-tying algorithm was introduced by Young et al. [55], and it evolved into a vital component of training large vocabulary speech recognisers. The main idea is to pool all context variants of a state, and then build a decision tree by successively splitting this set into two. For each step, the algorithm chooses one of the pre-defined acoustic questions in such a way that the resulting two non-overlapping sub-sets of the original state set $\mathcal{S}$ differ maximally. The algorithm measures this difference by using a likelihood-based decision criterion. The tree-based clustering has the important advantage of providing a mapping for unseen triphones as well. Although minor improvements to the algorithm like the automatic generation of the questions via clustering were proposed [61], the main scheme of the method proved so successful that it has remained unaltered ever since.

4.3.1 GMM-Based State Tying

Suppose that we have a set of states $\mathcal{S}$ that need to be tied, using the decision tree-based method of Young et al. [55]. Here, at each node, we have a set of questions, and each question can split $\mathcal{S}$ into two non-overlapping sub-sets depending on the answer to the question. Odell formulated a maximum likelihood-based decision criteria [62] and proposed a computationally efficient algorithm by approximating the splitting criterion as

$$L(\mathcal{S}) \simeq -\frac{1}{2}\big(\log[(2\pi)^K|\Sigma(\mathcal{S})|] + K\big)\sum_{s\in\mathcal{S}} N(s), \tag{4.1}$$

where $s \in \mathcal{S}$ are the individual states, $\Sigma(\mathcal{S})$ is the variance of data in $\mathcal{S}$, K is the dimension of the data and $N(s)$ is the number of examples (frames) in the training

data which belong to state s. Using this formula, we should choose the question q which maximizes the likelihood difference $\Delta L(q|\mathcal{S})$

$$\Delta L(q|\mathcal{S}) = \big(L(\mathcal{S}_y(q)) + L(\mathcal{S}_n(q))\big) - L(\mathcal{S}), \tag{4.2}$$

where $\mathcal{S}_y(q)$ and $\mathcal{S}_n(q)$ are the two subsets of $\mathcal{S}$ formed based on the answer to the question q. It can be seen that the likelihood values do not depend on the training observations themselves, but only on the variance over the training data corresponding to the states, and the raw number of frames belonging to each state. Although this assumption (regarding the variance of the feature vectors) fits in well with a system employing GMMs, in a HMM/DNN hybrid speech recogniser framework some other decision criterion might result in a more suitable set of tied states.

4.3.2 Clustering the CI DNN output

This approach, proposed by Senior et al. [52], is quite straightforward. They simply use the frame-level outputs of the auxiliary neural network as input for the state-tying procedure. The whole clustering process remains the same in every other respect. Senior et al. reported a slight improvement in the WER and, naturally, with this approach they were able to avoid the feature set mismatch among CD DNN training and the CD state-tying process. Despite this, as they used the original state-tying method of Odell [62], which relies on likelihoods of Gaussians, in our opinion their method can hardly be regarded as completely GMM-free.

4.3.3 Clustering the DNN hidden activations

In a parallel study Bacchiani and Rybach [57] proposed performing the clustering on the activations of the last hidden layer of the auxiliary CI NN. Although one cannot expect the activation vectors to be decorrelated (or to follow any predefined distribution), Bacchiani and Rybach were able to use them as inputs for the CD state-tying method of Young et al. The WERs they got were reported to be lower for smaller CD state sizes than by using the standard approach, but for larger state counts it was the other way around. They explained this by recalling that the frame-level CI labels were obtained by HMM/GMMs, causing a mismatch in the frame-level targets. Since we used a DNN-based flat start, there will be no such mismatch.

4.3.4 KL-divergence Based State Tying

This decision criterion was introduced by Imseng et al., who successfully applied it in their KL-HMM framework [58]. Here, we propose to use it during the CD state-tying

step. Next, we will give a brief description of this algorithm, based on articles [63] and [64].

Although the Kullback-Leibler divergence is known to be asymmetric, unfortunately there is no closed form of the symmetric KL-divergence-based cost function. Therefore we will apply the asymmetric KL-divergence between two posterior vectors z_t and y_s, defined as

$$D_{KL}(y_s|z_t) = \sum_{k=1}^{K} y_s(k) \log \frac{y_s(k)}{z_t(k)}, \tag{4.3}$$

where $k \in \{1, \ldots, K\}$ is the dimensionality index of the posterior distribution vector [65]. The KL-divergence is always non-negative and zero if and only if the two posterior vectors are equal. So instead of maximizing the likelihood, we will minimize the KL-divergence

$$D_{KL}(\mathcal{S}) = \sum_{s \in S} \sum_{f \in F(s)} \sum_{k=1}^{K} y_{\mathcal{S}}(k) \log \frac{y_{\mathcal{S}}(k)}{z_f(k)}, \tag{4.4}$$

where $\mathcal{S}$ is a set of states s, and $F(s)$ is the set of input vectors corresponding to state s. The posterior vector associated with the set $\mathcal{S}$ ($y_{\mathcal{S}}$) can be calculated as the normalised geometrical mean of the example vectors belonging to the elements of $\mathcal{S}$. That is,

$$\tilde{y}_{\mathcal{S}}(k) = \frac{\left(\prod_{s \in \mathcal{S}} \prod_{f \in F(s)} z_f(k) \right)^{\frac{1}{N(S)}}}{\sum_{k=1}^{K} y_{\mathcal{S}}(k)}. \tag{4.5}$$

After expanding and simplifying, we get [63]

$$D_{KL}(\mathcal{S}) = -\sum_{s \in \mathcal{S}} N(s) \log \sum_{k=1}^{K} \tilde{y}_{\mathcal{S}}(k), \tag{4.6}$$

so the KL divergence of a set of states $\mathcal{S}$ can be calculated based on the statistics y_s and $N(s)$ of the individual states.

For the splitting of a set of states $\mathcal{S}$, the straightforward option is to choose the question that maximizes the KL-divergence difference $\Delta D_{KL}(q|\mathcal{S})$:

$$\Delta D_{KL}(q|\mathcal{S}) = D_{KL}(\mathcal{S}) - \big(D_{KL}(\mathcal{S}_y(q)) + D_{KL}(\mathcal{S}_n(q)) \big).$$

4.3.5 Entropy-based decision criterion

The fourth approach we tested was proposed by Zhu et al. [59]. They also replaced the decision criterion of Eq. (4.1) with another formula that has no implicit GMM dependency. The key idea was to measure the inter-similarity of each merged cluster

by calculating the entropy of the examples belonging to it. The entropy of a K-dimensional probability distribution can be calculated as

$$H(p) = \sum_{i=1}^{K} p(i) \log p(i). \tag{4.7}$$

The probability distributions associated with each initial state (i.e. the y_s vectors) were estimated via the mean of the DNN outputs for all the frames associated with a given state. Then, for a set of states $\mathcal{S}$, the prototype probability vector ($y_{\mathcal{S}}$) was calculated as the arithmetic mean of the prototype (y_s) of the member states, weighted by the number of state occurrences ($N(s)$); from these values, the decision criterion used during state-tying can be calculated by using the entropy function, i.e.

$$D_E(\mathcal{S}) = -\sum_{s \in \mathcal{S}} N(s) \sum_{k=1}^{K} y_{\mathcal{S}}(k) \log y_{\mathcal{S}}(k). \tag{4.8}$$

4.4 Experimental Setup

In our experiments we employed DNNs with 5 hidden layers, each containing 1000 rectified neurons [14], while the softmax activation function was applied in the output layer. As input, FBank features were presented to the networks along with their first and second order derivatives. Decoding and evaluation was performed by a modified version of HTK [1].

The methods described in this chapter were tested on two databases. Firstly, the 81-hour long Wall Street Journal (WSJ) English read speech corpus [66] (specifically, the `si-284` set) was chosen to test the algorithms as it is a well known and widely used corpus. The recognisers were evaluated on the `eval92` and `eval93` test sets in the "open-vocabulary" (60K word vocabulary) test condition, using a pruned version of the standard trigram language model. We used the `eval93` set as our development set; i.e. we tuned the language model weight and the insertion penalty on it, and also chose the optimal number of tied states for each state-tying method based on the WER achieved on this set. Then, at the very end, we evaluated the models using the optimal meta-parameters on the `eval92` set as the test set.

We also used the 28 hour-long speech corpus of Hungarian Broadcast News [8], just like in the previous chapters. The whole corpus was utilised with the same partitioning as before: the training set was about 22 hours long, a small part (2 hours) was used for validation purposes, and a 4-hour part was used for testing.

We tested three approaches for flat start training (i.e. to get the frame-level phonetic targets for CD state-tying and CE DNN training). Since our goal was to create a GMM-free system, we evaluated the two algorithms presented in sections

4.2.1 and 4.2.2 and their combination for flat starting with DNNs. In these tests we always used five-hidden-layer CI DNNs. For the flat start method with iterative CE training (*"Iterative CE"*) we performed nine training-aligning iterations on the WSJ and four on the Hungarian database. DNN-based CD state tying was performed using the output and the alignments created by the final DNN. For MMI training (*"MMI"*) we also commenced with a randomly initialised CI DNN. After applying the discriminative sequence training method, the resulting DNN was used to create forced aligned labels and also to provide the input posterior estimates for KL clustering. In the last flat start approach tested, we first applied the sequence-discriminative method (i.e. "MMI"). Afterwards, we combined the two approaches; first we produced the alignments using the MMI network, then we trained another DNN with the CE criterion to supply both the final frame labels and the likelihoods for CD state-tying (*"MMI + CE"*). After the CI DNN training phase we tested the state-tying methods described in this chapter. To see how well the state clustering methods perform we also created CD states with the original GMM based method (*"MFCC + Likelihood "*). Keep in mind that in this case the frame alignments were still produced by a DNN.

In the case of the Hungarian corpus we also applied the standard GMM-based flat start training to produce initial time-aligned labels. To further improve the segmentation, we trained a shallow CI ANN using the CE criterion and re-aligned the frame labels based on the outputs of this ANN (we will refer to this approach as the *"GMM + ANN"* method). (In our early study we found that using a deep neural network for this re-alignment setup did not bring about any improvement [51].) After the realignment, only the KL-based state-tying algorithm was executed on the output of the CI ANN.

The main aim of this chapter is to compare various flat start strategies and state-tying methods. This is why, after obtaining the CD labels, the final DNN models were trained starting from randomly initialised weights and using just the CE criterion. Of course, it is possible to extend the training with a final refinement step using CD sequence-discriminative training, but it is out of the scope of this study.

4.5 English results

First, let us compare the three flat start strategies, Table 4.1 shows the WERs of the CI DNNs. It is clear that the iterative method, which optimised the frame level CE performed worst of all. Besides providing inferior WER, it also required far more time than the other two approaches. In total nine training and re-aligning iterations were required, afterwards the WER on the development set started to decrease. Surprisingly, the DNN trained with MMI yielded quite acceptable results, despite the fact that it was only a CI acoustic model. The combination of the two methods (MMI+CE)

Method	Dev.	Test
Iterative CE	28.63%	20.47%
MMI	15.78%	10.07%
MMI+CE	15.43%	9.64%

Table 4.1: WERs got by using different flat start methods on the WSJ.

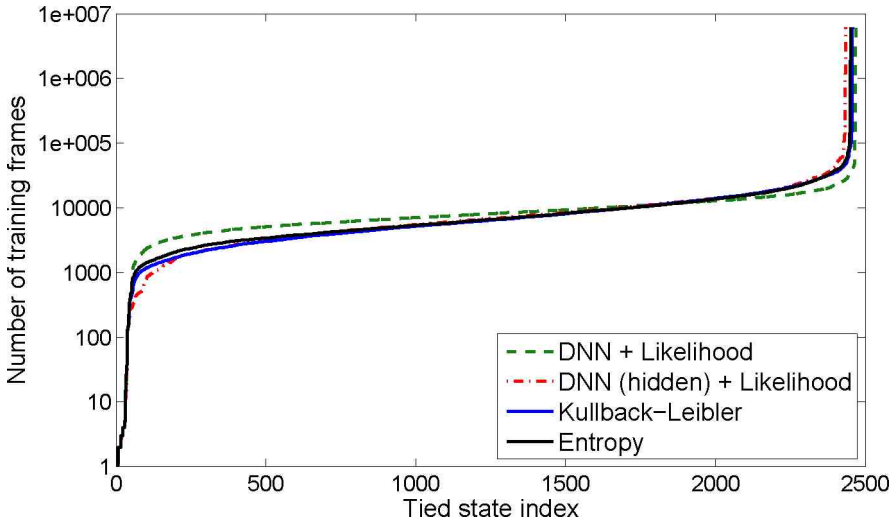

Figure 4.1: The number of training frames for the different state-tying methods for the case of about 2400 CD states.

offered some improvements on both the development and test set, but almost doubled the training time. The inferior performance of the iterative method can be explained by the "run-away silence model" effect, meaning that these networks became greatly biased towards the silence label, this caused a lot of deletion errors.

Figure 4.1 shows the distribution of the training classes after state-tying (using the MMI+CE flat-start strategy) for the case of roughly 2400 CD states. Besides noticing that the distribution produced by the different state-tying methods is quite similar, we should also note that using the original decision criterion with the DNN outputs as input (proposed by Senior et al.) resulted in the best balanced class distribution.

First, we would like to mention that a purely GMM based system achieves 12.74% and 9.46% on the development and test sets, respectively [67]. Table 4.2 lists the best WER scores got on the development set and the corresponding WER values obtained on the test set. All GMM-free state-tying methods achieved the best results with an MMI-trained DNN. We can also see that the Iterative CE strategy led to the worst results and interestingly the MMI+CE approach yielded worse results than the MMI based flat start.

Flat start strategy	Clustering method	Development	Test
Iterative CE	MFCC + Likelihood	11.02%	8.20%
	DNN + Likelihood	11.48%	7.64%
	DNN (hidden) + Likelihood	11.05%	7.81%
	Kullback-Leibler	10.47%	7.27%
	Entropy	10.24%	7.27%
MMI	MFCC + Likelihood	8.58%	6.13%
	DNN + Likelihood	8.7%	6.47%
	DNN (hidden) + Likelihood	8.85%	6.04%
	Kullback-Leibler	8.06%	5.72%
	Entropy	8.03%	5.92%
MMI + CE	MFCC + Likelihood	8.79%	5.97%
	DNN + Likelihood	9.14%	6.45%
	DNN (hidden) + Likelihood	9.43%	6.77%
	Kullback-Leibler	8.5%	6.15%
	Entropy	8.09%	6.20%

Table 4.2: WER values on the development and test sets got by using the different flat-start and CD state-tying methods.

Upon examining the results it can be seen that all GMM-free methods markedly outperformed the HMM/GMM system. Notice that on the development set the highest WER is around 11.48%, while it is 8.2% for the test set, so the same relative WER improvement corresponds to a smaller absolute improvement for the latter set. The two most basic approaches worked the worst of all: using the CI DNN outputs or the hidden activations with the standard state-tying decision criterion. While using MMI-based flat start and the outputs of the last hidden layer with the original state-tying method (proposed by Bacchiani and Rybach) led to slightly worse scores on the development set, it greatly outperformed the first approach on the test set. This approach is also justified by the fact that the activation vectors of a DNN are commonly used as features in several tasks such as speaker identification [68] and various image processing applications [69]. The standard state-tying method (MFCC + Likelihood) was quite competitive with those that did not change the decision criteria, meaning that the alignments produced by a DNN were better than those got by using a GMM.

The remaining two methods utilised some novel decision criteria instead of the

Gaussian-based, standard one; and this fact is clearly reflected in their performance. On the development set they achieved practically identical WER scores (8.06% vs. 8.03% for the Kullback-Leibler and the entropy-based decision criteria, respectively); they differed somewhat on the test set, but the difference is not statistically significant. Overall, by relying on the Kullback-Leibler-based decision criterion the WER scores were reduced by 0.8% compared to the basic approach of Senior et al., meaning a 12% improvement in terms of relative error reduction.

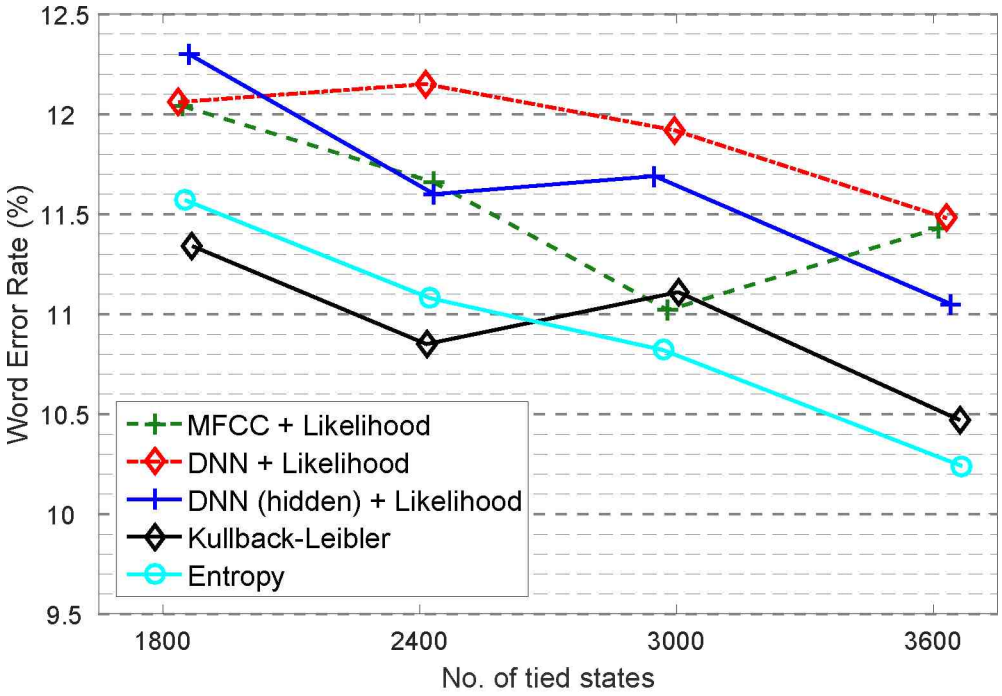

Figure 4.2: WER for the different state-tying approaches on the development set using the iterative flat start method.

Taking a closer look at figures 4.2-4.7, we can see the WER scores obtained as a function of the number of CD states. It can be observed that the two solutions that used the original state-tying algorithm, and the two which utilised a decision criterion designed for DNN outputs, are well separated, with the latter group producing consistently lower WER scores for both sets regardless of the number of tied states. The results of the MFCC + Likelihood method are a little hard to interpret as the curves of the development and test sets do not correlate, and sometimes behave quite differently. The most probable explanation for this is the inappropriate feature set issue: the CI DNN was trained using FBank features, but the state-tying algorithm used MFCC features. These results, in our opinion, confirm our hypothesis that besides changing the input of the CD state-tying algorithm, its behaviour should also be adapted to better suit DNNs, allowing them to achieve better results.

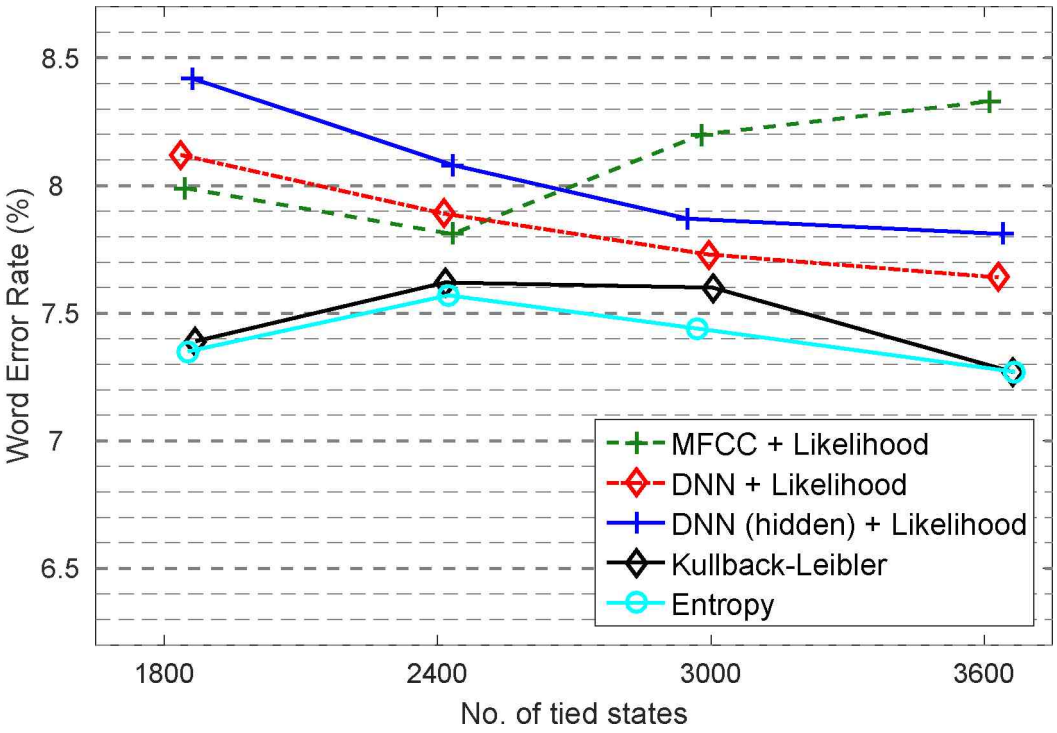

Figure 4.3: WER for the different state-tying approaches on the test set using the iterative flat start method.

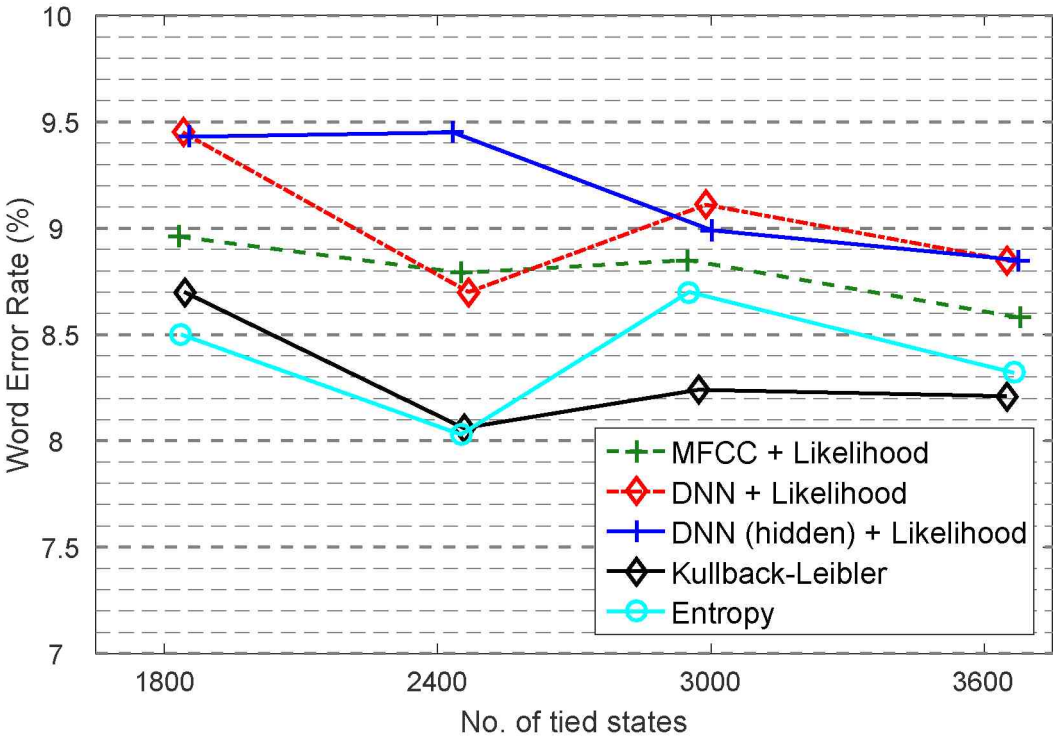

Figure 4.4: WER for the different state-tying approaches on the development set using MMI for flat start.

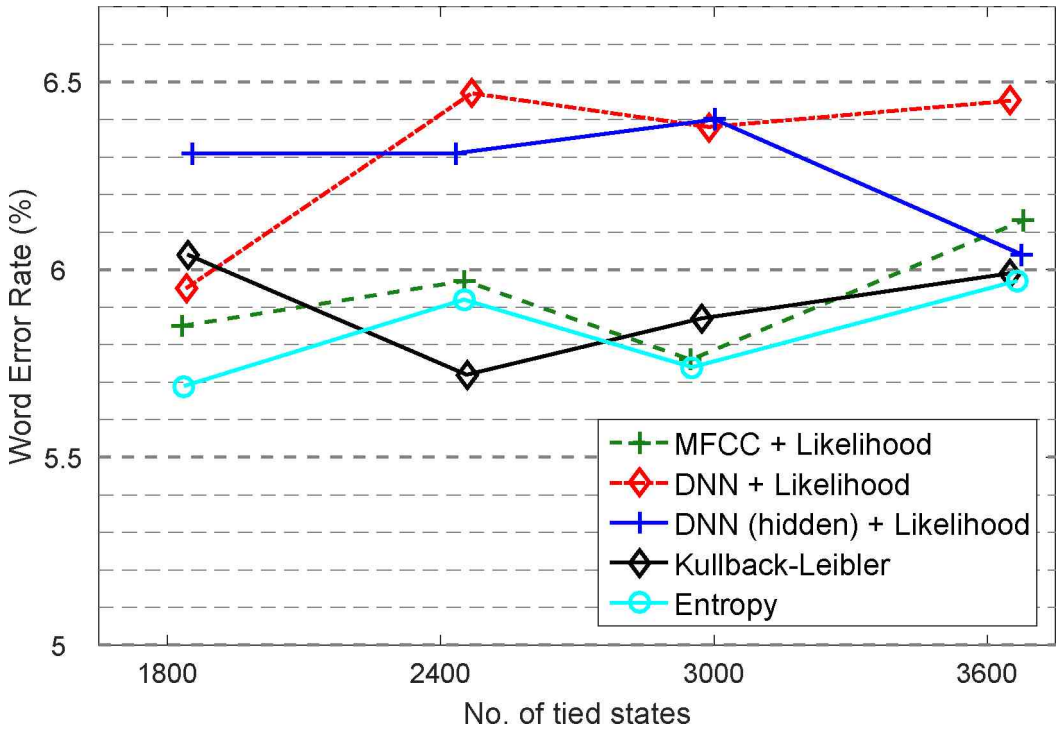

Figure 4.5: WER for the different state-tying approaches on the test set using MMI for flat start.

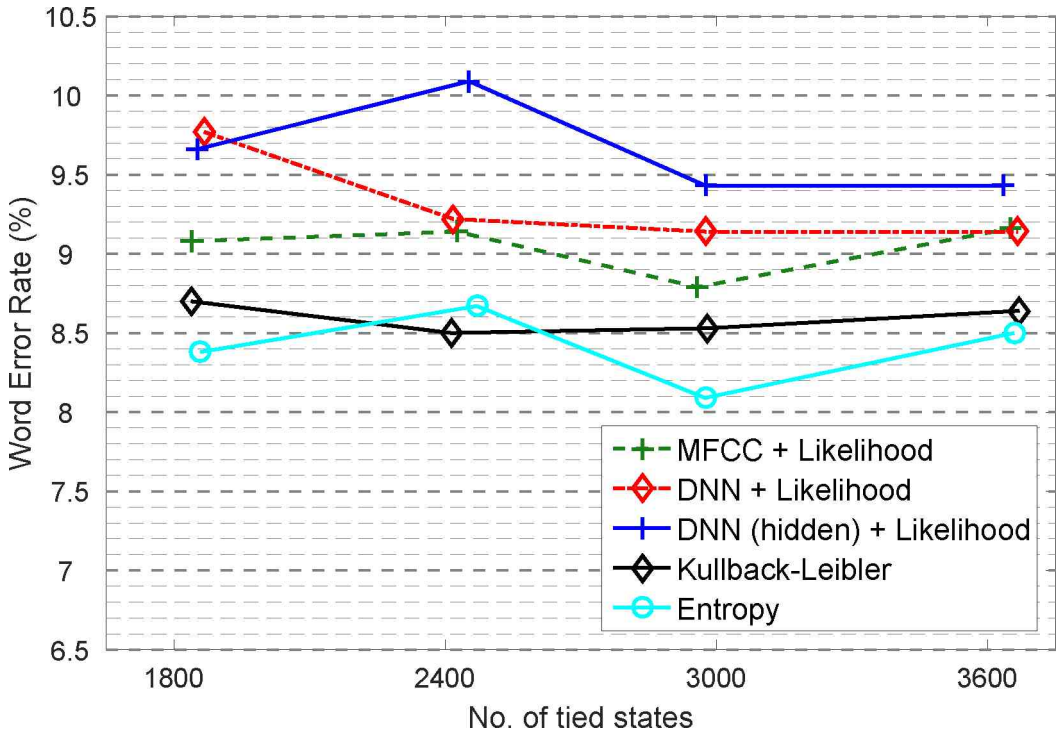

Figure 4.6: WER for the different state-tying approaches on the development set using MMI-CE for flat start.

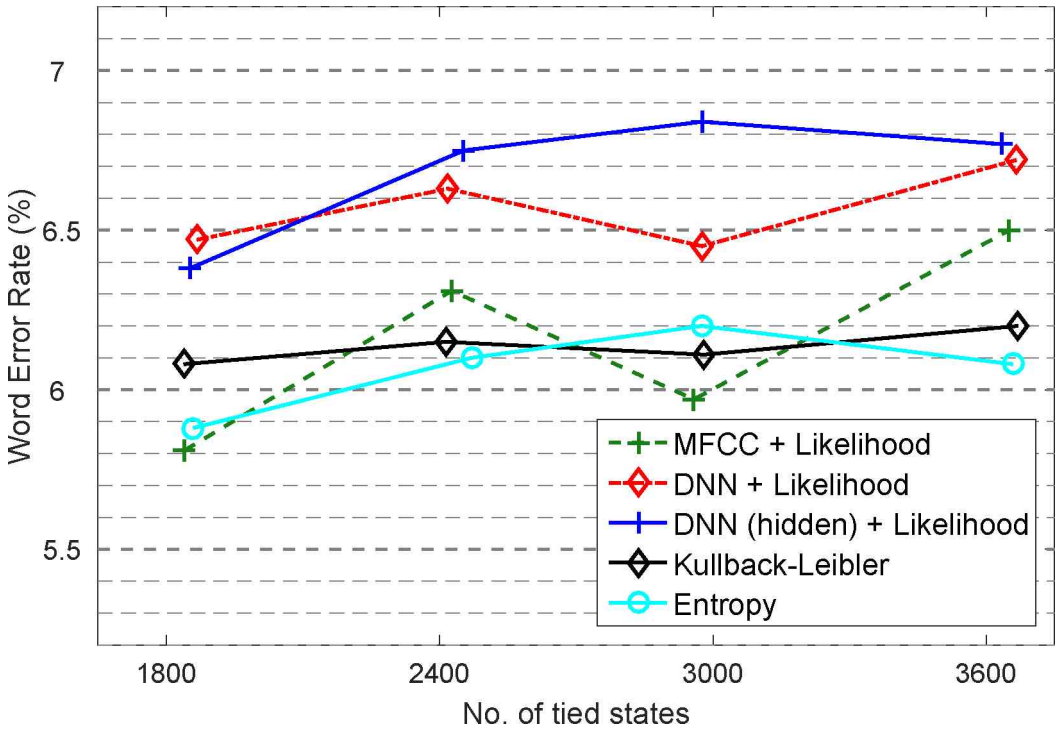

Figure 4.7: WER for the different state-tying approaches on the test set using MMI-CE for flat start.

As a further note, increasing the number of CD states helps those approaches that use the original, likelihood-based criterion; for the other two methods, however, optimality is achieved by having about 2400 states. On the test set, all four approaches seem to be quite insensitive to the number of tied states. Note that these inventory sizes appear to be smaller than those commonly used on the WSJ corpus, which, due to the lower computational requirements, is an improvement by itself.

4.6 Hungarian results

Figures 4.8 and 4.9 show the resulting WER scores as a function of the number of CD tied states on the Szeged Hungarian Broadcast News dataset. As can be seen, the MMI-based flat start strategy gave consistently better results than the iterative method in every case, just like before. We also observed that the final CD models which got their training labels from the MMI-trained DNN were more stable with respect to varying the number of CD states. Fine-tuning the labels of the MMI-trained DNN with a CE-trained DNN ("MMI" vs. "MMI+CE") again seems unnecessary, as it was not able to notably improve the results. This strongly suggests that sequence training yields both fine alignments and good posterior estimates.

Flat start method	state-tying method	WER %		No. of epochs
		Dev.	Test	
GMM + ANN	GMM	18.83%	17.27%	—
GMM + ANN	KL	17.12%	16.54%	—
Iterative CE	KL	16.81%	16.50%	48
MMI		16.50%	15.96%	13
MMI + CE		16.36%	15.86%	29

Table 4.3: *Word error rates (WER) for the different flat start strategies and the KL state-tying method.*

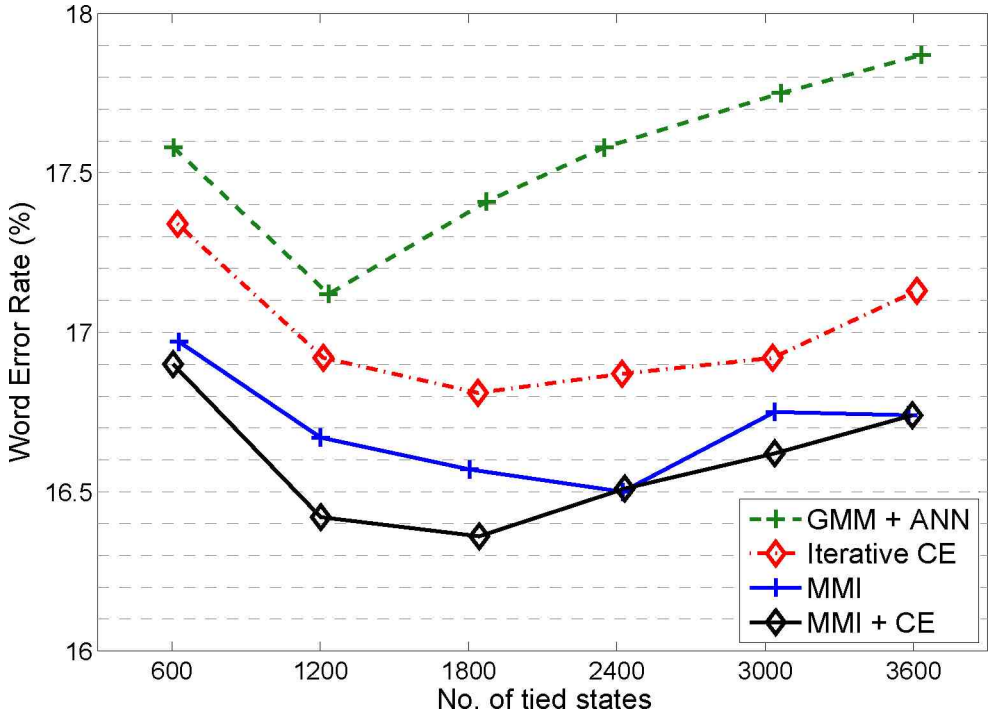

Figure 4.8: WER as a function of the number of KL-clustered tied states on the Hungarian development set.

Table 4.3 summarises the best WER values obtained on the development set, and the corresponding scores on the test set for the Hungarian corpus. The KL clustering method clearly outperformed the GMM-based state-tying technique. Comparing the alignment methods, we can see that relying on the alignments produced by the HMM/GMM resulted in the lowest accuracy score, in spite of the fine-tuning step that used an ANN. After setting the parameter configurations, the Iterative CE training method performed slightly worse than the MMI-based strategies. Unfortunately, for

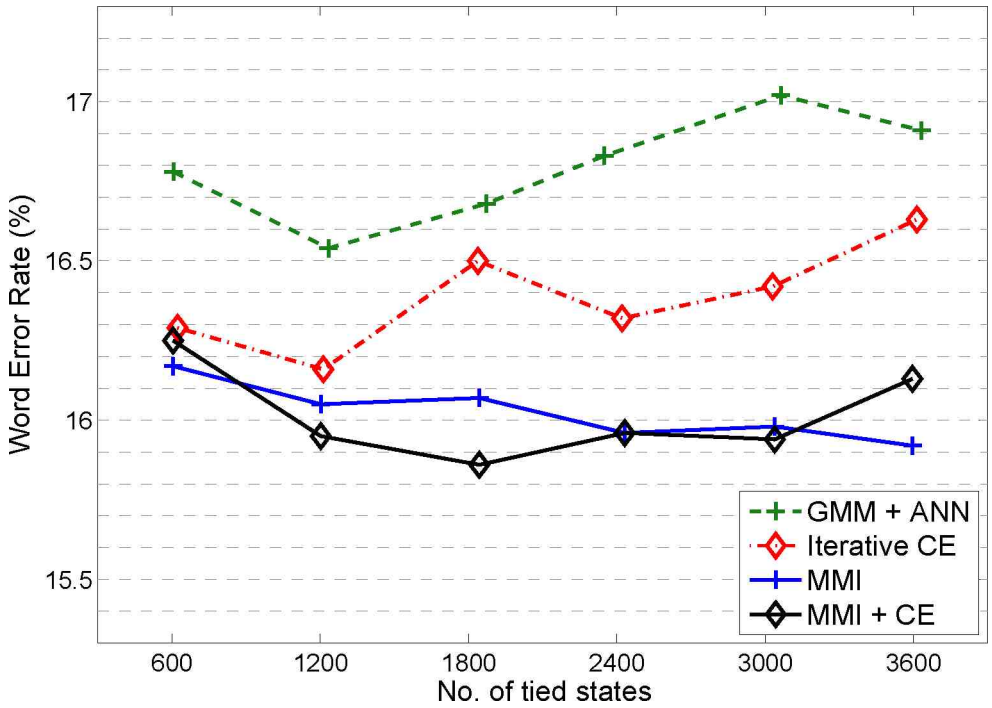

Figure 4.9: WER as a function of the number of KL-clustered tied states on the Hungarian test set.

the Iterative CE method the right number of training-aligning steps is hard to tune. For example, Zhang et al. performed 20 such iterations [53], while we employed only 4 iterations. In this respect, it is more informative to compare the training times, which are shown in the rightmost column of Table 4.3. We did not include the number of epochs for the "GMM + ANN" method, as the training procedure was radically different in that case. For our 28-hour dataset, 48 epochs were required by the four iterations of iterative CE flat start strategy, while MMI required only one-fourth of it. Although performing the forward-backward search adds a slight overhead to the MMI training process, it is clear that it was still much faster, even when the final CE re-alignment step was also involved (MMI+CE).

Measuring the training times in CPU/GPU time gives even larger differences in favour of the MMI method (3 hours vs. 16 hours). The reason is that for iterative CE flat start training we used a mini-batch of 100 frames (which we found optimal previously [51]), while for MMI whole utterances (usually more than 1000 frames) were used to update the weights, and this allowed better parallelisation on the GPU.

In our view, two modifications are crucial for the speed and stability of the proposed algorithm. The first one is that we use only CI phone models without phone language model, so we can very quickly update the numerator and denominator lattices after the processing of each sentence. This continuous refinement of the frame-level soft

targets obviously leads to a faster convergence. The only study we know of, which does not perform the re-alignment of the frame-level targets immediately after a training iteration, is that of Bacchiani et al. [70]. Their study focuses on describing their massively parallelised online neural network optimisation system, where a separate thread is responsible for the alignment of the phonetic targets, while DNN training is performed by the client machines. Besides the fact that in their model there is no guarantee that the alignment of phonetic targets are up-to-date, it is easy to see that their architecture is quite different from a standard DNN training architecture, making their techniques pretty hard to adapt. In contrast, our slight modifications can be applied relatively easily.

As regards stability, a known drawback of sequence training methods is that the same process is responsible both for aligning and training the DNN, which often leads to the "run-away silence model" issue [46]. That is, after a few iterations, only one model (usually the silence model) dominates most parts of the utterances, which is even reinforced with the next training step. To prevent the occurrence of this phenomenon, we monitored the error rate on a hold-out set during training. If the error increased after an iteration, we restored the weights of the network to their previous values and the learning rate was halved. In our experience, restoring the weights to their previous values and continuing the training using a lower learning rate can successfully handle this issue.

4.7 Word-Level Error Analysis of a Hungarian Automatic Speech Recognizer

Next, we will take a closer look at the typical word-level errors of our best GMM-free Hungarian speech recognition system. To achieve this, we selected one hundred utterances from the test set and the errors produced by the ASR system were manually annotated, then analysed. The word-level error rate in Automatic Speech Recognition (ASR) is traditionally measured by a metric based on edit distance, which relies on the exact match of word forms. Like most common techniques in ASR, this approach works well with the English language, but as we will see, for other languages such as agglutinative ones (like the Hungarian language) it may be suboptimal.

4.7.1 Analysing the Errors

To analyse the error types, we manually compared the ASR output and the correct transcription for a subset of the test set. First we automatically located the errors in the ASR output, and displayed them in a form along with one neighbouring word on

each side to provide a context for the human annotators. Error categories were set up by linguists, and each error occurrence was categorised manually by two human annotators.

Errors were first categorised based on linguistic criteria. For instance, when the only difference between the gold standard text and the output of the ASR system was just a space, causing only a slight change in meaning (if any) we regarded this as a *Compounding* error. Two examples of this are:

- *a két százmilliárdos tétel* [ɔ ke:t sa:zmillia:rdoʃ te:tɛl] (the two one.hundred.billion item) "the two items worth one hundred billions" vs. *a kétszáz milliárdos tétel* [ɔ ke:tsa:z millia:rdoʃ te:tɛl] (the two.hundred.billion item) "the item worth two hundred billions";
- *az exportdinamikája is* [ɔzɛksportdinɔmika:jɔ iʃ] (the export.dynamics-3SGPOSS too) "its export dynamics too" vs. *az export dinamikája is* [ɔz ɛksport dinɔmika:jɔ iʃ] (the export dynamics-3SGPOSS too) "the dynamics of the export too").

Another frequent error was that a sound was followed by another one of the same quality, which was treated as a long phoneme by the system. We created two subtypes for this category, one for the consonants (*Consec. consonants*), and the other for the vowels. Since the most common source of error with vowels was that a word ending in *-a* was followed by the definite article *a*, we called this type the *Two "a" sounds*. (e.g. *mondja bankszövetség* [monɟɔ ɔ bɔnksøvɛtʃe:g] (say-3SGOBJ bank.federation) "bank federation says" vs. *mondja a bankszövetség* [monɟɔ bɔnksøvɛtʃe:g] (say-3SGOBJ the bank.federation) "the bank federation says").

In many cases, the stem of the word was correctly recognised but its suffixes were not (*Incorrect suffix*): either the inflectional suffix was missing (e.g. a possessive suffix in *Mezőtúr polgármester* [mezø:tu:r polga:rmɛʃtɛr] (Mezőtúr mayor) "Mezőtúr mayor" vs. *Mezőtúr polgármestere* [mezø:tu:r polga:rmɛʃtɛrɛ] (Mezőtúr mayor-3SGPOSS) "the mayor of Mezőtúr"), or an incorrect one was assigned to the word (present vs. past tense in *szétdarabolják* [se:ddɔrɔboj:a:k] (cut.into.pieces-3PLOBJ) "they are cutting it into pieces" vs. *szétdarabolták* [se:ddɔrɔbolta:k] (cut.into.pieces-PAST-3PLOBJ) "they were cutting it into pieces").

In other cases, one word was absent from the ASR output (*Omitted word* category) (*terén erősítik* [tɛre:n ɛrø:ʃi:tik] (aspect-3SGPOSS-SUP improve-3PLOBJ) "they improve this in this aspect" vs. *terén ha erősítik* [tɛre:n hɔ ɛrø:ʃi:tik] (aspect-3SGPOSS-SUP if improve-3PLOBJ) "if they improve this in this aspect").

In two special cases the ASR output was correct, but it differed from the transcription. First, the *gold standard* may have contained an error (*a szennyezetett víztől* [ɔ sɛɲ:ɛzɛtɛtt vi:stø:l] (the polluted-TYPO water-ABL) "from the polluted water" vs. *a*

Error type	NE	NUM	OOV	Annot.	Total
Compounding	3	25	25	14	61
Two "a" sounds	0	0	0	0	11
Be/de change	0	0	9	0	9
Consec. consonants	0	0	0	0	5
Gold standard	0	0	1	1	1
Spelling	7	0	6	0	7
Is/és change	0	0	0	0	19
Omitted word	0	0	0	0	12
Incorrect suffix	7	3	20	0	91
Other error	96	6	114	0	185
Total	113	34	175	15	401

Table 4.4: Total number of each error type and each annotated word type

szennyezett víztől [ɔ sɛɲ:ɛzɛtt vi:stø:l] (the polluted water-ABL) "from the polluted water"). Second, the errors in the *Spelling* category were caused by the fact that the linguistic principles behind the creation of the transcript had some special aspects. For instance, named entities with irregular pronunciation were encoded according to the Hungarian orthographical norms but the ASR system provided the original spelling for them (*Magyar Helszinki Bizottság* [mɔɟɔr hɛlsinki bizottʃa:g] "Hungarian Helsinki Committee" vs. *Magyar Helsinki Bizottság*, the correct spelling, where the digraph *sz* denotes the phoneme [s]).

We found two other very common error types, both types being caused by replacing a word with a similar sounding one. In the case of the first category (*be/de change*), the ASR system replaced the word *be* [bɛ] "in" with the word *de*[dɛ] "but". The second category (*Is/és change*) contained errors when the two word *is* [iʃ] "too" and *és* [e:s] "and" were interchanged.

Apart from the error type categories, we also examined which error types were related to certain word types. We examined four word categories; namely the named entities (*NE*), the numbers (*NUM*), the out-of-vocabulary words(*OOV*) and words with annotation errors (*Annot.*). If any of the words in the local context of the actual error belonged to the given word category (e.g. one of the three words was *OOV*), we marked the given error occurrence as one related to the given word category.

Firstly, we examined whether the correct transcript contained a named entity (e.g. *Balogh* [bɔlog] (a Hungarian surname), *Fidesz* [fidɛs] (the name of a political party), *tálibok* [ta:libok] "Taliban"). Secondly, we checked whether it contained any numerals (e.g. *ezeréves* [ɛzɛre:vɛʃ] "a thousand years old", *kétmilliós* [ke:tmillio:ʃ]

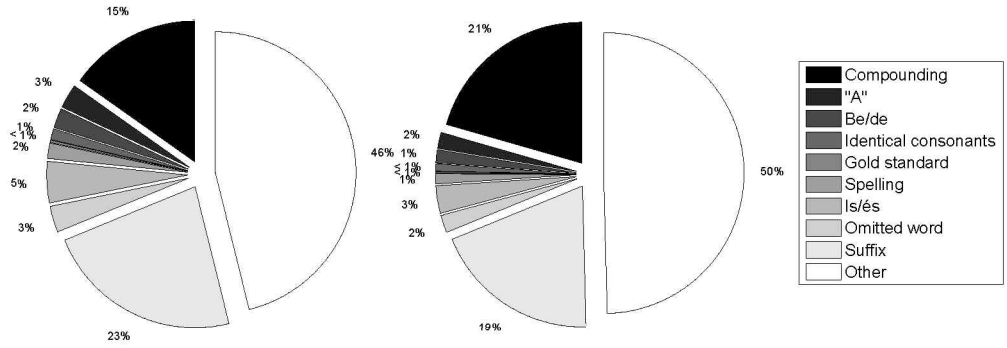

Figure 4.10: *Distribution of errors among the error categories, expressed in error (left) and word error (right) percentages.*

"(worth) two million", *ezerkilencszázötvenhatos* [ɛzɛrkilɛntssa:zøtvɛnhɔtoʃ] "of/from 1956"). Thirdly, we checked to see if any of the word forms was OOV. Lastly, we examined whether the transcript was correct, or if it contained some error (e.g. compounding error or typo).

Note that in the above approach, an error may affect several consecutive word occurrences, which are treated as one error instance. Naturally, as we use WER to measure the word-level error, these errors influence the final WER more than those which only affects one word. Furthermore, as each word type was treated independently of the others, in theory an error occurrence can be related to multiple word categories.

4.7.2 Results of the analysis

Figure 4.10 shows the distribution of error categories, expressed in terms of the ratio of errors and the ratio of word errors. The distribution of the error categories and the given word categories can be seen in tables 4.4 and 4.5. We can see that only slightly more that half of the error occurrences can be assigned to one of the meaningful categories, while about 46% of them fell into the "Other error" category. When measured in word errors, "Other errors" represent a slightly larger part – almost 50%; this can be traced back to the fact that for certain error types (e.g. be/de change, is/és change, two consecutive "a" sounds, omitted word, difference in spelling), one error occurrence typically affects only one word, while on average this value is around 1.5. At the same time, the ratio of compounding errors increased, as this error type leads to at least two word errors for each error occurrence.

Evidently, among the errors affecting named entities, a very common error type was that of spelling differences, and many errors were caused by incorrect suffices. This is quite logical, as the named entities appear quite rarely in the training text corpus, and their inflected forms are even less frequent. Still, most errors related to named entities

Error type	NE	NUM	OOV	Annot.	Total
Compounding	6	52	52	28	124
Two "a" sounds	0	0	0	0	11
Be/de change	0	0	9	0	9
Consec. consonants	0	0	0	0	5
Gold standard	0	0	1	1	1
Spelling	7	0	6	0	7
Is/és change	0	0	0	0	19
Omitted word	0	0	0	0	12
Incorrect suffix	11	5	32	0	116
Other error	157	11	188	0	299
Total	181	68	288	29	603

Table 4.5: Total number of each word error type and each annotated word type

fell into the error category of Other.

Error occurrences related to numerals mostly belonged to the compounding error category. A straightforward explanation would be that there are just too many (numeral) word forms possible which cannot be listed in the vocabulary, but surprisingly out of the 25 cases only 5 were OOV ones at the same time. The high frequency of compounding errors for numerals was probably because the language model allowed *both* versions (e.g. for the word *kétszázharmincezer* [ke:tsa:zhɔrmintsɛzɛr] "two hundred thirty thousand" both the word *kétszázharminc* [ke:tsa:zhɔrmints] "two hundred thirty" and the word *ezer* [ɛzɛr] "thousand" were present in the vocabulary). Interestingly, in 11 cases the compounding errors related to numerals were annotation errors at the same time.

Examining the error categories related to OOV words, compounding errors and using incorrect suffices altogether formed only one-fourth of the error occurrences, while the vast majority of these errors belonged to the category Other. The reason for this is probably that for these two kinds of errors at least a variation of the OOV word with a different suffix has to be present in the vocabulary. Annotation errors usually led to compounding errors, and in one case there was a typo in the transcription (*szennyezetett* instead of *szennyezett* "polluted").

Focusing on the error categories we can see that, for a large portion of compounding errors, some of the marked word types also occur; these form roughly 80% of compounding errors. The errors "be/de change" are always OOV errors, simply because the word "be" was missing from the vocabulary. Spelling errors affect only named entities, but it is surprising that in one case it was *not* an OOV error. The reason for

Category	NE	NUM	OOV	Annot.	Total
Named entity	113	0	99	0	113
Numeral	0	34	10	11	34
OOV	99	10	175	1	175
Annotation	0	11	1	15	15
Total	113	34	175	15	216

Table 4.6: Total number of errors concerning each annotated type and their combinations

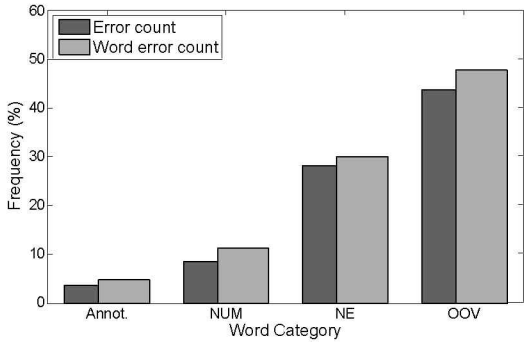

Figure 4.11: *Frequency of word categories, expressed in affected error occurrences and word errors.*

this is that besides the most common word form *Attilának* (Attila-DAT) "for Attila", an alternative form *Atillának* was also present in the vocabulary.

More than half of the error occurrences classed as *Other error* contained a named entity, and two-thirds of them had at least one OOV word. Examining all the errors we can see a similarly high ratio for these two word types; overall, 46% of the error occurrences contained at least one of the examined word types, although these gave 60% of the WER.

Figure 4.11 shows what proportion of the given word types were present in the error occurrences and word errors. As expected, a large part (almost 50%) of the errors were OOV; yet, there were many named entities (28-30%) and numerals (8-11%) present in the error occurrences as well. Tables 4.6 and 4.7 show the co-occurrence of the marked word categories. (Evidently, diagonal elements are the same as those in the Total row and column.) It can be seen that the vast majority (87%) of the error occurrences containing named entities are OOV errors as well; evidently, this ratio is much smaller (53%) than the other way around, as many other word forms (e.g. suffixed forms) may be frequently missing from the vocabulary. Roughly one-third of the error occurrences of numerals are also OOV or annotation errors. Also, notice how frequent the numerals are among annotation errors. This is probably due to the

Category	NE	NUM	OOV	Annot.	Total
Named entity	181	0	161	0	181
Numeral	0	68	22	22	68
OOV	161	22	288	1	288
Annotation	0	22	1	29	29
Total	181	68	288	29	360

Table 4.7: Total number of word errors concerning each annotated type and their combinations

complicated spelling of Hungarian numerals (as the words with dashes were split in the transcriptions, hence hyphenation errors appear as compounding ones).

Overall, a great amount of recognition errors simply represented a mismatch between the vocabulary and the transcriptions. Using the orthographic transcription for proper names helped us when creating the phonetic transcripts (and thus, in training the acoustic model), but these words were present in the vocabulary using a different spelling. Furthermore, there may have been a mismatch between the `origo` corpus that was used to build the language model, and the pronunciation dictionary (the Hungarian Pronunciation Dictionary), which led to a number of abbreviations (mostly names of political parties, being present quite frequently in broadcast news) missing from the vocabulary. And, for some mysterious reason, some common words (e.g. *be* "in", *legalább* "at least") were missing from the vocabulary.

Nevertheless, these errors might be responsible for at most 10% of WER, since 90% came from compounding errors, those of incorrect suffix, and of course the "Other" category. From this, perhaps the more interesting case is the high number of compounding errors, especially in the case of numerals, where the language model usually allows both versions. In such cases the ASR output is "practically" correct, so it can be read and understood very well, containing "only" some spelling error. This phenomenon is not a frequent one in English ASR, but as we have seen, in Hungarian (and probably for several other languages) it affects the WER to a notable extent. Of course, word sequences containing compounding errors cannot be regarded as correct ones; still, it would be sensible to treat them as less serious mistakes instead of omitting a word with an entirely different meaning. What WER does in practice, however, happens to be the opposite: compounding errors, by their nature, result in at least two word errors (e.g. one substitution and one insertion). In our opinion this highlights a language-dependent weakness of the de-facto standard WER metric.

4.8 Summary

In this chapter, I introduced a GMM-free method to train DNN based speech recognizers. In the previous chapter I proposed several modifications to the standard MMI sequence training method, which made it possible to train randomly initialised CI DNNs without forced aligned labels. After the flat start step I also compared the performance of four state clustering approaches (including the KL-divergence-based one, which was proposed by us) to create context-dependent tied states for DNN acoustic models. What was common in the four approaches is that they utilized the output of a context-independent neural network as their input. The experimental results showed that replacing the decision criterion used during state clustering is also beneficial. The results indicated that, compared to the standard procedure of iterative CE DNN training and re-alignment, the MMI based one was not only able to produce better WER scores, but also achieved a significant reduction in training times. By also utilising several new DNN-based state-tying methods, the whole training procedure of context-dependent HMM/DNNs became GMM-free.

Furthermore, we also examined the word error types that are common in the output of a standard ASR system built for the Hungarian language. For this, we collected the word errors and their local context, then we manually categorised and analysed them. We found that a large amount of word errors can be traced to OOV word forms, which is just what we expected. Nevertheless, compounding errors were surprisingly common. We found that the main reason for this is that the language model allows both word forms, and the acoustic model simply cannot decide which form is the correct one (as both solutions have the same phonetic transcript). This kind of error is judged to be a minor one by human readers, yet WER, which is based on the concept of exact word matching, treats these errors as more serious ones than substituting just one word with a completely different meaning. We found this issue quite common in Hungarian ASR, hence in the future we would like to have a new metric to measure the accuracy of Hungarian ASR systems.

In this chapter, the author regards the following as his main contributions:

- The introduction of a novel KL-divergence based state-tying algorithm;
- The application of the MMI-based training for the flat start training of CI DNNs;
- An experimental comparison of multiple GMM-free ASR soulutions.

The methods and results of this chapter were published in [51, 50, 71].

Chapter 5

Training Context-Dependent DNN Acoustic Models using Probabilistic Sampling

After exploring possible improvements in the flat start and state clustering phase, we now turn our attention to the CD training phase. In current HMM/DNN speech recognition systems, the purpose of the DNN component is to estimate the posterior probabilities of tied triphone states. In most cases the distribution of these states is uneven, meaning that we have a markedly different number of training samples for the various states. This imbalance in the training data is a source of suboptimality for most machine learning algorithms, and DNNs are no exception. A straightforward solution is to re-sample the data, either by upsampling the rarer classes or by dowsampling the more common classes.

In this chapter, we experiment with the so-called probabilistic sampling method that applies downsampling and upsampling at the same time, to improve the accuracy of CD acoustic models. For this, we define a new class distribution for the training data, which is a linear combination of the original and the uniform class distributions. As an extension to previous studies, we also propose a new method to re-estimate the class priors, which is required to reduce the mismatch between the training and the test data distributions introduced by re-sampling.

Using probabilistic sampling and the proposed modification we achieved relative word error rate reductions of 5% and 6% on the TED-LIUM and on the AMI corpora, respectively. We will also show that this re-sampling method can improve our GMM-free system.

5.1 Introduction

The imbalance in the class distribution poses a significant challenge to most machine learning algorithms [72], and DNNs are no exception. It is known that neural networks tend to become biased towards classes with more training examples, underestimating the posterior probabilities of the rarer classes [73]. Class imbalance is a typical problem in detection tasks, where usually only a small percentage of the training samples belong to the positive class [74]. The situation is even more difficult when the total amount of training data is already very low in itself.

In this chapter, we focus on the effect of class imbalance on the training of CD DNN acoustic models. At first glance, class imbalance is not an issue in speech recognition, as the frequency of the phones is quite balanced, and we have tremendous amounts of training data compared to some other machine learning tasks. However, we typically use context dependent (CD) phone models, and the number of tied states is allowed to increase when the size of the training corpus increases. We will show that the distribution of these CD target labels is far from uniform, meaning that many of the training samples belong to only a few classes, while many of the CD state classes are represented by just a few examples. While one would think that this causes problems only in low-resource scenarios, our experiments will show that the technique we propose may significantly improve the recognition results even in the case of fair-sized corpora.

The problem of class imbalance is typically tackled by applying re-sampling algorithms to the training data. In the simplest approach, the class-balance of the data is achieved by either reducing the number of the examples of the most common classes (*downsampling*) [75] or by presenting the rare examples more frequently (*upsampling*). In this chapter, we describe a more sophisticated algorithm called probabilistic sampling [76]. Probabilistic sampling offers a way of downsampling and upsampling at the same time by applying a two-step sampling process. For this, we define a new probability distribution over the classes, which determines how frequently the classes are chosen during re-sampling. The first step of the sampling process chooses a class based on this distribution. For the second step, a sample from the training vectors of the chosen class is selected following a uniform distribution. A simple solution to create a probability distribution over the classes is to take the linear combination of the original class distribution and the uniform distribution. This will result in a re-sampling process that has one free parameter, the weight λ of this linear interpolation. With $\lambda = 0$, we retain the original class distribution, while $\lambda = 1$ results in a uniform class sampling.

Tóth and Kocsor applied the probabilistic sampling method to a very small speech recognition task in 2005 in the framework of HMM/ANN hybrids, and they reported

improvements in the results [77]. As they worked only with monophone class labels, the main problem they tried to handle by probabilistic sampling was data scarcity. In 2015, Song et al. applied probabilistic sampling in the training of DNN acoustic models with context-dependent targets, and they obtained a significant reduction in the word error rate [78]. However, they performed their experiments on a low-resource task, using a corpus of only 4.5 hours of speech.

When discussing re-sampling methods in the framework of speech recognition, we should also mention the in-depth study of García-Moral et al., who applied a simple downsampling approach by discarding examples belonging to the more common classes. Although this made the ANN training process much faster, they experienced a slight drop in the accuracy scores [75]. Lastly, we should mention that over the past few years we have successfully used probabilistic sampling in detection-oriented paralinguistic tasks such as detecting the intensity of cognitive and physical load [74, 79, 80].

In Chapter 1, we mentioned the classic mathematical formulation of HMM/DNN hybrids states. To put it simply, the neural network outputs estimate the posterior distribution of the training labels, and the they can be incorporated in the HMM framework after a division by the class priors [20]. When probabilistic sampling is applied with uniform class sampling, Tóth and Kocsor [77] proved that there was no need to divide by the priors, as the network will approximate the class-conditional probabilities within a scaling factor.

Unfortunately, neither the authors of [77] nor [78] addressed the problem of intermediate distributions; that is, when the interpolation factor λ is between 0 and 1. García-Moral emphasizes that in such cases the posterior estimates require a proper scaling [75] after re-sampling the training data. To achieve this, here we propose to re-estimate the priors from the re-sampled training data, and divide the DNN outputs by these adjusted priors. Besides examining the effect of scaling by the various estimates of the class priors, we will also compare two different strategies for the random selection of the samples within a given class.

Our experiments show that with the proposed minor modifications probabilistic sampling can be used to improve the results of training CD DNN acoustic models, even in cases where large amounts of data are available. In the experiments we evaluated our method on the publicly available TED-LIUM corpus (release 1), which contains 118 hours of training data [81], and the public AMI corpus, which has a training set of 100 hours [82]. With the best λ we managed to achieve relative word error rate reductions between 5% and 6% on these corpora.

5.2 Probabilistic Sampling

The class distribution of CD state labels is a heavy-tailed distribution, meaning that the number of examples for each state differs significantly. Figure 5.1 shows the empirical distribution of the CD states on a logarithmic scale for the TED-LIUM corpus (the CD states were obtained using the Kaldi recipe [83]). As can be seen, a subset of the classes is significantly over- and under-represented, which might bias the DNN to favour certain classes and neglect some others. As a result, it generates imprecise posterior estimates for these classes, which usually leads to a higher word error rate (WER). One possible way to avoid this is to artificially balance the class distribution

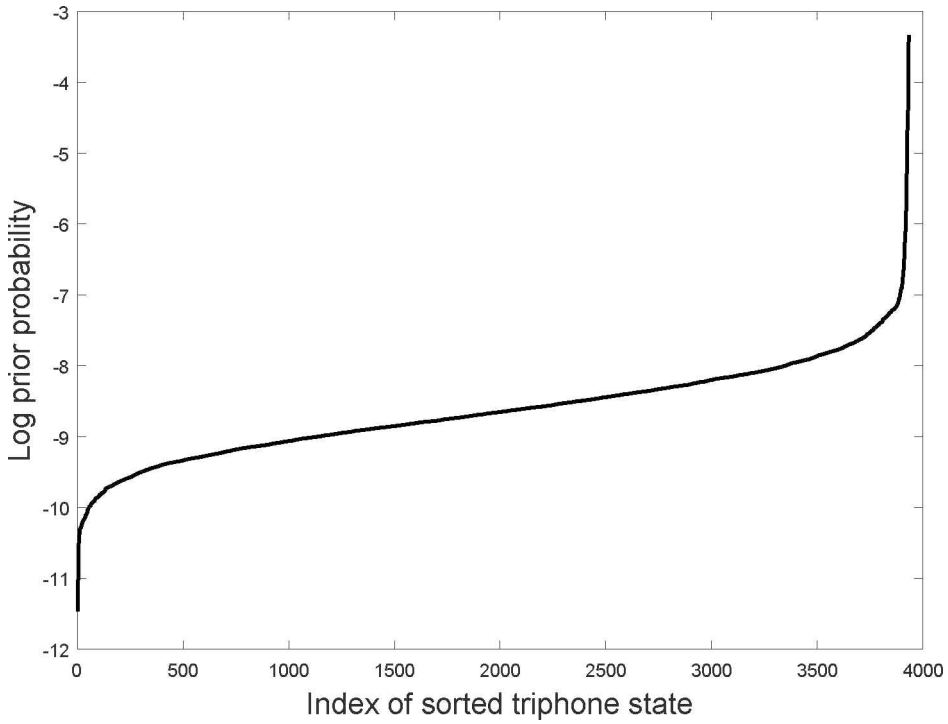

Figure 5.1: The distribution of tied CD states on a logarithmic scale in descending order (TED-LIUM corpus, Kaldi recipe)

by re-sampling the training set. Usually, we have no way of generating additional samples from a rare class, so balancing can be achieved by either reducing the number of examples belonging to the most common classes (downsampling) or by presenting the rare examples more frequently (upsampling).

Probabilistic sampling offers a third option by combining the two previous sampling approaches [76]. It applies a simple two-step sampling scheme; namely, first we select a class, then we pick a training sample belonging to this class. The first step requires us to assign a probability to each class, which determines how frequently it is selected. Here, we will use the following formula to define the sampling probability of the classes:

$$P(c_k) = \lambda \frac{1}{K} + (1 - \lambda) Prior(c_k), \tag{5.1}$$

where $Prior(c_k)$ is the prior probability of class c_k, K is the number of classes and $\lambda \in [0, 1]$ is a parameter. For $\lambda = 1$, we get a uniform distribution over the classes; and with $\lambda = 0$ we retain the original class distribution. Using intermediate λ values leads to a linear combination of these two distributions.

5.2.1 Selecting samples within the classes

Having chosen a class based on Eq. (5.1), we need to select a sample belonging to that class. During re-sampling our main goal is to modify the class distribution of the training data and leave the distribution of the training examples belonging to the same class unchanged (uniform). The simplest way to do this is to pick a random training vector within the class. However, as we perform only a few iterations on the training data, this strategy could have an undesired side-effect that it could change the distribution of the examples within the same class. The reason for this is that for some classes the re-sampling method may present the training vectors to the DNN unevenly, meaning that some examples might not be selected at all during the whole training process. We propose a very simple solution to remedy the problem. First, we define a random ordering of the examples belonging to the given class. Then, during training, we always select the next sample according to this ordering. This strategy ensures that the examples of the given class are presented evenly.

5.2.2 Adjusting the prior probability estimates

The standard practice for HMM/ANN hybrids is to divide the outputs of the DNN acoustic model by the class priors, in order to convert the posterior estimates into likelihood estimates. When applying probabilistic sampling, in theory, the division by the priors is required when $\lambda = 0$ (there is no re-sampling), and there is no need to divide with the priors when $\lambda = 1$ (uniform class sampling). The key theoretical question here is what to do in the intermediate cases ($0 < \lambda < 1$). Lacking theoretical results, Tóth and Song performed their evaluations by dividing the posterior estimates by the class priors or by using the neural network outputs directly, and found the optimal λ value experimentally [77, 78]. Here, we argue that the re-sampling of the training database requires us to properly adjust the prior probabilities. The reason is that by balancing the data we create a mismatch between the distribution of the training and the test sets. A simple and intuitive solution for the adjustment is to apply the class selection probabilities calculated using Equation (5.1) as class prior estimates.

This way, we can ensure that the adjusted priors estimate the class distribution of the re-sampled training data. In our experiments we evaluate our models with both the original and the adjusted prior estimates to empirically justify the significance of this adjustment.

5.3 Experimental Setup

Two large English speech databases were used to train the CD DNNs, namely the TED-LIUM and AMI corpus. The TED-LIUM corpus [81] is composed of 774 TED talks, containing 118 hours of speech overall: 82 hours of male and 36 hours of female speech. All recordings and their closed captions in this corpus were extracted from the TED website. The training data was automatically transcribed and only the development and test sets were transcribed manually (for more details, see [81]). As training targets we used 3933 CD labels, and the class distribution can be seen in Figure 5.1. We evaluated the trained DNN-based acoustic models using a 3-gram and a 4-gram language model as well.

AMI is a multi-modal corpus, which contains recordings of meetings [82]. All participants of the meetings speak in English, but only some of them are native English speakers, which leads to a high degree of variability in speech patterns. Here we used only the audio part of the corpus, specifically the recordings captured with the independent headset microphone (IHM). Following the Kaldi [83] recipe, the DNNs predicted the posterior scores of 3973 CD states, which had a similar class distribution to that of the TED-LIUM corpus.

We also used the Hungarian Broadcast News corpus [8]. For this database we made use of the best CD system (MMI-CE + KL) from the previous chapter to show that probabilistic sampling could also improve a GMM-free recogniser.

The acoustic model in our experiments was a DNN with 5 hidden layers, each containing 1000 rectified neurons [14], while we applied the softmax activation function in the output layer. The DNNs were trained using frame aligned labels and no sequence training was applied. As input, we used the 40-dimensional fMLLR features in the case of the TED-LIUM and AMI databases. We extracted the features by following the Kaldi recipe and the DNNs were trained on 11 neighbouring frames. The Hungarian recogniser used 15 neighbouring frames of FBank features as input, just like that described in the previous chapter. To train the DNNs we used our own deep learning framework [8], while the decoding and evaluation of the English corpora was performed with Kaldi.

To determine the effectiveness of the probabilistic sampling method, we tested λ values between 0.1 and 1.0 with a step size of 0.1. For each training iteration, we

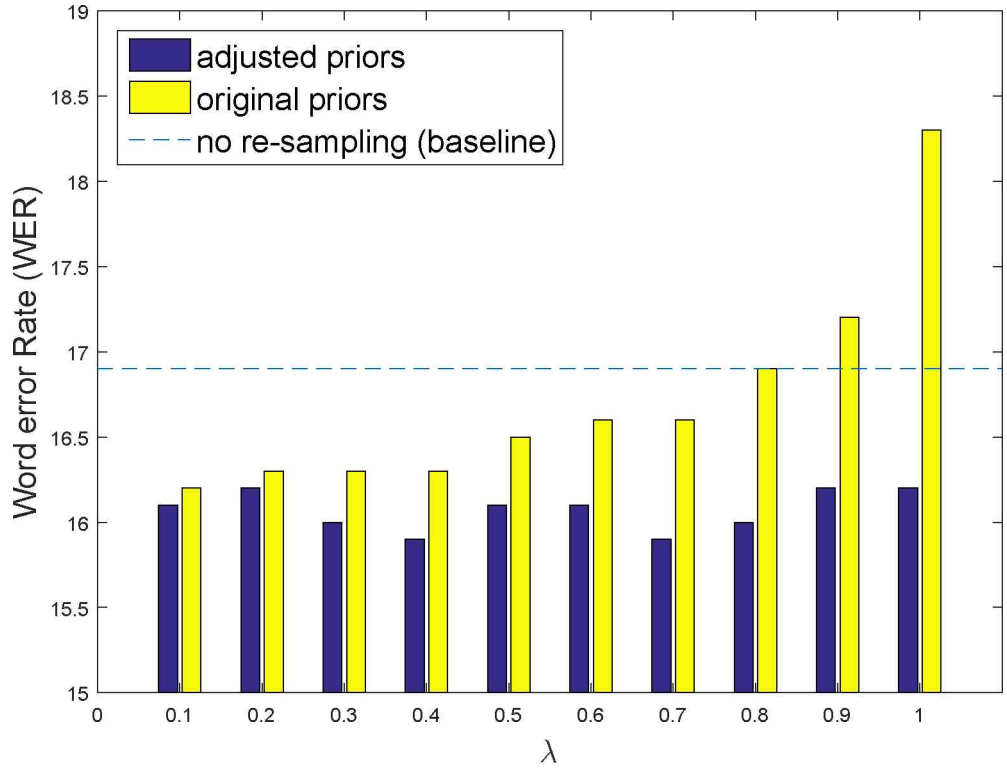

Figure 5.2: Word error rates got for the development set of the TED-LIUM corpus using a 3-gram language model and probabilistic sampling.

re-sampled the same amount of training vectors as that in the original data. All DNN models were evaluated with the division by the original or the adjusted priors to see the effectiveness of the adjustment.

5.4 Results

First, we compared the two sample selection approaches described in Section 5.2.1. We found that selecting training vectors within the classes with uniform sampling led to suboptimal models for some rare triphones. In our preliminary experiments we observed that this strategy led to a 1% increase in the frame error rates compared to that for the other selection method, and also resulted in a higher WER. As the selection method that applies a random ordering performed consistently better, we decided to apply it in all our experiments.

5.4.1 TED-LIUM

Figures 5.2 and 5.3 show the results we got with probabilistic sampling on the TED-LIUM corpus. Clearly, dividing the DNN outputs by the original priors gives worse

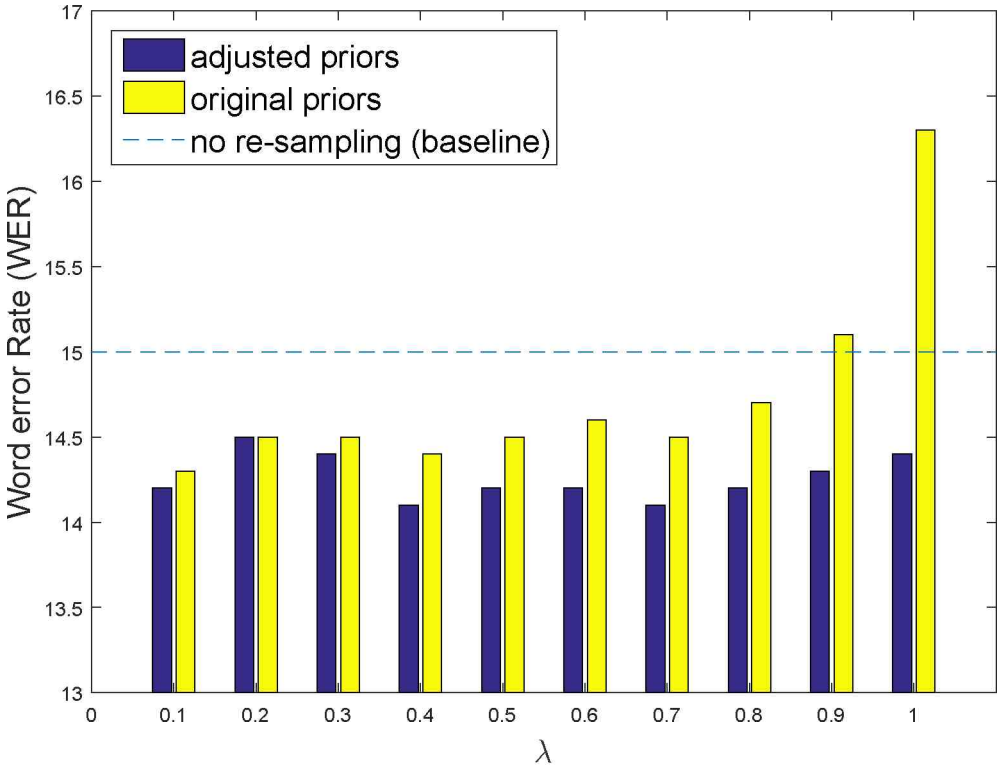

Figure 5.3: Word error rates got for the test set of the TED-LIUM corpus using a 3-gram language model and probabilistic sampling.

LM	Method	Dev WER		Test WER	
		original priors	adjusted priors	original priors	adjusted priors
3-gram	baseline	16.9	–	15.0	–
	$\lambda = 0.4$	16.3	15.9	14.4	14.1
4-gram	baseline	15.2	–	13.7	–
	$\lambda = 0.4$	14.7	14.4	13.0	12.9

Table 5.1: Best word error rates got with and without probabilistic sampling and dividing by the original and the adjusted priors.

results as λ increases, and we found that small λ values (here 0.4) work best. For small λ values, i.e. when the original distribution remains dominant in the class distribution of the new training data, both prior estimation strategies performed similarly, but as we increase λ above 0.5, the mismatch between the training and test sets caused a significant drop in recognition accuracy (even below the baseline).

When we adjusted the priors, the models became more robust and we got better results than the baseline for all λ values. The best result on the development set was attained using the adjusted priors and $\lambda = 0.4$; this network achieved a 14.1% WER

on the test set, which means a 6% relative error reduction compared to the baseline.

Table 5.1 summarises the best results on the TED-LIUM database. As can be seen, probabilistic sampling always yielded better results and with the prior adjustment we managed to improve the performance further. Using the 4-gram language model produced similar results to those achieved with the 3-gram model. The optimal value for the re-sampling parameter was 0.4, just like when the 3-gram language model was used.

5.4.2 AMI

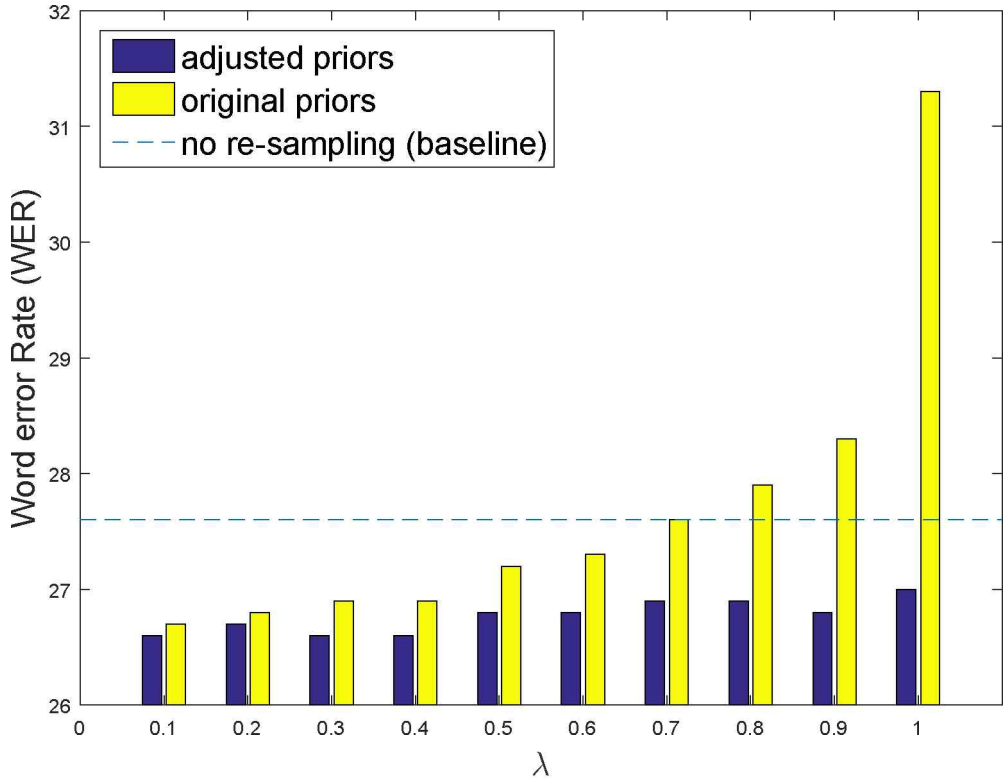

Figure 5.4: Word error rates got for the development set of the AMI corpus using probabilistic sampling.

On the AMI corpus the results follow a similar trend; the best results were achieved with the adjusted priors, and the division by the original priors resulted in a declining recognition accuracy for increasing λ. All DNNs trained with $\lambda \leq 0.7$ performed better than the baseline model both on the development and the test sets. The optimal value of λ was 0.1 when we divided by the original prior (26.7% WER on the development set and 27.4% on the test) and 0.1 or 0.4 when the adjusted priors were used. Both DNNs achieved a WER of 26.6% on the development and 27.3% on the test set. On the test set the best WER was 27.3%, which is far better than the baseline (28.6%), yielding

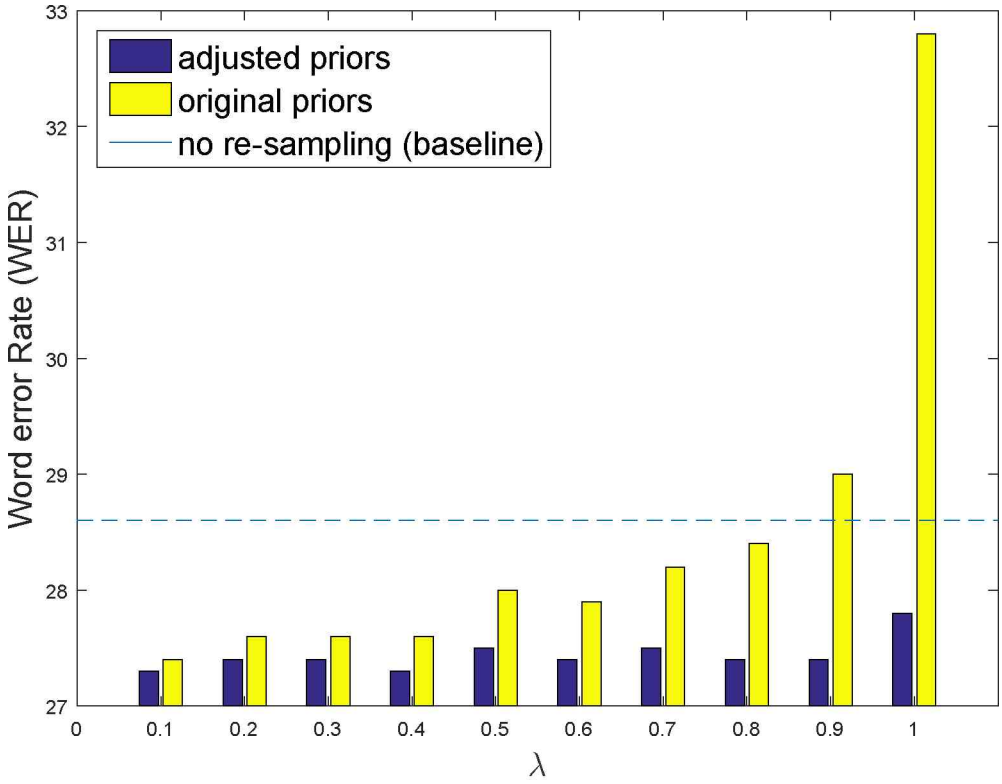

Figure 5.5: Word error rates got for the test set of the AMI corpus using probabilistic sampling.

a relative error reduction of 5%. We should mention that using uniform re-sampling with the original priors resulted in recognition results far below the baseline.

5.4.3 Improving GMM-free systems using probabilistic sampling

In the case of the Hungarian Broadcast News corpus we employed our previous best GMM-free CD system that made use of the MMI-based algorithm followed by one iteration of CE training for flat start. To create the CD targets the KL-divergence based clustering method was applied. The number of CD states was 1843, meaning that this corpus was less imbalanced than the other two; 22 hours of training data was available for 1843 classes, while for the English databases we had about 100 hours of data for roughly 4000 classes. Using $\lambda = 0.4$, the trained CD DNN gave a WER of 16.14% on the development set, which is better than the baseline 16.36%. However on the test set it achieved only a small improvement (15.79% vs 15.86%). The reason for this is probably the small amount of training data, we hypothesis that if the database had more speech data we would have seen improvements similar to the English corpora. To test this, we also applied probabilistic sampling to train the best GMM-free CD DNN

from the previous experiments using the Wall Street Journal (WSJ) corpus. The corpus has 81 hour of training data and the DNN had approximately 2400 output neurons. As a reminder, the best WERs achieved by us previously were 8.03% and 5.92% on the development and test sets, respectively. By applying probabilistic sampling, the new DNN managed to perform significantly better, yielding a WER of 7.6% on the development set and 5.44% on the test set. These results suggest that our hypothesis was correct and the sampling method works best if the amount of training data is sufficiently large.

5.4.4 Discussion

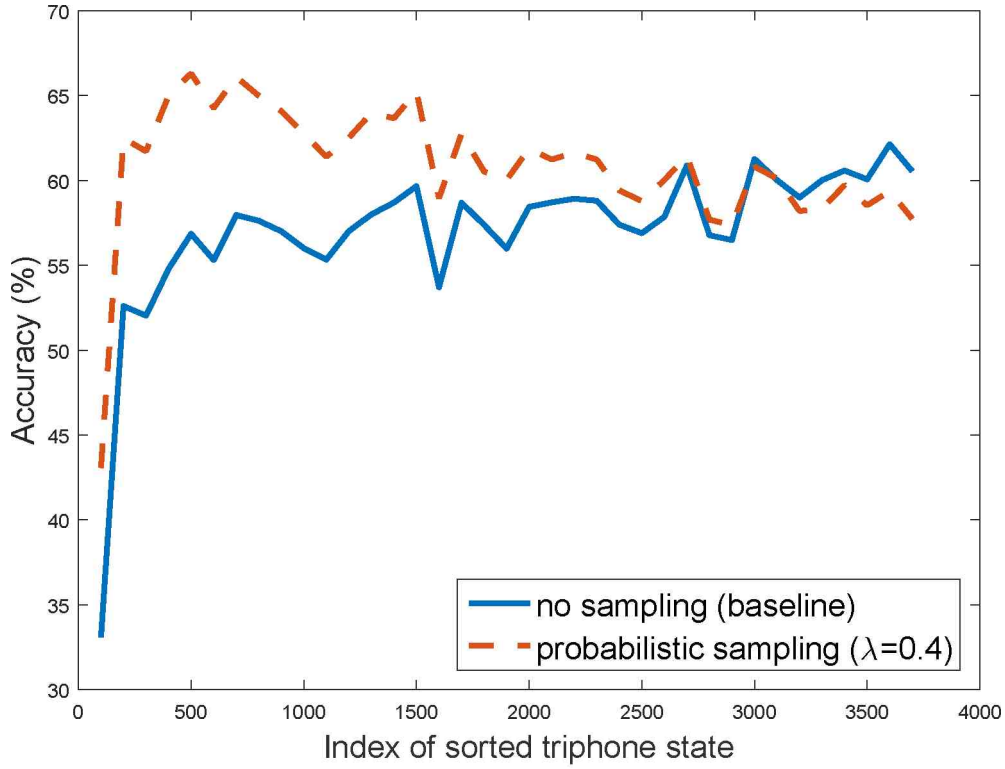

Figure 5.6: Averaged accuracy scores of sorted CD states obtained on the TED-LIUM development set with and without re-sampling.

To get an insight into why probabilistic sampling helps, we performed an analysis to learn how the accuracy of CD state classification varies as a function of state frequency. Figure 5.6 shows the average frame-level accuracy scores of the sorted CD states, and it compares the baseline method with the best model trained with re-sampling. The first thing to notice is that probabilistic sampling significantly improves the accuracy scores of the rare states (Index $\leq$ 1000), and even the frequent states are recognised more often. The downside of this improvement is the lower accuracy of those classes that

have the most training data. Interestingly, the accuracy of classes having an average amount of training data (middle part in the figure) also increased with probabilistic sampling; the reason is that they were less likely confused with the more frequent states.

As we saw, dividing the DNN outputs by the adjusted priors stabilized the results: for almost all λ values we got similar WE scores. If the original priors are used then a declining trend is observed as we move farther from the original distribution. The stability of this adjustment could be explained by the fact that it reduces the mismatch between the training and test data introduced by the re-sampling process.

5.5 Solving Paralinguistic Tasks using Probabilistic Sampling

For a long time the main focus of speech technology was Automatic Speech Recognition (ASR), but recently a new sub-area has emerged called computational paralinguistics. It seeks to extract and identify phenomena present in the audio signal other than the words uttered. The fact that since 2009 the Computational Paralinguistics ChallengE (ComParE) series takes place annually at the INTERSPEECH conference shows the importance of this new area. ComParE is an open challenge in the field of speech technology that deals with states and traits of speakers, as manifested in their speech. Every year, new highly relevant paralinguistic tasks are introduced in this competition series. Most of these tasks have only a limited amount of training data and a highly imbalanced class distribution. Luckily, the limited data is distributed among a few classes, so probabilistic sampling is applicable. We managed to apply DRNs, trained with probabilistic sampling and achieved good results in many of these challenges.

- In 2014, we created a system, whose goal was to detect the intensity of cognitive and physical load of the speaker [84]. Our DNN-based method consistently managed to outperform the baseline SVMs, yielding an unweighted average recall (UAR) of 63.05% on the Cognitive Load Challenge, and a UAR of 73.03% on the Physical Load Challenge [74].

- In 2016, we participated in the Deception Sub-Challenge [85] with a DRN that was trained using re-sampling [79]. The aim of this Sub-Challenge was the detection of deceit, using only the speech of the person in question. With DNNs alone, we managed to get a higher UAR value than the baseline by a mere 0.3%, but this is much less than the 3.6% improvement measured on the development

set. In our view this can be attributed to the limited amount of training data and a possible mismatch between the training and test data.

- In 2017, we experimented with the re-sampling method and applied it in the Addressee and Cold Sub-Challenge [80]. In the Addressee Sub-Challenge, one had to determine whether the adult speaks to a child or to another adult, and the Cold Sub-Challenge sought to separate healthy speakers from those who have a cold [86]. We should mention here that the baseline systems of these sub-challenges were very competitive (the fusion of three approaches). On the Addressee Sub-Challenge our approach yielded worse results than the baseline, but on the Cold Sub-Challenge it managed to significantly outperform the baseline. Furthermore, as our outstanding result proved to better than those of our competitors, we won this sub-challenge.

5.6 Summary

In this chapter, I demonstrated that CD DNN training can be improved by re-sampling the training data with probabilistic sampling. I also proposed a new method for re-estimating the class priors when using this sampling algorithm. The experimental results proved that this re-estimation is essential for remedying the mismatch between the training and the test data distributions introduced by the re-sampling step. These adjusted priors made the re-sampling method more robust, and the recognition results varied only slightly as the class distribution, with a bigger λ value was shifted towards a uniform distribution. Our experiments revealed that by using this modification, the recognition results dramatically improved, it gave relative error reductions between 5% and 6% on two fair-sized corpora (TED-LIUM and AMI).

In this chapter, the author regards the following as his main contributions:

- The use of probabilistic sampling during the training of CD DNNs;

- A new way to adjust the priors in the case of re-sampling;

The methods and results of this chapter were published in [87, 74, 79, 80].

Chapter 6

Summary

In this thesis, we proposed new HMM/DNN acoustic modelling techniques and evaluated them on large vocabulary speech recognition tasks. In Chapter 1 we briefly introduced the basic components of an automatic speech recognition system, such as the feature extractor, the HMM/DNN acoustic model and the language model. Here, we also described how neural networks work, and how they can be trained. In later chapters, we examined several training methods for HMM/DNNs and modified the algorithms it had inherited from the HMM/GMM system to better suit this new DNN-based model.

6.1 A Comparison of Deep Neural Network Training Methods for LVSR

The second chapter focused on comparing the performance of four DNN training algorithms. The first one is the original algorithm proposed by Hinton et al.[9], and the second one is called discriminative pre-training by Seide et al. [10]. Both of these methods apply a pre-training phase before they finetune the DNNs. Deep Rectifier Networks, our third approach differs greatly from the previous two in the sense that it modifies the activation of the hidden neurons instead of the training process. The fourth training algorithm that we examined is a regularization method called Dropout [11], which simply turns off neurons during training.

In our experiments we compared the recognition accuracies of these methods on a large vocabulary Hungarian recognition task. Our conclusion was that, although the four algorithms yielded quite similar recognition performances, rectifier networks achieved better accuracies and their training was considerably faster. Based on these facts in the rest of the thesis I just used rectifier networks.

6.2 Sequence Training Methods for Deep Rectifier Neural Networks in Speech Recognition

After determining our preferred choice of DNN, we turned our attention to the task of flat start training, which is the first step of training a speech recognition system. The goal of flat start is to get time-aligned context independent labels for the database. Our aim here was to compare two sequence training approaches that could be used to train randomly initialised DNNs without force-aligned labels. The first one was the Connectionist Temporal Classification (CTC) and the second one was the Maximum Mutual Information (MMI) method. Both of them were used to train DRNs. We proposed several modifications to the standard MMI method, which were essential to make it suitable for the flat start process.

In the experimental part, we evaluated the two methods on several phone recognition tasks. For all databases we tested, we found that the sequence training methods gave better results that those obtained with force-aligned training labels produced by an HMM/GMM system. From the experimental results, it was also clear that the MMI-based approach gave better results than the CTC-based one. Furthermore, DRNs trained with CTC could not produce forced-aligned labels. Based on these findings we concluded that MMI was the better algorithm for flat start training.

6.3 A GMM Free Training Method for Deep Neural Networks

Next, we adapted the state-tying algorithm with the goal of getting rid of its GMM dependency. The context-dependent states used to train DNNs are usually obtained using the standard tying algorithm, even though it is based on likelihoods of Gaussians, hence it is more appropriate for HMM/GMMs. Recently, however, several new refinements have been published which seek to adapt the state tying algorithm to the HMM/DNN hybrid architecture.

Some of the new methods change only the input of the clustering algorithm, feeding the output or the activations of the neurons in the last hidden layer to the clustering method while the whole state tying algorithm remains intact. Other studies proposed novel decision criteria as well for the clustering method, which better suit the new input provided by a DNN.

In this chapter, we proposed a KL-divergence-based approach. We evaluated it along with three other state-tying methods on the same LVCSR tasks, and compared their performance under the same circumstances. We combined them with the MMI-

based flat start method from the previous chapter, and showed that the whole training procedure of context-dependent HMM/DNNs can be carried out without using GMMs.

The experimental results confirmed that the MMI-based flat star approach is far better than the procedure of iterative CE DNN training and re-alignment. Furthermore, we saw that replacing the decision criterion used during state clustering is also beneficial for DNN training. Lastly, we examined the best Hungarian HMM/DNN system to see what type of errors are most common. For this, we collected the word errors and their local context, then we manually categorised and analysed them. Our conclusion was that a new metric is needed to measure the accuracy of Hungarian ASR systems, since the current one (WER) treats some errors more seriously than human readers do.

6.4 Training Context-Dependent DNN Acoustic Models using Probabilistic Sampling

In Chapter 5, we turned our attention to the CD training phase of the ASR system. In the current HMM/DNN speech recognition systems, the purpose of the DNN component is to estimate the posterior probabilities of tied triphone states. It is well known that the distribution of the CD states is uneven, meaning that we have a markedly different number of training samples for the various states. This imbalance in the training data is a source of suboptimality for most machine learning algorithms, and DNNs are no exception to this.

Here, we experimented with the so-called probabilistic sampling method that applies downsampling and upsampling at the same time, to improve the accuracy of CD acoustic models. This re-sampling method defines a new class distribution for the training data, which is a linear combination of the original and the uniform class distributions. As an extension to previous studies, we also proposed a new method to re-estimate the class priors, which is required to remedy the mismatch between the training and the test data distributions introduced by re-sampling.

Using probabilistic sampling we achieved relative word error rate reductions of 5% and 6%, respectively, on two fair-sized corpora (TED-LIUM and AMI). We also showed that this re-sampling method can improve our GMM-free system outlined in the previous chapter. Our experimental results strongly suggest that the re-estimation of the priors is essential to handle the mismatch between the training and the test data distributions introduced by the re-sampling step. These adjusted priors made the re-sampling method more robust, and the recognition results varied only slightly as the class distribution was shifted towards a uniform distribution.

6.5 Conclusions and future directions

In this thesis, we successfully adapted the standard methods of the HMM/GMM acoustic models to better suit the new HMM/DNN hybrid. We revised both the initial training phase (flat start) and the CD state-tying phase, and introduced new strictly DNN-based solutions to these problems. By combining these methods, we created a new training method that does not depend on GMMs at all. We also showed that the final training phase could be improved by employing a simple re-sampling method. On the Szeged Hungarian Broadcast News corpus, a traditional HMM/GMM gave a WER of 20.07%, the best DNN that relies on GMMs produced a WER of 16.59%; while our best GMM-free system managed to achieve a WER of 15.79%.

Naturally, many experiments have been left for the future, mainly due to lack of time or because they lay outside the scope of the present study. The following list presents some of the possible future research directions.

- For one, we should consider applying a new DNN type, namely the Convolutional Neural Network (CNN), since it has provided impressive results both in image processing and speech recognition.

- To extend the results of Chapter 3, it would be worth examining other sequence learning methods, such as minimum phone error (MPE) or state-level minimum Bayes risk (sMBR), and adept them so they are suitable for flat start training.

- It is worth investigating what would happen if we had more CD clusters in our GMM-free systems. The hypothesis here is that with more states we should get better results, of course, at the cost of the increased training and evaluation times.

- It would be interesting to learn how the CD DNNs trained with probabilistic sampling perform after a final sequence discriminative training phase, which is nowadays a common practice.

6.6 Key points of the Thesis

In the following a listing of the most important results of the dissertation is given. Table 6.6. summarises the relation between the theses and the corresponding publications.

I. The author compared the performance of four deep learning methods empirically; two of these methods were pre-training algorithms, the third one applied

	[26]	[49]	[50]	[51]	[71]	[87]	[74]	[79]	[80]
I.	•								
II/1.		•							
II/2.		•	•						
III/1.				•					
III/2.			•	•	•				
IV.						•	•	•	•

Table 6.1: Correspondence between the thesis points and the publications.

the rectifier activation function and the fourth was a regularisation technique called Dropout. The experiments were also carried out using a Hungarian speech corpus, and this study was among the first to apply a HMM/DNN system to Hungarian speech recognition. The results indicated that the new HMM/DNN systems can outperform the traditional HMM/GMM system significantly. The conclusion of the experiments was that, although the four algorithms yielded quite similar recognition performances, rectifier networks consistently produced the best results.

II/1. The CTC algorithm was originally proposed for the training of recurrent neural networks, but here the author showed that it can also be used to train conventional feed-forward networks. Using several corpora, deep rectifier networks were trained with the CTC method, in order to determine whether this approach was suitable for the flat start training step. The results led us to conclude that CTC can be used to train randomly initialised networks without time-aligned labels.

II/2. As a competitor, the MMI-based training algorithm was also examined. The author proposed several modifications to the standard MMI, to make it suitable for the task (flat start training). The experimental results showed that the modified MMI is a far superior alternative to CTC, for training randomly initialised networks without time-aligned labels.

III/1. The author created a new DNN-based state-tying method by changing the decision criterion used by the standard algorithm during the clustering step. Since this new state tying method uses posterior probability vectors produced by DNNs as input, KL-divergence seemed a logical choice for decision criterion. The experimental results also supported this view, as the new method markedly outperformed the original one.

III/2. By combining the MMI-based flat start training algorithm with the KL-divergence-based clustering method, the author built an ASR system that did not rely on GMMs. He compared this GMM-free solution with other recently proposed alternatives, and found that it was competitive with the other approaches used. Furthermore, the results demonstrated that the GMM-free systems were capable of producing better results than those that rely on GMMs.

IV. The author examined the probabilistic sampling method for the training of CD DNNs. He hypothesised that when the training data is re-sampled, the prior probability values need to be re-estimated. He justified this experimentally, and showed that re-sampling with adjusted priors greatly improves the performance of CD DNNs. This re-sampling algorithm was also applied with great success in several paralinguistic tasks.

Chapter 7

Summary in Hungarian

Ebben a dolgozatban az új, mély neuronhálós akusztikus modelleket vizsgáltuk és alkalmaztuk nagy szótáras beszédfelismerési feladatokban. Az első fejezetben röviden bemutattuk az automatikus beszédfelismerők alap komponenseit; a különböző jellemzőkinyerési módszereket, az új HMM/DNN hibridet alkalmazó akusztikus modellt és a nyelvi modellt. Szintén a bevezető fejezetben bemutattuk a mesterséges neuronhálók működését, illetve tanulási algoritmusukat. Az ezt követő fejezetekben megvizsgáltunk több mély neuronhálós tanítási módszert, majd megmutattuk, hogyan lehet a HMM/GMM modelltől örökölt algoritmusokat úgy módosítani, hogy azok jobban illeszkedjenek az új DNN alapú modellhez.

7.1 Mély neuronhálós tanítási módszerek összehasonlítása nagyszótáras beszédfelismerésben

A második fejezetben négy mély neuronhálós tanítási módszert hasonlítunk össze. Az első módszer a Hinton és társai által kidolgozott eredeti előtanító algoritmus [9], a második módszer pedig az úgynevezett diszkriminatív előtanítás, amelyet Seide és társai publikáltak [10]. Ezen két algoritmusban közös, hogy két fontos fázisból állnak; az előtanítás során inicializálják a neuronhálót, majd a második lépésben finomhangolják azt. A mély egyenirányított hálók, a harmadik módszer, amit megvizsgáltunk, jelentősen eltérnek a korábbiaktól, hiszen ebben az esetben nem a tanítási algoritmus módosul, hanem a rejtett neuronok aktivációs függvénye. A negyedik módszerként egy regularizációs technikát választottunk, az úgynevezett Dropout [11] algoritmust, melynek lényege, hogy tanítás során véletlenszerűen kikapcsolunk neuronokat a hálózatban. Ez a módszer nem egy önálló algoritmus, hanem más módszerekkel (bármelyik korábbival) kombinálva használható.

Kísérleteinkben ezen módszereket hasonlítottuk össze egy magyar nyelvű nagy

szótáras beszédfelismerési feladaton. Konklúzióként azt kaptuk, hogy mind a négy módszer elég hasonló eredményeket tudott elérni, de a legjobbnak az egyenirányított hálózatok bizonyultak, tekintve, hogy ezek érték el a legjobb felismerési pontosságokat és betanításuk is kevesebb időt igényelt. Ezen eredményekre alapozva, a dolgozatom további részében csak egyenirányított neuronhálókat alkalmaztam.

7.2 Mély egyenirányított neurális hálók tanítása szekvenciatanuló módszerekkel

Miután kiválasztottuk a legjobb mély tanítási módszert, a flat start nevű feladatra fordítottuk figyelmünket. Ezen feladat megoldása az első lépés minden beszédfelismerő rendszer létrehozása során. Ezen lépés lényege, hogy meghatározzuk a kontextusfüggetlen címkék időbeli illesztését. A hosszútávú célunk egy teljesen mély hálókon alapuló módszer kidolgozása volt, ezért ebben a fejezetben két szekvenciatanuló módszert hasonlítottunk össze, amelyek alkalmasnak tűntek a kezdeti kontextusfüggetlen modellek tanítására. A konnekciós temporális osztályozás (CTC) algoritmust vettettük össze a maximális kölcsönös információn (MMI) alapulóval. Mindkét vizsgált módszert mély egyenirányított hálók tanítására használtuk. Az alap MMI algoritmushoz több módosítást is javasoltunk, melyek lehetővé tették, hogy véletlenszerűen inicializált hálók tanítására használjuk ezt a módszert időben illesztett címkék nélkül.

A kísérleteink során különböző fonémafelismerési feladatokon hasonlítottuk össze a két módszert. Mindegyik adatbázis esetén azt találtuk, hogy a szekvenciatanuló módszerek jobban működtek mint a hagyományos rendszerek, amelyeket egy HMM/GMM által generált időben illesztett címkékkel tanítottunk. Az eredményekből az is egyértelműen kiderült, hogy az MMI módszer jobb eredményeket képes elérni mint a CTC algoritmus. A CTC algoritmus egy további hátránya, hogy a betanított hálók nem használhatók a címkék kényszerített illesztésére. Mindezeket figyelembe véve megállapítható, hogy az MMI módszer a legjobb választás a flat start lépés megoldására.

7.3 GMM-mentes mély neuronhálós beszédfelismerők

A 4. fejezetben az állapotkapcsolási algoritmust adaptáltuk, célunk a GMM függőségek eltávolítása volt. A környezetfüggő állapotokat általában a standard algoritmussal állítják elő, annak ellenére, hogy az algoritmus speciálisan a Gauss-görbék illeszkedését használja ki, így optimalitása egy mély hálós rendszerben megkérdőjelezhető. Az

utóbbi időben azonban több olyan állapotklaszterező algoritmust is publikáltak, amelyek megkísérlik a korábbi eljárást a mély neuronhálós modellezéshez igazítani.

Néhány új módszer csak a klaszterező algoritmus bemenetén változtat, azaz a klaszterezést a DNN kimenetén futtatják le, magát az algoritmust pedig egyáltalán nem módosítják. Más szerzők a bemenet kicserélésén túl a klaszterező eljárás döntési kritériumát is módosítják oly módon, hogy az jobban illeszkedjen a neuronhálós eloszlás-modellezéshez.

Ebben a fejezetben három különböző módszert hasonlítottunk össze a saját KL divergencián alapuló módszerünkkel, ugyan azon a nagy szótáras beszédfelismerési feladaton. Kombinálva ezen módszereket az előző fejezetben bemutatott MMI-alapú flat start módszerrel megmutattuk, hogy lehetséges HMM/DNN beszédfelismerőket tanítani GMM használata nélkül is.

A kísérleti eredményeink azt mutatták, hogy az MMI-alapú módszer sokkal jobban működik, mint a keresztentrópiás tanítást és újraillesztést iteráló módszer. Továbbá azt is láttuk, hogy célszerű a döntési kritériumot is lecserélni a klaszterező algoritmusban. Mindezeken túl azt is megvizsgáltuk, hogy a legjobb magyar beszédfelismerőnk milyen típusú hibákat vét leggyakrabban. Ehhez a teszthalmaz egy részén előforduló szószintű hibákat kigyűjtöttük, majd manuálisan kategorizáltuk és elemeztük. A vizsgálatok eredményeképpen megállapítottuk, hogy egy új hibametrikára lenne szükség magyar nyelvű beszédfelismerő rendszerek értékeléséhez, mivel a jelenleg használt metrika (WER) több hibát sokkal súlyosabbként kezel, mint az emberi annotátorok.

7.4 Kontextusfüggő mély neuronhálós akusztikus modellek tanítása valószínűségi mintavételezéssel

A 5. fejezetben a környezetfüggő akusztikus modellek tanítására fókuszáltunk. A manapság használatban lévő beszédfelismerőkben a DNN komponensek feladata, hogy állapotkapcsolt trifónok posterior valószínűségét becsüljék. A problémát az jelenti, hogy a címkék eloszlása nem egyenletes, így a gyakorlatban az egyes osztályokhoz tartozó tanítópéldák száma jelentősen eltér. A tanító adat egyenlőtlen eloszlása problémát jelent a legtöbb gépi tanuló algoritmusnak, ez alól a mély hálók sem kivételek.

A probléma megoldására a valószínűségi mintavételezés módszerét használtuk, amelynek előnye, hogy egyszerre alkalmazza az alul- és a felül-mintavételezést. Az adatbázis újramintavételezéséhez egy új osztályeloszlást definiál a módszer, ez az új eloszlás az eredeti és az egyenletes eloszlás lineáris kombinációjaként áll elő. A korábbi tanulmányokhoz képest mi a prior valószínűségek újraszámlálására is javasoltunk egy

módszert. Erre azért volt szükség, mert az adat újramintavételezése révén jelentősen eltért a tanító és a teszt adatbázis egymástól.

A valószínűségi mintavételezés segítségével 5% és 6% szószintű hibaarány redukciót sikerült elérnünk két nagy méretű adatbázison (TED-LIUM és AMI). Megmutattuk azt is, hogy ezzel a módszerrel a korábbi fejezetben bemutatott GMM-mentes rendszer is jobb eredményeket képes elérni. A kísérleti eredményeink alátámasztották azon sejtésünket is, hogy a prior valószínűségek újrabecslése kritikus az újramintavételezés miatt a tanító és teszt adat között fellépő különbség kezelése szempontjából. Ezek az újrabecsült priorok robusztusabbá tették a módszerünket, hatásukra a felismerési pontosságok csak csekély mértékben változtak, ahogy az egyenletes eloszlás felé mozgattuk az osztályok eloszlását a mintavételezés során.

7.5 Konklúzió és jövőbeli kutatási irányok

A dolgozatban bemutattuk, hogy a standard HMM/GMM rendszerhez kidolgozott módszerek hogyan adaptálhatóak az új HMM/DNN hibrid modellhez. Ehhez kidolgoztunk új, tisztán DNN alapú módszereket a kezdeti tanítási fázis (flat start) és az állapotkapcsolási lépés megoldására. Ezek összekapcsolásával sikeresen létrehoztunk egy új tanítási módszert, amely során nincs szükség GMM-ek használatára. Végül megmutattuk, hogy a végső tanítási lépés javítható egy egyszerű újramintavételező algoritmussal. A kísérleteink során felhasznált magyar nyelvű Szeged Híradós korpuszon egy hagyományos HMM/GMM 20.07%-os szószintű hibaarányt képes elérni. Az új hibrid módszer esetében, ami még változatlanul használja a megörökölt algoritmusokat, a szószintű hibaarány 16.59%-ra csökkent, míg a legjobb GMM-mentes módszerünk még ennél is jobb eredményt (15.79%) ért el.

Természetesen rengeteg további kísérletet lehetne még elvégezni, ezeket sajnos időhiányában a jövőbeli munkáink közé soroljuk. A következőkben felsorolunk néhány lehetséges jövőbeli kutatási irányt.

- Az elmúlt pár évben megjelent egy új típusú neuronháló, a konvolúciós neuronháló (CNN), amely jelentős sikereket ért el képfeldolgozásban és beszédfelismerésben. A kidolgozott módszereinket célszerű lenne kipróbálni ilyen típusú hálókkal is.

- A 3. fejezet kibővítése céljából más szekvenciatanuló algoritmusokat, például a minimális fonéma hiba (MPE) vagy minimális Bayes kockázat (sMBR) módszert is tervezzünk megvizsgálni.

- Érdekes kérdés, hogy vajon hogyan alakulna a GMM-mentes modelljeink pontossága, amennyiben a mostaninál több klaszter létrejöttét is engednénk. A

	[26]	[49]	[50]	[51]	[71]	[87]	[74]	[79]	[80]
I.	•								
II/1.		•							
II/2.		•	•						
III/1.				•					
III/2.			•	•	•				
IV.						•	•	•	•

Table 7.1: A tézispontok és a szerző publikációinak viszonya.

hipotézisünk, hogy több kontextus-függő állapot esetén jobb eredményeket tudnának elérni a hálók, természetesen ennek az ára a megnövekedett tanítási és kiértékelési idők lennének.

- Szintén megérné megvizsgálni, hogy a mintavételezéssel tanított hálók hogyan viselkednének, egy végső szekvencia-diszkriminatív tanítási lépés végrehajtása után.

7.6 Az eredmények tézisszerű összefoglalása

Az alábbiakban tézispontokba rendezve összegezzük a szerző kutatási eredményeit. A 7.6. táblázat összegzi a kutatásokból származó publikációk és az egyes tézispontok viszonyát.

I. A szerző kísérleti úton összehasonlított négy mély tanulásos módszert: két előtanításos algoritmust, az egyenirányított aktivációs függvényt és a Dropout nevű regularizációs technikát. A kiértékeléseket egy magyar nyelvű adatbázison is elvégeztük, az itt közölt eredmények, legjobb tudomásunk szerint, a legelső mély neuronhálós eredmények magyar nyelvű beszédfelismerésben. Az eredmények alapján megállapíthatjuk, hogy a HMM/DNN hibrid szignifikánsan jobban teljesít mint a hagyományos HMM/GMM. A végső konklúziója a kísérleteknek az volt, hogy mind a négy módszer elég hasonló eredményeket tudott elérni, de az egyenirányított hálók konzisztensen jobbnak bizonyultak a többi módszernél.

II/1. A szerző megmutatta, hogy a CTC algoritmust, amit eredetileg rekurens neuronhálók tanítására készítettek, fel lehet használni előrecsatolt hálók tanítására is. A kísérletek célja annak megállapítása volt, hogy ez a módszer alkalmas-e a flat start tanítási lépés elvégzésére, ezért mély egyenirányított neuronhálók lettek tanítva CTC algoritmussal, különböző adatbázisokon. Az eredmények azt

mutatták, hogy a CTC módszer alkalmas véletlenszerűen inicializált neuronhálók flat start tanítására.

II/2. A CTC algoritmus versenytársaként megvizsgálásra került az MMI algoritmus is. A szerző több módosítást is javasolt, hogy ezt a módszert alkalmassá tegye a flat start tanításra. Az összehasonlítás során egyértelművé vált, hogy az MMI sokkal jobb megoldás mint a CTC algoritmus véletlenszerűen inicializált neuronhálók tanítására időben illesztett címkék nélkül.

III/1. A szerző kidolgozott egy új, mély neuronhálós állapotkapcsolási algoritmust, a standard algoritmus döntési kritériumának lecserélésével. Tekintve, hogy a módszer bemenetként DNN által predikált posterior valószínűségi vektorokat kap, ezért döntési kritériumnak a KL-divergencia tűnt logikus választásnak. Ezt a kísérleti eredmények is alátámasztották, az új algoritmus lényegesen jobban teljesített, mint az eredeti módszer.

III/2. Az MMI-alapú flat start módszer és a KL-divergenciát alkalmazó állapot klaszterezési algoritmus kombinálásával a szerző egy teljesen GMM-mentes eljárást hozott létre. Ezt az új eljárást más, közelmúltban javasolt módszerrel hasonlította össze. A kísérletek során kiderült, hogy az új GMM-mentes módszerek jobb eredményeket képesek elérni mint azok, amelyek felhasználnak GMM-eket tanításuk során.

IV. A szerző megvizsgálta a valószínűségi mintavételező algoritmust és alkalmazta azt kontextusfüggő DNN tanításra. A hipotézise az volt, hogy a tanítóadat újramintavételezésével a prior valószínűségek újrabecslése szükségessé válik. Kísérleti úton igazolta ezt a sejtést és megmutatta, hogy újramintavételezéssel és a priorok helyes beállításával szignifikánsan javítható a mély hálók pontossága. A mintavételező algoritmust paralingvisztikus feladatokon is sikeresen alkalmazta.

Publications of the author

Publications accepted by the PhD School in Computer Science

Journal publications

- Gábor Gosztolya and Tamás Grósz. Domain adaptation of deep neural networks for automatic speech recognition via wireless sensors. *Journal of Electrical Engineering*, 67(2):124–130, 2016.

- Péter Bodnár, Tamás Grósz, László Tóth, and László G Nyúl. Efficient visual code localization with neural networks. *Pattern Analysis and Applications*, 21(1):249–260, 2018.

Conference publications

- László Tóth and Tamás Grósz. A comparison of deep neural network training methods for large vocabulary speech recognition. In *Proceedings of TSD*, pages 36–43. Springer Berlin Heidelberg, 2013.

- Gábor Gosztolya, Tamás Grósz, Róbert Busa-Fekete, and László Tóth. Detecting the intensity of cognitive and physical load using adaboost and deep rectifier neural networks. In *Proceedings of Interspeech*, pages 452–456, 2014.

- György Kovács, László Tóth, and Tamás Grósz. Robust multi-band asr using deep neural nets and spectro-temporal features. In *Proceedings of SPECOM*, pages 386–393. Springer International Publishing, 2014.

- Tamás Grósz, Gábor Gosztolya, and László Tóth. A sequence training method for deep rectifier neural networks in speech recognition. In *Proceedings of SPECOM*, pages 81–88. Springer International Publishing, 2014.

- Tamás Grósz and István Nagy. Document classification with deep rectifier neural networks and probabilistic sampling. In *Proceedings of TSD*, pages 108–115. Springer International Publishing, 2014.

- Péter Bodnár, Tamás Grósz, László Tóth, and László Nyúl. Localization of visual codes in the dct domain using deep rectifier neural networks. In *Proceedings of ANNIIP*. SciTePress, 2014.

- Tamás Grósz, Péter Bodnár, László Tóth, and László G Nyúl. Qr code localization using deep neural networks. In *Proceedings of MLSP*, pages 1–6. IEEE, 2014.

- Tamás Grósz, Róbert Busa-Fekete, Gábor Gosztolya, and László Tóth. Assessing the degree of nativeness and parkinson's condition using gaussian processes and deep rectifier neural networks. In *Proceedings of Interspeech*, pages 1339–1343, 2015.

- Gábor Gosztolya, Tamás Grósz, László Tóth, and David Imseng. Building context-dependent dnn acoustic models using kullback-leibler divergence-based state tying. In *Proceedings of ICASSP*, pages 4570–4574. IEEE, 2015.

- Gábor Gosztolya, Tamás Grósz, György Szaszák, and László Tóth. Estimating the sincerity of apologies in speech by DNN rank learning and prosodic analysis. In *Proceedings of Interspeech*, pages 2026–2030, 2016.

- Gábor Gosztolya, Tamás Grósz, Róbert Busa-Fekete, and László Tóth. Determining native language and deception using phonetic features and classifier combination. In *Proceedings of Interspeech*, pages 2418–2422, 2016.

- Gábor Gosztolya, László Tóth, Tamás Grósz, Veronika Vincze, Ildikó Hoffmann, Gréta Szatlóczki, Magdolna Pákáski, and János Kálmán. Detecting mild cognitive impairment from spontaneous speech by correlation-based phonetic feature selection. In *Proceedings of Interspeech*, 2016.

- Gábor Gosztolya, Tamás Grósz, and László Tóth. GMM-free flat start sequence-discriminative DNN training. In *Proceedings of Interspeech*, pages 3409–3413, 2016.

- György Kovács, Tamás Grósz, and Tamás Váradi. Topical unit classification using deep neural nets and probabilistic sampling. In *Proceedings of CogInfoCom*, pages 199–204. IEEE, 2016.

- Gábor Gosztolya, Róbert Busa-Fekete, Tamás Grósz, and László Tóth. DNN-based feature extraction and classifier combination for child-directed speech, cold and snoring identification. In *Proceedings of Interspeech*, pages 3522–3526, 2017.

- Tamás Gábor Csapó, Tamás Grósz, Gábor Gosztolya, László Tóth, and Alexandra Markó. DNN-based ultrasound-to-speech conversion for a Silent Speech Interface. In *Proceedings of Interspeech*, pages 3672–3676, 2017.

- Tamás Grósz, Gábor Gosztolya, and László Tóth. Training context-dependent DNN acoustic models using probabilistic sampling. In *Proceedings of Interspeech*, pages 1621–1625, 2017.

- Tamás Grósz, Gábor Gosztolya, and László Tóth. A comparative evaluation of GMM-free state tying methods for ASR. In *Proceedings of Interspeech*, pages 1626–1630, 2017.

- Tamás Grósz, Gábor Gosztolya, László Tóth, Tamás Gábor Csapó, and Alexandra Markó. F0 estimation for dnn-based ultrasound silent speech interfaces. In *Proceedings of ICASSP*, 2018.

Bibliography

1. S. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book*. Cambridge, UK: Cambridge University Engineering Department, 2006.

2. C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

3. W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *CoRR*, vol. abs/1610.05256, 2016.

4. Z. Tüske, P. Golik, R. Schlüter, and H. Ney, "Acoustic modeling with deep neural networks using raw time signal for lvcsr," in *Proceedings of Interspeech*, 09 2014, pp. 890–894.

5. D. Povey and G. Saon, "Feature and model space speaker adaptation with full covariance gaussians," in *ICSLP*, 2006.

6. M. J. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

7. D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

8. T. Grósz and L. Tóth, "A comparison of Deep Neural Network training methods for Large Vocabulary Speech Recognition," in *Proceedings of TSD*, 2013, pp. 36–43.

9. G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

10. F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proceedings of ASRU*, 2011, pp. 24–29.

11. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," in *CoRR*, vol. 1207.0580, 2012.

12. L. Tóth, B. Tarján, G. Sárosi, and P. Mihajlik, "Speech recognition experiments with audiobooks," *Acta Cybernetica*, pp. 695–713, 2010.

13. G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly *et al.*, "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

14. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proceedings of AISTATS*, 2011, pp. 315–323.

15. A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

16. N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary conversational speech recognition," Dept. Comp. Sci., University of Toronto, Tech. Rep., 2012.

17. G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained Deep Neural Networks for large vocabulary speech recognition," *IEEE Trans. ASLP*, vol. 20, no. 1, pp. 30–42, 2012.

18. V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of ICML*, 2010, pp. 807–814.

19. L. Tóth, "Phone recognition with Deep Sparse Rectifier Neural Networks," in *Proceedings of ICASSP*, 2013, pp. 6985–6989.

20. H. Bourlard and N. Morgan, *Connectionist Speech Recognition – A Hybrid Approach*. Kluwer Academic, 1994.

21. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of AISTATS*, 2010, pp. 249–256.

22. Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, pp. 153–160, 2007.

23. T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," pp. 901–909, 2016.

24. K. Abari, G. Olaszy, C. Zainkó, and G. Kiss, "Hungarian pronunciation dictionary on Internet (in Hungarian)," in *Proceedings of MSZNY*, 2006, pp. 223–230.

25. I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of ICML*, 2013, pp. 1139–1147.

26. L. Tóth and T. Grósz, "A comparison of deep neural network training methods for large vocabulary speech recognition," in *Proceedings of TSD*, 2013, pp. 36–43.

27. B. Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proceedings of ICASSP*, 2009, pp. 3761–3764.

28. G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep Neural Networks for acoustic modeling in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

29. A. Senior, G. Heigold, M. Bacchiani, and H. Liao, "GMM-free DNN training," in *Proceedings of ICASSP*, 2014, pp. 5639–5643.

30. A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Springer, 2012, vol. 385.

31. P. J. Werbos, "Backpropagation Through Time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

32. A. Graves, A.-R. Mohamed, and G. E. Hinton, "Speech recognition with Deep Recurrent Neural Networks," in *Proceedings of ICASSP*, 2013, pp. 6645–6649.

33. K. Rao, A. Senior, and H. Sak, "Flat start training of CD-CTC-SMBR LSTM RNN acoustic models," in *Proceedings of ICASSP*, 2016, pp. 5405–5409.

34. K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proceedings of Interspeech*, 2013, pp. 2345–2349.

35. P. Zhou, L. Dai, and H. Jiang, "Sequence training of multiple Deep Neural Networks for better performance and faster training speed," in *Proceedings of ICASSP*, 2014, pp. 5664–5668.

36. E. McDermott, G. Heigold, P. Moreno, A. Senior, and M. Bacchiani, "Asynchronous stochastic optimization for sequence training of Deep Neural Networks: Towards big data," in *Proceedings of Interspeech*, 2014, pp. 1224–1228.

37. A. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of Deep Belief Networks for speech recognition," in *Proceedings of Interspeech*, 2010, pp. 2846–2849.

38. G. Saon and H. Soltau, "A comparison of two optimization techniques for sequence discriminative training of Deep Neural Networks," in *Proceedings of ICASSP*, 2014, pp. 5604–5608.

39. S. Wiesler, P. Golik, R. Schüter, and H. Ney, "Investigations on sequence training of neural networks," in *Proceedings of ICASSP*, 2015, pp. 4565–4569.

40. D. Chen, B. Mak, and S. Sivadas, "Joint sequence training of phone and grapheme acoustic model based on multi-task learning Deep Neural Networks," in *Proceedings of Interspeech*, 2014, pp. 1083–1087.

41. X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, 1st ed. Prentice Hall PTR, 2001.

42. G. Gosztolya, A. Bánhalmi, and T. László, "Using one-class classification techniques in the anti-phoneme problem," in *Proceedings IbPRIA*, 2009, pp. 433–440.

43. X. He and L. Deng, *Discriminative Learning for Speech Recognition*. San Rafael, CA, USA: Morgan & Claypool, 2008.

44. D. Yu and L. Deng, "Chapter 8: Deep neural network sequence-discriminative training," in *Automatic Speech Recognition — A Deep Learning Approach*. Springer, 2014.

45. S. J. Rennie, V. Goel, and S. Thomas, "Annealed dropout training of deep networks," in *Proceedings of SLT*, 2014, pp. 159–164.

46. H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *Proceedings of ICASSP*, 2013, pp. 6664–6668.

47. S. S. Lamel L., Kassel R., "Speech database development: Design and analysis of the acoustic-phonetic corpus," in *DARPA Speech Recognition Workshop*, 1986, pp. 121–124.

48. L. Tóth, "Convolutional deep rectifier neural nets for phone recognition," in *Proceedings of Interspeech*, 2013, pp. 1722–1726.

49. T. Grósz, G. Gosztolya, and L. Tóth, "A sequence training method for Deep Rectifier Neural Networks in speech recognition." in *Proceedings of SPECOM*, Sep 2014, pp. 81–88.

50. G. Gosztolya, T. Grósz, and L. Tóth, "GMM-free flat start sequence-discriminative DNN training," in *Proceedings of Interspeech*, San Francisco, CA, USA, Sep 2016, pp. 3409–3413.

51. G. Gosztolya, T. Grósz, L. Tóth, and D. Imseng, "Building context-dependent DNN acousitc models using Kullback-Leibler divergence-based state tying," in *Proceedings of ICASSP*, 2015, pp. 4570–4574.

52. A. Senior, G. Heigold, M. Bacchiani, and H. Liao, "GMM-free DNN acoustic model training," in *Proceedings of ICASSP*, 2014, pp. 5639–5643.

53. C. Zhang and P. Woodland, "Standalone training of context-dependent Deep Neural Network acoustic models," in *Proceedings of ICASSP*, 2014, pp. 5597–5601.

54. D. Yu, L. Deng, and G. Dahl, "Roles of pretraining and fine-tuning in context-dependent DNN-HMMs for real-world speech recognition," in *Proceedings of NIPS*, 2010.

55. S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings of HLT*, 1994, pp. 307–312.

56. A. Senior, G. Heigold, M. Bacchiani, and H. Liao, "GMM-free DNN training," in *Proceedings of ICASSP*, 2014.

57. M. Bacchiani and D. Rybach, "Context dependent state tying for speech recognition using deep neural network acoustic models," in *Proceedings of ICASSP*, 2014, pp. 230–234.

58. M. Razavi, R. Rasipuram, and M. Magimai-Doss, "On modeling context-dependent clustered states: Comparing HMM/GMM, hybrid HMM/ANN and KL-HMM approaches," in *Proceedings of ICASSP*, 2014.

59. L. Zhu, K. Kilgour, S. Stüker, and A. Waibel, "Gaussian free cluster tree construction using Deep Neural Network," in *Proceedings of Interspeech*, Sep 2015, pp. 3254–3258.

60. W. Wang, H. Tang, and K. Livescu, "Triphone state-tying via Deep Canonical Correlation Analysis," in *Proceedings of Interspeech*, 2016, pp. 3444–3448.

61. K. Beulen and H. Ney, "Automatic question generation for decision tree based state tying," in *Proceedings of ICASSP*, 1998, pp. 805–808.

62. J. Odell, "The use of context in large vocabulary speech recognition," Ph.D. dissertation, University of Cambridge, 1995.

63. D. Imseng and J. Dines, "Decision tree clustering for KL-HMM," IDIAP Research Institute, Tech. Rep. Idiap-Com-01-2012, 2012.

64. D. Imseng, J. Dines, P. Motlicek, P. Garner, and H. Bourlard, "Comparing different acoustic modeling techniques for multilingual boosting," in *Proceedings of Interspeech*, 2012.

65. S. Kullback and R. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

66. D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proceedings of HLT*, 1992, pp. 357–362.

67. P. C. Woodland, J. J. Odell, V. Valtchev, and S. J. Young, "Large vocabulary continuous speech recognition using htk," in *Proceedings of ICASSP*, 1994, pp. II/125–II/128 vol.2.

68. P. Matějka, O. Glembek, O. Novotný, O. Plchot, F. Grézl, L. Burget, and J. H. Černocký, "Analysis of DNN approaches to speaker identification," in *Proceedings of ICASSP*, 2016, pp. 5100–5104.

69. Y. Bar, N. Levy, and W. L., "Classification of artistic styles using binarized features derived from a Deep Neural Network," in *Proceedings of ECCV*, 2015, pp. 71–84.

70. M. Bacchiani, A. Senior, and G. Heigold, "Asynchronous, online, GMM-free training of a context dependent acoustic model for speech recognition," in *Proceedings of Interspeech*, 2014, pp. 1900–1904.

71. T. Grósz, G. Gosztolya, and L. Tóth, "A comparative evaluation of GMM-free state tying methods for ASR," in *Proceedings of Interspeech*, 2017, pp. 1626–1630.

72. G. M. Weiss and F. Provost, "The effect of class distribution on classifier learning: an empirical study," *Technical Report, Rutgers Univ.*, 2001.

73. K. Andric and D. Kalpic, "The effect of class distribution on classification algorithms in credit risk assessment," in *Proceedings of MIPRO*, 2016, pp. 1241–1247.

74. G. Gosztolya, T. Grósz, R. Busa-Fekete, and L. Tóth, "Detecting the intensity of cognitive and physical load using AdaBoost and Deep Rectifier Neural Networks," in *Proceedings of Interspeech*, 2014, pp. 452–456.

75. A. I. García-Moral, R. Solera-Ureña, C. Peláez-Moreno, and F. Díaz-de-María, "Data balancing for efficient training of hybrid ANN/HMM automatic speech recognition systems," *IEEE Trans. Audio, Speech & Language Processing*, vol. 19, no. 3, pp. 468–481, 2011.

76. S. Lawrence, I. Burns, A. Back, A. Tsoi, and C. Giles, "Chapter 14: Neural network classification and prior class probabilities," in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 299–313.

77. L. Tóth and A. Kocsor, "Training HMM/ANN hybrid speech recognizers by probabilistic sampling," in *Proceedings of ICANN*, 2005, pp. 597–603.

78. M. Song, Q. Zhang, J. Pan, and Y. Yan, "Improving HMM/DNN in asr of under-resourced languages using probabilistic sampling," in *Proceedings of ChinaSIP*, 2015, pp. 20–24.

79. G. Gosztolya, T. Grósz, R. Busa-Fekete, and L. Tóth, "Determining native language and deception using phonetic features and classifier combination," in *Proceedings of Interspeech*, San Francisco, CA, USA, Sep 2016, pp. 2418–2422.

80. G. Gosztolya, R. Busa-Fekete, T. Grósz, and L. Tóth, "DNN-based feature extraction and classifier combination for child-directed speech, cold and snoring identification," in *Proceedings of Interspeech*, 2017, pp. 3522–3526.

81. A. Rousseau, P. Deléglise, and Y. Estéve, "TED-LIUM: an automatic speech recognition dedicated corpus," in *Proceedings of LREC*, 2012, pp. 125–129.

82. J. Carletta, "Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus," *Language Resources and Evaluation*, vol. 41, no. 2, pp. 181–190, 2007.

83. D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely,

"The Kaldi Speech Recognition Toolkit," in *Proceedings of ASRU*. IEEE Signal Processing Society, 2011.

84. B. Schuller, S. Steidl, A. Batliner, J. Epps, F. Eyben, F. Ringeval, E. Marchi, and Y. Zhang, "The INTERSPEECH 2014 computational paralinguistics challenge: Cognitive & physical load," in *Proceedings of Interspeech*, 2014, pp. 427–431.

85. B. Schuller, S. Steidl, A. Batliner, J. Hirschberg, J. K. Burgoon, A. Baird, A. Elkins, Y. Zhang, E. Coutinho, and K. Evanini, "The Interspeech 2016 computational paralinguistics challenge: Deception, sincerity & native language," in *Proceedings of Interspeech*.

86. B. Schuller, S. Steidl, A. Batliner, S. Hantke, E. Bergelson, J. Krajewski, C. Janott, A. Amatuni, M. Casillas, A. Seidl, M. Soderstrom, A. S. Warlaumont, G. Hidalgo, S. Schnieder, C. Heiser, W. Hohenhorst, M. Herzog, M. Schmitt, K. Qian, Y. Zhang, G. Trigeorgis, P. Tzirakis, and S. Zafeiriou, "The INTERSPEECH 2017 computational paralinguistics challenge: Addressee, Cold & Snoring," in *Proceedings of Interspeech*, 2017, pp. 3442–3446.

87. T. Grósz, G. Gosztolya, and L. Tóth, "Training context-dependent DNN acoustic models using probabilistic sampling," in *Proceedings of Interspeech*, 2017, pp. 1621–1625.