Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Radioelectronics

Continuous Speech Recognition using Advanced Deep Neural Networks

Master thesis

Bc. Martin Šubert

Study program: Electronics and communication
Specialisation: Audiovisual technique and signal processing
Supervisor: Doc. Ing. Petr Pollák, CSc.

Prague, May 2021

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Šubert Martin** Personal ID number: **457157**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Radioelectronics**

Study program: **Electronics and Communications**

Specialisation: **Audiovisual and Signal Processing**

II. Master's thesis details

Master's thesis title in English:

Continuous Speech Recognition using Advanced Deep Neural Networks

Master's thesis title in Czech:

Rozpoznávání spojité řeči s pokročilými strukturami hlubokých neuronových sítí

Guidelines:

1. Meet the principles of automated speech recognition (ASR) with a special focus on architectures based on deep neural networks (DNN).
2. Make a survey on currently used solutions for ASR with advanced structures of DNN and compare selected approaches of large vocabulary continuous speech recognition.
3. Using KALDI toolkit and available example scripts (recipes), realize an implementation of LVCSR based on selected approaches using available speech databases, analyze achieved recognition accuracy, and compare your results with other ones obtained and published by other authors. Based on KALDI conventions, create new example scripts (recipes) covering your designed solutions as a practical output of your work.

Bibliography / sources:

[1] X. Huang, A. Acero, H.-W. Hon: Spoken Language Processing. Prentice Hall, 2001.
[2] D. Yu, L. Deng. Automatic Speech Recognition A Deep Learning Approach. Springer-Verlag London. 2015
[3] M. Karafiát, et al. Multilingual BLSTM and speaker-specific vector adaptation in 2016 BUT Babel system. In 2016 IEEE SLT Workshop, San Diego, CA, 2016.
[4] J. Fiala: DNN-HMM Based Multilingual Recognizer of Telephone Speech. Diploma thesis, CTU FEE, 2016.

Name and workplace of master's thesis supervisor:

doc. Ing. Petr Pollák, CSc., Department of Circuit Theory, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **25.01.2021** Deadline for master's thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

doc. Ing. Petr Pollák, CSc. Supervisor's signature	doc. Ing. Josef Dobeš, CSc. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

Declaration

I hereby declare I have written this thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, May 2021

.....
Bc. Martin Šubert

Acknowledgements

I want to thank my supervisor, Doc. Ing. Petr Pollák, CSc., for his excellent assistance and support. He introduced me to the speech recognition field and provided me with the necessary tools to work on the experiments. His guidance taught me how to present scientific work properly.

Abstract

This thesis presents Automatic Speech Recognition (ASR) systems based on deep neural networks (DNN) with advanced structures and their implementations for the English and Czech languages using the Kaldi toolkit. The experiments use the newest Kaldi DNN nnet3 setup, which supports mentioned advanced DNN types. Moreover, two nnet3 models are adopted - the standard nnet3 models and the chain models, which are implemented as a part of the nnet3 DNN setup with the intention to decrease decoding time. The implementations work with DNN-HMM architecture with two neural network types, Time Delay Neural Network (TDNN) and Long Short-Term Memory (LSTM), using both nnet3 and chain models. Additionally, the Convolutional Neural Network (CNN) neural network structure is adopted using the chain model.

Experiments compare the accuracy of the DNN-HMM ASR models with the standard GMM-HMM approach, and the best accuracy was achieved with TDNN network with World Error Rate (WER) of 2.69 % for the English language, which was about 5% improvement over the standard GMM-HMM model. The best result for the Czech language was also accomplished with the TDNN network with a WER of 10.78 % (approximately 9% improvement over the GMM-HMM model). Secondly, the performance of DNN-HMM systems using GMM-HMM models trained with trigram and fourgram language models (LM) was analyzed as well as with a dictionary with and without silence and pronunciation probabilities. Finally, the accuracy improvements and the overall processing speed of the chain models over the standard nnet3 models were tested. The enhancement of accuracy was achieved both in TDNN and LSTM-TDNN chain models when more extensive improvements registered the TDNN chain model. The decoding time decreased for both DNN chain models when LSTM-TDNN decoded almost nine times faster than standard nnet3 implementation. LSTM-TDNN chain model also reduces the training time when the model was trained about 20 % faster. Nevertheless, the TDNN chain model had worse training speed when the network trained more than twice slower than the nnet3 variant.

Keywords: Deep neural networks, speech recognition, DNN-HMM, Kaldi, nnet3, chain models

Abstrakt

Tato diplomová práce prezentuje systémy automatického rozpoznávání řeči založené na hlubokých neuronových sítích (DNN) s pokročilými strukturami a jejich implementace pro anglický a český jazyk s použitím Kaldi nástrojů. Experimenty používají nejnovější verzi Kaldi DNN nnet3 nástrojů, které podporují zmíněné pokročilé DNN struktury. Byly použity dva nnet3 modely - standardní nnet3 model a tzv. chain model, který byl vytvořen jako součást nnet3 nástrojů za účelem snížit čas dekódování. Implementace pracuje s DNN-HMM architekturou se dvěma typy neuronových sítí, TDNN a LSTM, při použití nnet3 i chain modelů. Navíc byla použita CNN neuronová sít za použití chain modelu.

Experimenty porovnávají přesnost DNN-HMM modelu se standardním GMM-HMM přístupem, kdy nejlepší přesnost byla dosažena pomocí TDNN sítě s WER 2.69 % pro anglický jazyk, což bylo zhruba 5% zlepšení oproti standardnímu GMM-HMM modelu. Nejlepší výsledek pro český jazyk byl také dosažen s použitím TDNN sítě s hodnotou WER 10.78 % (přibližně 9% zlepšení oproti GMM-HMM modelu). Druhá část experimentů porovnávala zlepšení DNN-HMM systémů vycházejících z GMM-HMM modelů natrénovaných s trigramovým a čtyřgramovým jazykovým modelem, a dále se slovníkem s a bez pravděpodobnostmi ticha a výslovnosti. Poslední část experimentů porovnává přesnost a celkovou dobu zpracování za použití chain modelů a standardních nnet3 modelů. Zlepšení přesnosti lze vidět u TDNN i u LSTM-TDNN chain modelu, kdy větší přínos zaznamenala TDNN struktura. Doba dekódování poklesla u obou DNN struktur, kdy chain LSTM-TDNN model dekódoval téměř devětkrát rychleji než jeho implementace pomocí standardních nnet3 modelů. U LSTM-TDNN chain modelu navíc došlo ke snížení doby trénování, kdy byl model natrénován zhruba o 20 % rychleji oproti standardnímu nnet3 modelu. Nicméně u TDNN chain modelu byla doba trénování více než dvojnásobná oproti standardní nnet3 variantě.

Klíčová slova: Hluboké neuronové sítě, rozpoznávání řeči, DNN-HMM, Kaldi, nnet3, chain modely

Contents

List of Tables

List of Figures

List of Acronyms

AM Acoustic Model

ASR Automatic Speech Recognition

BTT Backpropagation Through Time

CNN Convolutional Neural Network

CTC Connectionist Temporal Classification

DBN Deep Belief Network

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DNN Deep Neural Network

FFT Fast Fourier Transform

fMLLR Feature Spaced Maximum Likelihood Linear Regression

GMM Gaussian Mixture Models

LDA Linear Discriminant Analysis

LM Language Model

LSTM Long Short-Term Memory

LVCSR Large Vocabulary Continuous Speech Recognition

MFCC Mel-Frequency Cepstral Coefficients

MLLR Maximum Likelihood Linear Regression

MLP Multilayer Perceptron

OOV Out of vocabulary

PCM Pulse-Code Modulation

PDF Probability Density Function

RBM Restricted Boltzmann Machine

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

TDNN Time Delay Neural Network

UBM Universal Background Model

VTLP Vocal Tract Length Perturbation

WER World Error Rate

WFST Weighted Finite-State Transducers

WSJ Wall Street Journal

Chapter 1

Introduction

Communication among humans seems to be one of the critical abilities to achieve particular progress in human life. As history shows, the way our ancestors were communicating changed and developed through thousands of years. Presumably, the most natural and effective form of communication among humans is speech when different languages arose throughout history depending on the culture or geographic region. Most of the languages also came with their written form, which helped to conserve ideas and messages.

Nowadays, the exponential development in the technology areas comes with a problem of human-to-machine communication. The methods human used to communicate with machines also developed over time and when the simple command buttons enhanced with more sophisticated keyboards. With the introduction of graphic interface, the mouse and touch screens arose their popularity, and they are nowadays still one of the most adopted ways to communicate with computers. Nevertheless, they have some limitations, e.g., obligation to use hands and paying more attention to the commands a person wants to enter into the system. This may be problematic when a person wants to adjust the GPS navigation while driving a car. Using speech to command the GPS seems to be a more suitable way. Therefore, research in ASR systems becomes one of the critical parts of today's technological progress. Speech recognition helps in many fields like IT, the industry sector, healthcare, and many more. Particular ASR applications can be presented with automatic transcriptions of meetings and the creation of movie captions or processing voice commands in the voice assistants. One of the latest usages of ASR is in the speech to speech translation systems.

The first step towards large vocabulary continuous speech recognition (LVCSR) systems comes with the GMM-HMM model proposed in [1] in 1980. This model uses statistical methods Gaussian mixture model (GMM) and Hidden Markov Model (HMM) and became the state-of-the-art approach of LVCSR systems. Later on, in the 1990s, Morgan and Bourlard proposed to replace the GMM in the hybrid GMM-HMM model with

artificial neural networks (ANN) [2]. The multilayer perceptron (MLP) in the hybrid ANN-HMM model should calculate the HMM state-posterior probabilities. Nowadays, models with integrated neural networks have become the leading ASR systems. Various DNN networks were implemented in the hybrid model DNN-HMM, including feed-forward networks such as TDNN, CNN, or variations of Recurrent Neural Network (RNN) like LSTM. One of the significant impacts of achieving good accuracy with these systems is the possibility of obtaining a larger dataset for training the ASR models. Finally, the so-called end-to-end models show promising results and are among the primary interests of speech recognition researchers.

This thesis aims to explore today's DNN based ASR approaches. The theoretical part introduces the DNN methods used in speech recognition, followed by the experimental part, which presents the DNN-HMM approach using the latest DNN Kaldi toolkit approach, nnet3 models. Moreover, the implementations contain DNN-HMM systems based on chain models. The Kaldi toolkit creators introduced the chain models [3] as part of nnet3 with an intention to decrease the decoding time, which may be essential in the online ASR applications.

This thesis is organized into seven chapters. Chapter 2 introduce the GMM-HMM based ASR system. The first part 2.1 describes the feature extraction process and the typical features used in the speech recognition process. The second part 2.2 presents the training and decoding principles and the models adopted in the processes. Chapter 3 firstly defines the DNN, training algorithms, and other methods corresponding with its practices. The second part 3.2 describes the DNN-HMM models with DNN types used in these systems. The last part of this chapter 3.3 introduces the end-to-end models. Chapter 4 discuss the DNN-HMM ASR system implementation. The ASR experimental setup is described in chapter 5 together with achieved results. Finally, chapter 6 discuss the conclusion of the thesis.

Chapter 2

GMM-HMM based speech recognition

The task of speech recognition is to find a word sequence corresponding to the input audio signal. Figure 2.1 shows the simple ASR system process. Implementation based on GMM-HMM is a basic approach of ASR, which proved to be a working LVCSR system. It is also the first ASR implementation used at the commercial level. Even though designs with (DNN) networks are replacing the GMM-HMM ASR systems, there are many reasons why they are still essential. An implementation with DNNs typically needs a more extensive database to train an accurate ASR system. Moreover, many ASR designs with DNNs use a GMM-HMM system to obtain frame-level reference in large training corpora. The system accuracy can then be improved because of the discriminative training of the DNNs.

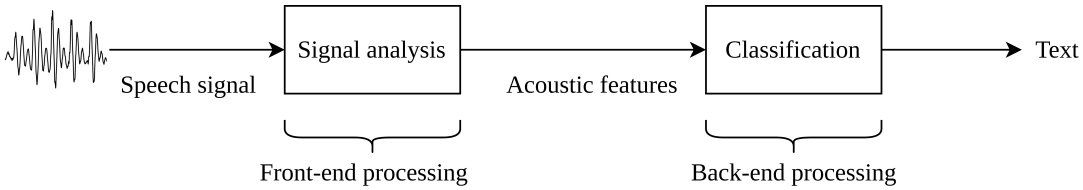

Figure 2.1: Basic ASR system

2.1 Front-end processing - signal analysis

Firstly, it is necessary to define the form of speech content information used during the recognition process. In ASR applications, the speech features are extracted from an audio recording to represent required information. Cepstral coefficients are mainly used as features for the ASR application because they benefit from the principle of speech production and perception knowledge [4].

2.1.1 MFCC - Mel-frequency cepstral coefficients

MFCC coefficients are typically the most frequently used features in ASR applications to represent the short-time signal information. MFCC uses the so-called mel scale, triangular bands with equal distances between each other, representing logarithmic characteristics at the frequency level. Using the logarithmic scale for the frequencies comes from the principle of the human auditory system. The process of obtaining MFCCs is shown in figure 2.2.

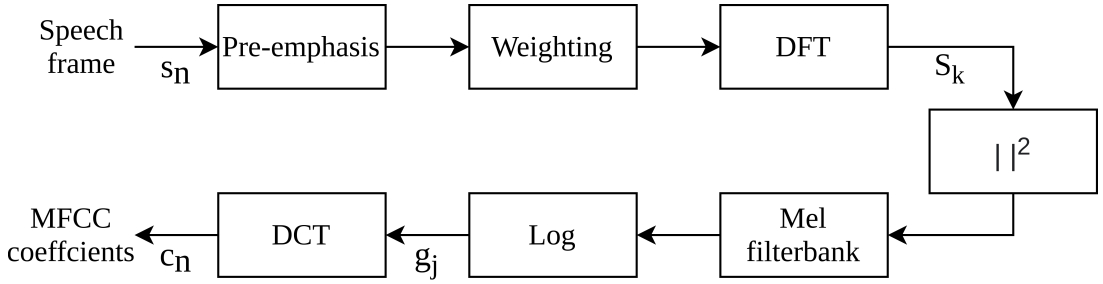

Figure 2.2: Extraction of MFCC features

Due to the non-stationarity of a speech signal, the features are extracted from short-time frames, considered stationary. The standard frame's length is 25 ms with approximately 10 ms frame step size [5]. Standardly, pre-emphasis is applied to compensate for the attenuation of high-frequency components during speech generation. Next, Hamming window is used to minimize spectral leakage, which appears during the discrete Fourier transform (DFT) computation. Then, an auditory-based filter bank is applied to compute the smoothed power spectrum, which models the logarithmic dependency of frequency perception in the human auditory system. Specifically, the logarithmic energies g_j are calculated using a formula (2.1) where $S[k]$ is the DFT spectrum, $H_{mel,j}$ is the frequency response of j-th filter, N is the number of samples of a particular frame, and M is the number of filter bank's bands. The cepstral coefficients c_n are computed using discrete cosine transform (DCT) with formula (2.2), where M is the number of MFCC coefficients. Typically, 13 MFCC coefficients are used to represent the signal features in GMM-HMM systems. However, the many DNN based ASR models use so-called high resolution features where 40 MFCC coefficients are used. The 0-th coefficient contains information about the power of the signal, and the rest of the coefficients represent the shape of the magnitude spectrum.

$$g_j = log \sum_{k=0}^{N/2} |S[k]|^2 H_{mel,j}[k], \qquad j = 1, 2, ..., M \tag{2.1}$$

$$c_n = \sqrt{\frac{2}{J}} \sum_{j=1}^{J} g_j \cos\big(\frac{\pi n}{J}(j-0.5)\big), \qquad j = 1, 2, ..., M \tag{2.2}$$

2.1.2 Differencial features

Additional to basic MFCC features, the delta and delta-delta features are typically used to represent the long temporal context information which may significantly improve the accuracy of speech recognition [6]. These features represent the evolution of static features [7] and can be added to the feature vector along the MFCCs. The delta d_n coefficients can be calculated using formula (2.3), where K is the size of the contextual window, and c_n is the feature frame.

$$d_n = \frac{\sum_{k=1}^{K} k * (c_{n+k} - c_{n-k})}{2 * \sum_{k=1}^{K} k^2}, \tag{2.3}$$

2.1.3 Feature normalization

Several aspects can bring unwanted variability and noise to the extracted features and lower the ASR system's robustness. Firstly, the convolution noise caused by the non-stationary recording environment and speaker diversity, and secondly, the additive noise. The normalization techniques are proposed to eliminate these factors. Commonly used Cepstral Mean and Variance Normalization (CMVN) method is based on subtraction of average cepstrum and its scaling. Normalization is typically performed at both utterance or speaker level.

2.1.4 LDA features

An additional approach to obtain temporal context information is to stack static features of many frames into one more dimensional vector. However, two main problems prevent using the features in the GMM-HMM model. Firstly, the features are correlated, and secondly, the dimension of the vector is too large [8]. The Linear Discriminant Analysis (LDA) is a statistical method that can be used to achieve the decorrelation of the features and lower the vector's dimension. The LDA maps features from n-dimensional space to m-dimensional space, where $n > m$.

2.1.5 Speaker adaptation techniques

Speaker adaptation methods are used to obtain speaker-dependent features, leading to increased accuracy of speech recognition systems. One of the widely used speaker adaptation

techniques is the Maximum Likelihood Linear Regression (MLLR). Using this method, the Gaussian means in the ASR system are adopted to maximize the particular speaker's data likelihood. The feature-spaced MLLR (fMLLR) is an alternative method to the MLLR, where the transformation is executed directly on the acoustic feature vectors [9]. The final vector of size 40 represents speaker-adapted features.

2.2 Back-end processing - classification

The purpose of back-end processing is the decoding process that determines the output recognized text form of input speech. The decoder uses an acoustic model (AM), lexicon, and language model, which provide information about phone-level, word-level, and sentence-level matching, respectively, as shown in figure 2.3. The decoding process works based on several computational methods such as HMM and Weighted Finite-State Transducers (WFST). This chapter describes these particular models and methods.

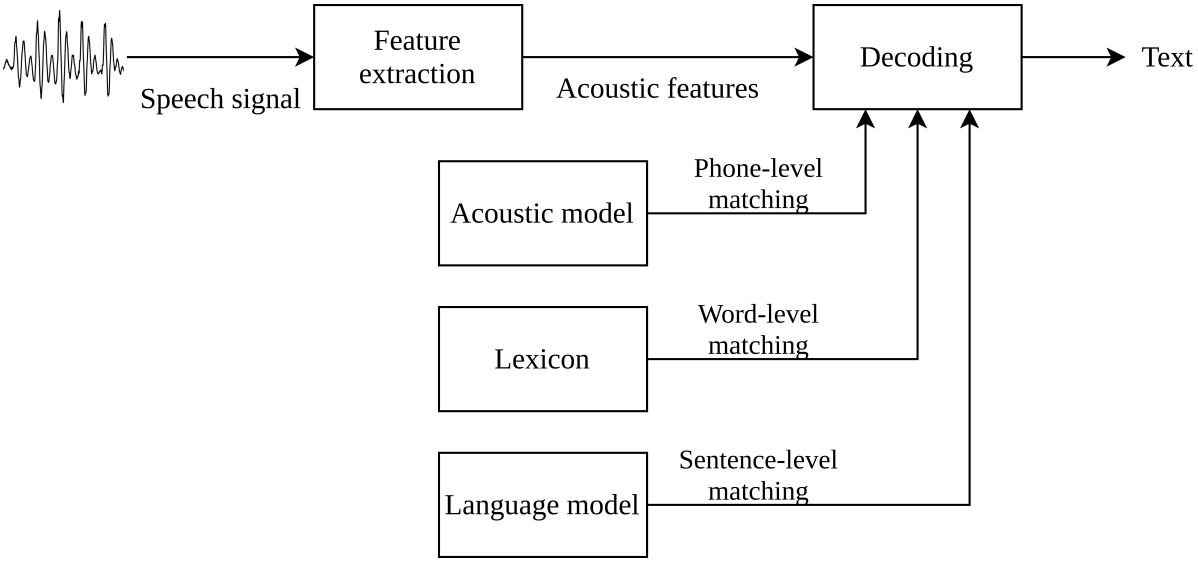

Figure 2.3: Structure of ASR system with models

The process of speech recognition may be expressed mathematically by finding the word sequence W with the maximal a posteriori probability when on the input of the system is the feature vector $X = (x_1, x_2, ..., x_N)$ using formula

$$W = arg \max_{W} P(W|X). \tag{2.4}$$

This equation can be rewritten using Bayes rule as

$$W = arg \max_{W} \frac{P(X|W)P(W)}{P(X)}. \tag{2.5}$$

The first term $P(X|W)$ is a probability of a sequence of extracted features X based on a sequence of words W. This part is represented by the *AM*. The second part of the numerator $P(W)$ is an apriori probability of a word sequence without acoustic information. This part is acquired from the *language model*. The last part of the equation $P(X)$ is an apriori probability of a sequence of feature vectors [10].

2.2.1 Acoustic model

The AM expressed by probability $P(X|W)$ in formula (2.5) represents the phone units of a language based on the extracted acoustic features X from the input signal. The acoustic features use the statistical Gaussian mixture model (GMM) for their representation. Following the equation (2.6), each acoustic phone feature can be expressed by Gaussian probability distribution $p(x|\lambda)$, determined by a mean value and the covariance matrix, where $N(x, \mu_i^s, C_i^s)$ is a n-dimensional Gaussian function given by vector of mean values μ and covariance matrix C.

$$p(x|\lambda^s) = \sum_{i=1}^{M_s} c_i^s N(x, \mu_i^s, C_i^s), \tag{2.6}$$

Hidden Markov Model

The HMM is a left-to-right model consisting of several states connected sequentially, one after another. The model estimates the transitions among the states where the state index remains or increases depending on the time change. The HMM modelling of the most probable state sequence Q^* can be mathematically express by equation (2.7), where $\mathbb{Q}$ represent all possible HMM state sequences $Q = \{q_1, q_2, ..., q_3\}$, the term $p(q_t|q_{t-1})$ referred to the transmission probability and $P(x_t|q_t)$ to the emission probability.

$$Q^* = \underset{Q \in \mathbb{Q}}{\operatorname{argmax}} P(Q, X) = \underset{Q \in \mathbb{Q}}{\operatorname{argmax}} \prod_{t=1}^{T} p(q_t|q_{t-1}) P(x_t|q_t) \tag{2.7}$$

Figure 2.4 shows the three states HMM model. The three states $q = 2, 3, 4$ can express the beginning, middle, and end parts of each phone. The state $q = 1$ and $q = 5$ manage the chaining of HMM models in word or sentence decoding. The emission probability $P(x_t|q_t)$ is modelled by the GMM models, where two parameters, the mean and the covariance matrix, define the Gaussian distribution. The HMM model computes the transmission probabilities $p(q_t|q_{t-1})$, which define the probability of remaining in the current state or moving to the following one. Two assumptions have to be satisfied in order to use the HMM model. Firstly, successive observations are conditionally independent of the past observations and states. Secondly, the state series is the first-order Markov chain [11].

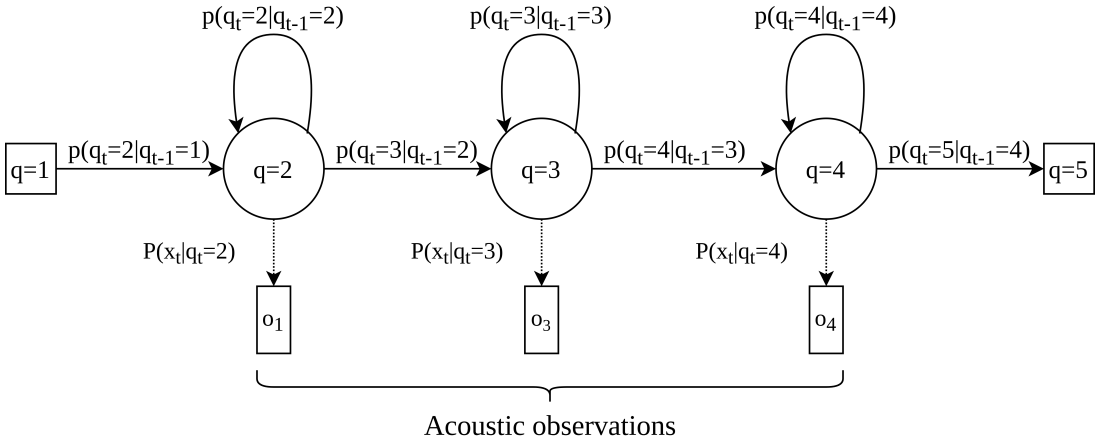

Figure 2.4: Three-state HMM model

Even though speech recognition systems using GMM-HMM models had great success, several weaknesses have already been known since these applications started to use the models [12]. One of the assumptions that a speech pattern is produced by a piece-wise stationary process, with instantaneous transitions between stationary states [13], is in discrepancy to how the human speech is generated - by continuous movements in the vocal tract. Secondly, the assumption that the probability of being in a state s at time t depends only on a state at time *t-1*. This property of HMM might be problematic since the actual state may depend on more past states in speech.

Phonetic Decision Trees

LVCSR systems use decision trees for clustering context-dependent HMM states [14]. The first step of training is calculating the monophone model, which does not depend on any other phone (context-independent model). The next stage of training builds the context-dependent model where the actual phone depends, for example, on the left and the right phone (so-called triphone model). The number of possible triphones is quite large, and each triphone has a different probability of occurring in the training data. Decision tree clustering helps to establish the context-dependent model more efficiently.

2.2.2 Lexicon and dictionary

The ASR system considers the dictionary as a list of words that can occur in the input speech. Once the dictionary does not include a word in the system's input, the *Out of vocabulary* world (OOV) notation is used. Sequences of phonemes can express human spoken words. Thus, each word needs its transcription into text characters. Lexicon models each word in the dictionary into the corresponding sequences of phones (a text form of phonemes).

2.2.3 Language model

The task of the language model is to find an apriori probability $P(W)$. It represents the occurrence probability of a word without any acoustic information. These probabilities are computed for all sentences listed in the training set. The most popular model is the *n-gram model* when the probability is computed for a given word based on limited words before, e.g. for trigram model the probability of occurrence of a word w_i is $P(w_i|w_{i-1}, w_{i-2})$. There is also a special model called uni-gram when the probability of a word w_i does not depend on information from any other word. The probability is computed as the word's quantity of occurrence divided by the number of words in the training text corpus.

2.2.4 Decoding

A decoder aims to take the speech representatives from the front-end part and create a corresponding written form. In other terms, the decoder maps the most probable sequence of words to make the final transcript of input audio speech.

Weighted Finite-State Transducers

Nowadays, the typical method used for decoding is with (WFST). It uses the AM, language model, and lexicon to establish the final decoded text. The WFST uses a static recognition graph $HCLG = H \circ C \circ L \circ G$ from a composition of particular automata. H represents the actual HMM topology, C is a context-dependency transducer from context-dependent phones to context-independent phones, L stands for lexicon, which maps the words from the input phonemes, and G is a probabilistic grammar, which sequences the words in the most probable order based on the language model. A transducer uses a composition operation that combines the representations. For instance, a pronunciation lexicon can be combined with the word-level grammar and create the phone-to-word transducer. The composition operation allows combining all levels of the ASR transducers into an integrated transducer [15].

The integrated transducer is optimized in two steps: determinization and minimalization. There are several additional optimization methods, but determinization and minimalization are mainly adopted. The determinization purpose is to eliminate redundant paths in the composed transducers, which reduce the recognition time. Moreover, it reduced the transducer size and improve the efficiency of composition [15]. This process is performed at each composition step of every pair of transducers. The final integrated transducer N can be expressed with equation

$$N = \pi_\epsilon(\det(\widetilde{H} \circ \det(\widetilde{C} \circ \det(\widetilde{L} \circ G)))), \tag{2.8}$$

where π_ϵ is an erasing operation that replaces the auxiliary distribution symbols by ϵ's. $\widetilde{L}$ is a modified lexicon L with auxiliary phone symbols. The auxiliary phone symbols are added at the end of a word and help distinguish two different words with the same pronunciation. Using $\widetilde{L}$ implies adopting context-dependency transducer $\widetilde{C}$ that pairs the context-independent auxiliary symbols from the modified lexicon $\widetilde{L}$. Finally, $\widetilde{H}$ is modified HMM representation H that allows to map the auxiliary context-dependent phones from $\widetilde{C}$ into a new distribution name. The minimization can be express with formula (2.9), where the min operator is added. The minimization algorithm further decreases the size of the integrated transducer.

$$N = \pi_\epsilon(\min(\det(\widetilde{H} \circ \det(\widetilde{C} \circ \det(\widetilde{L} \circ G))))), \tag{2.9}$$

2.2.5 Forced alignment

Forced alignment occurs during the decoding when a model of a sentence is aligned to the given transcription. The main principle of the alignment is shown in the figure 2.5. The computation typically uses the Viterbi algorithm, which estimates the likelihoods of passing through particular paths in the decoding graph. Subsequently, the backtracing finds the maximum likelihood path. The graph contains information both from the AM and the utterance content transcript represented by grammar. In this matter, it is crucial to choose the pronunciation best matches the observed one. The forced alignment assumes knowing the content of the sentence. If the corpus used for training does not contain a reference transcription, the alignments are computed as the decoder's best path.

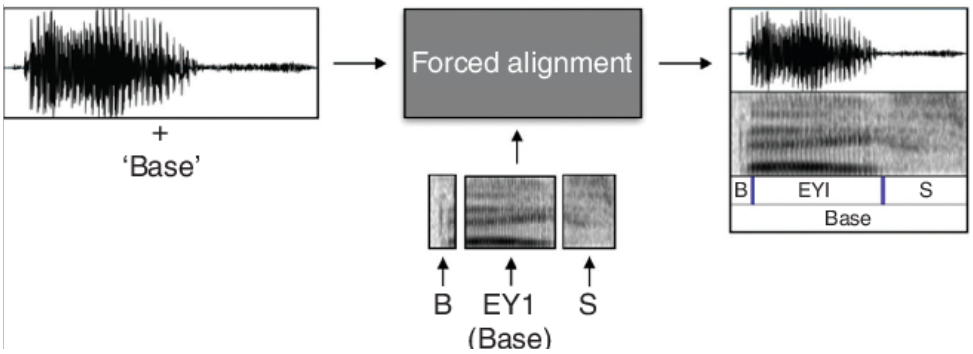

Figure 2.5: Forced alignment method [16]

Chapter 3

DNN within speech recognition

There are various approaches of DNN usage in the implementation of the speech recognition systems. It can help to obtain better-extracted features using unsupervised training. In that case, the features are extracted from one of the hidden layers of the neural net. Another typical scheme is the DNN-HMM structure, where the DNNs produce the posterior probabilities over HMM states. Therefore, the DNNs replace GMMs in the standard GMM-HMM approach. Several enhancements improve the DNN-HMM design, such as using different neural network settings, a different structure such as LSTM, CNN, and similar. The most recent implementation uses an end-to-end DNN approach, when the neural net input is the speech signal or its spectrogram and in the output is the transcript, recognized text.

3.1 Definition of DNN

A basic DNN is based on a multilayer perceptron (MLP) with numerous hidden layers. Figure 3.1 shows a DNN with one input layer, three hidden layers, and one output layer. The size of an input layer depends on the neural network inputs. In the DNN-HMM model, the neural network receives the extracted features, so the input layer size is the same as the feature vector. The hidden layer's size varies based on a particular application's needs and is typically more expansive than the input layer. The expected size of the hidden layer is from 500 up to 10000 neurons, when the number of neurons typically decreases with fewer hidden layers in DNN. The output layer size depends on the number of classification classes.

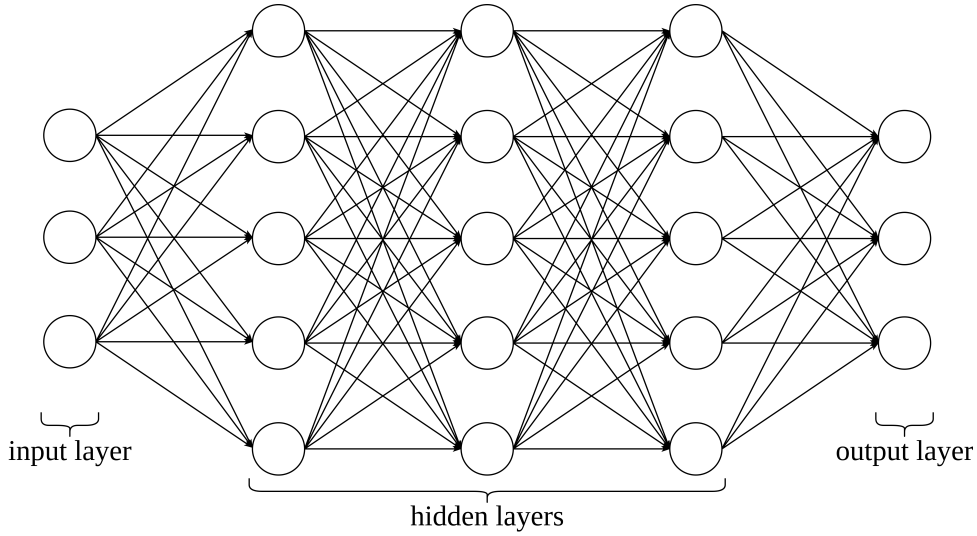

Figure 3.1: DNN with three hidden layers

The so-called activation function defines each neuron of the hidden layer j, which typically is a logistic function. The most common are sigmoid, giving values 0 or 1, and a tangent function, giving -1 or 1. The output of the neuron y_j can be calculated with equation (3.1) using the sigmoid function or with formula (3.2), which applies the tangent function.

$$y_{j_{sigmoid}} = \frac{1}{1 + e^{-x_j}}, \tag{3.1}$$

$$y_{j_{tangent}} = \frac{e^{x_j} - e^{-x_j}}{e^{x_j} + e^{-x_j}}. \tag{3.2}$$

The x_j is the total input to the neuron j and can be mathematically described using the formula

$$x_j = b_j + \sum_i y_j w_{ij}, \tag{3.3}$$

where b_j is the bias of neuron j, i is the index of the layer before and w_{ij} is the connection weight between layer i and j. The output of the DNN characterizes the a posteriori probabilities computed by the softmax activation function p_j described by equation

$$p_j(x_j) = \frac{e_{x_j}}{\sum_{i=1}^{J} e^{x_i}} \tag{3.4}$$

where index j belongs to the output layer, and the probabilities are calculated for the neuron potentials x_i.

3.1.1 Training process

The DNN model has two unknown parameter - the weights w_{ij} and the bias b_j. These parameters are estimated during the training process, which can be defined by a training criterion and a learning algorithm [12].

The most common training criterion in the ASR applications is the cross-entropy [17] criterion. The cost function C is then the cross-entropy between the reference output d and the output of the softmax function p for each training stage j expressed with formula (3.5), where the reference probabilities are the supervised information used to train the DNN classifier [18].

$$C = -\sum_j d_j \log p_j, \tag{3.5}$$

The training process can use the backpropagation of cost function's derivatives. This method may be computationally challenging with a large dataset. The solution for this problem is to divide training data into smaller batches, and the derivatives are computed within them. Another improvement brings the Stochastic Gradient Descent (SGD) method by adding a momentum coefficient α which smooths the gradient. The actual weight w_{ij} for batch t can be then computed using using formula (3.6), where ϵ is the learning rate and $0 < \alpha < 1$.

$$\Delta w_{ij}(t) = \alpha \Delta w_{ij}(t-1) - \epsilon \frac{\partial C}{\partial w_{ij}(t)}, \tag{3.6}$$

The biases can be computed with the same method as the weights, looking at them as the connections from neurons that always value one [18].

3.1.2 Pre-training

DNNs with many hidden layers and units per layer are very flexible, and they can be trained for very complex and nonlinear relationships between input and output. However, this fact also comes with disadvantages. It permits to train false regularities as an accidental characteristic of the training dataset, leading to an overfitting problem [18].

The DNN-HMM model uses supervised training in the training process of the neural network. Therefore, the DNN needs a reference of the output that the network should provide. In this matter, the GMM-HMM model is trained in the first places, providing the context-dependent states from the forced-alignment process as the references. At this moment, the input feature vector and the reference output of the neural network are known. Furthermore, the weights and the biases of DNN need to be initialized before the training starts.

According to [12], the neural network's initialization has a significant impact on the resulting model. There are several approaches to initialize such a network. The first practice is to set the weights and biases values randomly. Random initialization is necessary because neurons in the DNNs are symmetric and interchangeable [12]. However, using the random initialization may not lead to the best-resulting model, and overfitting can occur while tackling a difficult task. The overfitting can be reduced by using large datasets [19], but this results in a longer training process time.

Another strategy is to use advanced initialization techniques, including restricted Boltzmann machines (RBM) and deep belief networks (DBN), which utilize the training data's information. The goal is to learn one layer of feature detectors from the previous layer acting as it already obtained the training data. The pre-training process finds the region of weight-space, making better progress during the discriminative fine-tuning and significantly reducing the overfitting [20].

Restricted Boltzmann machines

The RBM is defined by one layer of stochastic binary visible units representing the input data connected to the layer of stochastic binary hidden units that learn the significant nonindependence between the visible units [21], as it can be seen in figure 3.2. The connections are only between units in the visible layer and the units of the hidden layer.

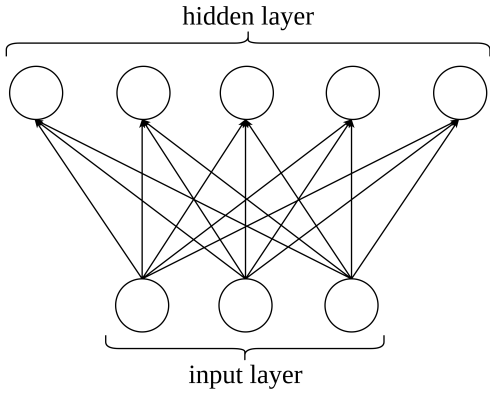

Figure 3.2: RBM network

The RBM algorithm distributes energy to every configuration of visible units, vector $v \in \mathbb{R}^{N_v \times 1}$, and hidden units, vector $h \in \{0,1\}^{N_h \times 1}$, where the N_v and N_h are the number of visible and hidden units, respectively [12]. The ASR applications usually use the Gaussian–Bernoulli (GRBM) when the input units may take real values occurring in the MFCCs. The mathematical equation of the GRBM energy can be computed as

$$E(v,h) = \frac{1}{2}(v-a)^T(v-a) - b^T h - h^T W v, \tag{3.7}$$

as seen in [12] where the $W \in \mathbb{R}^{N_h \times N_v}$ is the weight matrix representing the connection between visible and hidden units, $a \in \mathbb{R}^{N_v \times 1}$ is the bias vector of visible units and $b \in \mathbb{R}^{N_h \times 1}$ is the bias vector of hidden units.

Using the energy function, each connection between a visible unit and a hidden unit obtain a calculated probability. Moreover, the posterior probabilities $P(v|h)$ and $P(h|v)$ necessary for the learning process are defined by equation (3.8) and (3.9), respectively, where $\sigma = (1 + e^{-x})^{-1}$ is the element-wise logistic sigmoid function, $\mathcal{N}$ is a Gaussian, and I is the appropriate identity covariance matrix [12].

$$P(v|h) = \sigma(Wv + b), \tag{3.8}$$

$$P(h|v) = \mathcal{N}(v; W^T h + a, I), \tag{3.9}$$

Deep belief networks

Once the first RBM is trained, the hidden layers calculated states could be used as the input data into a new RBM network representing the hidden-to-hidden layer of DNN. Following this principle, it is possible to chain many RBM networks to produce a multilayer generative model, the so-called (DBN) network [18]. The DBN is used as unsupervised training to initialize the DNN network. This initialization leads to higher accuracy compared with the random initialization [22].

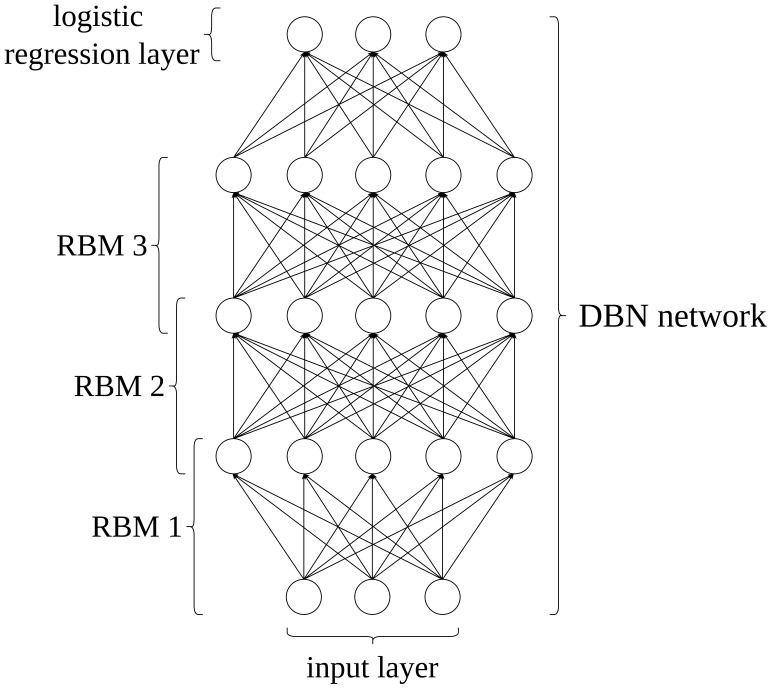

Figure 3.3: DBN network

3.1.3 Data augmentation

Data augmentation techniques in ASR helps to increase the amount of training data in certain corpus. The usage of common used technique as Vocal Tract Length Perturbation (VTLP) [23] showed improved robustness of ASR system trained with speech data corupted with noise [24] or better acuracy of phoneme recognition [23].

An alternative method to VTLP, the speed-perturbation, is proposed in [25]. The input audio signal $x(t)$ is subjected to time wrapping by a factor α. The resulting output signal $x(\alpha t)$ has shifted frequency components of the $x(\hat{\omega})$ by value of frequency ω [25] which corresponds approximately to a shift of the spectrum in the mel spectrogram [26]. These changes in the mel spectrogram are similar to the VTLP method. The speed-perturbation techniques also make changes in the duration of the audio signal and therefore prolong the number of frames in the utterance [25].

3.2 DNN-HMM models

A typical DNN requires fixed-size inputs, which is a problem since the speech signals are time series. Therefore, the speech signal's variable length must be considered if the speech recognition system uses the DNN in its implementation. One of the solutions to this problem is using DNN in combination with HMM. In this design, the HMMs models the speech signal while the DNNs compute the observation probabilities. The DNN takes the acoustic observations as an input to the neural network, and in the output, each neuron represents the posterior probability of continuous density HMM's state [12]. Figure 3.4 shows the DNN-HMM model architecture. The early systems with the DNN-HMM model used context-independent phones as labels for DNN training and worked well for smaller vocabulary corpora. Later on, the DNN achieve training with context-dependent triphones, the so-called senons, which led to promising results even in LVCSR [27].

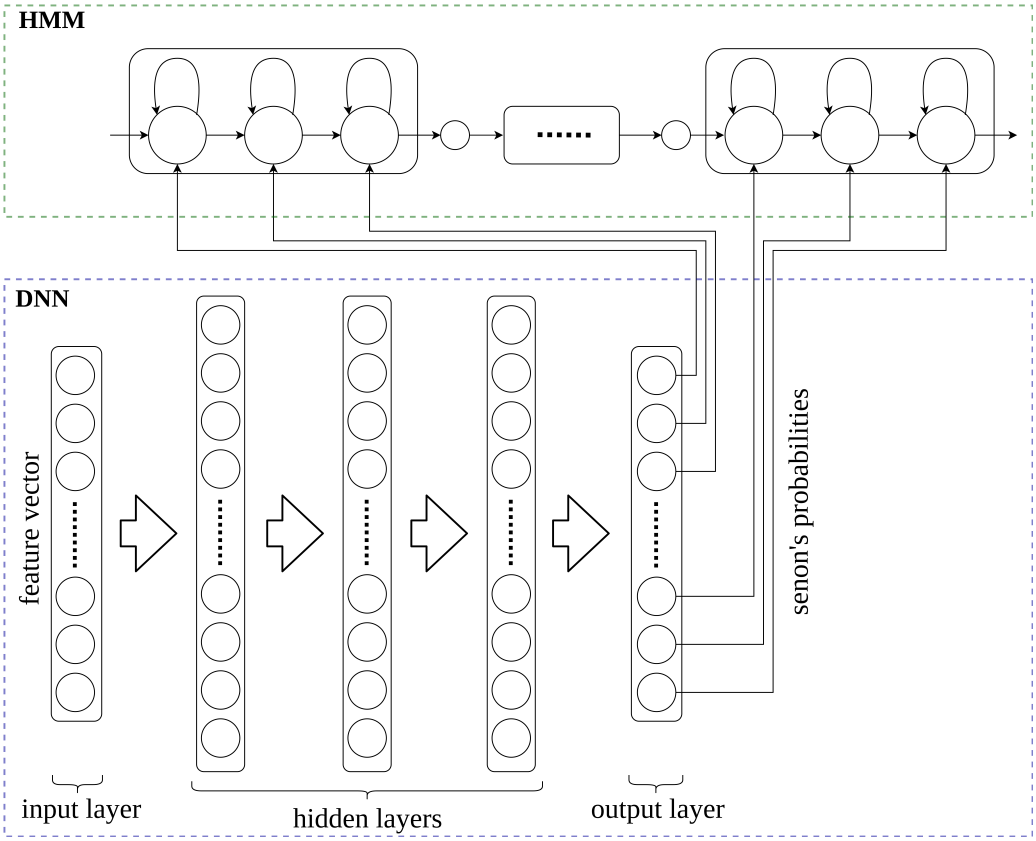

Figure 3.4: DNN-HMM model architecture

3.2.1 Features in DNN-HMM

The DNN input layer usually obtains a vector of short-time features such as MFCC with delta and delta-delta features. The neural networks may also improve using the information of the speaker adaptation features. However, the fMLLR and other speaker adaptation methods need two-pass decoding [28] which is challenging to use in online ASR applications. iVectors are widely used in speaker recognition [29] and may also be used to determine the speaker adaptation features for speech recognition purposes. They capture both the speaker and environment information, which helps rapid and discriminative adaptation of the DFT [30].

The acoustic feature vector can be considered as a sample, generated with a Universal Background Model (UBM) [31], represented as a GMM with diagonal covariance Gaussians [32]. There is an assumption that between GMM speaker-dependent means and speaker-independent means is a linear dependency. The extraction of the iVector is computed based on this assumption and is described more in detail in [32]. The most common method of using iVectors in DNN training is to extract iVectors for each speaker or utterance and then concatenate it to the acoustic feature vector of the corresponding speaker or utterance [33]. The resulting vector is introduced in the input of the DNN.

3.2.2 Time delay neural network

Each layer consists of units with an activation function as in standard DNN. The neurons in TDNN have added delays D_1 up to D_N, where N is the total number of delays. Every input I_m will be multiplied by the weights of each delay w_{id}, plus the weight of the undelayed unit w_i. For the TDNN presented in figure 3.5, 30 weights will be necessary for the number of delays $N = 2$ and number of inputs $M = 10$. Each input will be processed in three different time points - one for the undelayed unit and two for the delayed units. The logistic function comes after the summation (sigmoid function σ is used in this example).

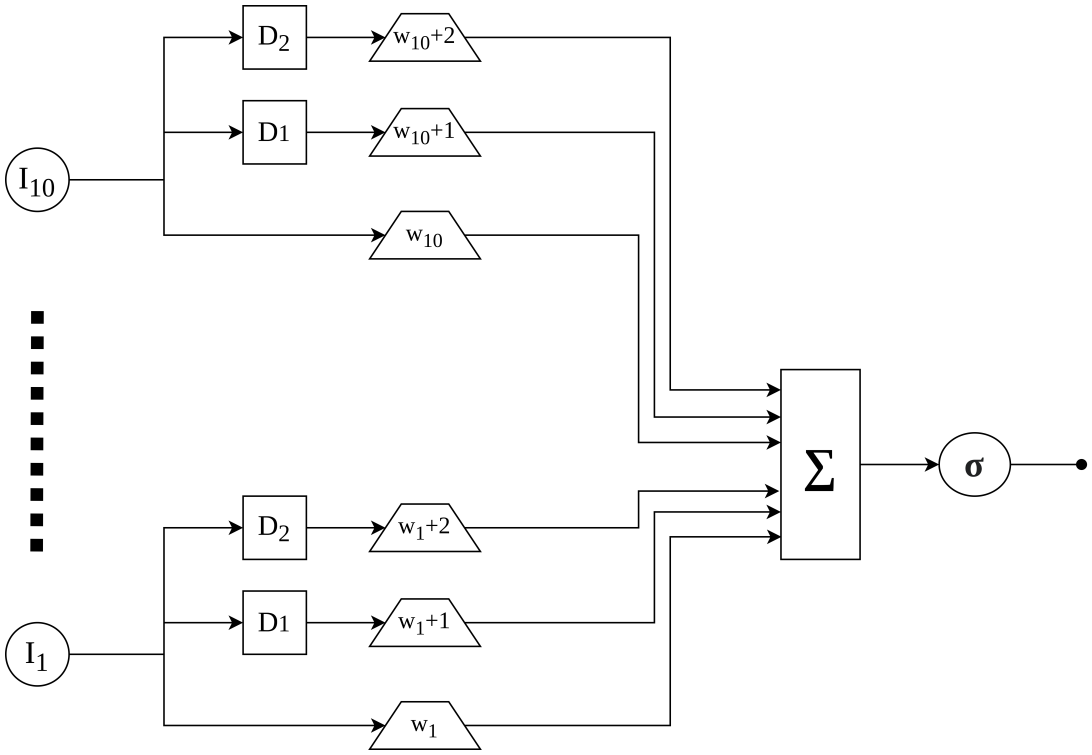

Figure 3.5: Principle of TDNN network

As mentioned in [34], in the standard DNN, the initial transforms are learned for the entire temporal context. The TDNN structure is different in the way that the initial transforms are trained just from a limited context when the deeper layers process the hidden activations from a broader temporal context [34]. Every layer of the TDNN has a different resolution which increases with the deeper layer.

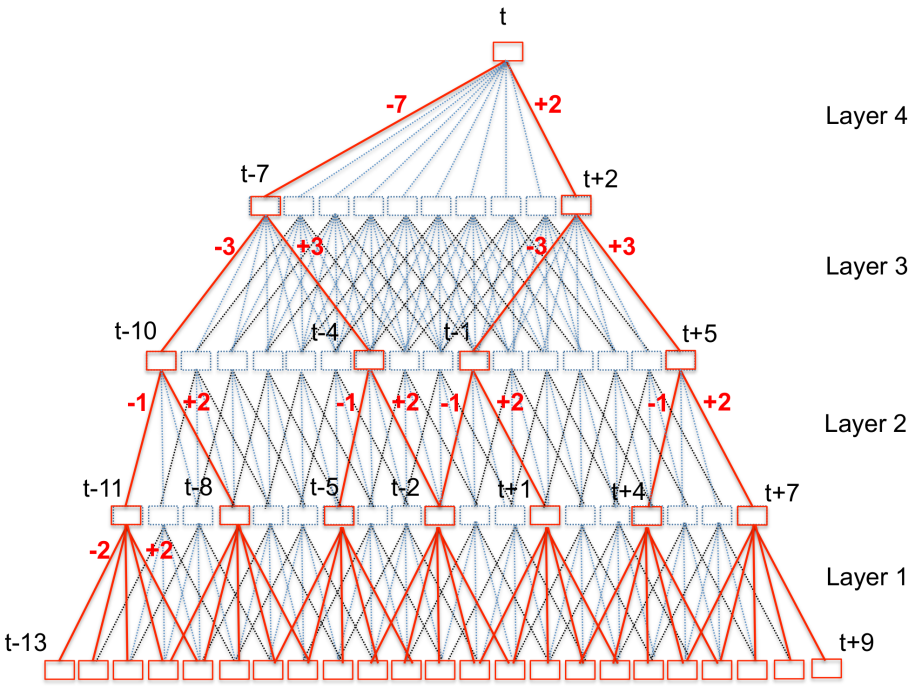

Figure 3.6: TDNN with sub-sampling (red) and without sub-sampling (blue) [34]

The hidden activations are calculated at all time steps, which comes with overlaps and redundancy between adjacent steps. The standard TDNN structure can be improved using sub-sampling proposed in [34]. The neighbouring activations can be sub-sampled if there is a correlation between them. Instead of splicing together the temporal windows of frames, this approach lets to have gaps between the frames. The difference between the typical TDNN and the TDNN with sub-sampling is shown in the figure 3.6

As the training algorithm for TDNN can be used standard backpropagation method with stochastic gradient descent (SGD) updates [34], [35].

3.2.3 Recurrent neural network

Recurrent Neural Networks (RNNs) focus on working with memory when the network's outputs depend on the inputs and the previous states. The dependency in prior states is a question of thorough training of the network. An easy example of RNN is shown in figure 3.7, where x_t is the input of the network, A is one part of RNN, and h_t is the output of the particular layer A. The loop in the layer A can be understood as multiple copies of classic neural networks as shown in figure 3.8.

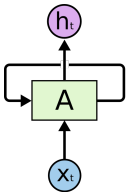

Figure 3.7: Simple part of RNN [36]

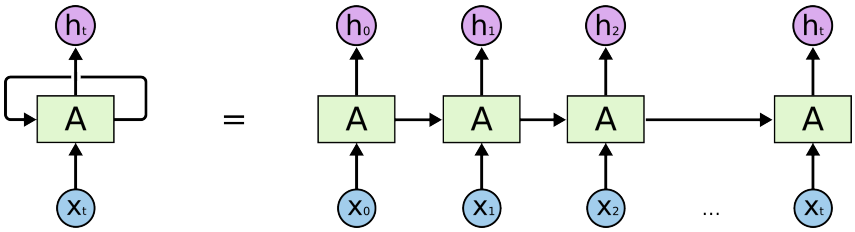

Figure 3.8: RNN expressed as a sequence of neural networks [36]

The computation of the current state h_t is similar to classic MLP with difference that activations come from both the actual input x_t and the delayed hidden layer activation h_{t-1}. Therefore, the current state h_t can be expressed using formula (3.10), where W_{hh} are the weights hidden-to-hidden layer, W_{hx} are the weights input-to-hidden, σ is the activation function and b_h is a bias. The output of the RNN y_t is computed as in MLP following the formula (3.11), where W_{yh} are the weights hidden-to-output.

$$h_t = \sigma(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \tag{3.10}$$

$$y_t = W_{yh}h_t, \tag{3.11}$$

Backpropagation through time

The learning process and network's weights updating are performed by backpropagation, although with some minor upgrades since the network deals with a sequence in time. The basic concept of backpropagation through time (BTT) is to unfold the RNN into a series of feed-forward networks, as shown in figure 3.8. The first step initiates the input layer and the initial hidden layer states. The weights are randomly set. After that, it is possible to execute forward and backward propagation and compute the gradients. The gradients are then averaged, and the weights are continuously updated at the same time. This process should be repeated for all the elements of the input sequence.

The BTT method has a problem dealing with long-term dependencies when the gradient flow will decay sharply through non-linear operations [37]. This phenomenon is called the vanishing gradient problem.

3.2.4 Long short-term memory

Long-Short Term Memory network (LSTM) is a special kind of RNN which can deal with the long-term dependencies causing the vanishing gradient problem. The LSTMs are explicitly designed to learn long-term dependencies.

All the RNNs are possible to unfold into a sequence of classic neural networks. The difference in LSTM is the structure of the neural network module. The typical RNN network would have a simple *tanh* layer as shown in figure 3.9, where the yellow rectangle is a single neural network layer.

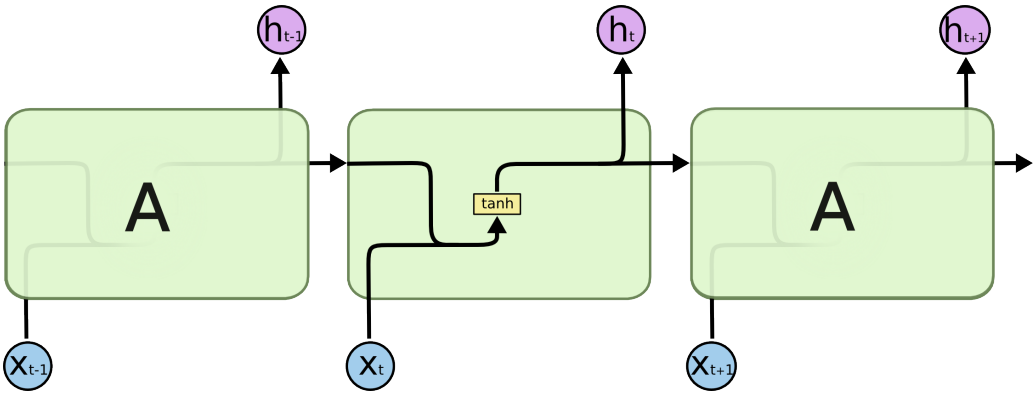

Figure 3.9: Structure of RNN module [36]

Cell structure

The LSMT's module structure is more complex, consisting of four neural network layers as displayed in figure 3.10. Inside the module, four neural network layers are separately trained. Moreover, there are different operations among the separate outputs of each layer. The core value is then the state of the cell C_t. This state is passed through the whole chain of the network, and each cell modifies it. Because of the more complex cell structure, it is possible to control the cell states and outputs much better than in the classic RNN.

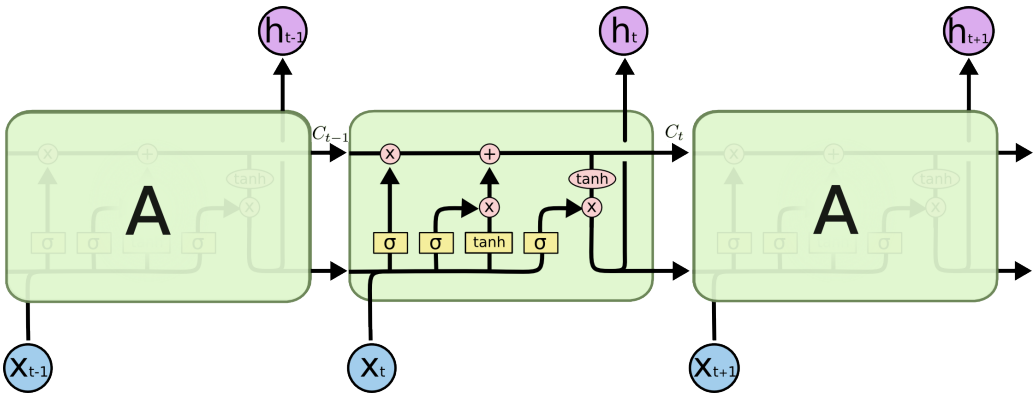

Figure 3.10: Structure of LSTM module [36]

Forward propagation inside the cell

The process in the cell starts with two inputs h_{t-1} and x_t going to the first layer with sigmoid activation function σ as shown in figure 3.11. This part is also called *forget gate layer*, and it decides what information will be removed from the state cell C. The output of this layer f_t can be calculated with the formula (3.12), where the W_f is the weight matrix, and b_f is the bias.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3.12}$$

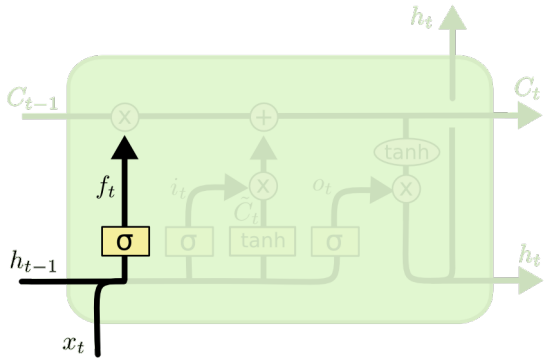

Figure 3.11: Forget gate layer of a cell [36]

The following two so-called *input layers* (figure 3.12) decide about the new information which will be added to the cell state. The equation (3.13) express the output i_t of the first sigmoid layer. This layer decides which values will be updated in the cell state. The output of the second tanh layer $\hat{C}_t$ can be estimated with formula (3.14), where the candidates of new values in the cell state are calculated.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \tag{3.13}$$

$$\hat{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C). \tag{3.14}$$

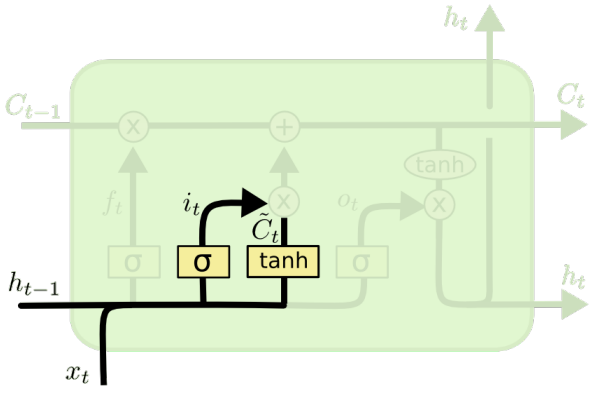

Figure 3.12: Input layers of a cell [36]

The outputs of these two layers are multiplied and sum to the state cell. Figure 3.13 shows the particular operations between the variables from the forget gate layer and the input layers. Then, the new cell state C_t is calculated using the equation (3.15), where $*$ is the element-wise multiplication of the vectors.

$$C_t = f_t * C_{t-1} + i_t \hat{C}_t \tag{3.15}$$

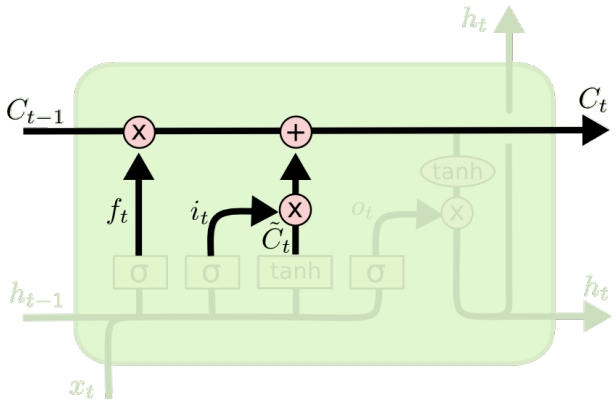

Figure 3.13: A cell state C_t updating [36]

The last sigmoid layer, shown in figure 3.14, influences both the cell state and the output of the cell h_t. The computation is performed in two steps defined by formulas (3.16) and (3.17).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.16}$$

$$h_t = o_t * \tanh(C_t) \tag{3.17}$$

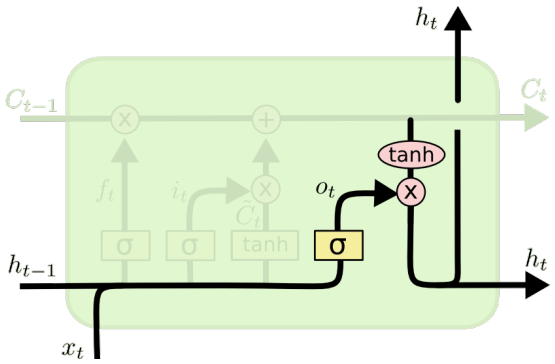

Figure 3.14: Fourth layer of a cell [36]

3.2.5 Convolutional neural network

A convolutional neural network (CNN) is a type of DNN that comes with a new network structure. The two new layers, the *convolutional* and the *pooling* layer, are used alternatively to the hidden layer in the standard DNN. Typically the input layer of the CNN obtains so-called *feature maps* instead of feature vectors. This different input type comes from image processing, where the neural network deals with 2D objects. The feature map can be considered for ASR purposes as a spectrogram with time and frequency axis using static, delta, and delta-delta features.

Locality

One of the fundamental added values of CNN is exploiting the locality characteristic. Each phoneme may have a various decomposition of energy throughout local bands on the frequency axis. Combining filters working on different local frequency regions can lead to better recognition of a particular phoneme. Using local filters also help the robustness against ambient noise, especially if the noise appears in a specific frequency band.

The CNN inputs have to preserve the locality in both the time and frequency axis to acquire stated advantages. The time axis input consists of several frames with a vast context and does not bring locality problems. The difficulties come in the frequency axis where the standard MFCC cannot be applied because DCT projects the spectral energies into a new basis that may not keep locality. Therefore, the frequency axis input can be represented with so-called MFSC features that use the log-energy computed directly from the MFCC (with no DCT) and their delta and delta-delta features [38].

Input layer structure

There are several ways how to arrange the speech features into the feature maps in the input layer. The first option is to organize each MFSC feature type (static, delta, delta-delta) into its feature map. In this case, individual feature type is represented along the frequency axis, as frequency bands, and along the time axis as frames within each context window. The normalization of frequency and time axis is accomplished using two-dimensional convolution. This structure is shown in figure 3.15(b).

The second structure of the input feature maps is displayed in figure 3.15(c) where the feature map is a vector for each feature type for each frame. So if there are 15 frames per window and three feature types, the total number of feature maps in the input layer will be 45, each with a dimension of the number of filter banks (in this example, 40 frequency bands). This arrangement uses one-dimensional convolution along frequency [38] and will be considered for the rest of the chapter for a description of CNN.

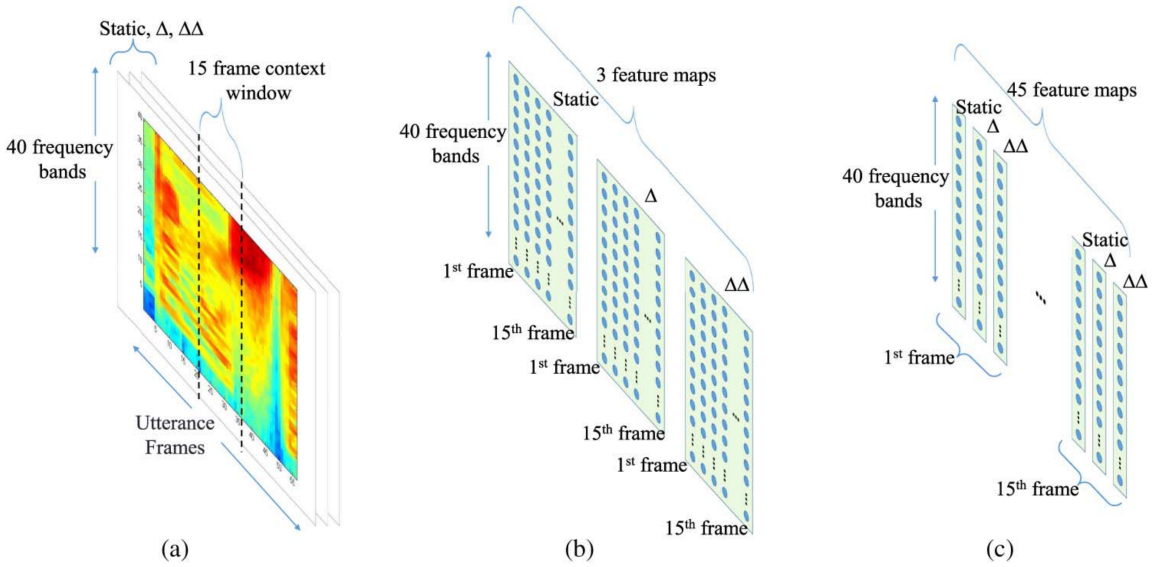

Figure 3.15: CNN structures

Convolution layer

The convolution layer also consists of many feature maps which are connected with the input layer. The principal difference from the hidden layer in standard DNN is that each convolution unit takes input only from the input's local region. Therefore, the unit of the convolution layer represents specific features of the local area. The convolution layer units can be built into various feature maps, which allows them to share the same weights but take input from different places of a previous layer [38].

Each input feature map O_i (I input feature maps in total) is connected to J feature maps in the convolution layer Q_j. The connections are represented by $I \times J$ local weight matrices $w_{i,j}$. The convolution operation deals with the mapping, and for one-dimensional input feature maps, the units of the convolution feature map are computed as

$$q_{j,m} = \sigma\left(\sum_{i=1}^{I}\sum_{n=1}^{F} o_{i,n+m-1} w_{i,j,n} + w_{0,j}\right), \qquad (j = 1, ..., J) \tag{3.18}$$

where σ is the sigmoid function, F is the filter size representing the number of filter bands in every input feature map that each convolution layer unit takes as input, $o_{i,m}$ is the m-th unit of the i-th input feature map O_i, $q_{i,m}$ is the m-th unit of the j-th convolution feature map Q_i, and $w_{i,j,n}$ is the n-th element of the weight vector $\mathbf{w}_{i,j}$ representing the connection between i-th input feature map and the j-th convolution feature map. The equation (3.18) can be expressed in matrix form with formula 3.19, where $*$ symbol is the

convolution operator, O_i is the i-th input feature map and $\mathbf{w}_{i,j}$ local matrix weight.

$$Q_j = \sigma\Big(\sum_{i=1}^{I} O_i * \mathbf{w}_{i,j}\Big), \qquad (j = 1, ..., J) \tag{3.19}$$

The number of feature maps in the convolution layer defines local weight matrices' quantity in the described convolution mapping. In real applications, many local weight matrices are identical or very similar and can be removed. Therefore, the networks lower the number of convolution layers.

Pooling layer

The pooling layer is another layer of CNN that always comes in pair with a convolution layer. It also consists of the same number of feature maps as the corresponding convolution layer. However, the feature maps' size is smaller to decrease the resolution. The pooling layer generalizes the corresponding convolution layer features, which reduce the variation among speakers [39]. The reduction can be performed using the max-pooling function to units in a particular local region specified by a pooling size parameter [38]. The max-pooling can be computed as

$$p_{i,m} = \max_{n=1}^{G} q_{i,(m-1)\times s+n}, \tag{3.20}$$

where G is the pooling size, s is the shift size, which defines adjacent pooling windows overlap.

Training CNN

The learning of the CNN network uses the backpropagation algorithm as standard DNNs. However, there are modifications in order to handle the sparse connections and weight sharing. The basic weight matrix W is replaced by a large sparse weight matrix $\hat{\mathbf{W}}$ [38]. Following the equation (3.19), the convolution output can be computed with $\hat{\mathbf{W}}$ as

$$\hat{\mathbf{q}} = \sigma(\hat{\mathbf{o}}\hat{\mathbf{W}}). \tag{3.21}$$

The input feature vector $\hat{\boldsymbol{o}} = [\boldsymbol{v_1}|\boldsymbol{v_2}|...|\boldsymbol{v_M}]$ consists of row vectors $\boldsymbol{v_m}$, where M is the total number of frequency bands. Since the formula (3.21) is mathematically the same as the output of hidden layer in the standard DNN, the update of $\hat{\mathbf{W}}$ can be computed using backpropagation as

$$\Delta\hat{\mathbf{W}} = \epsilon \cdot \hat{\mathbf{o}}\text{`}\mathbf{e}, \tag{3.22}$$

where $\boldsymbol{e}$ is the error vector, and ϵ is the learning rate. Updating the shared weights adds a sum over the weights in their update stage as

$$\Delta w_{i,j,n} = \sum_{m} \Delta \hat{\mathbf{W}}_{i+(m+n-2)\times I, j+(m-1)\times J}. \tag{3.23}$$

3.3 End-to-end models

The rise of speech recognition applications starts to require running the ASR systems on largely adapted devices that are challenging to handle because of the size of the lexicon and language model with its corresponding WFST. A different approach to implementing an ASR system with only DNNs is eliminating the need for the WFST and LM. These designs are called end-to-end systems, and they map input acoustic feature sequences into the corresponding sequence of labels.

End-to-end ASR systems consist of a number of components combined together. The components are particular DNN network structures when the DNN type used in the system can vary based on the implementation. Applying RNN networks, the usage of LSTM showed promising results [40], [41] as well as Gated Recurrent Unit (GRU) networks [42]. The CNN neural networks were also successfully implemented in the end-to-end systems with encouraging results [43], [44].

3.3.1 Connectionist Temporal Classification

A basic approach of an end-to-end ASR system can be constructed with neural network layers stacked together with the Connectionist Temporal Classification (CTC) loss. This architecture does not need to align the data in advance but requires just trained output of the network since it uses supervised training [45]. The output of the CTC model is a probability distribution of corresponding labels and the so-called blank symbol. Each spike in the speech corresponds to a classification. The parts of the speech without any spikes are represented with the blank symbol [45]. The posterior probability of the output label sequence $P(y|x)$ can be computed as

$$P(y|x) = \sum_{\hat{y} \in B(y,x)} \prod_{t=1}^{T} P(\hat{y}_t|x), \tag{3.24}$$

where B represents all possible alignments, y is the output label sequence, $\hat{y}$ is the output label sequence after removing all blank symbols, and x is the network input. The equation

(3.25) determine the CTC loss $\mathbb{L}$.

$$\mathbb{L} = -E[\log(P(y|x))]. \tag{3.25}$$

The model uses the forward-backward recursion algorithm assuming conditional independence [46]. Figure 3.16 shows the architecture of the CTC loss model.

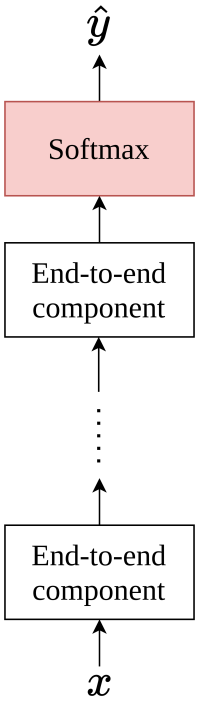

Figure 3.16: CTC loss end-to-end model

3.3.2 Attention model

The attention model uses an attention mechanism that computes the similarity score between the decoder and every encoder frame [45]. The attention mechanism is represented in the model architecture by another component inserted between the encoder and decoder, as shown in figure 3.17. The output of the attention component is the context vector entering into the decoder. Mathematically, the attention process can be express with equations

$$e_{m,l} = energy(h_m^{enc}, h_{l-1}^{dec}), \tag{3.26a}$$

$$a_{m,l} = softmax(e_{m,l}), \tag{3.26b}$$

$$\mathbf{c}_l = \sum_{m=0}^{M-1} a_{m,l} h_m^{enc}, \tag{3.26c}$$

where $e_{m,l}$ is the energy between encoder hidden output h_m^{enc} at time m and decoder hidden output h_{l-1}^{dec} at the label index $l-1$. Formula (3.26c) computes the final context vector $\mathbf{c}_l$ used as the input to the decoder.

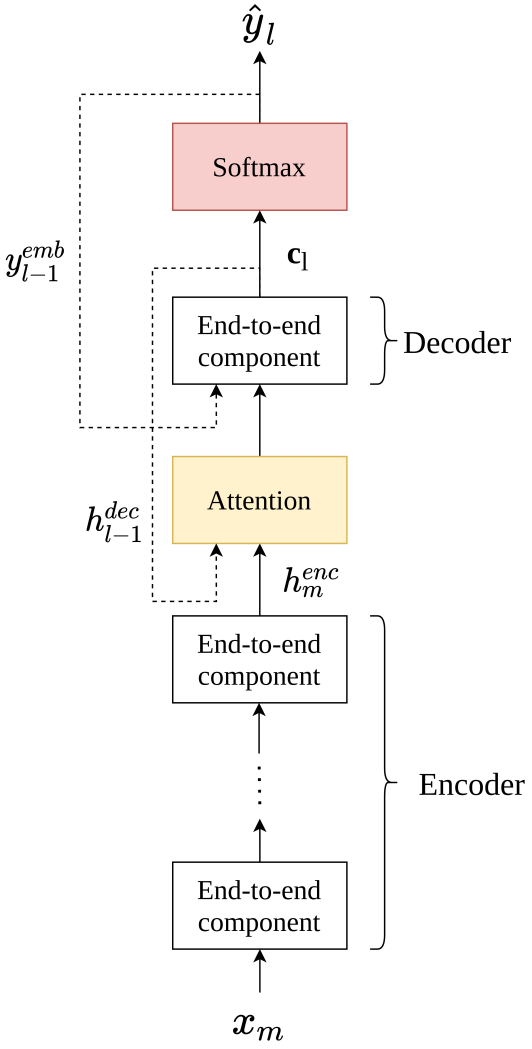

Figure 3.17: Attention model

Chapter 4

DNN-HMM ASR system implementation

The experimental ASR system was implemented using Kaldi, a free, open-source toolkit for speech recognition licensed under the Apache License v2.0. Kaldi is written in C++ language and supports many state-of-the-art designs such as standard methods using the GMM-HMM model or systems with DNN. The fundamental advantages of the Kaldi toolkit are the ability to use finite-state transducers, extensive linear algebra, and its non-restrictive license [3]. Kaldi contains many recipes that serve as examples of how to work with Kaldi tools using bash scripts. Each part of the recognition system is implemented in a separate script. Therefore, it is easier to modify the recipe for a new dataset or adjust parameters or methods based on different preferences.

The DNN-HMM ASR experimental setups use the standard *s5* Kaldi recipe for LVCSR. The overall process is to train the standard GMM-HMM model and obtain triphone models with their WFST graphs. Later, DNN networks are trained using the triphones model, and a new decoding process is performed. The DNN part works with the latest Kaldi recipes of neural networks nnet3. Additionally to the standard nnet3 models, the chain models are used for several implementations. Two DNN architectures, TDNN and LSTM, are adopted in both the standard nnet3 and chain models. Additionally, one more neural network, CNN, is selected using the chain models.

4.1 Recipe for LVCSR

This section describes the implementation of the DNN-HMM LVCSR system used for the experiments of this thesis. The implementation is divided into four stages, where stages 1-3 define the GMM-HMM model and stage 4 describe the training and decoding process of the DNN-HMM system.

4.1.1 Stage 0: Data preparation

The first step in the experimental ASR system implementation is the data preparation which consists of a definition of train, development, test data, creation of language models and lexicons. The corpora's data is divided into several groups of speakers with its utterances. The data preparation process generates list files for each data group necessary later in the ASR process. The lists contain information about the speaker's ID, corresponding utterances, transcript of the utterances, paths to original audio files, and similar.

The LMs are built using the standard n-gram model for train and test datasets. Moreover, several versions of LMs are used in the ASR system when the differences are in the adopted dictionary, which may contain silences or OOV words. The implementation with the Czech corpora uses one silence phone, silence (*SIL*), whether the implementation with the English corpora uses three silence phones, silence (*SIL*), spoken noise (*SPN*) and non-spoken noise (*NSN*).

Firstly, the LM for the train dataset is created. The folder representing the LM for training contains lists of words and phones and a file *topo*, which represents the states and transitions among states for HMM model. In the experimental setups, three and five states were used to describe silence and non-silence phones, respectively. Further, *L.fst* file is created for the WFST method in the decoding process based on the lexicon. Then, the LMs for each test dataset are created with additional *G.fst* file representing the grammar G in the WSFT algorithm. The LM is firstly built without the silence probabilities. Later, the silences are added into the lexicon, and another LM is created.

4.1.2 Stage 1: Feature extraction

The implemented ASR system uses MFCC features. The features extraction process begins with a definition of the frame length, where 25 ms frames are used with 10 ms shifts. The preemphasis, offset elimination, and multiplication of Hamming window are performed for each frame. Next, the power spectrum is computed using fast Fourier transform (FFT). The mel bank consists of 23 overlapping triangular bins with equally spaced centers in the mel-frequency domain. The cutoff frequency of mel frequency filter banks is from 20 Hz to 7800 Hz. The energy is calculated in each of these mel bins and takes the log value of the energies. Next, the 13 cepstral coefficients are computed using the cosine transform. The sampling frequency is 16 kHz. Additionally, the CMVN technique described in section 2.1.3 is applied. The normalization is done at the speaker level.

4.1.3 Stage 2: Acoustic modeling in GMM-HMM model

The experimental ASR system uses forced alignment discussed in chapter 2.2.5 to optimize the AM parameters. This alignment is executed in each cycle of the training process. There are several AMs used in the ASR implementation. Each model use the MFCC features normalized with CMVN technique with some extra features. The monophone AM *mono* was expanded into context-dependent triphone models *tri1*, *tri2*, *tri3*, and *tri4*. The AMs also differ in which LM is used. There are both LM without pronunciation and silence probabilities (*sp*) and without them (*nosp*). The overview of the features used in a particular AM is shown in the table 4.1. All the feature types are explained in chapter 2.1.

Model	Features	LM
mono	MFCC	nosp
tri1	MFCC, delta, delta-delta	nosp
tri2	MFCC, LDA, MLLT	nosp
tri3	MFCC, LDA, MLLT, SAT	nosp
tri4	MFCC, LDA, MLLT, SAT	sp

Table 4.1: Overview of AMs used in ASR implementation

4.1.4 Stage 3: Decoding in GMM-HMM model

The next stage of ASR implementation creates the HCLG graph used in the decoding process. The finite-state transducer accepts the probability density function (PDF) IDs in the input and produces word IDs in the output. Once the HCLG graphs for each model are created, the decoding process calculates the final transcript.

4.1.5 Stage 4: DNN training and decoding

The GMM-HMM model is trained at this stage and can provide the necessary alignments for the DNN-HMM model training. The experimental ASR implementations use the latest Kaldi DNN recipes, nnet3 models. Two DNN architectures are adopted in the experimental setups using nnet3 models, TDNN and LSTM. Moreover, the DNN-HMM chain models introduced as part of nnet3 are applied. There are three DNN architectures selected using the chain models, TDNN, LSTM, and additionally CNN network.

Chain models

The chain models use three times smaller frame rate at the neural net output, which lowers the computational time during the test stage. Therefore, the systems should be about three times faster to decode. Furthermore, the system's accuracy is supposed to be approximately 5 % better and the training stage a bit faster. The training is based on the maximum mutual information (MMI) method [47] without the needs of lattices [48]. The training process performs a full forward-backward algorithm on a decoding graph determined from a phone n-gram LM. The HMM topology is transformed to allow the HMM traversal in one state obligated because of the reduced frame rate. So far, the HMM transition probabilities are fixed without any further training. Additionally, a phone LM is generated from the training data phone alignments necessary for the FST creation.

Data augmentation and iVector creation

Before the DNN training process starts, the data and features are adapted to the purpose of DNN training. Firstly, the data augmentation is performed using the speed-perturbation technique. The MFCC features with CMVN normalization are computed for these low-resolution speed-perturbed data necessary for the alignment. The low-resolution MFCCs have dimension of 13 and follow the configuration described in 4.1.2. Secondly, the volume-perturbation is executed on the training data, which helps the trained DNNs to be more invariant to the test data volume. Next, the high-resolution MFCC features with CMVN normalization are determined. The process of high-resolution MFCCs extraction is the same as for the low-resolution MFCCs but with a configuration shown in the table 4.2. The feature vector keeps all the 40 MFCCs which provide the same information as the filterbank features. However, the MFCCs are more efficiently compressible because of the lower correlation.

Parameter	Value	Significance
–use-energy	false	using average of log energy
–num-mel-bins	40	number of mel bins
–num-ceps	40	number of cepstral coefficients
–low-freq	20	low cutoff frequency
–high-freq	-400	high cutoff frequency

Table 4.2: Configuration of high-resolution MFCCs extraction

The second part of the process creates and trains the iVector. In the first step, about a quarter of the high-resolution data are dedicated to training a diagonal UBM model. The principal components analysis (PCA) [49] transform is computed using these data. The diagonal UBM is then trained with 512 Gaussians. Finally, the iVector extractor is trained using all the speed-perturbed data. The iVector dimension is set to 100. The iVector is also extracted for the test data, but in such case, no speed-perturbed data are needed.

TDNN architecture

Firstly, the standard nnet3 model architecture of TDNN is adopted in the experimental setup. The input to the TDNN network consists of two types of features: 40-dimensional MFCC features and 100-dimensional iVector. The iVector is concatenated with five MFCC feature vectors in one 300-dimensional feature vector. Further, the component's decorrelation is performed with the LDA method with an output vector of size 650, which is the input to the TDNN network. Figure 4.2 left shows the nnet3 TDNN network structure with five TDNN blocks. The TDNN block layers are shown in figure 4.1 where each layer proceed following operations: affine transformation, ReLU activation, and batch normalization, respectively.

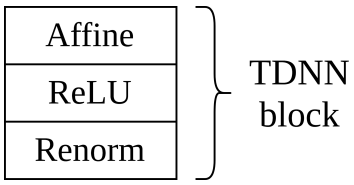

Figure 4.1: TDNN block layers

As described in chapter 3.2.2, TDNN blocks perform 1-dimensional convolution on input vectors at time t. The first TDNN block does not apply any convolution, the second and third process the input vectors at times $t-1$, t, $t+1$, the fourth block at times $t-3$, t, $t+3$, and the fifth block at times $t-6$, $t-3$, and t. The output block applies the affine transform and the log-softmax function and produces a vector of dimension 3344.

The training proceeds with the cross-entropy method. Training parameters remain with their default values except for *–use-gpu* which is modified from value "yes" to value "wait". This option allows training the neural network with fewer GPU devices than the number of jobs during the training. The GPU must be set to the exclusive mode in order to use the "wait" option. Finally, the decoding process is done in the standard way as in the GMM-HMM model with corresponding test data.

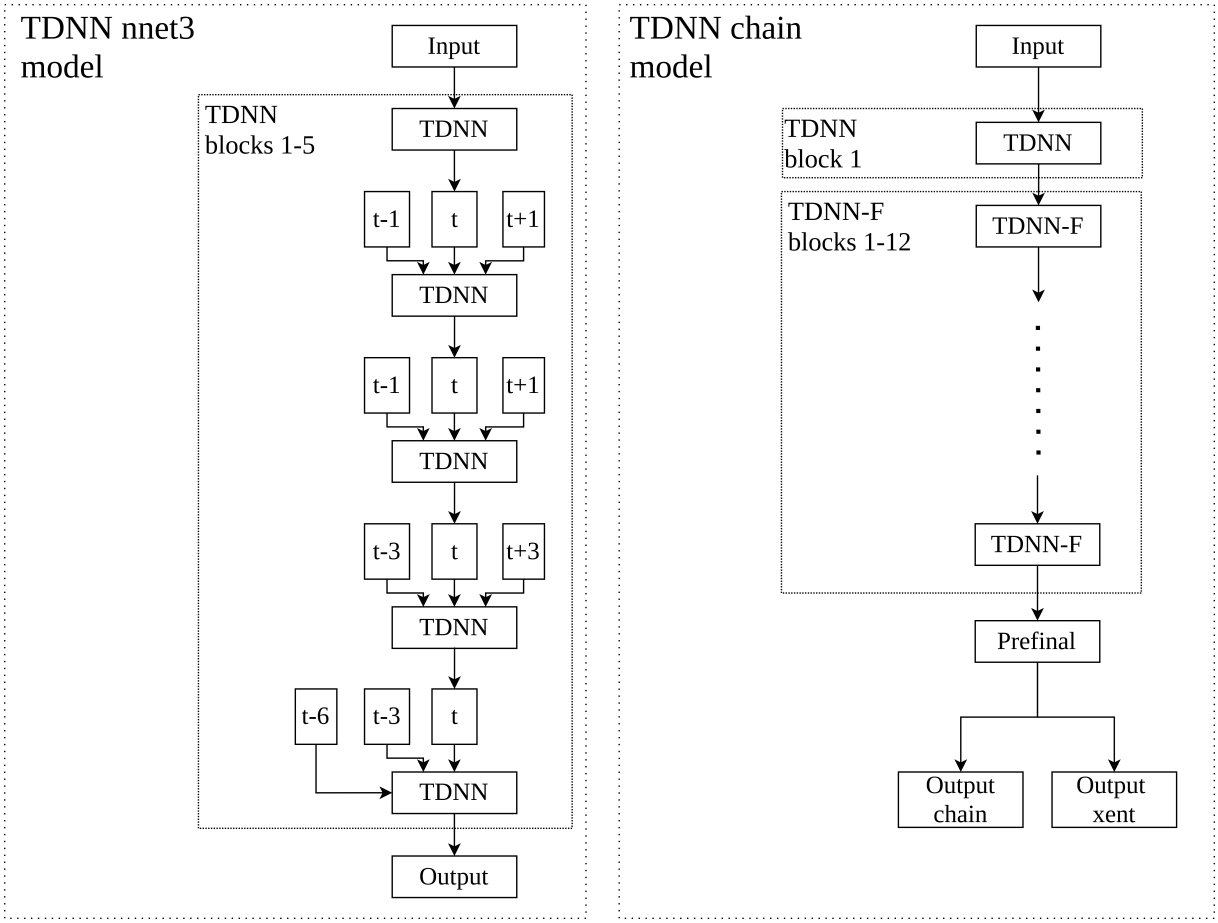

Figure 4.2: TDNN network structure

Figure 4.2 right shows the TDNN architecture using the chain models. The first TDNN block with a hidden layer's dimension of 1024 follows twelve factored TDNN blocks (TDNN-F) whose structure is based on the TDNN but is trained from a random start with one of the two factors of each matrix constrained to be semi-orthogonal [50]. Figure 4.3 shows the difference in structure of TDNN and TDNN-F block. The TDNN-F block introduces a linear bottleneck layer where the weight matrix from the standard hidden layer to the bottleneck layer is supposed to be semi-orthogonal. The dimension of the TDNN-F hidden layer is 1024, with a bottleneck layer dimension of 128. The output consists of two blocks: the first block based on lattice-free MMI criteria (chain) and the second output block using cross-entropy criteria (xent). Each output block produces a vector with a dimension of 2840.

The training process of chain-based TDNN differs in several ways described early in chapter 4.1.5. A new LM model is built based on the chain topology, which is later utilized to create a phonetic decision tree. The decision tree works with the alignments of speed-perturbed data obtained during the data augmentation and iVector creation process. The decoding with chain models is the same as in the standard nnet3-based TDNN.

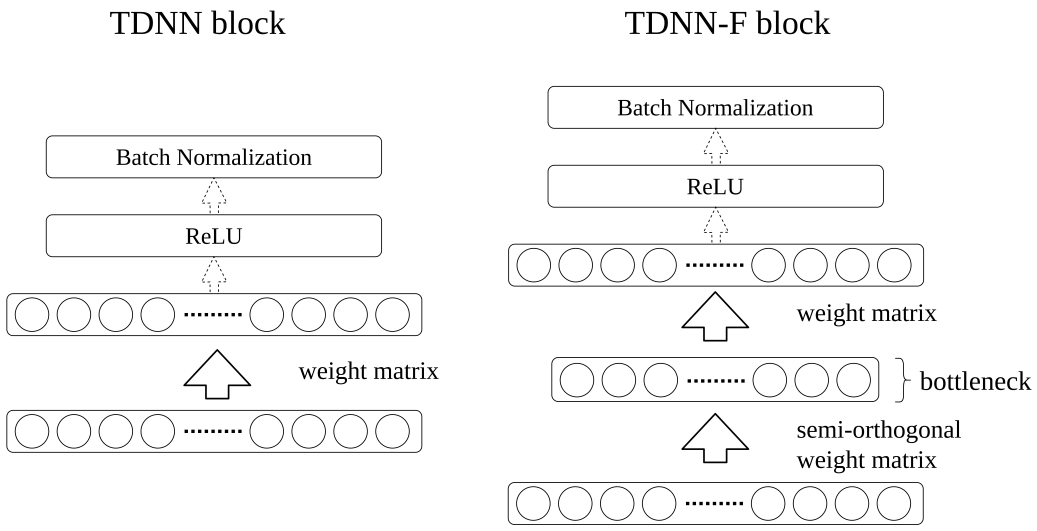

Figure 4.3: TDNN and TDNN-F block structure

LSTM-TDNN architecture

The LSTM-TDNN network is adopted using the standard nnet3 models has the same 300-dimensional input feature vector as the TDNN. Figure 4.4 left represents the LSTM-TDNN architecture with six TDNN blocks of dimension 520. The TDNN block has the same composition as in the standard TDNN network shown in figure 4.1. Additionally, three LSTM blocks are added to the network. Each of the LSTM blocks has a cell dimension of 520 and uses the so-called projected LSTM (LSTMP) [51]. The LSTM block contains both recurrent and non-recurrent projection layers with a dimension of 130. The output block performs the affine transform and the log-softmax function and returns a 3344-dimensional vector.

The network training is performed with the cross-entropy and produces AMs in output chunks. This approach differs from the feed-forward TDNN training, which produces individual outputs. Additionally, the shrinkage stage is added, which scales the model parameters when the derivative averages at the non-linearities are below the threshold. Training parameters are set to their defaults value except for the *–use-gpu*, which is changed to "wait". The decoding process is done in the standard way as in the TDNN model with corresponding test data.

The LSTM-TDNN architecture using the chain models is shown in figure 4.4 right. It is the same feature vector with a dimension of 300 as in the standard nnet3 LSTM-TDNN network. The first part of the network constitutes four TDNN blocks with a dimension of 448. Next, the LSTM block with a cell dimension of 384 has the chain model structure with a 256-dimensional bottleneck layer. Finally, two blocks compose the network output. The chain output block uses the lattice-free MMI criteria and outputs a 2832-dimensional

vector, while the xent output block operates on a cross-entropy objective with an output vector of the same dimension 2832. Training is similar to the nnet3 models with a few modifications to satisfy the chain model's criteria. The decoding process persists without any changes.

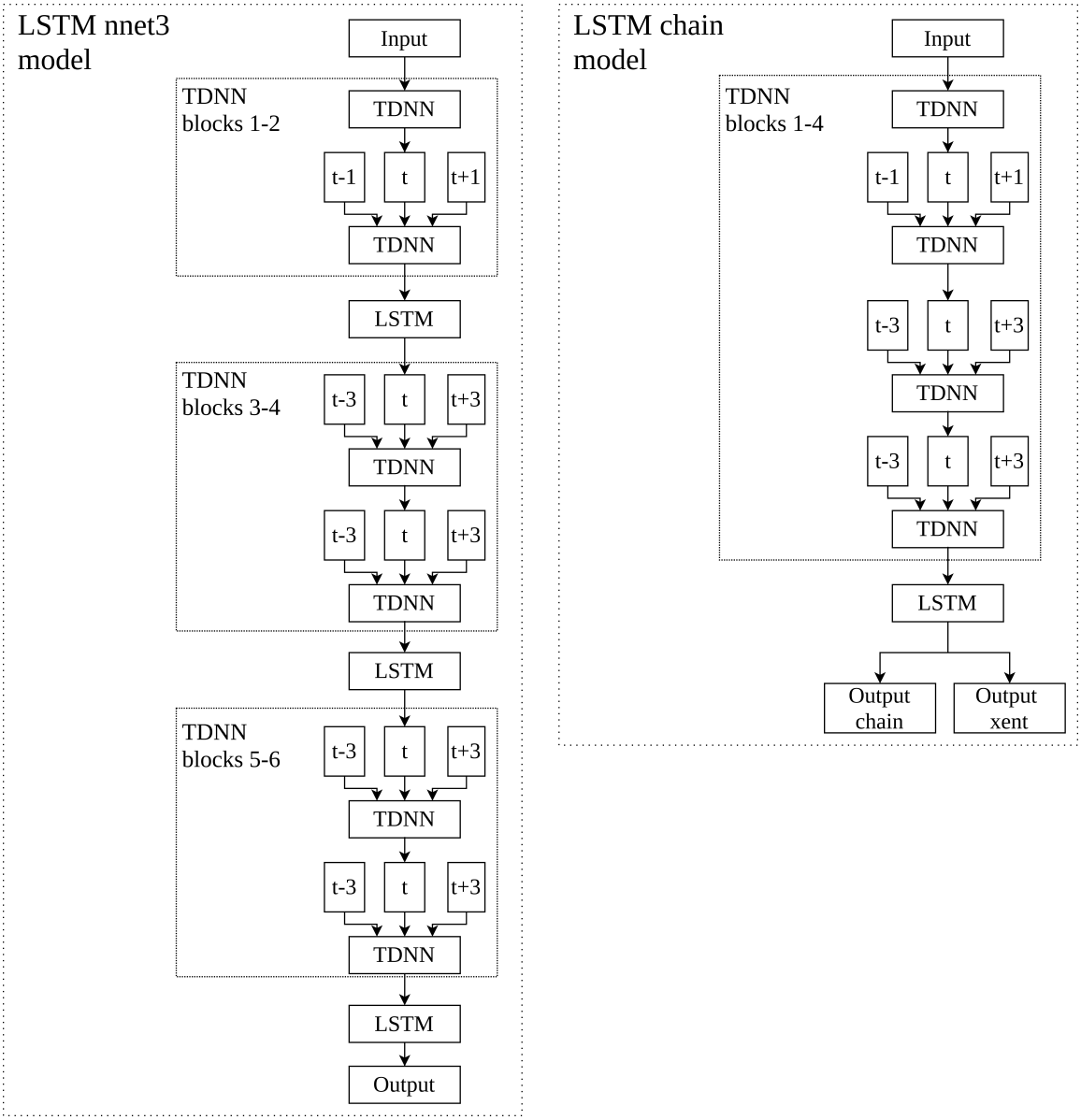

Figure 4.4: LSTM-TDNN network structure

CNN-TDNN architecture

The CNN-TDNN network is implemented only for the chain models. The input of the CNN-TDNN network consists of two types of features, the iVector, and the Mel filterbanks. Since the standard input feature vector is with MFCC features, there is an additional layer on the network input which converts the MFCC features into Mel filterbanks using IDCT transform. Moreover, the feature vector is transformed and sorted into the

matrix form. The CNN-TDNN network receives an input matrix containing values from iVector and Mel filterbanks vectors.

The network architecture consists of six CNN blocks followed by nine TDNN-F blocks as shown in figure 4.5. Each CNN block size is represented with a matrix with a height (rows) and a number of filters (columns). The size of the matrix can differ in the input and the output of a particular CNN block. Additionally, parameters 'time offset' and 'height offset' define the context, i.e., which frames are processed. There are three components used in the CNN block: convolution, ReLU, and batch normalization. The convolution component applies time and feature space convolution on the input matrices, followed by operation ReLU and batch normalization. The TDNN-F blocks have a dimension of 1024 with a bottleneck size of 128. The network output works with two types of output blocks. The chain output block with lattice-free MMI criteria produces a 2840-dimensional vector, and the xent output block with cross-entropy criteria produces a vector of a dimension 2840. The training process is the same as in the TDNN chain model training with settings adapted to the CNN-TDNN network. The training parameters are left with their default values. The decoding process persists without any changes.

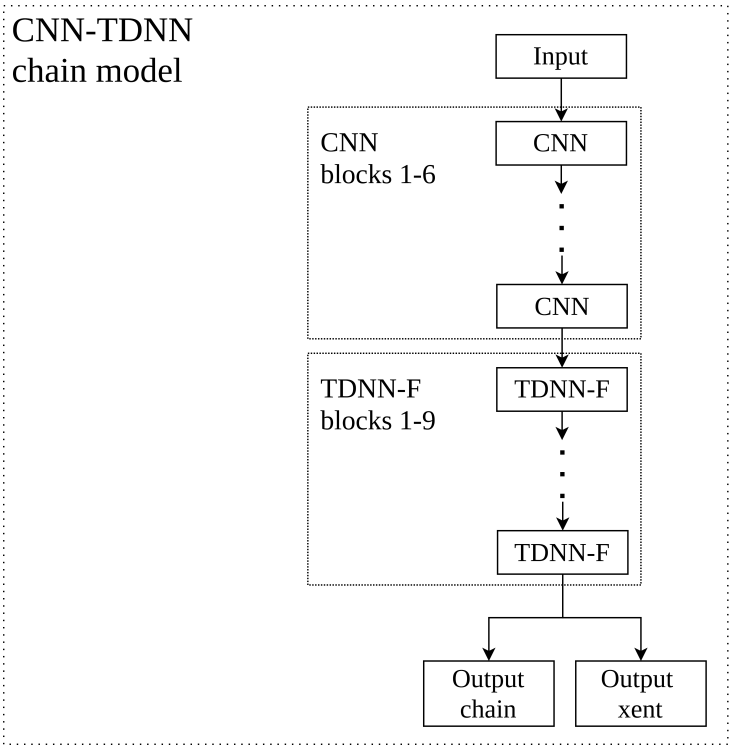

Figure 4.5: CNN-TDNN network structure

Chapter 5

Experimental part

The experiments were realized for English and Czech languages using recipes described in the previous chapter. The nnet3 recipes used in the experiments are primarily enhanced for the English Wall Street Journal (WSJ) corpus. Then, the recipes were adapted so that they could also be used for the Czech corpora. The evaluation was done using the standard WER metric, which works on the word level and can be calculated as

$$WER = \frac{S + D + I}{N} \times 100, \tag{5.1}$$

where S stands for substitution, which represents the count of correct words replaced by incorrect ones in the recognized sentence, D (deletion) expressed when a valid word is missing in the recognized sentence, I (insertion) represents words added to the sentences even though they do not belong there, and N is the number of words in the reference transcript.

An ASR setup for the English language uses one corpus, WSJ, divided into train and test datasets. Data for Czech languages are constructed from more corpora joined together. A train dataset consists of TEMIC and SPEECON corpora. Moreover, data from the SPEECON corpus are divided and also used in the test dataset along with CtuTest, and CzLecDSP corpora.

The experimental part aims to testify the accuracy of DNN-HMM over the standard GMM-HMM models for the English and Czech language. Additionally, investigate the improvements in DNN-HMM systems when the DNN is trained with alignments from the GMM-HMM system with different model parameters. Lastly, compare the efficiency of the latest Kaldi DNN models, the nnet3, and the chain models in the speed and accuracy domain.

5.1 Results for various DNN-HMM ASRs

The first part of the experiments compares the accuracy of the DNN-HMM models for English and Czech languages. The results evaluate three DNN types, TDNN, LSTM, and CNN. Additionally, the DNN-HMM models are compared with the GMM-HMM model, whose computed alignments were used for the DNN training. The GMM-HMM model works with trigram LM and *tri3* triphone AM described in table 4.1.

5.1.1 DNN-HMM systems for English

This chapter introduces the adopted corpus used in the DNN-HMM system with the English language, followed by a section with achieved results. The GMM-HMM model was trained using all data from the train and test datasets presented later in this chapter by table 5.2 and 5.3, respectively, while the DNN network of the DNN-HMM model was trained using only the *Train284* train dataset. The results were obtained for two test dataset, *Dev93* and *Eval92* also presented later in this chapter.

Adopted corpora

Wall Street Journal (WSJ) dataset used in the implementation consists of close-talking microphone English recordings. According to the WSJ design [52], corpus supports both speaker-dependent and speaker-independent training. There is an equal portion of verbalized and non-verbalized punctuation, different speaker adaptation materials, equal numbers of male and female speakers with various voice quality and dialect. Most of the utterances are recorded in read mode when the speaker is asked to read newspaper text paragraphs. Dataset also contains a small amount of spontaneous speech for comparison. Test data are constructed in order to examine the performance of speaker-dependent and speaker-independent modification and variable size of lexicon with open and closed vocabulary to test so-called out-of-vocabulary model [52].

The dataset is divided into two subsets *wsj0* and *wsj1* as shown in table 5.1. The data are further combined and organized into train and test groups. Table 5.2 and 5.3 shows the distribution of train and test dataset, respectively.

	wsj0	ws1	wsj0 + ws1
Speakers	84	200	284
Utterances	7240	30276	37516

Table 5.1: WSJ data structure

	Train284	Train84
Speakers	282	83
Utterances	37318	7138

Table 5.2: WSJ train datasets

	Dev92		Dev93		Eval92		Eval93	
	5k	20k	5k	20k	5k	20k	5k	20k
Speakers	10	10	10	10	8	8	10	10
Utterances	410	403	513	503	330	333	215	213

Table 5.3: WSJ test datasets

Results

Table 5.4 presents the resulting WER for each experimental ASR system implemented for the English language. The results of a model DNN (nnet2) written in *italics* are taken over from [53] and were computed for the DNN-HMM model with a simple feed-forward DNN network using older Kaldi nnet2 models. Overall, the decoding of *Eval92* test dataset had better WER than the *Dev93* dataset, except the DNN (nnet2) model. Each DNN-HMM model within nnet3 had better accuracy than the GMM-HMM model when the best result was achieved for *Eval92* dataset using TDNN (chain) architecture with WER of 2.68 % and for *Dev93* dataset using CNN-TDNN (chain) architecture with WER of 4.44 %.

Model	Dev93	Eval92
DNN (nnet2)	*7.02*	*7.25*
GMM-HMM	11.16	7.25
TDNN (nnet3)	6.64	3.70
TDNN (chain)	4.68	**2.68**
LSTM-TDNN (nnet3)	6.80	3.97
LSTM-TDNN (chain)	5.57	3.14
CNN-TDNN (chain)	**4.44**	2.69

Table 5.4: WER results for English test datasets

5.1.2 DNN-HMM systems for Czech

The second part of the DNN-HMM comparison experiment is realized for the Czech language. The data consisted of various Czech corpora describe more below. The train dataset included two corpora defined by table 5.5 and was used for the GMM-HMM training and the DNN-HMM training. The LM uses Czech National Corpus (CNK) dictionary with a size of 340000 words. There were three test datasets (presented by table 5.6) used to evaluate both the GMM-HMM and DNN-HMM models.

Adopted corpora

SPEECON is a Czech corpus with recordings from 550 adult speakers, taped in various environments (office, public areas, cars, and so) using four microphones (headset, close distance, medium distance, and far distance). The experiments use only the recordings from the headset microphone. The audio files use a 16 kHz sampling frequency and a raw 16bit Pulse-Code Modulation format [54]. Data from the SPEECON corpus was used both for training and testing the implementations. The SPEECON corpus was combined with the Czech corpora described more below to obtain a larger dataset. Table 5.5 and 5.6 shows the distribution of train and test data, respectively.

	SPEECON	TEMIC
Speakers	225	301
Utterances	60877	12724

Table 5.5: Train datasets for experiments with Czech language

	SPEECON	CtuTest	CzLecDSP
Speakers	24	40	8
Utterances	699	577	995

Table 5.6: Test datasets for experiments with Czech language

TEMIC corpus consists of Czech language recordings from 1000 speakers. The audio files were recorded in a car with two channels and three different microphones. Even though the recording was done in various car conditions (engine turn on/off, driving), all the experiments in this thesis use just the recording from a standing car with engine turn off and the headset microphone to match with the SPEECON dataset. Both the training and testing stages of the experiment use data from the TEMIC dataset as shown in the table 5.5 and 5.6.

CtuTest is a private corpus of Czech Technical University in Prague. The dataset contains read speech of sentences from a journal with various topics with a total length of approximately one hour which is relatively more minor than other corpora used in the experiments. Therefore, the CtuTest database is used for testing purposes. The recordings use 16 kHz sampling frequency and 16 bit linear Pulse-Code Modulation (PCM) format.

CzLecDSP designed by Czech Technical University in Prague consists of audio recordings from lectures about digital signal processing. It is a dataset with spontaneous speech, but the utterance's content is more formal. The audio files were recorded using 16 kHz sampling frequency and 16 bit linear PCM format. The implementation uses this dataset only for the testing stage.

Results

Table 5.7 presents the resulting WER for each experimental ASR system implemented for the Czech language. The results of a DNN (nnet2) model written in *italics* are taken over from [55] and were computed for the DNN-HMM model with a simple feed-forward DNN network using older Kaldi nnet2 models. The best result for the Czech language was also accomplished with the TDNN network with WER 10.78 % for the CtuTest dataset, which is approximately 9 % improvement compared to the GMM-HMM model.

Model	CtuTest	SPEECON	CzLecDSP
GMM-HMM	*17.0*	*23.4*	*41.3*
DNN (nnet2)	*15.2*	*21.1*	*37.4*
GMM-HMM	19.68	23.11	47.18
TDNN (nnet3)	18.83	18.33	31.60
TDNN (chain)	**10.78**	**11.55**	25.98
LSTM-TDNN (nnet3)	17.39	18.74	31.53
LSTM-TDNN (chain)	13.71	13.26	32.23
CNN-TDNN (chain)	11.21	**11.55**	**21.86**

Table 5.7: WER results for Czech test datasets

All implementations using nnet3 and chain models had better results than the DNN nnet2 model except when using the CtuTest dataset, where TDNN and LSTM-TDNN standard nnet3 models had worse results than the simple DNN nnet2 model. The reference GMM-HMM model for nnet2 using the CtuTest dataset is better trained than the GMM-HMM model used for nnet3 and chain models. Therefore, they may be harder to train

in order to achieve better results. The GMM-HMM for nnet2 used CtuCopy tool [56] for the feature extraction while the GMM-HMM model for nnet3 used standard recipe **make_mfcc.sh** included in the Kaldi toolkit, which may be one of the reasons for getting different WER results in the GMM-HMM models. However, the chain models achieved significantly better results even though they were trained with worse reference GMM-HMM model.

The DNN-HMM system for the Czech language achieved significant accuracy gains over the GMM-HMM model as the DNN-HMM system for the English WSJ corpus. Hence, the adaptation of the DNN-HMM models to the Czech corpora can be considered adequate and effective.

5.2 Training variants of DNN-HMM

The second part of the experiments examines the GMM-HMM models trained with different LM and analyzes their influence on the resulting DNN-HMM system accuracy. There were used two DNN types in the DNN-HMM model, TDNN and LSTM, to evaluate these experiments. The results were computed for the English language using WSJ corpus with the same data structure described in section 5.1.1.

5.2.1 Impact of the n-gram model

Two n-gram LMs were used in this experiment - the trigram and fourgram models. Firstly, the GMM-HMM model decodes with a trigram LM model to obtain the reference alignments for the DNN-HMM model training. Secondly, the fourgram LM for the GMM-HMM model is created using lattice LM rescoring [57] in ARPA format. Then, the GMM-HMM model decodes with the rescored LM to obtain reference alignments for the DNN-HMM training. The experiments use the tri3 AM.

Table 5.8 presents the achieved results using WER metric. Using the fourgram model resulted in better accuracy in all tested systems for both test datasets. The improvements are not huge when WER always decreased less than 1 %.

	GMM-HMM		TDNN		LSTM-TDNN	
LM type	Dev93	Eval92	Dev93	Eval92	Dev93	Eval92
tri-gram	9.40	5.42	6.64	3.70	6.80	3.97
four-gram	8.56	4.52	6.00	3.14	6.04	3.56

Table 5.8: WER comparison of models using LM with tri-gram and four-gram model

5.2.2 Impact of silence and pronunciation probabilities

This part evaluates the impact of the training model with a dictionary with (*nosp*) and without (*sp*) silence and pronunciation probabilities. The GMM-HMM model is trained using the particular trigram LM and generates the reference alignments for the DNN-HMM model.

Table 5.9 shows the results for each LM settings using WER metric. The improvements in accuracy can be seen in each example of the DNN-HMM model. However, the WER difference between *nosp* and *sp* dictionary is not significant, and the usage of silence and pronunciation probabilities had an even smaller impact than the fourgram LM.

	GMM-HMM		TDNN		LSTM-TDNN	
LM type	Dev93	Eval92	Dev93	Eval92	Dev93	Eval92
nosp	9.40	5.42	6.64	3.70	6.80	3.97
sp	9.33	5.37	6.34	3.62	6.16	3.83

Table 5.9: WER comparison of models using dictionary with and without silence and pronunciation probabilities

5.3 Comparison of nnet3 and chain models

This section compares two Kaldi DNN models: the standard nnet3 model and the chain model. The results evaluate the accuracy changes between these two models and their differences in training and decoding time. One of the main benefits of using chain models instead of the standard nnet3 should be its significant reduction in decoding time which may be very important in online ASR applications.

5.3.1 Model processing time

Table 5.10 presents the resulting times of training and decoding stages for two DNN structures, TDNN and LSTM, using the WSJ corpus. Our results confirm the assumption that the chain models decode faster than the nnet3 models.

	TDNN		LSTM-TDNN	
	Train time	Decode time	Train time	Decode time
nnet3	8.95 h	6.10 m	9.08 h	19.05 m
chain	19.35 h	2.03 m	7.30 h	2.25 m

Table 5.10: Training and decoding time of nnet3 and chain models

The decoding time of a chain model with a TDNN network decreased approximately three times than the implementation based on nnet3. However, the training time is more than twice longer for the TDNN chain model. Even though the decoding time is usually more critical in real applications, this disadvantage of longer training time may be considered. Chain models using the LSTM network come with an even more significant improvement with more than eight times faster decoding time than the nnet3 implementation. Moreover, the training time decreased approximately by 20 %.

5.3.2 Model accuracy

This section compares the accuracy of nnet3 and chain models with TDNN and LSTM systems using the WER metric. Table 5.11 presents the WER rates for Dev93 and Eval92 dataset of English WSJ corpus. The accuracy improvements can be seen for both TDNN and LSTM-TDNN chain models. The WER declined not more than 1 % for chain models, except the TDNN chain model using the Dev93 test dataset, where the WER decreased almost by 2 % compared to the TDNN nnet3 model. Table 5.12 shows the WER results for Czech corpora. The chain models achieved better accuracy except for the LSTM-TDNN model for the CzLecDSP dataset when the chain models had slightly worse results than the standard nnet3 model. Generally, the chain models with the TDNN network accomplish greater improvement in the accuracy over the nnet3 models than the models with the LSTM-TDNN network.

	TDNN		LSTM-TDNN	
	Dev93	Eval92	Dev93	Eval92
nnet3	6.66	3.58	6.79	3.85
chain	4.68	2.68	5.57	3.14

Table 5.11: WER comparison of nnet3 and chain models for English

	TDNN			LSTM-TDNN		
	CtuTest	SPEECON	CzLecDSP	CtuTest	SPEECON	CzLecDSP
nnet3	18.83	18.33	31.60	17.39	18.74	31.53
chain	10.78	11.55	25.98	13.71	13.26	32.23

Table 5.12: WER comparison of nnet3 and chain models for Czech

Chapter 6

Conclusions

The first part of this thesis introduced the theory of GMM-HMM based speech recognition and DNN networks used in the ASR systems. Two DNN based speech recognition systems, the DNN-HMM and end-to-end, were presented with their properties and possible architectures. The DNN-HMM ASR system was further studied and implemented using the Kaldi toolkit. The DNN-HMM implementations used the newest Kaldi DNN nnet3 recipes with TDNN, LSTM, and CNN networks.

Inspired by the literature, the thesis investigates the following problems: Firstly, the possible accuracy improvements of the DNN-HMM over the GMM-HMM models for English and Czech languages. The implementation for this experiment is primarily built for English corpus WSJ and then further adapted to Czech corpora. Therefore, the experiment should also reveal whether adaptation to the Czech language results in accuracy augmentations. The experimental results confirmed the accuracy increase of the DNN-HMM model over GMM-HMM for all DNN types when the implementations with TDNN network achieved best results with WER 2.68 % with the English corpus and 10.78 % with the Czech corpus. However, models with the CNN network achieved very comparable results. The DNN-HMM system for the Czech language achieved similar accuracy gains over the GMM-HMM model as the DNN-HMM system for the English WSJ corpus. Therefore, the adaptation of the DNN-HMM models to the Czech corpora can be considered adequate and effective. Secondly, test the impact on the accuracy of the DNN-HMM model trained with alignments from GMM-HMM using LM with different n-gram models and LM with a dictionary with and without silence and pronunciation probabilities. The results showed slight improvements using LM with fourgram over the trigram-based LM and using LM with the dictionary with silence and pronunciation probabilities. However, the accuracy changes of the DNN-HMM model correlated with the changes of the GMM-HMM model used to train the DNN. Therefore, the modification of LM seemed to have just minor significance. Lastly, compare the accuracy and processing time of the DNN-HMM

model implemented using standard nnet3 models and nnet3 chain models. The systems based on chain models achieved considerably better results for all DNN types. Decoding time was significantly reduced for all DNN-HMM systems using the chain models when the LSTM-TDNN model decoded almost nine times faster than implementation using standard nnet3 models. Additionally, the training time also decreased approximately by 20 % for the LSTM-TDNN network. However, the model with a TDNN network trained more than twice slower.

Due to the time and hardware limitations, the thesis did not contain the experiments with the end-to-end systems. The Kaldi toolkit includes some recipes for end-to-end systems that could be considered to use for experimental implementations. An interesting experiment could compare the accuracy of the end-to-end systems with DNN-HMM models with the same corpora since the size of the training dataset is critical for DNN based ASR system. Further, several Python-based toolkits are developed combining the Kaldi toolkit with Python-based layers. The PyTorch-Kaldi speech recognition toolkit [58] presents a different approach for implementing the DNN-HMM ASR system using the Kaldi for the feature extraction, label computation, and decoding, while the PyTorch-Kaldi toolkit manages the DNN network part. Another tool is PyKaldi [59] providing Python wrappers for the C++ code in the Kaldi toolkit. Combining Kaldi and Python can deliver further advantages in building particular applications, which may be more user-friendly for commercial use.

Bibliography

[1] T. J. Watson, F. Jelinek, L. Bahl, and R. Mercer, *Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech*, ser. Research reports // IBM. IBM Thomas J. Watson Research Division, 1974. [Online]. Available: `https://books.google.cz/books?id=agPvPgAACAAJ`.

[2] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Jan. 1994, ISBN: 978-1-4613-6409-2. DOI: `10.1007/978-1-4615-3210-1`.

[3] 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 *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Catalog No.: CFP11SRW-USB, IEEE Signal Processing Society, Dec. 2011.

[4] J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong, *Robust automatic speech recognition: A bridge to practical applications*. Jan. 2015, pp. 1–286.

[5] G. Sarada, T. Nagarajan, and H. Murthy, "Multiple frame size and multiple frame rate feature extraction for speech recognition", Jan. 2005, pp. 592–595, ISBN: 0-7803-8674-4. DOI: `10.1109/SPCOM.2004.1458529`.

[6] J. Pinto, S. Prasanna, B. Yegnanarayana, and H. Hermansky, "Significance of contextual information in phoneme recognition", Jan. 2007.

[7] B. E. D. Kingsbury and N. Morgan, "Perceptually inspired signal processing strategies for robust speech recognition in reverberant environments", Ph.D. dissertation, 1998, ISBN: 0599224762.

[8] R. Haeb-Umbach and H. Ney, "Linear discriminant analysis for improved large vocabulary continuous speech recognition", in *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 1992, 13–16 vol.1. DOI: `10.1109/ICASSP.1992.225984`.

[9] Z. Zajíc, L. Machlica, and L. Müller, "Refinement approach for adaptation based on combination of MAP and fMLLR", Sep. 2009, pp. 274–281, ISBN: 978-3-642-04207-2. DOI: `10.1007/978-3-642-04208-9_39`.

[10] U. Kamath, J. Liu, and J. Whitaker, *Deep Learning for NLP and Speech Recognition*, 1st. Springer Publishing Company, Incorporated, 2019, ISBN: 3030145956.

[11] "Readings in speech recognition", in, A. Waibel and K.-F. Lee, Eds., San Francisco: Morgan Kaufmann, 1990, ISBN: 978-1-55860-124-6. DOI: https://doi.org/10.1016/B978-0-08-051584-7.50003-6.

[12] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated, 2014, ISBN: 978-1-4471-5778-6. DOI: 10.1007/978-1-4471-5779-3.

[13] J. Holmes and W. Holmes, *Speech Synthesis and Recognition 2nd edition*. Taylor & Francis Group, 2003, ISBN: 0-203-48468-1.

[14] S. J. Young and P. C. Woodland, "State clustering in HMM-based continuous speech recognition", *Computer Speech and Language*, 1994.

[15] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition", *Comput. Speech Lang.*, vol. 16, no. 1, Jan. 2002, ISSN: 0885-2308. DOI: 10.1006/csla.2001.0184. [Online]. Available: https://doi.org/10.1006/csla.2001.0184.

[16] L. MacKenzie and D. Turton, "Assessing the accuracy of existing forced alignment software on varieties of British English", *Linguistics Vanguard*, vol. 6, 2020.

[17] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks", *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 2345–2349, Jan. 2013.

[18] G. Hinton, l. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *Signal Processing Magazine, IEEE*, vol. 29, pp. 82–97, Nov. 2012. DOI: 10.1109/MSP.2012.2205597.

[19] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, Big, Simple Neural Nets for Handwritten Digit Recognition", *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, Dec. 2010, ISSN: 0899-7667. DOI: 10.1162/NECO_a_00052. eprint: https://direct.mit.edu/neco/article-pdf/22/12/3207/842857/neco_a_00052.pdf. [Online]. Available: https://doi.org/10.1162/NECO_a_00052.

[20] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation", ser. ICML '07, New York, NY, USA: Association for Computing Machinery, 2007, 473–480, ISBN: 9781595937933. DOI: 10.1145/1273496.1273556. [Online]. Available: https://doi.org/10.1145/1273496.1273556.

[21] G. E. Hinton, "Training products of experts by minimizing contrastive divergence", *Neural Comput.*, vol. 14, no. 8, 1771–1800, Aug. 2002, ISSN: 0899-7667. DOI: 10.1162/089976602760128018. [Online]. Available: https://doi.org/10.1162/089976602760128018.

[22] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition", *Signal Processing Magazine*, 2012.

[23] N. Jaitly and E. Hinton, "Vocal tract length perturbation (VTLP) improves speech recognition", 2013.

[24] M. Gales, A. Ragni, H. AlDamarki, and C. Gautier, "Support vector machines for noise robust ASR", Jan. 2010, pp. 205 –210. DOI: 10.1109/ASRU.2009.5372913.

[25] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition", in *INTERSPEECH*, 2015.

[26] J. Anden and S. Mallat, "Deep scattering spectrum", *IEEE Transactions on Signal Processing*, vol. 62, no. 16, 4114–4128, 2014, ISSN: 1941-0476. DOI: 10.1109/tsp.2014.2326991. [Online]. Available: http://dx.doi.org/10.1109/TSP.2014.2326991.

[27] H. Bourlard, N. Morgan, C. Wooters, and S. Renals, "CDNN: A context dependent neural network for continuous speech recognition", *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 1992.

[28] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted MMI for model and feature-space discriminative training", in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 4057–4060. DOI: 10.1109/ICASSP.2008.4518545.

[29] N. S. Ibrahim and D. A. Ramli, "I-vector extraction for speaker recognition based on dimensionality reduction", *Procedia Computer Science*, vol. 126, pp. 1534–1540, 2018, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.08.126. [On-

line]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918314042.

[30] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1713–1725, 2014. DOI: 10.1109/TASLP.2014.2346313.

[31] D. Povey, S. Chu, and B. Varadarajan, "Universal background model based speech recognition", May 2008, pp. 4561 –4564. DOI: 10.1109/ICASSP.2008.4518671.

[32] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors", in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 55–59. DOI: 10.1109/ASRU.2013.6707705.

[33] V. Gupta, P. Kenny, P. Ouellet, and T. Stafylakis, "I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription", in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6334–6338. DOI: 10.1109/ICASSP.2014.6854823.

[34] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts", in *INTERSPEECH*, 2015.

[35] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989. DOI: 10.1109/29.21701.

[36] C. Olah. (Aug. 2015). "Understanding LSTM networks", [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 05/08/2021).

[37] T. He and J. Droppo, "Exploiting LSTM structure in deep neural networks for speech recognition", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5445–5449. DOI: 10.1109/ICASSP.2016.7472718.

[38] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition", *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 22, no. 10, 1533–1545, Oct. 2014, ISSN: 2329-9290. DOI: 10.1109/TASLP.2014.2339736. [Online]. Available: https://doi.org/10.1109/TASLP.2014.2339736.

[39] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition", May 2012, pp. 4277–4280, ISBN: 978-1-4673-0045-2. DOI: 10.1109/ICASSP.2012.6288864.

[40] K. Kim, K. Lee, D. Gowda, J. Park, S. Kim, S. Jin, Y.-Y. Lee, J. Yeo, D. Kim, S. Jung, J. Lee, M. Han, and C. Kim, "Attention based on-device streaming speech recognition with large speech corpus", Dec. 2019. DOI: 10.1109/ASRU46091.2019.9004027.

[41] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S.-y. Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices", in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6381–6385. DOI: 10.1109/ICASSP.2019.8682336.

[42] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation", Jun. 2014. DOI: 10.3115/v1/D14-1179.

[43] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions", Sep. 2019, pp. 3785–3789. DOI: 10.21437/Interspeech.2019-2460.

[44] J. Park, Y. Boo, I. Choi, S. Shin, and W. Sung, "Fully neural network based speech recognition on mobile and embedded devices", in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, Red Hook, NY, USA: Curran Associates Inc., 2018, 10642–10653.

[45] S. Wang and G. Li, "Overview of end-to-end speech recognition", *Journal of Physics: Conference Series*, vol. 1187, p. 052 068, Apr. 2019. DOI: 10.1088/1742-6596/1187/5/052068.

[46] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks", in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, New York, NY, USA: Association for Computing Machinery, 2006, 369–376, ISBN: 1595933832. DOI: 10.1145/1143844.1143891. [Online]. Available: https://doi.org/10.1145/1143844.1143891.

[47] L. Bahl, P. Brown, P. Souza, and R. Mercer, "Maximum mutual information estimation of hidden markov parameters for speech recognition", vol. 11, May 1986, pp. 49 –52. DOI: 10.1109/ICASSP.1986.1169179.

[48] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlíček, Y. Qian, K. Riedhammer, K. Veselý, and T. Vu, "Generating exact lattices in the WFST framework", *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, Mar. 2012. DOI: 10.1109/ICASSP.2012.6288848.

[49] S. Shabani and Y. Norouzi, "Speech recognition using principal components analysis and neural networks", in *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, 2016, pp. 90–95. DOI: 10.1109/IS.2016.7737405.

[50] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks", Sep. 2018, pp. 3743–3747. DOI: 10.21437/Interspeech.2018-1417.

[51] L. Huang, J. Sun, J. Xu, and Y. Yang, "An improved residual LSTM architecture for acoustic modeling", Aug. 2017.

[52] D. B. Paul and J. M. Baker, "The design for the wall street journal-based CSR corpus", in *Proceedings of the Workshop on Speech and Natural Language*, ser. HLT '91, Harriman, New York, USA: Association for Computational Linguistics, 1992, ISBN: 1558602720. DOI: 10.3115/1075527.1075614. [Online]. Available: https://doi.org/10.3115/1075527.1075614.

[53] D. Povey, *Results*, Feb. 2017. [Online]. Available: https://github.com/kaldi-asr/kaldi/blob/master/egs/wsj/s5/RESULTS.

[54] P. Pollák and Černocký Jan, "Czech SPEECON adult database", vol. 104, 2004.

[55] P. Mizera, "Applying articulatory features within speech recognition", PhD thesis, Czech Technical University in Prague, 2019.

[56] P. Fousek, P. Mizera, and P. Pollak, *Ctucopy feature extraction tool*. [Online]. Available: http://noel.feld.cvut.cz/speechlab/.

[57] E. Chung, H.-B. Jeon, J.-G. Park, and Y.-K. Lee, "Lattice rescoring for speech recognition using large scale distributed language models", in *Proceedings of COLING 2012: Posters*, Mumbai, India: The COLING 2012 Organizing Committee, Dec. 2012, pp. 217–224. [Online]. Available: https://www.aclweb.org/anthology/C12-2022.

[58] M. Ravanelli, T. Parcollet, and Y. Bengio, "The pytorch-kaldi speech recognition toolkit", in *In Proc. of ICASSP*, 2019.

[59] D. Can, V. R. Martinez, P. Papadopoulos, and S. S. Narayanan, "Pykaldi: A python wrapper for kaldi", in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*, IEEE, 2018.

[60] M. Záruba, "Modern techniques of speaker recognition based on GMM and DNN", Master thesis, Czech Technical University in Prague, 2017.

[61] J. Silovský, "Generative and discriminative classifiers in the tasks of text-independent speaker recognition and diarization", M.S. thesis, Technical University of Liberec, 2011.

[62] M. Lakosil, "DNN-based voice activity detector", Master thesis, Czech Technical University in Prague, 2017.

[63] K. Veselý, "Semi-supervised training of deep neural networks for speech recognition", PhD thesis, Brno University of Technology, 2017.

[64] C. Chakraborty and P. Talukdar, "Issues and limitations of HMM in speech processing: A survey", *International Journal of Computer Applications*, vol. 141, no. 7, 13–17, 2016. DOI: 10.5120/ijca2016909693.

[65] D. Silingas and L. Telksnys, "Specifics of hidden markov model modifications for large vocabulary continuous speech recognition", *Informatica, Lith. Acad. Sci.*, vol. 15, pp. 93–110, Jan. 2004. DOI: 10.15388/Informatica.2004.048.

[66] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review", *IEEE Access*, vol. 7, pp. 19 143–19 165, 2019. DOI: 10.1109/ACCESS.2019.2896880.

[67] M. Ernestus, L. Kočková-Amortová, and P. Pollak, "The nijmegen corpus of casual Czech", in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 365–370. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2014/pdf/134_Paper.pdf.

[68] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition", *CoRR*, vol. abs/1402.1128, 2014. arXiv: 1402.1128. [Online]. Available: http://arxiv.org/abs/1402.1128.

[69] S. Sen, A. Dutta, and N. Dey, *Audio Processing and Speech Recognition: Concepts, Techniques and Research Overviews*, 1st. Springer Publishing Company, Incorporated, 2019, ISBN: 9789811360978.

[70] A.-L. Georgescu, H. Cucu, and C. Burileanu, "Kaldi-based DNN architectures for speech recognition in Romanian", in *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, 2019, pp. 1–6. DOI: 10.1109/SPED.2019.8906555.

[71] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall, 2001, ISBN: 0130226165.

[72] M. Karafiát, M. K. Baskar, P. Matějka, K. Veselý, F. Grézl, and J. Černocky, "Multilingual BLSTM and speaker-specific vector adaptation in 2016 but babel system", in *2016 IEEE Spoken Language Technology Workshop (SLT)*, 2016, pp. 637–643. DOI: 10.1109/SLT.2016.7846330.

[73] J. Fiala, "DNN-HMM based multilingual recognizer of telephone speech", Diploma thesis, Czech Technical University in Prague, 2016.