Multilingual Modulation by Neural Language Codes

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik

des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Markus Müller

aus Zweibrücken

Tag der mündlichen Prüfung: 29.6.2018

Erster Gutachter: Prof. Dr. Alexander Waibel

Zweiter Gutachter: Prof. Dr. Laurent Besacier

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe, sowie dass ich die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des KIT, ehem. Universität Karlsruhe (TH), zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 3. Mai 2018

Markus Müller

Abstract

Multilingual Speech Recognition remains one of the great challenges in speech processing. Each of the 7,000+ languages of the world and also many different accents and dialects require their own acoustic model to achieve acceptable recognition performance. Even though the human sound inventory is finite and the same phone symbols are used to transcribe speech across languages, each language and accent imposes its own coloring to the sounds. Many approaches towards multilingual adaptation have been proposed, but they generally are inferior to monolingually trained models and still require language specific data and manual tuning.

In this thesis, we propose a different approach based on a large multilingual model, which is adapted rapidly using learned *language codes*. Extracted via an auxiliary neural network, the language codes encode language properties which are used to gate the activity of neurons in the multilingual acoustic model network. Using such a large multilingual model and this language modulation, we are not only able to achieve parity with monolingually trained models, but even surpass their performance.

This thesis is developed in several stages to investigate the effectiveness of the proposed methods. We built two types of systems using either a classical approach with multiple explicitly modeled components or a novel approach based on a single bi-directional Long Short-Term Memory (BiLSTM) network. In traditional systems, the networks are trained frame-wise using a pre-computed alignment between the sequence of acoustic feature vectors and the targets. Using BiLSTM networks in combination with the Connectionist Temporal Classification (CTC) loss function, the network itself learns the alignment between the sequence of input vectors and output targets. We begin with a classical speech recognition system which has multiple components: an acoustic model, a language model, a pronunciation dictionary and

an acoustic front-end for signal pre-processing. In state-of-the-art speech recognition systems, some (or all) of these components are based on neural networks.

Additive Language Codes. Initially, we use an approach similar to speaker adaptation. But instead of appending a low-dimensional representation of speaker properties to the acoustic features, we here append the language identity information using one-hot encoding. Adding such a code to the acoustic features of the pre-processing and the acoustic model improves the recognition accuracies. Next, we attempt to develop a low-dimensional representation of language properties. To extract such a code, we train a feed-forward network for language identification. This network is composed of multiple layers, with a bottleneck layer as second-to-last layer. After training, the layers after the bottleneck are discarded and the output activations of the bottleneck are taken as Language Feature Vectors (LFVs). Supplying this language code instead of the one-hot encoded language identity increases the performance further.

Multiplicative Language Codes. Beyond additive codes, we wish to maximize the language adaptation of the system using neural network modulation with multiplicative language codes. For this reason, we prefer to do experiments using RNN/CTC based setups. As these systems are based on single BiLSTM networks, fewer components are modeled explicitly, e.g. no context-dependent polyphones as in traditional systems are required. Adapting the neural network in this all-neural architecture enables better language adaptation, as all implicitly learned components are adapted as well. This architecture maximizes the impact of language adaptation and is therefore ideal for studying its effects, even though traditional systems do achieve lower word error rates (WERs).

For integrating language codes into BiLSTM networks, we evaluate two approaches. Appending them to the acoustic features results in gains similar to traditional systems. But language properties are not as signal related as speaker or channel properties. Adding these language codes at the same place as the acoustic features is therefore not optimal. Based on prior work on speaker independence (Meta-Pi), we propose a method called "modulation" to enable the deeper integration of language codes into the network architecture. By using the language codes to

gate the outputs of BiLSTM cells, the learned features of the network are modulated depending on language properties. This stimulates the network in developing feature detectors based on language codes. Supplying language codes this way enables the quick adaptation of the neural network based acoustic model to languages, and improves the recognition accuracy further compared to the simple addition.

Adaptive Neural Language Codes. As an alternative to large monolithical networks, we also investigate the modulation involving multiple language specific networks as part of a giant network superstructure, similar to Meta-Pi networks. We adapt this method to our setup by pre-training monolingual subnets and an ancillary Neural Language Code (NLC) network. During the joint training of the network superstructure the weights of all subnets are updated. The ancillary network learns to adapt the language codes to code more useful representations of language properties. By applying this method, we do not only achieve parity with the monolingual setup (24.2% WER), but are able to improve the recognition accuracy further (23.5% WER).

Zusammenfassung

Systeme zur automatischen Spracherkennung haben in den letzten Jahren enorme Fortschritte in der Erkennungsleistung gemacht, doch es gibt weiterhin viele ungelöste Probleme. Eines dieser Probleme ist die Multilingualität. Jede der über 7000 Sprachen in der Welt, ebenso Dialekte, benötigen ein eigenes akustisches Modell um eine akzeptable Erkennungsleistung zu erzielen. Obwohl das Inventar an menschlichen Lauten begrenzt ist und es ein sprachenübergreifendes phonetisches Alphabet gibt, prägt jede Sprache den Phonemen ihre spezielle Färbung auf. Grund dafür sind koartikulations Artefakte, welche beim Übergang zwischen Phonemen entstehen. In der Vergangenheit wurden verschiedene Ansätze zur multilingualen Adaption vorgestellt, aber multilingual trainierte Modelle erreichen im Allgemeinen nicht die Leistung von Modellen, welche auf nur einer Sprache trainiert wurden. Wir verfolgen einen Ansatz, bei welchem ein großes, multilinguales Modell trainiert und mittels eines Sprachencodes adaptiert wird. Diese Sprachencodes werden mittels eines zusätzlichen neuronalen Netzes extrahiert und kodieren Sprachenmerkmale, welche dazu benutzt werden, um die Ausgaben von Neuronen im multilingualen Netz zu steuern. Basierend auf dem Sprachencode werden die Ausgaben von Neuronen entweder verstärkt oder abgeschwächt. Dank der Kombination eines großen multilingualen Modells und dieser sogenannten Modulation erreicht unser System nicht nur die Leistung eines monolingual trainierten Modells, sondern übertrifft sie sogar noch.

Ein Spracherkennungssystem besteht aus mehreren Komponenten, beispielsweise dem akustischen Modell, Sprachmodell, Aussprachewörterbuch oder der Signalvorverarbeitung. Bei Systemen welche dem Stand der Technik entsprechen bestehen diese Komponenten aus neuronalen Netzen. Ein Beispiel sind akustische

Modelle, bei welchen mehrschichtige neuronale Netze (Deep Neural Networks, DNNs) zum Einsatz kommen. Um ein solches Netz beispielsweise an Sprecher zu adaptieren, fügt man eine niedrigdimensionale Repräsentation von Sprechereigenschaften an die akustische Merkmale an. Dies erlaubt es dem Netz, sich an Sprechereigenschaften einzustellen. Zur Adaptation an Sprachen verwenden wir einen ähnlichen Ansatz.

Additive Sprachencodes. Für ein erstes Experiment wählen wir einen naiven Ansatz und kodieren die Eingabesprache als Bitvektor. Jedes Bit entspricht einer Sprache und ist gesetzt, sofern diese Sprache die Eingabesprache ist. Dieser Code wird an die akustischen Merkmale angehängt und wir beobachten so bereits eine Verbesserung bei multilingual trainierten Systemen. Jedoch bildet ein solcher Code lediglich die Identität der Sprache ab, nicht aber deren Eigenschaften. Als nächstes haben wir daher eine niedrigdimensionale Repräsentation von Spracheneigenschaften entwickelt. Um einen solchen Code zu erzeugen, trainieren wir ein neuronales Netz darauf, Sprachen zu erkennen. Dieses Netz enthält als vorletzte Schicht ein sogenanntes Bottleneck, eine Schicht welche im Vergleich zu den übrigen Schichten des Netzes nur sehr wenige Neuronen beinhaltet. Nach dem Training werden die Schichten nach dem Bottleneck verworfen und die Ausgaben des Bottlenecks als Sprachencode (Language Feature Vectors, LFVs) verwendet. Angehängt an die akustischen Merkmale steigert dieser Code die Erkennungsleistung des multilingualen Systems gegenüber dem Bitvektor mit der Spracheninformation.

Multiplikative Sprachencodes. In jüngster Zeit rücken Spracherkennungsysteme, welche auf einem einzigen neuronalen Netz basieren, in den Fokus der Forschung. Diese bestehen üblicherweise aus bi-direktionalen Long Short-Term Memory (BiLSTM) Netzwerken, welche ein mächtiges Werkzeug zur Sequenzmodellierung sind. In traditionellen Spracherkennungssystemen werden die Netzwerke frame-weise trainiert. Dafür werden in einem zusätzlichen Schritt vorab die Zielsymbole an den akustischen Merkmalsvektoren ausgerichtet. Beim Einsatz von einem BiLSTM Netzwerk, welches mittels der Connectionist Temporal Classification (CTC) Fehlerfunktion trainiert wird, entfällt dieser zusätzliche Schritt: Das Netzwerk wird direkt auf den Ausgabesymbolen trainiert und lernt die Ausrichtung selbstständig.

Ein solches System beinhaltet außerdem weniger explizit modellierte Komponenten, beispielsweise keine kontextabhängigen Polyphone eines traditionellen Systems. Das Netz lernt solche Aspekte implizit zu modellieren. Die Adaptation des zentralen neuronalen Netzes eines solchen Systems ermöglicht daher eine bessere Adaption an Sprachen, da die implizit gelernten Aspekte ebenfalls mit adaptiert werden. Diese Systemarchitektur maximiert die Effekt der Sprachenadaption und ist daher ideal um seine Auswirkungen zu messen, obwohl traditionelle Systeme im Allgemeinen geringere Wortfehlerraten (WER) aufweisen.

Zur Integration der Sprachencodes in BiLSTM Netzwerke vergleichen wir zwei Ansätze. Das Anhängen an die akustischen Merkmale führt zu ähnlichen Verbesserungen wie bei traditionellen Systemen. Aber Spracheneigenschaften sind nicht so signalbezogen wie beispielsweise Sprecher- oder Kanaleigenschaften. Das Hinzufügen der Sprachencodes an der gleichen Stelle wie die akustischen Merkmale ist daher nicht optimal. Im Rahmen von Meta-PI Netzwerken wurde eine Methode namens Modulation vorgestellt. Sie erlaubt die tiefere Integration von Sprachencodes in die Netzwerkarchitektur. Anhand der Sprachencodes werden die Ausgaben von BiLSTM Einheiten basierend auf Spracheneigenschaften entweder verstärkt oder abgeschwächt. Dies stimuliert das Netzwerk Merkmalsdetektoren in Abhängigkeit von Sprachencodes zu lernen. Die Integration der Codes auf diese Art erlaubt die schnelle Adaption des auf neuronalen Netzen basierenden akustischen Modells auf Sprachen. Der Einsatz dieser Methode führt zu einer besseren Erkennungsleistung gegenüber dem Anhängen der Sprachencodes an die akustischen Merkmale.

Adaptive neuronale Sprachencodes. Der zweite Aspekt von Meta-PI Netzen ist der Einsatz von vortrainierten Teilnetzen in einem großen Supernetzwerk. Wir haben diesen Ansatz für unseren Anwendungsfall adaptiert und monolinguale Teilnetze, sowie ein Hilfsnetzwerk zur Extraktion neuronaler Sprachencodes (NLC) trainiert. Während des gemeinsamen Trainings als Teil des Supernetzwerks werden die Parameter aller Teilnetze mit optimiert. Das NLC Hilfsnetz lernt die Sprachencodes so zu adaptieren, dass sie Spracheneigenschaften extrahieren, welche bei der Spracherkennung ein besseres Ergebnis liefern. Durch die Anwendung dieser Methode erreicht unser System nicht nur die gleiche Leistung wie ein monolingual trainiertes (24.2% WER), sondern sogar ein besseres Erkennungsergebnis (23.5% WER).

To my parents.

In gratitude.

Acknowledgements

I would like to thank every one who has supported and helped me during the work on my thesis. First and foremost, I want to thank Prof. Alex Waibel for making me part of his team and letting me perform the research leading to my thesis, for his advice and support, for our discussions, and for being a rich source of new ideas. Working at the Interactive Systems Laboratories has been a real joy. I would also like to extend my thanks to Prof. Laurent Besacier for showing interest in my thesis and for being my co-advisor. I enjoyed working with Laurent during the JSALT workshop in 2017 and the BULB project.

With Alex being the "father" of this thesis, I also have thank my "big brother", Sebastian Stüker. Always available for discussions, he gave important advice and contributed very much to the success of my thesis. I also have to thank Sarah Fünfer, with who I shared an office during my first years at the institute. Thank you for being the best office mate ever.

A special thank goes to Elizabeth Salesky, Florian Deßloch and Jan Niehues for proof-reading parts of my thesis. My thanks also extend to all the past and present members of the ASR team with whom I have worked over the years: Jonas Gehring, Michael Heck, Christian Mohr, Huy Van Nguyen, Bao Quoc Nguyen, Thai Son Nguyen, Matthias Sperber, and Thomas Zenkel. Additionally to the ASR team, I would like to thank the members of the MT and Dialog teams with whom I had many fruitful discussions during my time at the lab: Jan Niehues, Eunah Cho, Thanh-Le Ha, Ngoc Quan Pham, Teresa Herrmann, Mohammed Mediani, Maria Schmidt, and Stefan Constantin.

My thanks also go out to all secretarial, technical and administrative staff. First and foremost Silke Dannenmaier and Margit Rödder, Bastian Krüger, Franziska

Vogel, Virginia Roth and Mirjam Simantzik. I would like to thank our colleagues in Pittsburgh: Florian Metze, Alan Black, Susanne Burger, Ramon Sanabria, Yajie Miao, Jae Cho and Eric Riebling. Thank you for welcoming me during my stays at the CMU. I am also thankful to Odette Scharenborg and Emanuel Doupoux for including me at the JSALT workshop in 2017, and to all members of the Rosetta team, especially Pierre Godard.

At last but not least, I would like to thank my parents for always supporting and encouraging me on my journey. Thanks go also to Marcel Noe who got me interested in studying computer science in Karlsruhe. I would like to also thank my relatives and friends for their support and interest, notably Kurt Werle.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multilingual Speech Recognition is one of the most challenging AI problems. Each language and even different accents require their own acoustic model to obtain the best recognition performance. Even though an universal phonetic alphabet exists, each language and accent has is own coloring or "twang". Many adaptive approaches have been proposed, but they are generally inferior to monolingually trained models. In this work, we propose a novel approach for language adaptation. Using an all-neural multilingual architecture, we are able to not only reach parity with monolingually trained models, but surpass their performance.

1.1 Main Contribution

The main contribution of this thesis is a novel neural network adaption technique. A large multilingual model is *modulated* by the codes generated by an ancillary network trained on an auxillary task. This ancillary network learns to code useful language properties. This language code is used to gate the activity of neurons in the large network, which is trained to recognize speech from multiple languages. By applying these codes, the network is stimulated to learn features based on language properties which allow for rapid language adaptation. While we study this method in the regimen of multilingual speech recognition, it can also be used in other areas, e.g. to adapt to speaking modes or dialects.

1.2 Overview and Structure

We first use traditional speech recognition systems with multilingual acoustic models. They feature neural network based multilingual components as part of the acoustic front-end for feature pre-processing and as part of the acoustic model. We study neural network adaptation methods for adapting networks to languages. Inspired by speaker adaptation of neural networks which uses a low-dimensional feature vector encoding speaker and channel properties, we use a similar approach for language adaptation by appending language codes to the acoustic features, as outlined in Chapter 6. First, we use a naive approach by providing only the language identity as one-hot encoded vector to the network. This improves the performance but this vector does not represent language properties. We therefore shift towards language codes which encode language properties. To extract such a code, we train a neural network for language identification. This network features a narrow bottleneck layer and the output activations of this layer are used as language code after training. Using this language representation improves the performance more than using the one-hot encoded language identity. While we could close the gap between mono- and multilingual systems with this approach further, we still observe a loss in performance when training acoustic models for multiple languages.

Language properties are not as signal related as speaker or channel characteristics. Adding language codes deeper into the network architecture should therefore improve the performance, as we study in Chapter 7. We also transition from a traditional system architecture to an all-neural architecture. While traditional setups feature many explicitly modeled components, all-neural approaches learn to model aspects implicitly. Adapting the network to languages adapts the learned features as well and the adaptation has a larger effect. With this system architecture, we use multiplicative codes, inspired by Meta-PI networks which feature Meta-PI connections to gate the activity of neurons. These connections apply a weight to the output of a neural unit by multiplication with a coefficient. We call this type of adaptation *modulation*. We modulate the output of a hidden layer in our network architecture with language codes which improves the performance over the simple addition of the codes to the acoustic features at the input layer of the network.

Once extracted, we keep the language codes fixed during system training. Using adaptive codes is a better approach, as the code is modified to improve the overall system performance. In Chapter 8, we study using such adaptive codes in our network architecture. Inspired by Meta-PI, we create a network superstructure which combines multiple subnets which were pre-trained on different tasks. We use monolingually pre-trained subnets and a neural language code network. During the joint optimization, the parameters of all networks are updated. This includes the language codes which are updated for optimal multilingual performance. Using this approach, we could not only reach parity with monolingual acoustic models, but also improve the recognition accuracy further.

Chapter 2

Theory and Background

2.1 Automatic Speech Recognition

Automatic speech recognition (ASR) is the translation of speech into the corresponding word sequence. This is a difficult problem, because speech is a highly variable biosignal. There are variabilities on multiple levels: signal, phonetic and linguistic. Special methods have been proposed to mitigate these variabilities. This section provides a brief overview, [HAH01] provides a more detailed description.

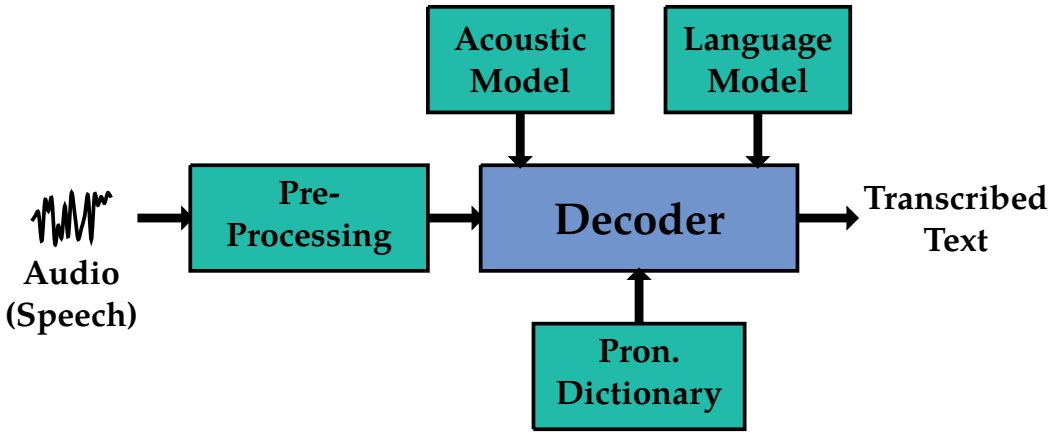

Figure 2.1: Components of an ASR system.

Figure 2.1 shows a block diagram of a typical ASR system. It features components like, e.g. the pre-processing of the raw audio signal, the acoustic model, language model and the pronunciation dictionary. ASR can be described by the following

fundamental equation:

$$\hat{w} = argmax_W P(W|X) \tag{2.1}$$

With $\hat{w}$ being the word sequence, W being words and X the acoustic representation thereof. $argmax_w$ determines the most likely word sequence, given the acoustic representation X. Using the Bayes rule, the equation can be rewritten as in:

$$\hat{w} = argmax_W P(W|X) = argmax_W \frac{P(X|W) \cdot P(W)}{P(X)} = argmax_W P(X|W) \cdot P(W) \tag{2.2}$$

The term $P(X)$ can be omitted as we are only interested in the most likely word sequence and $P(X)$ becomes constant when applying the $argmax$. Components of an ASR system can be found in this formula, e.g. the acoustic model $P(X|W)$ or the language model $P(W)$.

Audio Pre-processing

The first step in the speech recognition pipeline is the feature pre-processing. Input is the digital representation of an analogue audio signal containing speech. The pre-processing extracts speech features out of the raw audio signal, thereby performing a dimensionality reduction as well as to address variability. Ideally, the extracted features should be robust towards noise, channel distortions and different speakers. Typical features are Mel frequency cepstral coefficients (MFCCs) [DM80], the mean variance distortion response (MVDR) [MR97] or the logarithmic Mel scaled spectrum (logMel) [SVN37]. All these features operate in the frequency domain instead of the time domain, which requires a transformation of the signal. The Fourier transform [Fou22] is used to transform the signal into the frequency domain. In order to retain the temporal structure of the signal, windowing is applied. Typical window sizes are 16ms or 32ms, which provide a good trade-off between frequency and time resolution. Within each window, the signal is considered to be stationary.

There exist so-called tonal languages, where pitch and tone carry semantic information, e.g. Vietnamese or Cantonese. Special features were developed for this type languages. In this work, we used the fundamental frequency variation (FFV) [LHE08], as well as a pitch tracker [Sch99]. Estimating the fundamental frequency is difficult because of ambiguity. The speech signal, as it is modulated by the human vocal

tract, contains harmonics and depending on the acoustic conditions, such a harmonic may be wrongly recognized as the fundamental frequency.

Pronunciation Dictionary

The task of speech recognition is to generate a transcript from speech. Systems therefore need to know which tokens to recognize. Training systems directly on words is possible [SLS16], but requires a massive amount of data. Adding new words also proves difficult in such a system architecture. Therefore, smaller units are generally chosen to model the acoustics. The pronunciation dictionary provides the mapping from words to these tokens, the acoustic units. Typically, phonemes are being used to model the pronunciations of words. A common set of phones is the international phonetic alphabet (IPA) [Ass99]. Derived from it are SAMPA [W$^+$97] and an extension to it X-SAMPA [Wel95] which encodes the IPA symbols using only ASCII[1] characters.

It is also possible to build ASR systems using only the written representation of the target language. By using so-called graphemes as acoustic units, systems infer the letter-to-sound rules themselves. Depending on the language, automatically learning these rules is difficult and the resulting system produces more recognition errors. Given enough training data, systems can also be trained to recognize words as acoustic units [SLS16].

Acoustic Model

The acoustic model is a central part of ASR systems. It estimates $P(X|W)$, the conditional probability that the sequence of acoustic feature vectors X is produced by the word sequence W. The challenge is to build a model which closely reflects the acoustic properties of the language spoken. Phonemes are not uttered isolated, but in context of other phonemes, which results in co-articulation artifacts. Context-dependent acoustic units therefore improve the recognition accuracy in traditional systems.

There are two types of acoustic models: traditional systems use an HMM based model [Rab89], whereas novel approaches are based entirely on neural networks [GFGS06, CJLV16]. HMM based models can be further divided into

[1] American Standard Code for Information Interchange

GMM/HMM and DNN/HMM based systems. Although DNN/HMM based setups are the state-of-the-art approach, GMM/HMM based systems still play in important role. They are well researched and because the knowledge is modeled explicitly instead of implicitly by a neural network, they are required for certain adaptation techniques [BBdSM86, PKK+08, GW96]. GMM/HMM systems are also required for bootstrapping DNN/HMM systems, especially for determining the context-dependent polyphone models, which are context-dependent phone models [YW93, HHL89].

Language Model

The language model estimates the prior probability $P(W)$ of a given word sequence. It is an additional source of information and helps to address acoustic ambiguity. Without a proper language model, speech recognition systems tend to hypothesize acoustically similar, but wrong word sequences:

> This machine can recognize speech/This machine can wreck a nice beach
> European elections campaign/European election scum paying
> Show me a new display/Show me a nudist play

To compute the prior probability of a word sequence, it is broken down into the product of the probability of each word, given the word history. These probabilities are difficult to estimate, because not all word histories are seen during training. An approximation are fixed word histories, so-called N-grams. By limiting the word history to n words, a more robust estimation of the probabilities is possible. Depending on the length of the word history considered, the N-grams are called unigrams ($n = 1$), bigrams ($n = 2$), trigrams ($n = 3$), 4-grams ($n = 4$), etc. Language models typically use 3-5 grams, depending on the amount of training data. But even with N-grams, not every word is encountered in all possible contexts in the training data. Additional techniques like back-off are used to fall back to shorter word histories [KN95, CG96].

2.1.1 Evaluation Metrics

Multiple objective metrics were developed to assess the performance of speech recognition systems. We outline two of them: the word error rate (WER) and the average term weighted value (ATWV).

Word Error Rate

The word error rate (WER) is a very widespread measure in the field of automatic speech recognition. It is based on the Levenshtein minimal editing distance and computed by counting the operations required to transform the hypothesis of the system into the reference transcript. As outlined in [HAH01], chapter 9.2, the errors of an ASR system can be classified into 3 types:

Substitutions: The ASR system misrecognizes one word for another.

Deletions: The system omits a word in the hypothesis.

Insertions: An additional word not present in the reference is recognized.

The WER is computed in the following way:

$$WER = \frac{\#subs + \#dels + \#ins}{\#words\ in\ the\ reference} \tag{2.3}$$

A perfect system would achieve a WER of 0%, whereas a bad system could generate a hypothesis with more words than are present in the reference and thereby could even exceed a WER higher than 100%. For logographic or syllabic languages, related measures like character error rate (CER) or syllable error rate (SER) exist, which are computed in the same manner.

Average Term Weighted Value

Average term weighted value (ATWV) is a measure to evaluate keyword spotting (KWS) systems. It was originally developed by Fiscus [FAGD07] for the NIST 2006 Spoken Term Detection evaluation. The errors a KWS system generates are thereby weighted based on the frequency of the keywords: Missing or false detecting a rare keyword weights higher compared to errors on frequent keywords. Using arbitrary weights per keyword is also possible. A perfect system would achieve a score of 1.0, a system outputting nothing a score of 0.0. A bad system is able to obtain a negative score by outputting many false positives.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a powerful machine learning method dating back half a century ago. In 1957, Rosenblatt formulated the perceptron algorithm [Ros57]. Inspired by biology, a perceptron features one or multiple inputs, biases, an activation function and output(s). In recent years, due to increased computing capabilities and the emergence of specialized hardware for efficient computation, ANNs have become very popular. A single perceptron can only be applied to linear separable problems, which was the main criticism of Minsky [MS69]. More complex problems, e.g. the exclusive disjunction ("XOR"), can also be solved, but do require a so-called multi-layer-perceptron (MLP) with non-linear activation functions, which is a more sophisticated network architecture. Using linear activation functions would allow to collapse the multiple layers and no advantage would be gained.

Being a data driven method, ANNs do not require explicit modeling of knowledge, but do learn implicitly. The components to be modeled are the network architecture, as well as the representation of the input features and output targets. Choosing a good representation of the data is vital. To train a neural network, an input vector is input into the network and a forward pass of the data through the network is computed. At the output layer, an error function is used to compute the error based on the output of the network and the corresponding target vector. To determine how the weights should be updated, error backpropagation [RHW85] is used. The derivative of the error function is computed to obtain the gradients of this function for all weights of the network. Gradient descent is used to apply the weight updates based on the computed loss.

2.2.1 Feed Forward Neural Networks

MLPs are also called feed forward neural networks (FFNNs). The neurons in such a network are organized in layers. While the neurons of two layers are fully connected, there are no connections of neurons within the same layer. Using a non-linear activation function is required, because the layers could otherwise be collapsed and the advantage of using multiple layers would vanish. Networks of this type are used for frame-level classification, where the current classification result only depends on the current input frame, typically including a certain context around this frame. There

are special variants like time-delay neural networks (TDNNs) [WHH+87a, WHH+87b, WHH+88], which aim at modeling context in an efficient manner.

2.2.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of networks which are a powerful at sequence modeling [Elm90], or time-series in the scope of ASR. They feature recurrent connections which allow to transfer information between adjacent time-steps. There are two configurations possible: The information can either flow only forward in time, or in both directions: forward and backward. Such networks are called bi-directional [SP97]. Instead of single frames, RNNs are trained on sequences, using backpropagation through time [Wer90].

Long Short-Term Memory Networks

While RNNs are able to model context, they are limited in capturing long term dependencies due to the vanishing gradient problem [BSF94, HBFS01]. Long short-term memory (LSTM) networks were proposed to mitigate this problem [HS97, Hoc98]. By using internal memory cells, these networks are able to preserve information over longer distances. Each LSTM cell has 3 gates for controlling the hidden state: 1) a forget gate which determines how much information is preserved, 2) an input gate to decide how much new information should be stored and 3) an output gate to control how much of the internal state is output. Figure 2.2 (from [Gra13]) shows the information flow within an LSTM cell, including peephole connections [GSS03]. LSTM networks exist also in a bi-directional variant (BiLSTM).

2.2.3 Activation Functions

A multitude of activation functions exists. They can be divided into two groups: continuous and discrete. In addition to these functions, there is a special function, called "softmax" [Bri90]. It features a non-linearity, but in addition conditions the output in such a way, that it can be interpreted as a probability distribution:

$$\varphi(\mathbf{x})_j = \frac{e^{\mathbf{x}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}_k}} \tag{2.4}$$

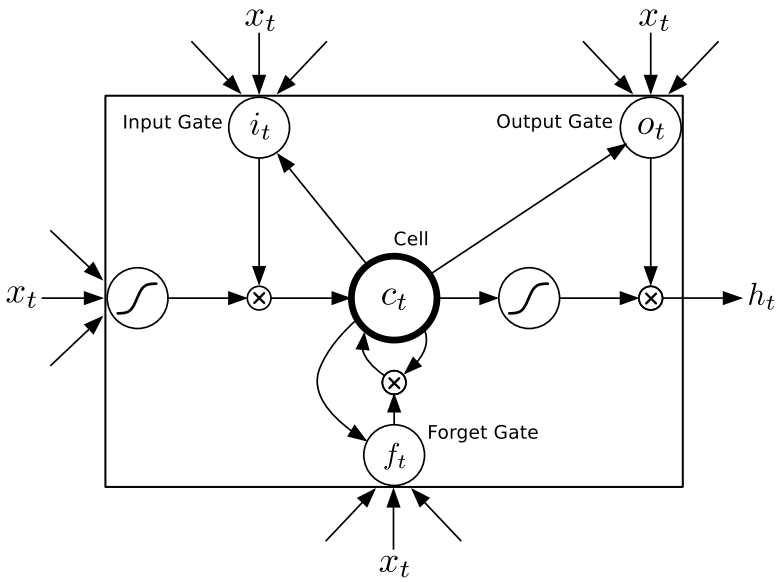

Figure 2.2: LSTM cell, from [Gra13]

with K being the total number of neurons in the layer and $\mathbf{x}$ being the summed and weighted input of the neuron j. The output of all neurons is conditioned to sum to 1, with the output of each neuron being in the range of $[0, 1]$. A refinement is shown in formula 2.5, called temperature or tempered softmax:

$$\varphi(\mathbf{x})_j = \frac{e^{\frac{\mathbf{x}}{\mathbf{T}}_j}}{\sum_{k=1}^{K} e^{\frac{\mathbf{x}}{\mathbf{T}}_k}} \tag{2.5}$$

It allows to adjust the spikiness of the output by the scaling the activation $\mathbf{x}$ with a temperature coefficient T. Setting this value to 1 corresponds to a regular softmax.

2.2.4 Error Functions

In order to train an ANN, a loss function is required. It computes the error between the output of the network and the actual value. Based on the task, an appropriate loss function has to be selected.

Mean Squared Error

This function computes the mean squared error between the output and the ground truth. This function is suitable if multidimensional, real valued outputs are required. It is not suitable for classification problems with hard targets encoded using, e.g., one-hot encoding, because the error signal generated by this function does not account for the type of error.

Cross Entropy

To train networks on classification tasks, using cross-entropy loss is suitable for this type of problems [B$^+$95].

Connectionist Temporal Classification

Recurrent neural networks are a powerful tool for sequence classification tasks. In these scenarios, a series of feature frames $I = i_0, i_1, \ldots, i_m$ are input into a neural network and the system is supposed to output a series of output tokens $O = o_0, o_1, \ldots, o_n$. Input frames and output sequences can vary in length, whereas in the regimen of speech recognition typically $m > n$. The ratio $m : n$ also varies. The traditional approach of training neural networks for speech recognition would be to pre-compute an alignment and assigning a label to each frame. The network would then be trained frame wise to predict the label of each time step. But for most tasks, only the correct sequence of output tokens is relevant but not the frame wise classification.

The connectionist temporal classification (CTC) loss function [GFGS06] allows neural networks to be trained directly on the sequence of output symbols O (taken from an alphabet L), without the need of a pre-computed alignment. CTC introduces the following concepts: a) a special blank label as additional output of the network b) an operator (called B in [GFGS06]) which performs the many-to-one mapping of the network outputs, removing blank labels as well as repeated labels. The blank label is required to account for the imbalance between the length of the sequence of feature frames and the sequence of output tokens.

As defined in [GFGS06], let L'^T denote the set of sequences with length T over the alphabet $L' = L \cup \{\text{blank}\}$. With y_k^t defining the activation of output unit k at time t,

we can compute the probability of a sequence π given an input sequence x of length T:

$$p(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L'^T \tag{2.6}$$

Applying the B operator, we can map $L'^T \mapsto L^{\leq T}$. The following example outputs would be considered equal ("$-$" denoting the blank label):

$$B(aa-b----cccc) = B(abbbb---c---) = B(aaaa----bc--) = abc \tag{2.7}$$

B can be used to define the conditional probability of a given labeling $l \in L^{\leq T}$ by accumulating the probabilities of all paths corresponding to it:

$$p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x) \tag{2.8}$$

Based on this, a classifier can be constructed which can be used for network training. Thereby, a forward backward algorithm similar to the one used for HMMs [Rab89] is used. Further details can be found in [GFGS06].

2.2.5 Update Functions

After updates to weights and biases have been computed, they need to be applied to the network. Here, again, multiple strategies can be used.

Stochastic Gradient Descent

One method is Stochastic Gradient Descent (SGD) [RM51]. Based on a learning rate, the weights and biases are being updated. There exist several improvements including a momentum term that varies the learning rate dynamically based on the error gradients. The basic idea behind the introduction of a momentum term is to make the training more robust by building up momentum as long as the gradient's direction does not change. This prevents lingering in a local minima and thereby lowers the overall error rate.

Update with (Nesterov) Momentum

An extension was proposed in the way how the momentum term is computed. Nesterov proposed a method [Nes83, Nes13] for applying momentum to gradients, which could also be applied in the context of neural networks. Modifying the gradients based on this method leads to faster convergence of the networks [SMDH13].

Family of Ada Methods

Another family of update methods has been proposed: Adam [KB14], Adagrad [DHS11, Dye13] and Adadelta [Zei12]. While SGD features the learning rate as free parameter, these methods do not require a manually set learning rate (although this is possible). Based on the computed loss, the learning rate is being scaled dynamically.

2.2.6 Parameter Initialization

Initializing the parameters of a network carefully does not only enable faster converging, but also results in a local minimum which is closer to the global one. The parameters should be initialized with respect to the activation function chosen to prevent neurons from saturating early. Different methods have been proposed to set the parameters initially. One approach is to select the parameters randomly, but condition the values based on certain criteria, e.g., based on the used activation function. The default initialization scheme draws values from a uniform distribution in the range of $[-\sigma; \sigma]$ with

$$\sigma = \frac{1}{\sqrt{|H|}} \quad \text{with} \quad |H| = \#\text{ of hidden units in layer} \tag{2.9}$$

The default method used in PyTorch samples values from a normal distribution, conditioned towards the number of neurons in each layer.

Glorot

Another approach is Glorot [GB10], which conditions the random values based on the input and output degree of each neuron by drawing samples from a uniform (U) or

normal (N) distribution.

$$a = \sqrt{\frac{12}{fan_{in} + fan_{out}}} \qquad \text{Weights} \quad = \quad U[-a, a] \tag{2.10}$$

$$\sigma = \sqrt{\frac{2}{fan_{in} + fan_{out}}} \qquad \text{Weights} \quad = \quad N[0, \sigma] \tag{2.11}$$

Pre-Training

Pre-training serves the purpose to pre-condition randomly initialized network weights. One possibility to pre-train networks would be Restricted Boltzmann Machines (RBMs) [YS11]. Another method for initializing the parameters is greedy (layer wise) pre-training [HOT06, BLP+07]. This training step is unsupervised and trains the network in the notion of a de-noising auto-encoder [VLBM08, GMMW13]. Artificial noise (called "salt and pepper noise") is added to the input features and the network is trained to reconstruct the data without noise. This forces the network to learn feature detectors based on the internal structure of the data.

By using this method, networks are constructed greedily on a layer-per-layer basis where each layer of the network is added and trained individually, with the weights frozen once the training of the layer at hand is finished. To reconstruct the original input, each layer has an inverse layer as counterpart which reverts the transformation applied to the data. The most common variant is to train such de-noising auto-encoders with tied weights between each layer and its counterpart. As this method is unsupervised, it does not require labeled data. In the regimen of ASR, no transcripts or forced alignments are required which allows the use of data where no such annotations exist.

2.2.7 Regularization

With many, often several millions, parameters, neural networks are prone to overfitting. One possibility to avoid this is the use of more data, which is potentially not available. Another option is to reduce the size of the model. With fewer parameters to train, the model is less likely to overfitting. But a smaller model would potentially be not as powerful. Several regularization methods were developed to avoid overfitting.

Dropout Training

One commonly used technique is called dropout training [HSK+12]. During training, connections between neurons are being dropped on a random basis with a certain probability. This way, neurons are less likely to co-adapt because the network features a different configuration during each training step. Training a network using dropout can be considered as training a multitude of networks in parallel, all with linked parameters. Dropout training introduces a parameter for the likelihood of a connection being dropped. Depending on the task, a dropout rate of up to 0.5 can prove beneficial. Connections are only dropped at training time. During inference, no connections are dropped, but the output of each neuron is being attenuated by the dropout factor to account for the input of the additional (not dropped) connections. Dropout training prevents co-adaptation between neurons in two ways: First, it prevents neurons from learning similar features, and second, it prevents neurons at higher layers to compensate for errors learned in lower layers which in turn forces the neurons at lower layers to not produce errors.

Gradient Clipping

Another technique is gradient clipping, where the computed gradients are limited to a certain threshold [GBC16]. This limits the impact of exploding gradients which may occur during training.

2.2.8 Learning Rate Scheduling Methods

To further optimize the training, methods for adapting the (initial) learning rate have been proposed. By adjusting the learning rate, the network training could converge faster and/or reach a local minimum which is closer to the global minimum. Multiple techniques have been proposed.

Fixed Scheduling

One method uses a fixed schedule where the learning rate is being adjusted based on the number of epochs trained. But being a static method, it does not account for the

network's progress. It further needs to be estimated prior to the training process, based on expert knowledge.

Exponential Decay

Instead of updating the learning rate at a fixed interval, there exists a method called exponential decay which decreases the learning rate after each epoch by a fixed factor. A good analogy for this would be playing golf. At the beginning, a long distance cup is chosen to play the ball into the broad direction of the hole. With every subsequent shot, the distance to the hole decreases and so cups for driving the ball for shorter distances are chosen. With respect to the learning rate, given a decreasing error, a smaller learning rate should be chosen instead of applying updates which are too big and would overshoot the minima.

Newbob

There exists yet another method, combining static learning rate scheduling with exponential decay, called "newbob" [MB90]. Based on the progress the network shows w.r.t. the error rate on the development set after each epoch, different stages are being selected. The training starts using a fixed learning rate. Once the decrease of the error rate falls below a certain threshold, newbob switches to exponential decay. The training continues in this mode until the observed delta of the error rate drops below the second threshold. After this threshold is met, the training stops. In total, this method introduces two additional parameters: The first threshold to determine the switch from a static learning rate to the exponential decay and the second threshold to stop the training.

Newbob+

Based on the idea of newbob, we used a similar method which we call "newbob+". With this approach, we attempted to eliminate the need for setting thresholds, at the cost of training time: We start by selecting an initial learning rate and monitor the drop in error rate after each epoch. Once the error rises, we restart the network training using the best set of parameters so far and decrease the learning rate by a fixed factor. In the

notion of exponential decay, we chose a learning rate decay of 0.5. The network training is stopped once a decrease in learning rate does not result in a decrease in error rate.

2.3 Speech Recognition Using Neural Networks

Neural network have been utilized in any part of speech recognition systems [Kil15]. In this work, we focus on using ANNs as part of the pre-processing and acoustic modeling.

2.3.1 Bottleneck Features

In Section 2.1, we have already covered traditional methods for feature extraction. Pre-processing pipelines extracting features can be enhanced by the use of neural networks. State-of-the-art speech recognition systems use bottleneck features (BNFs) [YS11, MKC+11, SKR12]. As input, traditional features like MFCCs, MVDRs or logMel scaled features are used. Frame stacking is applied to input a window covering a context of typically 5-7 frames in each direction into the network. As shown in Figure 2.3, the network features multiple hidden layers, each of which is typically 1,000 to 2,000 neurons wide. The second-to-last layer is a very narrow layer with less than 100 neurons. The purpose of this so-called "bottleneck" layer is to force the network to develop a low-dimensional feature representation, thereby discarding information irrelevant to speech recognition.

The network is trained in two steps: Pre-training and fine-tuning. We pre-trained the network greedy layer-wise, as described in Section 2.2.6. After pre-training, the narrow bottleneck layers and another wide layer are added and the whole network is trained using frame-wise state labels generated by an existing ASR system. After training, all layers after the bottleneck are discarded and the output activations of the bottleneck layer are taken as acoustic features. It is also possible to use a network architecture based on Time Delay Neural Networks (TDNNs) [WHH+87a, WHH+87b, WHH+88], which were also called shifting DBNFs [NGM+14].

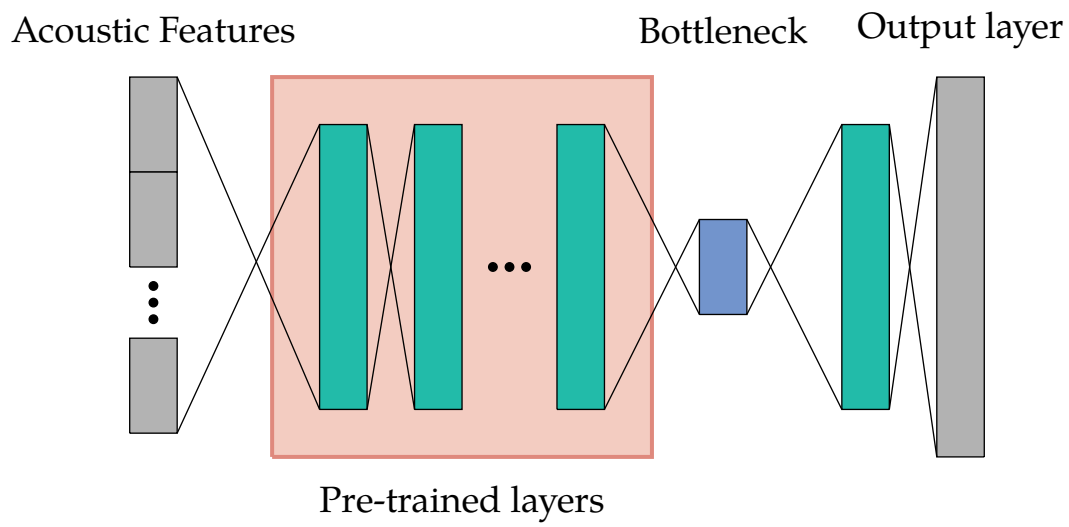

Figure 2.3: Network for bottleneck feature (BNF) extraction.

2.3.2 DNN Acoustic Models

DNNs can also be used in traditional system's acoustic models. Instead of estimating the phone posterior probabilities using GMMs, a DNN can be used instead [BM94, SLCY11, SLY11]. Trained on context-dependent phone state targets, the network outputs a probability distribution over phone states [DYDA12]. Figure 2.4 shows a combination of two DNNs in a hybrid system. The first DNN is trained to extract BNFs which are then fed into the DNN of the acoustic model for estimating phone posterior probabilities. The training procedure is the same as for BNF networks. It is divided into a pre-training and fine-tuning phase.

2.3.3 DNN Acoustic Model Adaptation

Neural networks can be adapted to various conditions by supplying additional features. These features will typically be appended to the acoustic input features of the network. Speaker adaptation is a very common example of adapting neural networks in the regimen of ASR. So-called "i-vectors" which encode speaker and/or channel properties are used [DDK+09, DKD+11, GBM+11]. Such speaker adapted networks achieve lower WERs [SSNP13]. As speaker and channel characteristics are strongly signal related, directly shifting the acoustic features based on i-vectors using a neural

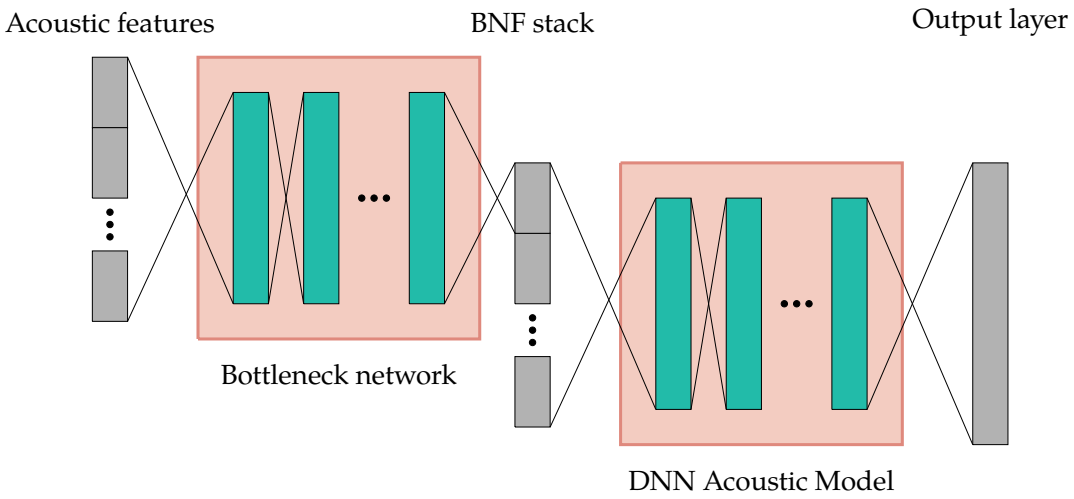

Figure 2.4: Overview of the network architecture used in hybrid DNN/HMM systems. Starting with the stack of the acoustic features, BNFs are extracted, stacked and used as input to the DNN acoustic model, which computes the phone posterior probabilities.

network improves the performance [MZM14b] even more. This method can also be applied to compensate for channel properties [MM15, GMNM17].

2.4 Speech Recognition Using Recurrent Neural Networks

In the same way as feed-forward neural networks can be used for acoustic modeling, recurrent neural networks can be used alternatively [ZSN16, ZDV+16], and replacing DNNs with RNNs reduces the WER. In addition to training traditional systems, a novel system architectures using the Connectionist Temporal Classification (CTC) loss function [GFGS06] for training recurrent neural networks recently gained substantial research interest. ASR can be considered a sequence labeling task. Based on a sequence of acoustic feature frames, a sequence of output tokens needs to be predicted. In traditional DNN/HMM based systems, a DNN is trained to predict a label for each acoustic frame. This done independently for each frame, and no sequential dependencies between frames are taken into account. An existing ASR system is required for bootstrapping. It generates a force alignment between the audio and the transcripts, which is kept fixed during neural network training. While this approach is successfully used to build ASR systems, it does not take temporal dependencies into account. Only the final sequence of output tokens is relevant, whereas the actual timing

information (or the frame-wise alignment) does not matter. CTC is a novel method for training RNNs by not requiring a pre-defined alignment between a recording and its transcript. It trains the networks in such a way that it an alignment is discovered automatically. RNNs are very powerful in learning sequence tasks and therefore very well suited.

As pointed out in [GFGS06], traditional approaches based on Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs) are the dominant frameworks for sequence labeling tasks. Their drawbacks are, that they require task specific knowledge, e.g. the design of state models for HMMs or the assumption that HMM observations are independent of each other. RNNs on the other hand do not require knowledge about the data or task at hand. They feature an internal state which provides a powerful, general mechanism for time-series modeling. In addition, RNNs are robust against temporal and spatial noises. Such perturbations occurs especially in the scenario of speech recognition, imposed by e.g. different speaking styles or conditions.

2.5 Articulatory Features

Phonemes can be described by the configuration of the articulators in the human vocal tract. Articulatory features characterize this configuration. They are the atomic units describing speech sounds produced by the vocal tract. The human sound inventory is limited by it. The International Phonetic Association published the International Phonetic Alphabet (see Figure 2.5) which organizes the phones based on their articulatory features. Two major classes can be distinguished: Vowels and consonants, each characterized by different properties:

Vowels are articulated without obstructions in the vocal tract with the vocal cords vibrating. Relevant articulatory features are the position of the tongue and the rounding of the lips.

Consonants are articulated with constrictions in the airflow. Features for characterization are the place and manner of articulation as well as the voicing.

While speaking, human articulators are in constant motion, transitioning between phones in an asynchronous manner. As a consequence, the canonically defined targets

may not be reached. The thereby introduced co-articulation artifacts depend on the phonetic context and may account for language specific colorings. More details can be found in [HAH01]. Articulatory features can be used to improve the performance of ASR systems [MW02]. Training feature detectors multilingually does improve the performance [SSMW03, SMSW03]. A more recent study of articulatory features and their application in ASR can be found in [MSN+14]. DNN acoustic models trained for ASR do learn articulatory feature detectors implicitly [NSM15].

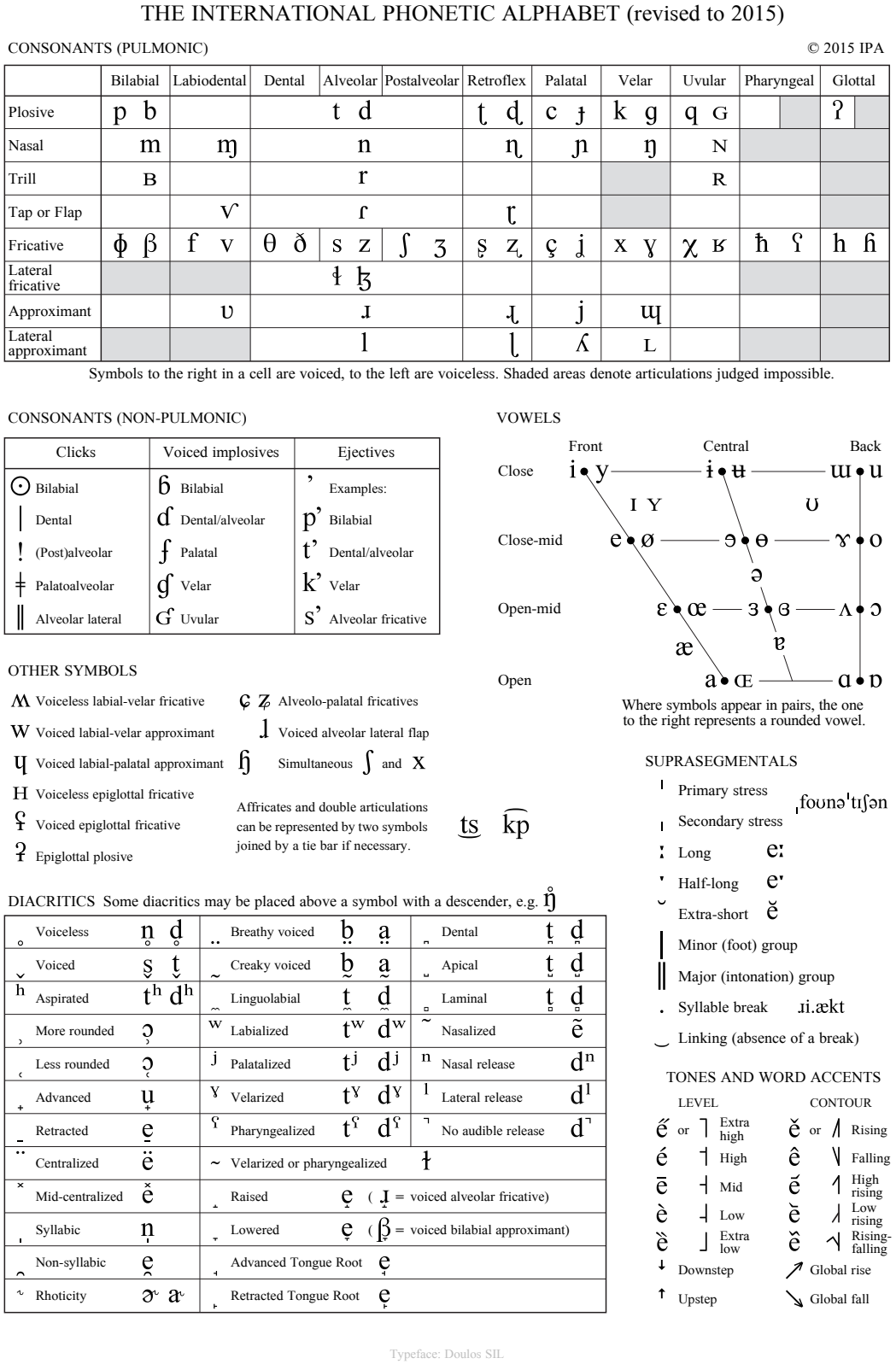

Figure 2.5: IPA Chart, http://www.internationalphoneticassociation.org/content/ipa-chart, available under a Creative Commons Attribution-Sharealike 3.0 Unported License. Copyright © 2015 International Phonetic Association.

2.6 Multilingual Speech Recognition

Multilingual speech recognition poses several challenges [WGT+00]. Training a system jointly on a combination of languages requires special adaptation techniques to account for language specific idiosyncrasies. We first describe traditional systems and how their components are adapted to multiple languages and then provide an overview of multilingual all-neural approaches.

2.6.1 GMM/HMM Based Setups

Traditional systems feature explicitly modeled components which need to be adapted. Proposed methods include efficient bootstrapping of systems for new languages [SW97] by first training a multilingual systems and then mapping the multilingual phones to language dependent ones. This method of bootstrapping enabled system building for new languages in low-resource conditions efficiently [SW98b] by the reduction of the number of parameters and system complexity by training a monolithic system. GMM/HMM based systems typically use polyphones as acoustic modeling units. As not all possible polyphones are seen during training, it is necessary to reduce the set of polyphones based on available training data. Language specific information can be utilized during this step [SW98a]. By adapting the polyphone decision tree to new languages using, e.g. ML-Mix or ML-Tag [SW00, SW01], acoustic models can be estimated for the target language given a variety of source languages. But this adaptation is limited to the polyphones seen during training. Approaches for cross-lingual adaptation were proposed [Stü08a], also by including articulatory features as language universal feature detectors [Stü08b]. Such methods can also be applied if limited data is available [Stü09].

2.6.2 DNN/HMM Based Setups

Traditional speech recognition systems use neural networks at two places to model the acoustics: The front-end for audio pre-processing to extract acoustic features and the acoustic model itself. Both parts can be trained using data from multiple languages.

Multilingual Bottleneck Features

Neural networks for BNF extraction can be trained multilingually. Because of the additional data and larger variety in speakers and acoustic phonetic sound inventories, the network parameters can be estimated more robustly and the extracted features show better results. There are multiple ways of integrating data from additional source languages into the training process. One approach is to train the hidden layers jointly, but use language dependent output layers. While the shared hidden layers learn to extract features for multiple languages, the output layers, being language dependent, only learn language specific features. Multilingual bottleneck (ML-BNF) features are language independent [VKG+12]. Using a single output layer and dividing it into language dependent blocks is also possible [GKV14].

Multilingual DNNs

Different strategies for training DNN based acoustic models multilingually were proposed. The DNN can be pre-trained multilingually [SGR12, SLF+08]. It is also possible to parallelize this training by working on distributed copies of the network [HVS+13]. Multilingual training of the ML-BNF and the DNN does improve the performance even further [VBMS12]. Another proposed method is to train the DNN on a number of languages in turn [GSR13]. For low-resource scenarios with only a limited amount of transcribed training data, using a combination of data from multiple source languages increases the recognition performance [MM13]. It is also possible to use untranscribed data semi-supervised increases the recognition accuracy [TSCH13]. Given that training on more data increases the training time of a system, different approaches to speed up the training process were proposed [MZM14a].

2.6.3 All-neural Based Approaches

In addition to approaches using traditional speech recognition systems, multilingual systems using an all-neural architecture were proposed as well. To the best of our knowledge, we were the first to propose using a global set of acoustic units [MSW17b] with a CTC/RNN setup, which we then improved in multiple iterations [MSW17c, MSW18b, MSW18c]. A similar approach was proposed [KS17], but with the difference of providing the language identity explicitly. There was also a different

approach [TSW+17] for this task proposed using and end-to-end architecture with an attention mechanism [CJLV16]. Using attention in combination with CTC was also proposed [WHH17] and refined [SWH+18] for better code switching. Similar to our study on data selection [MSS+14] (see Chapter 5), an analysis using an RNN/CTC based approach was carried out with similar findings [DSMB18].

2.7 Meta-PI Networks

Meta-PI [HW90, HW92] networks were proposed to build a neural network based modular classifier. The title of the original paper[1] summarized the two key aspects of Meta-PI networks: a) the distributed knowledge representation and b) multisource pattern recognition. The original work proposed a setup for speaker independent phoneme recognition, where subnets were trained speaker specific. The paper defines multiple scenarios:

"source independent" describes a classifier which is able to classify all sources correctly, not only ones seen during training.

"multi source" refers to the recognition of speech from speakers which were encountered during training.

"source dependent" The system is trained and tested on data from a single speaker. This scenario can be considered to be the gold standard, as it typically shows how good the recognition performance can be under optimal circumstances.

The main concept of Meta-PI is to assemble a classifier using submodules. As shown in Figure 2.6, it features multiple subnets, called "modules". Those modules were trained speaker specific. The outputs were then merged based on the output of another network, the "source id net". This network is trained to modulate the outputs of the subnets to achieve the best classification performance possible. The subnets itself remained unchanged.

The key aspect of the "Meta-PI combinational superstructure" are Meta-PI connections which allow to modulate the outputs of neurons with a coefficient. Based

[1] The Meta-Pi Network: Building Distributed Knowledge Representations for Robust Multisource Pattern Recognition

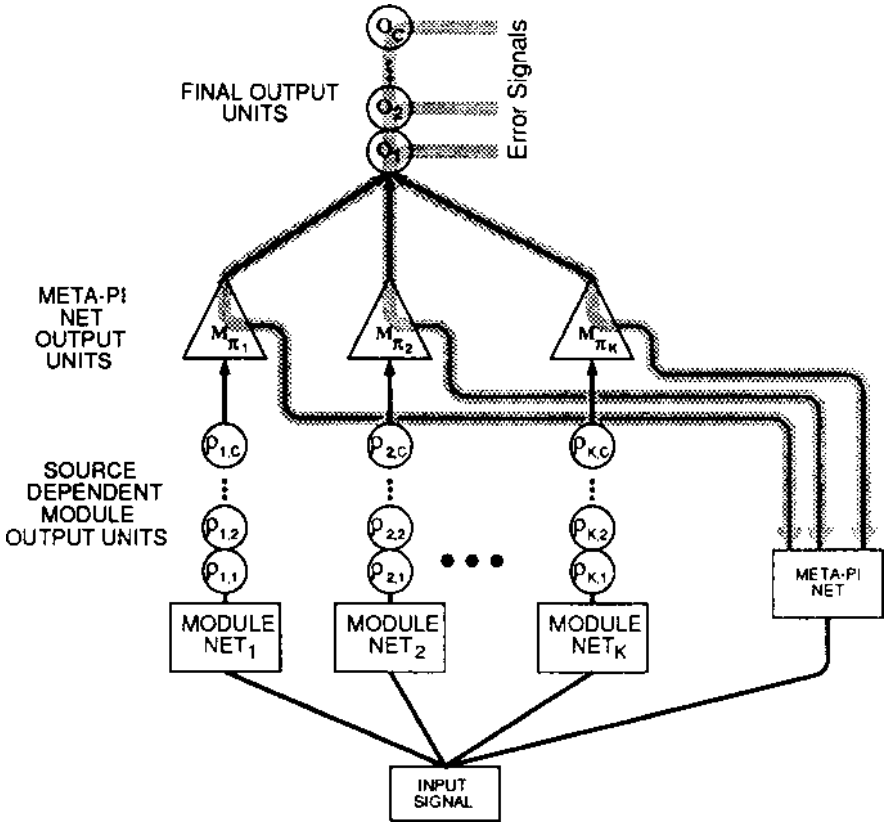

Figure 2.6: Meta-PI combinational superstructure, taken from [HW92].

on these connections, the outputs of individual subnets are being integrated into the architecture. Outputs of multiple subnets are combined in this way to obtain a global classification result. Two modes of operation exist: It is either possible to select the output of the best subnet for a given training sample, or to combine the outputs of all subnets to make a decision. The latter approach has the advantage ob being more robust towards errors in a single subnet. To compute the final outputs O_i, the Meta-PI network weighs the outputs of the k individual subnets $p_{k,i}$ in the following manner:

$$O_i = \frac{1}{\hat{\mu}} \sum_{k=1}^{K} p_{k,i} \cdot M_{\pi k} \quad \text{with} \quad \hat{\mu} = \sum_{k=1}^{K} M_{\pi k} \tag{2.12}$$

The Meta-PI network outputs one coefficient $M_{\pi k}$ per source network k. Using this architecture, speaker independent phoneme recognition using a combination of speaker dependent networks is possible [HIW90].

A similar approach to Meta-PI was proposed after [JJNH91]. In the scope of our work, the distributed knowledge representation are multiple language dependent components. Recognizing speech from multiple languages can be considered to be a multisource pattern recognition problem. Each language is thereby a single source.

Chapter 3

Experimental Setup

In this chapter, we will outline our experimental setup. Starting with data sets, we will also provide an overview of the tasks used for evaluation, as well as short descriptions of all the toolkits used.

3.1 Data Sets

In this work, we used data sets from 3 different projects, covering a wide variety of acoustic conditions. The most challenging data set originates from the BABEL project and consists of telephone recordings. Data from Euronews is of a better acoustic quality, but the available annotations are limited. Data sets used from the BULB project contain only very little data, which is challenging as well.

3.1.1 IARPA BABEL

Data used from the IARPA BABEL[1] project covers a total of 10 languages, as shown in Table 3.1. For each language, 100h of transcribed data is available. There exist two official splits of the training data: A limited language pack (LLP) with only 10h of data and a full language pack (FLP) containing all available data. The data was collected and carefully annotated by Appen[2] for this project. The annotations were provided

[1] https://www.iarpa.gov/index.php/research-programs/babel, accessed: 2018-03-29
[2] http://www.appen.com, accessed: 2018-03-29

both in the native script of the language and in Romanized form. A pronunciation dictionary based on SAMPA phonemes was supplied as well.

The recordings are narrowband telephone speech, sampled with 8kHz. As not only landlines, but also cellphones were used, the quality of the recordings varies, with some featuring heavy distortions introduced by data compression. The data was gathered in the field and as such does contain different types of background noises, e.g. street noise, animal sounds or people talking in the background. These channel conditions renders speech recognition difficult, even though the annotation is excellent.

Table 3.1: Overview of available data from the BABEL project

Language	Language Family	# Speakers	Total Hours
Assamese	Indo-European	790	52.6
Bengali	Indo-European	751	58.2
Haitian Creole	(French) Creole	697	53.9
Lao	Tai-Kadai	731	54.3
Pashto	Indo-European	1028	159.6
Tagalog	Austronesian	3051	73.2
Tamil	Dravidian	778	61.3
Turkish	Turkic	993	71.9
Vietnamese	Austroasiatic	1042	87.5
Cantonese	Sino-Tibetan	420	67.6
Zulu	Niger-Congo	767	52.6

3.1.2 Euronews Corpus

This corpus was created as part of the EU-BRIDGE project, which focused on developing automatic transcription and translation services[1]. The corpus [Gre14] consists of TV broadcast news recorded from the Euronews TV station. Given that all recordings originate from the same station, the acoustic conditions should be similar across languages. As shown in Table 3.2, the corpus features data from 10 languages, with 70h of data available per language. Annotations were created semi-automatically using a speech recognition system. Noises were annotated only in a very basic way with a single noise marker indicating all types of noises like,

[1] https://www.eu-bridge.eu/, accessed: 2018-03-29

e.g. music, non-human sounds, hesitations, false starts. No additional resources like pronunciation dictionaries were provided.

Table 3.2: Overview Euronews Corpus

Language	Audio Data	# Recordings
Arabic	72.1h	4,342
English	72.8h	4,511
French	68.1h	4,434
German	73.2h	4,436
Italian	77.2h	4,464
Polish	70.8h	4,576
Portuguese	68.3h	4,456
Russian	72.2h	4,418
Spanish	70.5h	4,231
Turkish	70.4h	4,385
Total	715.6h	44,253

3.1.3 Data Collected by BULB

The BULB project ("Breaking the Unwritten Language Barrier") [SAAD+16, ASAD+16] addresses the documentation of unwritten languages. It is focused on 3 less-resourced and mostly unwritten African languages from the Bantu family: 1) Basaa which is spoken in Cameroon, 2) Myene spoken in Gabon and 3) Mbosi which is spoken in Congo-Brazaville. The project has 3 goals: a) collection of data from those 3 langugages, b) Generation of automatic transcripts at phoneme level and the French translation thereof, c) development of tools based on the data and generated alignments aimed at supporting linguists in their work. For our experiments, we used data from Basaa and Mbosi.

Mbosi

Mbosi (also spelled "Embosi") is spoken in the region of Congo-Brazaville by 150k speakers. A detailed description of this corpus can be found in [GAAD+18]. The data was obtained using an established workflow [BHAL14]: First, recordings were created in the field, then later re-spoken by a native speaker in a controlled environment

and also translated into French. The data was phonetically transcribed using a non-standard graphemic form close to the language's phonology. Force alignments between the recordings and the transcription were created using an ASR system.

Basaa

Basaa is spoken by 300k speakers in the center and littoral provinces of Cameroon [SF17]. The data collection was done in a similar manner to Mbosi and a detailed description of this corpus can be found in [HMM+18]. Two data collection efforts were made. In the first one, local radio broadcasts were recorded, whereas in the second one elicited speech was recorded. Our experiments are based on only the first one, as it was available earlier.

3.2 Pronunciation Dictionaries

While pronunciation dictionaries were provided with the BABEL data, the Euronews corpus featured only transcriptions. In order to generate pronunciation dictionaries with a consistent set of acoustic units across languages, we used the MaryTTS [ST03] text-to-speech (TTS) system. It provided an interface which could be queried with the grapheme sequences of words and would return the corresponding phoneme sequence. While such automatically generated pronunciation dictionaries do not match the quality of manually created ones, they do however perform reasonably well. Given the available languages in the Euronews corpus and MaryTTS, we identified a set of 6 languages (English, French, German, Italian, Russian, Turkish) for which pronunciations and data would be available.

3.3 Tasks

There are more than 7,000 living languages in the world [JE05]. These languages can be divided into three categories. Tier 1 languages are high-resource and well-researched languages, but there are very few of these. Tier 2 languages are documented, but do lack sufficient resources to train ASR systems. There exists a long tail of tier 3 languages

which are undocumented. In this work, we addressed languages from each category and choose multiple tasks for evaluating our language adaptation methods.

3.3.1 Acoustic Unit Discovery

One of the first steps in documenting an unknown language (tier 3) is to determine the phoneme inventory. This is the goal of the acoustic unit discovery (AUD). As part of the BULB project we addressed the discovery of phone-like units. Our pipeline consists of multiple steps, and one is the extraction of articulatory features (AF). We studied methods to improve the extraction of AFs in multilingual scenarios using language adaptation techniques in Chaper 4.

3.3.2 Low-resource ASR

Methods for low-resource ASR address tier 2 languages who are documented, but do lack sufficient resources to build ASR systems with a good recognition performance. The availability of transcribed recordings for the acoustic model training is an important factor. A common approach is to use data from additional so-called "source languages" for acoustic model training. Source languages should be chosen regarding the target language. We evaluated methods for language selection in Chapter 5 to maximize the performance on the target language. [BBKS14] provides a general overview of methods for under-resourced languages.

3.3.3 Multilingual ASR

Given enough training data (tier 1), training an acoustic model monolingually results in the best performance. If speech from multiple languages is to be recognized, a common approach is to combine multiple monolingual ASR systems with a system for language identification. The output of the ASR systems is then switched based on the detected language. An alternative approach is to use a multilingual acoustic model which is trained on multiple languages. While is results in improvements for low-resource languages, the joint training on high-resource languages does not match the monolingual baseline. We address this issue as main topic in this thesis by studying multiple adaptation techniques in Chapters 6 to 8.

3.4 Toolkits

Janus Recognition Toolkit

Our traditional speech recognition systems were built using the Janus Recognition Toolkit (JRTk) [ea94] which features the IBIS decoder [SMFW01]. It is being developed at the Carnegie Mellon University (CMU) and at the Karlsruhe Institute of Technology (KIT). In addition to training ASR systems, we also used the audio pre-processing pipeline of JRTk for the extraction of features for all our experiments.

Theano

Feed-forward neural networks were trained using a setup based on Theano [BLP+12, BBB+10]. Theano being recently discontinued at the time of writing was one of the first toolkits with support for automatic differentiation. It allowed for writing code in Python which would then be compiled to run on either CPUs or GPUs. While this resulted in fast execution, the drawback of this approach is that runtime debugging required special methods.

Lasagne

Lasagne [DSR+15] built on Theano and provided a lightweight abstraction layer to ease handling of neural networks without the need for starting from scratch and implementing, e.g. recurrent or LSTM layers. We used it to train our LSTM networks for articulatory feature detection.

PyTorch

PyTorch [PGC+17] is a novel machine learning library, which provides Python bindings to Torch [CKF11]. It does not require a special compilation step like Theano. Being more recent, it also features a better integration with up-to-date CUDA[1] versions which result in faster processing times. It was used for training the RNN/CTC based ASR systems.

[1] CUDA (Compute Unified Device Architecture), a framework for general purpose GPU programming by Nvidia

DyNet

The dynamic neural network toolkit (DyNet) [NDG+17] is developed at the Carnegie Mellon University. It is designed for networks having dynamic structures and was used for training the RNN based language models.

Chapter 4

Acoustic Unit Discovery for Language Documentation

While the main focus of this work is multilingual ASR, we also applied the adaptation methods which we will present to a related task, the discovery of acoustic units for undocumented languages. There are more than 7,000 living languages in the world. The long tail of these languages are only spoken by a minority. They are not documented and in danger of becoming extinct. Language documentation is required in order to preserve those languages and the cultural heritage linked to them. The vast majority of undocumented languages do not have established writing systems. This renders documentation, which is a very time-consuming process, even more difficult. By using natural language processing (NLP) technology, we aim to improve this process by supporting linguists. This field of computational linguistics is relatively new research area. Fully automating the process of language discovery is next to impossible, as shown in [KM14]. Many language-specific phenomena create idiosyncrasies, with cases where even linguists disagree about certain features languages have.

Our aim is therefore not to fully automate the entire process of language documentation, but to develop tools to support linguists during their field work. One step in the documentation process is the discovery of the acoustic units of a language. We attempted to derive this set of acoustic units based on entirely unsegmented recordings of speech from the language at hand. Working without

traditional resources like pronunciation dictionaries or writing systems poses certain interesting challenges.

4.1 Our Approach Towards Acoustic Unit Discovery

There exist multiple approaches to acoustic unit discovery and a special "zero resource" challenge was created to evaluate them. Run first in 2015 [VTS+15, VAJD16] and then in 2017 [DCB+17], multiple methods for discovering linguistic units from raw speech in an unknown language were proposed. Here, we propose a three step pipeline for the discovery of linguistic units:

Phone Boundary Detection We first use a component to detect the boundaries of phone-like units. The other parts of our pipeline operate on this segmentation.

Feature Extraction For each segment, we extract features which are the basis for the clustering. It is important to extract features which are able to perform robustly in a cross-lingual scenario. In this work, we mainly focus on this step.

Clustering Based on the extracted features, we cluster the segments to derive the set of units. The clustering algorithm needs to determine the size of the set of phone-like units.

This process modularizes our pipeline and allows us to change individual parts, as well as to incorporate feedback from linguists at each stage. In the remainder of this chapter, we will discuss our approaches towards each step of the pipeline.

4.2 Phone Boundary Detection

There exist different methods to the detection of phone boundaries. One is to look for changes in the signal [SWE09]. Other methods include using Bayesian approaches, as in [yLG12, OBČ16]. RNN-based approaches have been proposed as well [MRTD17, WCL17]. We used a setup based on an ASR system, which was modified to recognize single phones. This approach was first proposed using only a monolingual speech recognition system for English [MB14, BSMB15], which we then extended using an ASR system with a multilingual set of acoustic units [VMH+16].

By using a system trained on only one language, the segmentation quality depends to a great extent on how well the source language fits the target language. Choosing the best source language given a particular target language is important, but making this determination is difficult to do in an unsupervised way. We therefore opted to train a system on multiple languages [MSW16a]. Such a setup should balance out the differences between individual languages and be less dependent on the performance of a single language. Even though choosing the best language for a given language in an oracle experiment would most likely result in a better segmentation, the multilingual system should generate a segmentation of a higher quality than one trained on a single, poorly-matched language.

We trained a multilingual system using data from 5 languages, in the same manner as described in Section 6.2. The DNN/HMM based system was trained like a regular ASR system, but was modified afterwards to recognize single phonemes. We used a special pronunciation dictionary with one "word" per phone and no language model. After decoding, we discarded the phone identities and retained only the boundaries. In addition to this baseline system, we also trained an adapted multilingual system using language codes (LFVs, see Section 6.2). As outlined in the following chapters, these codes improved performance of multilingual ASR systems, as well as for cross-lingual phone recognition (see Section 6.2.4), and we show them to also be beneficial for this task.

4.2.1 Evaluation of Segmentation Quality

To evaluate the quality of the segmentation, a common measure in literature [SWE09, yLG12, QSM08] is the F_1-Score. It is the harmonic mean between precision and recall:

$$F_1\text{-Score} = \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.1}$$

Human speech is very ambiguous. The transition of the articulators in the human vocal tract between two phones is continuous, with co-articulation artifacts observed at the boundaries. It is therefore common to allow for a fault tolerance of 20ms. This margin of error displays a good trade-off between the ambiguity of speech and being too permissive. In addition, the ground truth may also contain some noise, due to the aforementioned properties of speech.

4.2.2 Experimental Setup

We evaluated our method using data from Basaa [HMM+18] (see Section 3.1.3), a language from the Bantu family. It is spoken by approx. 300,000 speakers from center and littoral regions of southern Cameroon. The data consists of radio broadcasts that were re-spoken by a native speaker in a quiet environment. There exists no common orthography, as Basaa is an unwritten language. The recordings were phonetically transcribed by a linguist. The subset used consisted of 1h of speech. The phonetic transcription did not contain any time markers for phone boundaries.

We therefore created them using an ASR system. Given that there was only one hour of data available, training a system entirely on Basaa was not feasible, as the recognition quality would have been too low. Therefore, we used a pre-trained multilingual ASR system and adapted it to Basaa. We first mapped the phones of the phonetic transcription to the phonetic inventory of the ASR system. As the phone sets differed and not all Basaa phones had a matching counterpart in the multilingual system, we mapped the phones as best as possible manually to existing ones. The system was then re-trained using only Basaa data to adapt the acoustic model. The adapted system was then used to force align the phonetic transcripts to the recordings, and those alignments were taken as references in our experiments. But, using only 1h of speech data is not enough to fully adapt the ASR system to the new language, hence the created references do not represent a gold standard segmentation, but rather a "silver" standard with some alignment errors.

4.2.3 Results

The results [MSW16a] of the automatic segmentation are shown in Table 4.1. By applying LFVs as language codes, improvements for precision, recall and F_1-score were observed. This language adaptation technique, applied in this cross-lingual scenario, is able to improve upon the segmentation quality.

4.2.4 BiLSTM Based Segmentation

ASR systems are trained to recognize the best word sequence, but not to detect the precise timing when words or phones are uttered. We therefore evaluated an approach based on neural networks trained to recognize phone boundaries directly [FMSW16].

Table 4.1: Overview of results for cross-lingual phoneme segmentation. The F_1-Score shows 3.7% relative improvement.

System	Baseline	with LFV
Precision	0.520	**0.542**
Recall	0.515	**0.532**
F_1-Score	0.518	**0.537**

For this task, we opted for a BiLSTM based architecture, as the recurrent nature of such networks enables them to model temporal dependencies. We trained the network on data from the Euronews corpus, using data from 5 languages (French, German, Italian, Russian, Turkish). To evaluate the performance, we used both English data from Euronews as in-domain data, and data from Basaa, which was recorded using different acoustic conditions. The results are shown in Table 4.2 for English and 4.3 for Basaa. For the English and Basaa data, the BiLSTM based approach outperformed the ASR system.

Table 4.2: Results for cross-lingual phoneme segmentation on English.

System	Precision	Recall	F-Score
ASR based	0.67	0.73	0.70
BiLSTM	**0.74**	**0.84**	**0.79**

Table 4.3: Results for cross-lingual phoneme segmentation on Basaa.

System	Precision	Recall	F-Score
ASR based	0.54	0.53	0.54
BiLSTM	**0.68**	**0.72**	**0.69**

4.3 Articulatory Feature Extraction

After segmentation, the next step in our pipeline is the extraction of features for each segment. The discovery of phone-like units should be as language independent as possible. Therefore, we opted for a language-universal setup which is restricted as little as possible by the acoustic units of the source languages. Looking at the definition

of phonemes, they describe the configuration of the articulators in the human vocal tract. By directly detecting individual articulatory features (AFs), we achieve greater flexibility, as we are not limited to the AFs configurations imposed by the phones of our source languages.

There are different types of AFs, each with different numbers of classes. In total, we used 7 different AF types, 4 for vowels and 3 for consonants. Based on previous work [Met05], we used discretized versions of continuous features, like tongue position in the x and y direction, by dividing the continuous space into three buckets, either top, middle, bottom or front, middle, back. As each articulatory feature only applies to either vowels or consonants, we added an additional class to each feature representing "does not apply". The AF definitions used and the number of classes per feature are shown in Table 4.4. In addition, we optionally added a feature detector to detect the type of the phone (ptype).

Table 4.4: Overview of articulatory feature types used

Type	# Classes	Description
cplace	8	Place of articulation
ctype	6	Type of articulation
cvox	2	Voiced
vfront	3	Tongue x position
vheight	3	Tongue y position
vlng	4	Type of vowel
vrnd	2	Lips rounded
ptype	4	Type of phone

In order to generate AF training data for our experiments, we used ASR systems in combination with a TTS system. First, the ASR systems were trained for each source language, using our default method as described in Section 6.1.1. After training, we force aligned the training utterances to the audio in order to create frame-level phone labels. Based on the AF definitions embedded in MaryTTS' language definition files, we mapped phones to AFs and obtained frame level AF labels for our training data. The acoustic model of the trained ASR system used 3 sub-phone states per phone: beginning, middle and end. Similar to [Met05, MW02], we selected only the middle frames for training, because the articulators are more stable at the center of a phone. At

the phone boundaries, co-articulation artifacts are likely to occur, as shown in Figure 4.1, where extracted AFs are shown.

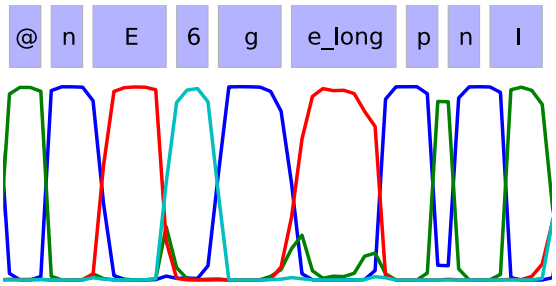

Figure 4.1: Example of extracted articulatory features, height of the tongue (vheight).

4.3.1 Articulatory Features for Low-Resource ASR

In a first approach [MSW16c], we evaluated the use of feed-forward neural network-based AF detectors in an ASR setup. As targets, we used the AF classes for each AF type, encoded using one-hot encoding. We trained individual networks for each AF, as we wanted to prevent co-adaption between AF detectors by learning that certain AF combinations are more likely to occur than others. Learning probabilities of AF combinations is an issue, because the target language may show a different distribution of AF combinations. We also applied multitask learning by sharing the hidden layers between networks, but with AF-specific output layers. The network configuration is shown in Figure 4.2.

Multilingual Articulatory Features

All systems were trained on data from the Euronews corpus (see Section 3.1.2). For this set of experiments, we used the full data set (70h) for German, French and Turkish. We simulated English as a low-resource language by using only a limited amount of data (10h). As input features to our networks, we used our default pre-processing pipeline for ASR systems and extracted logMel and tonal features using a window of 32ms and a frame-shift of 10ms.

As baseline (setup 1), we trained multilingual AF detectors on 70h of data from French, German and Turkish. For each feature, a separate network was trained. We

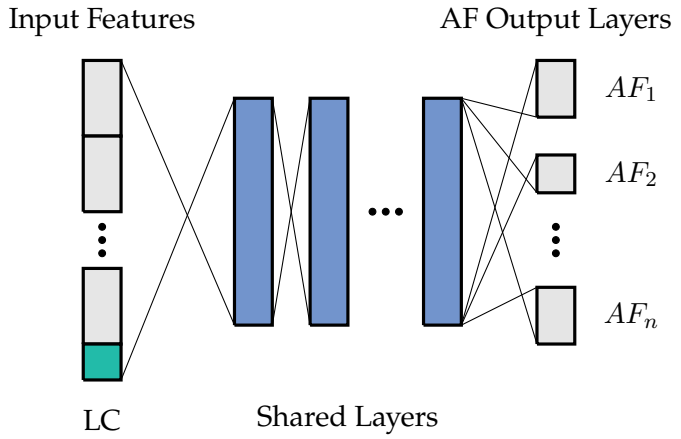

Figure 4.2: Network architecture of DNN based articulatory feature detectors using multi-task learning (MTL): the hidden layers were shared, while individual output layers for each articulatory feature were used. Language codes (LC) were optionally supplied.

then added LFVs to the input features (setup 2). In addition, we evaluated the use of multi-task learning (setup 3) by sharing the hidden layers, while using individual output layers per AF. The results for consonants (Table 4.5) and vowels (Table 4.6) show that the frame error rate (FER) can be lowered by suppling LFVs to the networks. Each AF network benefited from the additional features and outperformed the baseline. Using multi-task learning showed mixed results, with two consonant features (cplace, ctype) exhibiting higher FERs compared to the setup with individual networks.

Table 4.5: FER of AFs for consonants on the validation set. Networks were trained using 70h from French, German and Turkish. The addition of LFVs decreases the error in setup 2. Using MTL shows mixed results and does not improve the FER for all AFs (setup 3).

Setup	LFV	MTL	cplace	ctype	cvox	ptype
1	–	–	8.4	8.2	7.8	14.8
2	+	–	**7.0**	**6.8**	6.3	12.7
3	+	+	7.3	6.9	**6.2**	**12.6**

Multilingual Pre-Training

Next, we evaluated the performance on data from the target language (English) only. We chose setup 2 from the previous section, using individual networks for each AF as well as LFVs. As a baseline, we trained networks using 10h of data from each language

Table 4.6: FER of AFs for vowels on the validation set. Networks were trained using 70h from French, German and Turkish. The addition of LFVs decreases the error in setup 2. Using MTL shows mixed results and does not improve the FER for all AFs (setup 3).

Setup	LFV	MTL	vfront	vheight	vlng	vrnd
1	–	–	7.2	7.9	7.3	6.2
2	+	–	5.8	**6.6**	5.7	5.0
3	+	+	**5.7**	**6.6**	**5.5**	**4.9**

(English, French, German, Turkish), denoted as setup 1 in Tables 4.7 and 4.8. Our training process consists of two steps: pre-training and fine-tuning. In setup 2, we pre-trained the networks using 70h of data from French, German and Turkish. Then, two fine-tuning steps were performed. First, using only data from French, German and Turkish. Next, an additional fine-tuning step using data from all 4 languages was performed. As shown in Tables 4.7 and 4.8, by pre-training the networks on the 3 additional source languages, the performance on the target language can be improved for all AFs.

Table 4.7: Consonants: Classification error of AFs using different training schedules. Networks that were already trained on 3 languages and then fine-tuned again with data from 4 languages (setup 2) show better results than using only 10h of data from 4 languages (setup 1).

Setup	3L pre-train	cplace	ctype	cvox	ptype
1	–	9.1	9.7	9.5	16.4
2	+	**8.8**	**8.2**	**8.2**	**15.2**

Table 4.8: Vowels: Classification error of AFs using different training schedules. Networks that were already trained on 3 languages and then fine-tuned again with data from 4 languages (setup 2) show better results than using only 10h of data from 4 languages (setup 1).

Setup	3L pre-train	vfront	vheight	vlng	vrnd
1	–	8.8	7.9	8.3	6.0
2	+	**7.8**	**7.2**	**7.5**	**5.3**

ASR Using Articulatory Features

Next, we compared using AF detectors to traditional feature extraction for ASR systems. Features like logMel-scaled cepstral coefficients transform the raw audio signal into a low-dimensional representation of features emphasizing the parts of the signal relevant for speech recognition. We used a combination of logMel coefficients combined with tonal (FFV [LHE08] and pitch [Sch99]) features (see Section 2.1), denoted as "lMel+T" in the tables. The dimensionality of these features is 54 (40 dimensional logMel and 14 dimensional tonal features). AFs on the other hand model explicit features describing the configuration of the human vocal tract. In addition, AFs are divided into two groups for vowels and consonants, which results in only half of the detectors encode features relevant for detecting the current phone (as it is either a vowel or a consonant). Stacking the outputs of all AF networks resulted in a 39 dimensional feature vector.

To evaluate both feature extraction methods, we built multilingual ASR systems, trained on 10h of data from 4 languages (English, French, German, Turkish). We first evaluated using only one type of feature. For the baseline experiments, we did not supply language codes. As shown in Table 4.9, using AFs (setup 2) over lMel and tonal features (setup 1) does not improve the performance. Supplying language codes (LFVs) to these systems (setup 3 and 4) does lower the WER. Replacing AF nets with ones which have been pre-trained using more data from the source languages (setup 5) increases the performance, but does not match the setup using lMel+T features. One reason could be that each articulatory feature applies to only one type of phone (vowel or consonant). At each point in time, only the outputs of feature detectors for one class can be used to classify the sound, as the detectors for the other class are supposed to output "does not apply". Out of the 39 dimensional feature vectors, only half of the coefficients do carry information. In contrast to this, traditional features are agnostic to the type of phone encountered and do always output a 54 dimensional vector encoding audio features.

Combining Input Features

Using articulatory features exclusively showed no improvement in performance over traditional features, so we next evaluated different ways to combine both. Keeping the

Table 4.9: Comparison of WERs using different system configurations. Using only AFs does not improve the performance (2). Using LFVs for both lMel+T (setup 3) and AFs (setups 4 and 5) based systems improves the performance. However, systems based on AFs (4,5) did not improve beyond the lMel+T baseline (setup 1).

Setup	Features	LFV	WER
1	lMel+T	–	20.2%
2	AF (3L)	–	22.6%
3	lMel+T	+	18.7%
4	AF (3L)	+	21.8%
5	AF (4L)	+	20.2%

ASR setup identical to the previous experiment, we now evaluated adding either AFs trained on only 10h per language, or pre-trained using 70h of data from the source languages. As baseline, we used a system with lMel+T only. All setups used LFVs as language codes for multilingual adaptation. The results are shown in Table 4.10. Adding AFs trained on three languages (French, German, Turkish) did not improve performance (setup 2). However, using AFs which were fine-tuned using data from all 4 languages lowered the WER to 18.5% (setup 3).

Table 4.10: Adding AFs to acoustic features results in a slightly improved WER over the baseline.

System	Input Features	WER
1	lMel+T	18.7%
2	lMel+T + AF(3L)	19.0%
3	lMel+T + AF(4L)	18.5%

System Combination

We combined the outputs of different setups using Confusion Network Combination (CNC) [MBS00] in a final experiment. CNC is a technique for combining outputs of ASR systems, similar to ROVER [Fis97], but instead of operating on the final output, it combines the intermediate confusion networks. In Table 4.11, we first provide the WERs of each system individually. In addition to using lMel+T, we also trained a baseline system on MFCC and MVDR features (M2). LFVs were supplied to all

systems. Setups 4 through 6 show the WERs of combining each pair of two systems. In all cases, the WER drops from 18.7% to 18.1%. Combining the outputs of all 3 setups (setup 7) further reduces the WER down to 17.3%. This shows that the systems built using a different feature extraction pipeline do complement each other. This diversity leads to each system producing different errors, which are cancelled out by the system combination.

Table 4.11: Evaluation of different system combinations. Using AFs in combination with either system yields identical results to lMel+M2 in system combination. The three systems are additive, however, and combining all 3 systems results in the lowest WER.

Setup	lMel	M2	AF	WER
1	+	−	−	18.7%
2	−	+	−	18.7%
3	−	−	+	20.2%
4	+	+	−	18.1%
5	−	+	+	18.1%
6	+	−	+	18.1%
7	+	+	+	17.3%

Monolingual Phone Sets

As a contrastive experiment, we evaluated our approach using a system with a language-dependent phone set. This system was trained in the same manner, but using a DNN AM with language-dependent output layers. As shown in Table 4.12, the baseline of such a system is better compared to a system with multilingual phoneme inventory. By combining the system output of the best system combination in the previous section, we could decrease the WER from 16.5% to 15.7% which corresponds a relative improvement of 5%.

Table 4.12: Contrastive experiments using a language dependent phone set. By the addition of the best system combination from the previous section, an improvement over the baseline could be achieved.

System	WER
Baseline	16.5%
CNC	**15.7**%

4.3.2 Cross-lingual AF Extraction

Next, we evaluated our approach for AF extraction in a cross-lingual setting. Using data from 4 languages (English, French, German, Turkish), we first trained AF networks on 3 languages, leaving English out for the cross-lingual evaluation ("EN CL"). In comparison, we trained AF networks on all 4 languages ("EN ML"). The results for consonant features are shown in Table 4.13, and features for vowels in Table 4.14. Detecting AFs for an unknown language does result in higher error rates. The increase can be attributed to the acoustic phenomena not seen in one of the training languages (French, German, Turkish). Also the TTS system used to establish the mapping may feature language and voice dependent mappings of phones to articulatory features.

Table 4.13: Evaluation of classification performance on English, either multilingually (EN ML) or cross-lingual (EN CL). Cross-lingual recognition results in higher error rates. Results for consonants are shown.

Setup	cplace	ctype	cvox
EN ML	8.34	8.01	8.22
EN CL	15.93	15.60	14.29

Table 4.14: Evaluation of classification performance on English, either multilingually (EN ML) or cross-lingual (EN CL). Cross-lingual recognition results in higher error rates. Results for vowels are shown.

Setup	vfront	vheight	vlng	vrnd
EN ML	7.71	7.52	7.49	5.23
EN CL	14.07	14.43	14.96	9.49

4.3.3 BiLSTM Based AF Detection

We improved the existing approach using bi-directional LSTM networks [MFSW17a]. For speech recognition, it was demonstrated [ZSN16] that using bi-directional LSTM networks for acoustic modeling does improve the performance. The setup used multiple BiLSTM layers and sequences of fixed length. We adapted the proposed setup for AF detection. The network was trained to classify the center frame given a sequence of frames as input: To classify a frame x_i, the network was fed a sequence of frames with a fixed length ranging from x_{i-ctx} to x_{i+ctx}, where ctx indicates the

context width. The network architecture is shown in Figure 4.3. Each layer has two sublayers with 500 cells for each direction. The outputs from the two sublayers are concatenated and forwarded to the next layer. At the final BiLSTM layer, only the final sequential output is forwarded to the output layer, as we wanted to obtain only a single target value for the entire sequence of input features. The final feed-forward layer then computed the classification result. We call operating a RNN in this manner "fixed sequence mode". As input features, we used our default ASR pipeline which uses lMel and tonal features, extracted using 32ms windows with 10ms frame shift. For training the networks, we used newbob+ scheduling (see Section 2.2).

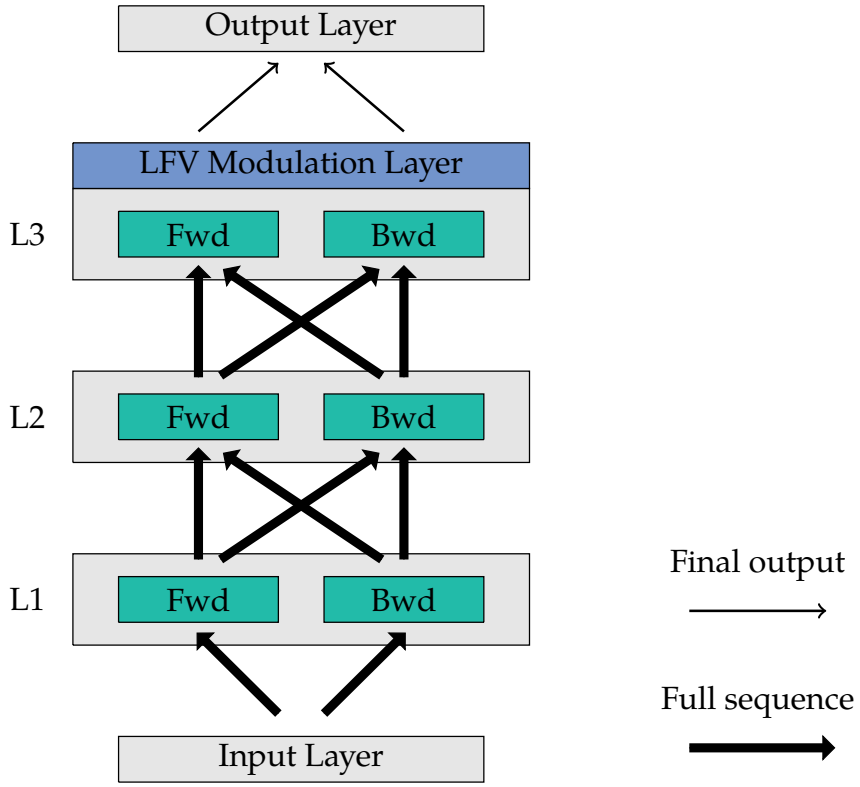

Figure 4.3: BiLSTM network architecture. The full sequence gets propagated through all BiLSTM layers. At the last layer, only the final output is retained, being modulated and forwarded to the output layer.

As baseline configuration, we were using a context width of 6 frames, a batchsize of 256, cross-entropy loss and Adam for computing the weight updates. In a first set of experiments, we evaluated different methods for computing the parameter updates, the output configuration of the recurrent layers, the mini batch size and

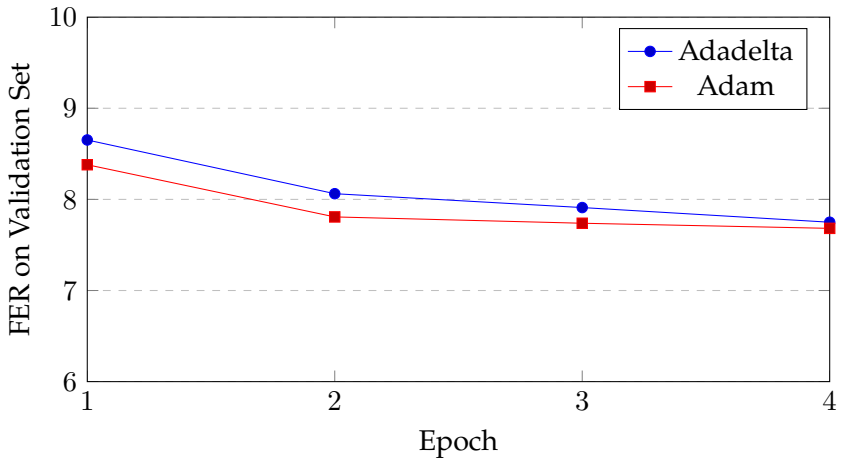

Figure 4.4: Comparison of FER using Adam and Adadelta for updating the weights.

context size. For these experiments, we trained only a single AF network to detect the place of articulation for consonants (cplace) for 4 epochs. Although a network is not fully trained after 4 epochs, these experiments should allow to optimize the hyperparameters evaluated.

Parameter Updates

We evaluated two methods for parameter updates: Adadelta and Adam. For both methods, the default parameter configurations were used. As shown in Figure 4.4, using Adam results in the error rate dropping marginal faster compared to Adadelta. After 4 epochs, the frame error rate dropped to 7.7% using Adam, in comparison to 7.8% when using Adadelta. With differences being this little, we continued using Adam from our baseline setup.

RNN Layer Output Configuration

As we operated the RNN in "fixed sequence mode", there are different options for forwarding the outputs from the final BiLSTM layer to the feed-forward layer. Our baseline configuration forwards only the final outputs of the forward and backward layers to the output layer. But forwarding the output for the entire sequence is also possible. A comparison in FERs of both techniques is shown in Figure 4.5. Providing the full output to the feed-forward layer resulted in higher error rates. After 4 epochs

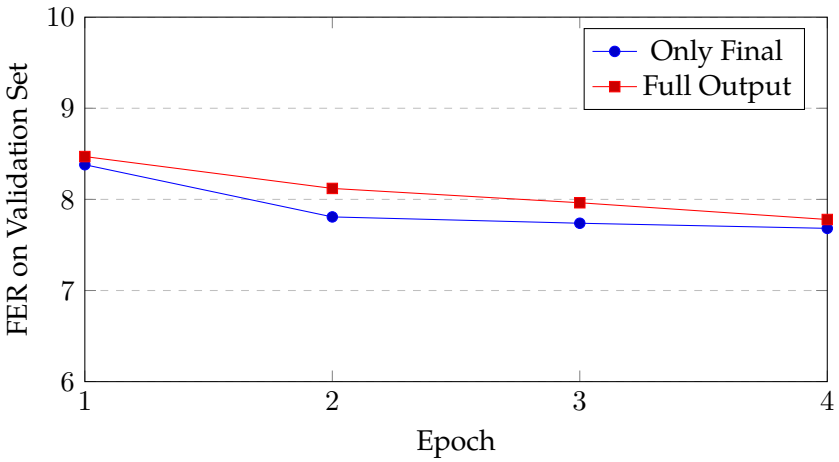

Figure 4.5: BiLSTM Output Configurations: Using the entire or only the final output.

of training, using only the final output does result in a FER of 7.7%, whereas taking the entire output results in 7.8%. Using only the final output reduces the dimensionality of the input to the feed-forward network and so the number of parameters in this layer. This makes the network less prone to overfitting. With the differences being marginal here as well, we continued using the baseline setup which uses only the final output. In addition, this allowed the framework used to apply further optimizations during training which reduced the training time.

Mini-Batch Size

Next, we evaluated different mini-batch sizes for applying the updates. Using larger batches results in fewer updates which are averaged over a larger number of samples. Smaller batches update the network parameters more frequently, but the updates are more localized as they have seen fewer training examples. In our experiment, we evaluated increasing the size of the mini-batch updates from 256 to 1024 and 2048. As shown in Figure 4.6, there is little to no difference between mini-batch sizes, with all conditions resulting in a FER of 7.7%. Hence, we did not change our default mini-batch size and kept it at 256 samples.

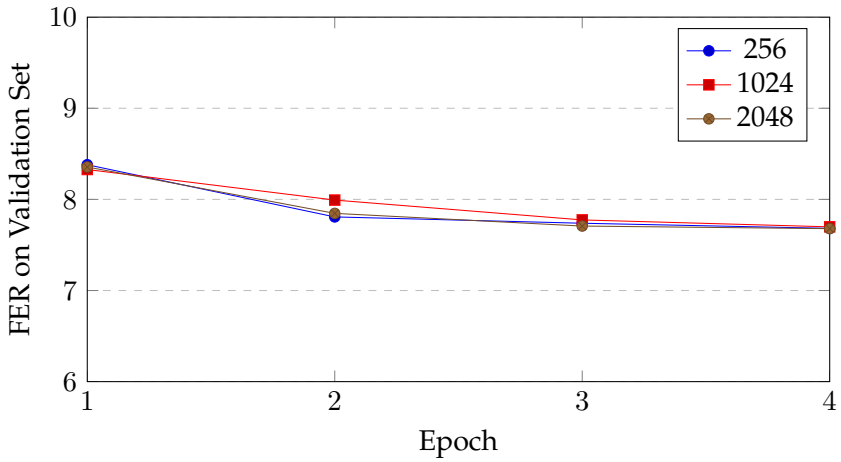

Figure 4.6: Comparison of FER using mini-batches of size 256, 1024 and 2048.

Context Size

As final parameter, we evaluated using different context sizes (here: sequence lengths) for BiLSTMs. We first used the same context (6 frames) for both the feed-forward network as well as the BiLSTM based setup. As shown in Figure 4.7 and Table 4.15, using a context of 6 frames does result in worse results for the BiLSTM based setup (7.7% vs. 7.4%). Increasing the context size (sequence length) did improve the accuracy of the BiLSTM setup, but did not improve the feed-forward network based setup. In total, using the best BiLSTM based setup resulted in a FER of 5.9%, whereas using the best DNN configuration had a FER of 7.4%. The BiLSTM based setup benefited more from increasing the context size. Due to technical limitations[1], we could not increase the context size beyond 15 frames.

Table 4.15: Classification error of different context lengths, evaluated using FFNNs as well as BiLSTM based NNs.

Network Type	Context	FER
DNN	6	7.4
DNN	15	8.8
BiLSTM	6	7.7
BiLSTM	10	6.3
BiLSTM	**15**	**5.9**

[1] Available GPU memory and training time

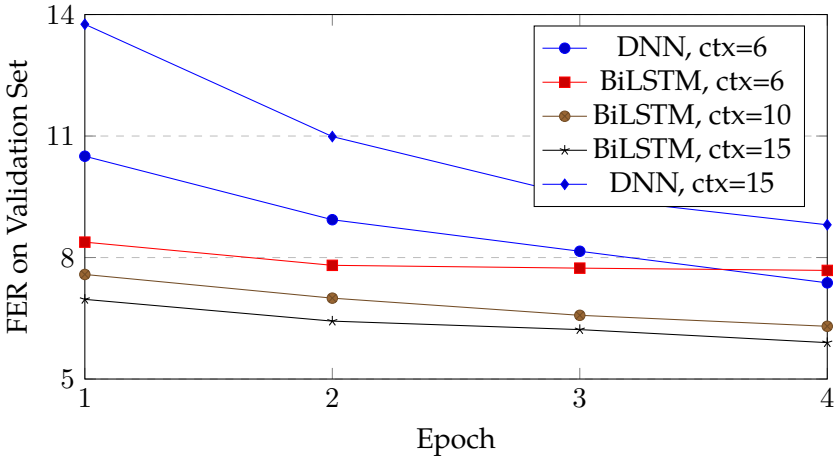

Figure 4.7: FER of different context sizes, comparing DNNs and BiLSTMs

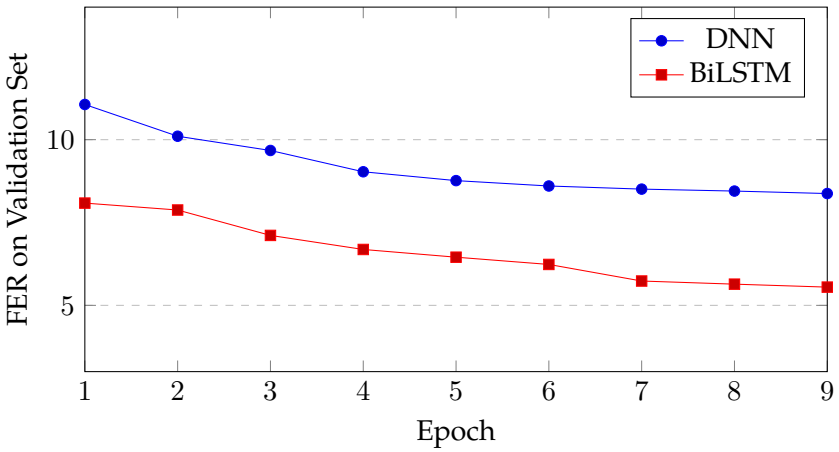

Figure 4.8: FER of the best DNN (context = 6) and BiLSTM (context = 15) setup during training over several epochs.

Multilingual Results

Based on the optimal configuration (context of $+/-$ 15 frames, minibatch size of 256, Adam, forwarding only the final output of the last BiLSTM layer), we trained AF classifiers for each AF, using a combination of 3 languages (French, German, Turkish). We omitted English as this language was used to determine the hyper parameters. As shown in Tables 4.16 and 4.17, using BiLSTMs instead of feed-forward networks did result in an improved recognition accuracy across all AFs. BiLSTMs are able to effectively use a larger context window (sequence length).

Based on the optimal configuration of DNNs and BiLSTMs, we trained networks of each kind for more epochs. As shown in Figure 4.8, the gap in FERs between the two network architectures remains stable.

Table 4.16: Classification error of AFs trained on German, French and Turkish using 70h per language. The results show the FER on the validation set.

Network Type	cplace	ctype	cvox
DNN	8.4	8.2	7.8
BiLSTM	5.7	6.4	7.1
Relative Gain	33%	22%	9%

Table 4.17: Classification error of AFs trained on German, French and Turkish using 70h per language. The results show the FER on the validation set.

Network Type	vfront	vheight	vlng	vrnd
DNN	7.2	7.9	7.3	6.1
BiLSTM	6.1	6.0	6.9	5.7
Relative Gain	16%	25%	6%	7%

4.3.4 Neural Modulation Enhanced AF Detection

We further improved our setup by applying neural modulation [MFSW17b, MSW17a], as we will propose in Section 7.3. Using the same network architecture, we modulated the outputs of the last recurrent layer, as shown in Figure 4.9. The modulation was applied in a manner similar to Meta-PI [HW92]. But instead of training the mixture weights after the main model, we derived language feature vectors (LFVs) prior to the

training (see Section 6.2). The network was then trained with these features, which allowed it to adapt to language properties. Modulation can be considered similar to dropout, but instead of randomly omitting connections, the outputs of the units will be gated systematically by multiplication with language codes which stimulates the network to learn features depending on language properties. The number of LSTM cells per layer (504) was chosen to the be a multiple of the dimensionality of the LFVs (42), because the output of each LSTM cell will be modulated with one coefficient. Both the dimensionality of the output and the language code have to match.

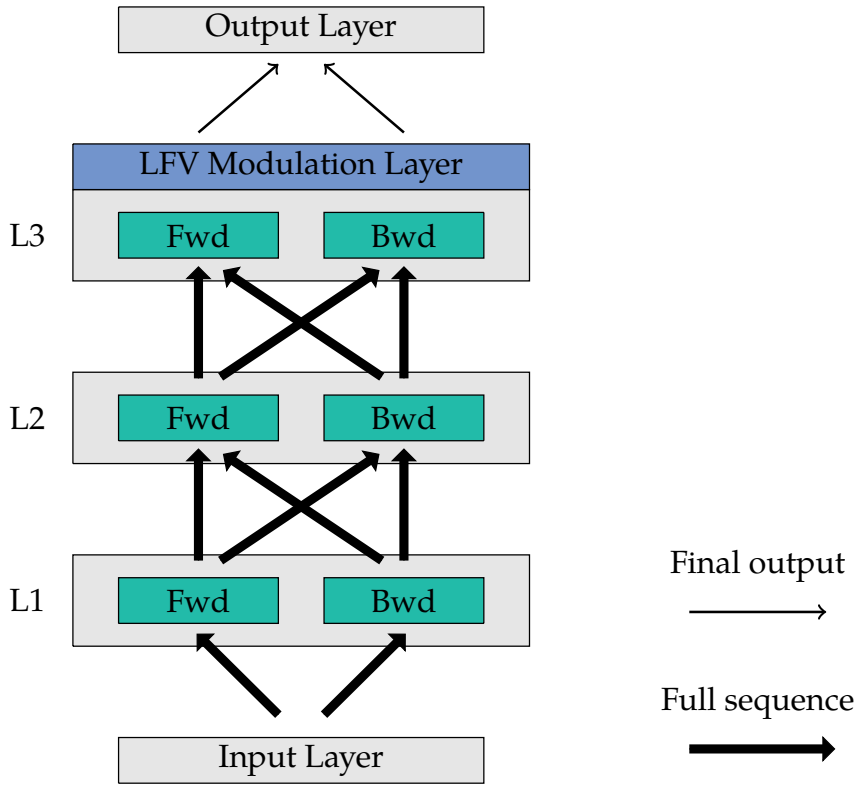

Figure 4.9: BiLSTM network architecture with modulation. The full sequence is propagated through the BiLSTM layers. Only the final output is retained after the final BiLSTM layer, being modulated and forwarded to the output layer.

The results are shown in Tables 4.18 for consonants and 4.19 for vowels. As contrastive experiments, we included results using feed-forward neural network based setups. All articulatory feature detectors benefit from adding language codes, improvements were observed by the use of LFV as additional features. Using BiLSTMs modulated with LFVs resulted in the lowest frame error rates.

Table 4.18: Classification error of AFs for consonants, being trained on German, French and Turkish using 70h per language. The results show the FER on the validation set.

Network Type	cplace	ctype	cvox
DNN	8.4	8.2	7.8
DNN + LFV	7.0	6.7	6.3
BiLSTM	5.7	6.4	7.1
BiLSTM + LFV	**5.0**	**5.3**	**5.0**

Table 4.19: Classification error of AFs for vowels, being trained on German, French and Turkish using 70h per language. The results show the FER on the validation set.

Network Type	vfront	vheight	vlng	vrnd
DNN	7.2	7.9	7.3	6.1
DNN + LFV	5.8	6.6	5.7	5.0
BiLSTM	6.1	6.0	6.9	5.7
BiLSTM + LFV	**4.8**	**5.2**	**4.6**	**4.0**

4.4 Articulatory Feature Based Clustering

As last step in our pipeline, we clustered the segments based on extracted AFs to derive a set of acoustic units. Performing this clustering process is a difficult task because of the variability in speech and unknown idiosyncrasies of the language at hand. Example methods for clustering segments are hierarchical clustering [MB14, BSMB15], the Dirichlet process [HSN16c, HSN16b, HSN16a] or the Chinese Restaurant Process [Ald85]. In this first approach, we used k-Means clustering [Har75]. This method requires the number of classes to be known beforehand, which we pretended to be known. We therefore could assess whether the proposed workflow would be suitable in general, given the number of classes.

In a first experiment, we compared the clustering performance using AF extractors trained multi- and crosslingually. We extracted AFs for each segment. By stacking the outputs of all AF detectors, we obtained a 36 dimensional feature vector. For each segment, we extracted AFs only for the inner third of frames and averaged the vectors to obtain a single feature frame per segment. The restriction to the inner third is due to co-articulation artifacts (see Section 4.3.1). We clustered all segments given the actual number of classes. Figure 4.10 shows the confusion matrix of the reconstructed set

of phone-like units for English. The AFs were trained multilingually on 4 languages (English, French, German, Turkish). English, the test language, was included in the training data. For the second setup (Figure 4.11), we removed English from the set of training languages and trained the detectors only on 3 languages (French, German, Turkish). Both figures show a similar result, but deriving a phone set crosslingually (Figure 4.11) shows an image which is more perturbed, containing more noise and more outliers. But the diagonal is still clearly visible.

4.4.1 Evaluation of Features for Clustering

We evaluated different features to cluster the segments, using features extracted in a cross-lingual manner. Multilingual bottleneck features (ML-BNFs) and articulatory features were used. The ML-BNFs were trained using a combination of 5 languages (French, German, Italian, Russian, Turkish), as described in Section 6.1.1, and the AFs were trained using 3 languages (French, German, Turkish). The quality of the clustering was determined using the adjusted mutual information (AMI) score [VEB10]. Computing this score requires the ground thruth. We therefore evaluated this approach supervised using English data. For this evaluation, we again used k-Mean clustering and computed the scores using a varying amount of classes, ranging from 20 to 80. Figure 4.12 shows the results: The scores using ML-BNFs plateau over a wide range of class counts, whereas the articulatory feature based results show generally higher scores and a peak at 33 classes. The scores for both 33 and 38 classes are given in Table 4.20. Given that the actual number of classes is 38, peaking at 33 does indicate that the clustering approach is able to derive a sub-optimal, but still reasonable set of acoustic units. As not all acoustic phenomena present in English were encountered during training with the source languages, the peak at a lesser class count may account for this.

Table 4.20: AMI Score for clusterings using either ML-BNFs or AFs.

Feature Type	33 classes	38 classes
ML-BNFs	0.397	0.394
AFs	0.489	0.481

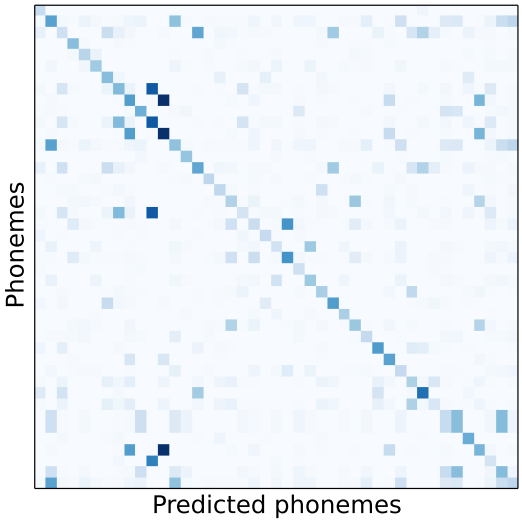

Figure 4.10: Multilingual phoneme mapping: Mapping AFs to English phoneme targets. The system was trained on DE, EN, FR and TR.

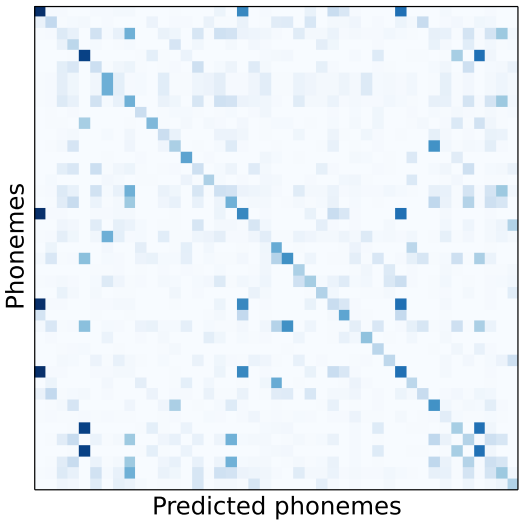

Figure 4.11: Crosslingual phoneme mapping: Mapping AFs to English phoneme targets. The system was trained on DE, FR and TR.

4.4.2 Unsupervised Evaluation on Mbosi

As final experiment, we evaluated our pipeline using data from Mbosi [GAAD+18] (see Section 3.1.3), a language from the Bantu family. The data was pre-processed in the same manner as the Basaa dataset: Recordings collected in the field were re-spoken by a native speaker in a controlled enviroment. Phonetic transcriptions were provided

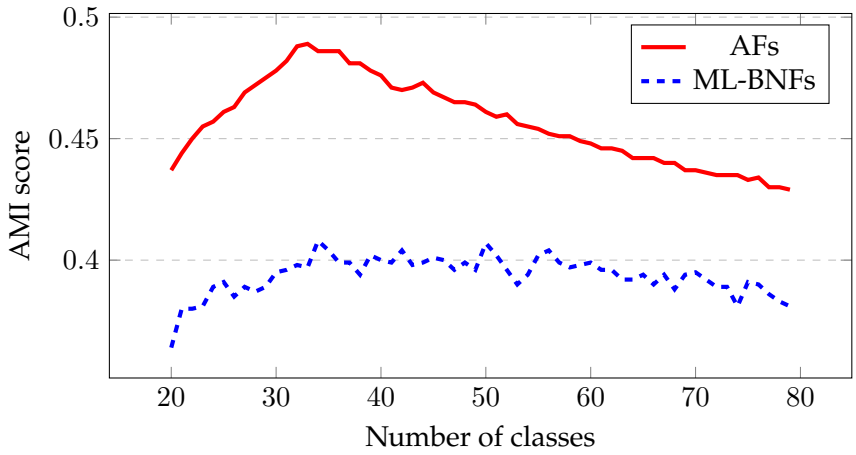

Figure 4.12: Comparison of adjusted mutual information (AMI) scores using different class counts for k-Means clustering, with features based on ML-BNFs and AFs

as well. For the unsupervised evaluation, we used the Mel Cepstral Distorion (MCD) score [MTSC01]. This metric, which is used to evaluate TTS systems, can also be used in the regimen of acoustic unit discovery [BSMB15]. First, a TTS system [TBC98] is trained [BL03, Bla06] based on transcripts created using the discovered units. Next, the trained TTS system is used to reconstruct the original recording and the distortion is determined by the MCD score.

Because of the size of the dataset and as we are only evaluating the quality of the acoustic units, we trained and evaluated the TTS on the same data. While this is not a fair evaluation of the TTS as such, it should provide an indication of the quality of the discovered units. As contrastive experiment, we trained a TTS on the manually created phonetic transcripts. As shown in Table 4.21, the MCD score using the manual transcripts is lower compared to using the inferred units to synthesize Mbosi speech. While the performance of the system based on the derived units is slightly worse, it shows that these units can be used to synthesize Mbosi speech.

Table 4.21: Comparison of MCD Scores for different conditions

System	MCD Score
Manual Transcripts	5.25
Acoustic Unit Discovery	5.78

4.5 Conclusion

We presented an approach towards the unsupervised discovery of acoustic units. Our pipeline consists of 3 steps: segmentation, articulatory feature extraction and clustering. Each of these components can be optimized independently. In this work, we focused on the extraction of articulatory features. By using additive and multiplicative language codes, we improved the performance of the feature detectors. Future work includes the optimization of the clustering process. Possible improvements are the use of a better distance measure which takes the properties of articulatory features into account by, e.g. applying weights to certain features.

If transcripts using phone-like units are available, they can be used to discover words [GAAD+16, OGB+18]. It is also possible to build speech-translation systems which directly translate the acoustic input into an other language without a written representation of the source language in between [Stü09].

Chapter 5

Language Selection

As outlined in Section 2.6.2, training neural networks for multilingual bottleneck feature extraction (ML-BNF) on a mixture of languages improves the performance. Especially in low-resource conditions, the extracted features become more robust due to the training on more data. We studied three aspects to optimize the ML-BNF training [MSS+14]: a) at which training stages data from additional source languages should be used, b) effects of language selection and amount of data used and c) the optimal set of source languages given a target language.

5.1 Experimental setup

For these experiments, we used data from the IARPA BABEL project (see Section 3.1.1), which covers many low-resource languages. As shown in Table 5.1, we used a total of 10 languages. During the course of the BABEL project, language resources were released at certain intervals, with Tamil being the latest addition at the time of these experiments. We therefore selected it as target language with the aim of analyzing the use of different source language combinations. The table also shows the amount of phones each source language shared with the target language. Depending on the source language, the sound inventory of the target language is covered to a varying degree.

Table 5.1: Language overview, including the language family, size of phone set and amount of phones each language shares with Tamil

Language	Language Family	# Phones	# Phones w/ Tamil
Tamil	Dravidian	34	-
Assamese	Indo-European	50	20 (59 %)
Bengali	Indo-European	51	21 (62 %)
Haitian Creole	(French) Creole	32	17 (50 %)
Lao	Tai-Kadai	41	20 (59 %)
Pashto	Indo-European	43	24 (71 %)
Tagalog	Austronesian	46	20 (59 %)
Turkish	Turkic	41	25 (74 %)
Vietnamese	Austroasiatic	68	18 (53 %)
Cantonese	Sino-Tibetan	37	14 (41 %)
Zulu	Niger-Congo	47	16 (47 %)

Baseline

As a baseline, we trained a system using the Tamil LLP data set only. We first built a context-independent system from scratch using a flatstart approach to bootstrap the acoustic models, as described in [SMNW14]. Trained for several iterations, the pipeline consisted of incremental splitting of Gaussians training (MAS) [KFN98], followed by optimal feature space training (OFS), which is a variant of *semi-tied covariance* (STC) [Gal99] training using a single, global transformation matrix. After 6 iterations of context-independent training, we trained a context-dependent system using 2,000 models. In preliminary experiments, we determined a size of 2,000 context-dependent models to result in the best performance for the given amount of data. Using this system, we generated labels for training the networks for BNF extraction.

Network architecture

The network for extracting BNFs featured 5 hidden layers, with 1,000 neurons each. The second to last layer was the bottleneck layer with only 42 neurons. The network was trained in two stages. We first pre-trained the network in the notion of a de-noising auto-encoder using Gaussian noise and a corruption rate of 0.2. Newbob learning rate scheduling was applied, starting the exponential learning rate decay if the frame error rate on the validation set decreased by less than 0.005 between two epochs. The training

was stopped when the FER decreased by less than 0.0001 between two epochs in the exponential phase.

Additional languages

We studied the effect of adding auxiliary languages. As aforementioned, we evaluated 3 different aspects: a) language diversity by using more languages in contrast to more data (Section 5.2), b) stages at which the data is included (Section 5.3) and c) language selection (Section 5.4). We thereby focused on the pre-processing pipeline and trained GMM/HMM based systems in the same manner while evaluating different training schedules for the ML-BNFs. The target language of all experiments is Tamil.

5.2 Combination of a Single Language with Tamil

In this first set of experiments, we used 40h of speech data from a single additional source language in combination with Tamil LLP (10h). By running these experiments, we assessed the performance gains resulting from adding each available language individually, which also served as a baseline for later experiments. By using only one additional language, the choice of languages does have the largest impact on the system performance. To increase this effect, we chose to use 4 times as much data from the additional language compared to the target language. As shown in Table 5.2, the observed gains differ depending on the language.

Table 5.2: Tamil LLP plus additional 40h of another language. The last column shows the amount of phones each language shares with Tamil

Language	WER	ATWV	# Phones w/ Tamil
Tamil (Baseline)	82.6	2.67	-
+ Assamese	82.7	3.00	20
+ Bengali	81.5	3.26	21
+ Haitian Creole	81.5	3.82	17
+ Lao	82.3	2.97	20
+ Pashto	81.5	3.48	24
+ Tagalog	82.0	3.40	20
+ Turkish	**81.3**	**3.96**	**25**
+ Vietnamese	86.5	-1.34	18
+ Cantonese	83.3	1.53	14
+ Zulu	84.6	-0.04	16

While some languages improved the system performance, others had a negative effect. This emphasizes the need to carefully select the source language(s) depending on the target language. The amount of phones shared between the auxiliary and the target language is, within certain limits, an indication of the improvement in performance. Turkish, which shares the most phones with Tamil, does lead to the highest improvements. On the other hand, languages like Vietnamese or Cantonese decrease the performance relative to the baseline. Despite the fact that these are tonal languages, they do share less phones with Tamil. One exception is Haitian Creole, which only shares 17 phones, but displays similar improvements as languages sharing more phones.

Data Mixtures

In a second experiment, we evaluated adding different amounts of data from a single source language, using the 4 languages (Haitian Creole, Lao, Assamese and Bengali) that were released during the second year of the BABEL project (Option Period 1, OP1). We trained the ML-BNFs using multilingual pre-training and shifting (see Section 2.3.1) steps. The Tamil system in Table 5.2 (WER: 82.6%, ATWV: 2.67) is the baseline. As the results in Table 5.3 indicate, using all available data (70h, FLP) from the source languages results in only marginal improvements. This maybe due to the unbalanced data ratio of 7:1, as we are using only 10h from the target language. Using only 40h does

improve the performance, leading to the best results for Bengali and Haitian Creole as additional source languages. While the composition of the data is still unbalanced, a ratio of 4:1 is more leveled than 7:1. Using only the LLP data set (10h), the data is balanced and the best results for Assamese and Lao were observed. Using a more leveled data set does improve the performance, but as both a ratio of 4:1 and 1:1 lead to the best results in different languages, further experiments are needed to determine the optimal mixture of data.

Table 5.3: Use of different amounts of data in combination with Tamil LLP. The number on the left denotes WER, the one on the right ATWV.

Language	FLP	40h	LLP
Assamese	82.4% / 2.54	82.7% / 2.37	**82.0% / 3.28**
Bengali	82.0% / 2.61	**81.5% / 3.26**	81.7% / 3.03
Hait. Creole	82.2% / 2.30	**81.5% / 3.82**	81.6% / 3.14
Lao	82.5% / 2.20	82.3% / 2.97	**81.6% / 3.31**

5.3 Methods of Using Data from Additional Languages

Next, we evaluated different methods of adding data during the training process. Neural networks are trained in multiple steps. In our setting, we perform pre-training, fine-tuning, shifting and optionally another fine-tuning step. At each step, either mono- or multilingual data can be used. We assessed 3 different cases: a) using multilingual data only during pre-training, b) using multilingual data during pre-training, fine-tuning and shifting, c) the same setup as in (b), but with an additional fine-tuning step at the end using only monolingual data.

The multilingual data was taken from all 4 OP1 languages. While we limited ourselves to this set of languages for this analysis, an evaluation based on all available languages within year 1 and 2 of the BABEL project can be found in the next section (5.4). From the target language (Tamil), we used the LLP with 10h hours of data, whereas 40h of data per language were used from the additional source languages. This results in up to 160h of additional training data.

We varied the training parameters in two dimensions: a) the training strategy, by adding multilingual data during different training steps and b) adding more source languages. We evaluated the setups using both WER and ATWV. The results are shown

in Table 5.4. Baseline is the Tamil system in Table 5.2 (WER: 82.6%, ATWV: 2.67). Using multilingual data during pre-training (a), shifting (b) in combination with a final fine-tuning step (c) results in the best performance throughout the different language combinations. Each network is first trained to extract multilingual features which are then re-fined for the target language.

Regarding the aspect of adding more languages, we chose to add each language individually to the set of training languages, in the order to increase the phoneme coverage of the target language. As the results indicate, adding more languages gradually improves the recognition accuracy.

Table 5.4: Tamil LLP plus additional source languages (**H**aitian Creole, **L**ao, **A**ssamese and **B**engali) and training methods: a) ML pre-training, b) ML pre-training and shifting, c) additional fine-tuning on Tamil LLP after shifting. The number on the left denotes WER, the one on the right ATWV.

	H	**H+L**	**H+L+A**	**H+L+A+B**
a)	82.5% / 2.18	83.3% / 1.47	82.3% / 2.93	82.2% / 2.42
b)	81.5% / 3.82	81.2% / 3.63	80.8% / 4.06	80.6% / 4.05
c)	81.2% / 3.85	80.7% / 4.13	80.8% / 4.34	**79.9% / 5.05**

5.4 Combining Multiple Languages

After establishing the optimal multilingual training strategy, we looked into the impact of language selection on the system performance given a set of available languages. Following our initial experiments in Section 5.2, we selected two sets with four languages each: The best and worst fitting ones, determined by the gains in WER and ATWV. The languages chosen are shown in Table 5.5.

Table 5.5: Overview of languages fitting best and worst to Tamil. The best fitting languages are sorted starting with the best fitting one, the worst fitting languages are starting with the worst fitting one.

Best fitting	**Worst fitting**
Turkish	Vietnamese
Haitian Creole	Zulu
Pashto	Cantonese
Bengali	Assamese

With this experiment, we aim at answering the following questions: a) will the combination of the best four languages result in additive gains, b) how big is the difference between the best and worst fitting languages, c) what is the impact of the amount of data per language and d) does using data from more languages improve the WER and ATWV more than using the same amount of data from fewer or only a single language. We trained systems using the optimal ML training strategy (see previous Section 5.3) and carried out the same set of experiments for both groups of languages.

The results using the best fitting languages are shown in Table 5.6. Adding these source languages results in better performance with each new language. Regarding the amount of data added, the series of experiments with a fixed amount of 40h of additional data in total displays the same behavior and similar improvements. Adding all 4 languages shows the best results. Although the results using only 40h of additional data over 160h in total are slightly lower (79.9% WER versus 79.7% WER), one conclusion is that adding data from more languages is more important than adding simply more data from a smaller set of source languages. Using 40h of only a single language decreases the WER to 81.3%, whereas adding the same amount but from a combination of 4 language lowers the WER to 79.9%.

Table 5.6: Use of additional languages (**T**urkish, **H**aitian Creole, **P**ashto and **B**engali) with either 40h of data **p**er **l**anguage or 40h in total for all additional languages. The number on the left denotes WER, the one on the right ATWV. The last column shows the amount of phonemes shared with Tamil.

Language	**40h p. l.**	**40h total**	**# Phones w/ Tamil**
Baseline	82.6% / 2.67	82.6% / 2.67	-
T	81.3% / 3.96	81.3% / 3.96	25
T+H	81.0% / 4.50	80.9% / 4.21	25
T+H+P	80.3% / 5.41	80.5% / 4.66	28
T+H+P+B	79.7% / 5.65	79.9% / 5.52	28

Using the second set of languages, a similar trend in improvements can be observed, see Table 5.7. The absolute numbers are slightly lower compared to the best fitting languages, which can be explained by the languages individually having a lower performance in combination with the target language. Using a combination of 4

languages yields the best performance, and differences between 160h and 40h are only marginal.

Table 5.7: Use of additional languages (**V**ietnamese, **Z**ulu, **C**antonese and **A**ssamese) with either 40h of data **p**er **l**anguage or 40h in total for all additional languages. The number on the left denotes WER, the one on the right ATWV. The last column shows the amount of phonemes shared with Tamil.

Language	40h p. l.	40h total	# Phones w/ Tamil
Baseline	82.6% / 2.67	82.6% / 2.67	-
V	86.5% / -1.34	86.5% / -1.34	18
V+Z	82.4% / 1.98	82.5% / 1.91	20
V+Z+C	82.0% / 3.05	81.9% / 3.19	22
V+Z+C+A	81.6% / 3.90	81.7% / 3.85	24

5.5 Conclusion

We have evaluated multiple training techniques for neural network based multilingual speech recognition systems using different sets of source languages in combination with Tamil as the target language. Regarding the three aspects studied, based on the results we conclude that a) using data from additional languages during all training stages leads to the best results. Regarding the aspect b) of language selection and amount of data used, the best strategy is to use data from a larger set of source languages. If the total amount of data a system is trained on is kept identical, then selecting data from more languages improves the performance. The ML-BNFs become more robust by training on more languages. This observation also holds true for all-neural ASR systems (see Section 2.4) trained using the CTC loss function [DSMB18], where a larger variety of training languages improves the recognition accuracy as well.

To select of the optimal set of source languages (aspect c), one strategy is to base the decision on the number of phones shared with the target language, but our results also showed that this does not hold true in all cases. Selecting languages based on their individual performance regarding the target language resulted in better performance.

Chapter 6

Language Adaptation by Additive Language Codes

In the previous Chapter 5, we demonstrated that speech recognition systems benefit from using multilingual data during training. In this section, we propose a first approach towards a multilingual adaptation of neural networks in speech recognition systems. While neural networks display improvements from adding additional languages, adapting the networks to those languages should improve their performance even further.

There are two major approaches to train neural networks for multilingual speech recognition: 1) Share the hidden layers between languages, but use language-specific output layers. This is a typical approach used in multi-task learning, where multiple related tasks are learned jointly. Language-dependent information is encoded by using one output layer per language. 2) Share all layers between languages by having a multilingual output layer which features a joint set of acoustic units, combined from all languages.

It has been demonstrated that neural networks do benefit from using additional information sources. One example is speaker adaptation by the use of "i-vectors" [SSNP13]. They encode speaker and/or channel characteristics in a low-dimensional representation. Appending them to the acoustic features results in lower WERs as the networks are able to adapt to those characteristics, which renders them more robust.

Similar to this adaptation method, we propose an approach for adapting networks to different languages by providing features encoding language properties.

6.1 Adaptation Using Language Identity

In this first set of experiments [MW15], we evaluated encoding the language identity information (LID) as feature vector in a naïve manner using an one-hot encoding. In our scenario, we used data from 6 languages, resulting in a 6-dimensional vector with each dimension representing one language. As shown in Figure 6.1, if the input language is, e.g. English, then the coefficient for English is set to 1, while the others are set to 0. Providing the LID to the network renders it aware of the different languages and it is able to learn language- related features, similar to speaker-related features, that allow for better adaptation. This approach provides a first indication of whether neural networks benefit from language information in a multilingual scenario, but it does not take differences or similarities between languages into account.

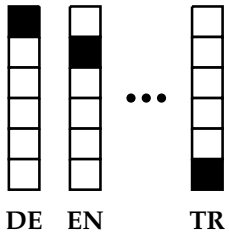

Figure 6.1: Naïve approach to encode language features by using the language identity only.

6.1.1 Experimental Setup

We conducted this series of experiments on the Euronews Corpus, see Section 3.1.2. In total, we used data from 6 languages (English, French, German, Italian, Russian, Turkish), with English as the target language. The languages were selected based on the availability of pronunciation dictionaries (see Section 3.2).

We evaluated the use of LID in two different cases: a) using adapted ML-BNF features with GMM/HMM based acoustic models and b) using adapted ML-BNF features with DNN/HMM based acoustic models. Figure 6.2 shows two possibilities

for adding the LID code: the early fusion by appending the LID prior to the ML-BNF network, and the late fusion, where the LID code is appended to the stack of ML-BNFs which is used as input to the DNN of the DNN/HMM acoustic model. For both conditions, we built systems using either a shared phone set or language-dependent phone sets. The setup with language-dependent phone sets featured networks with shared hidden layers and language-dependent output layers. As contrastive experiment, we trained an unadapted GMM/HMM based system, as well as a monolingual system on English only.

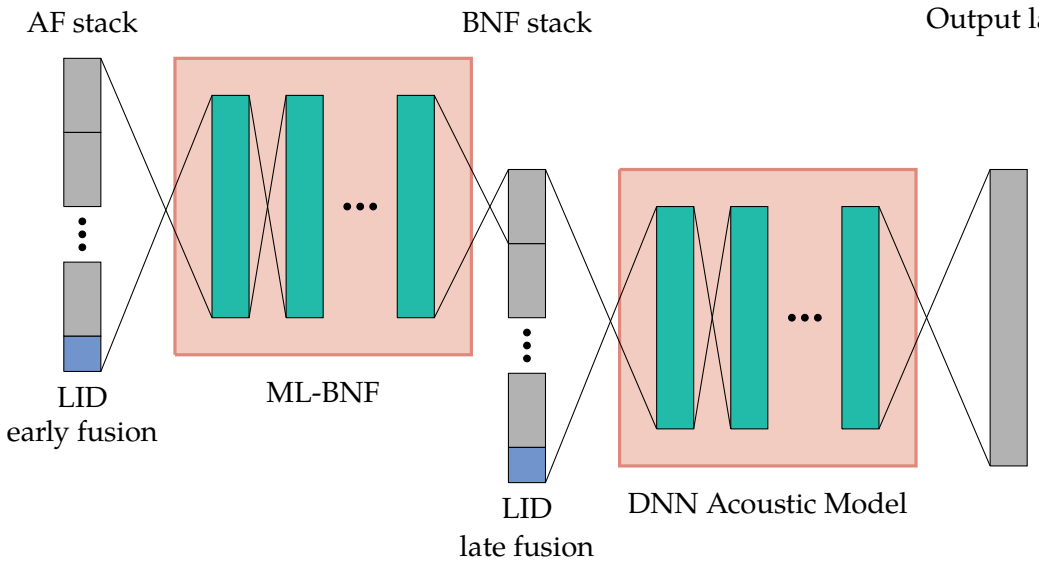

Figure 6.2: Overview of the network architecture used in our setup. We first stack the acoustic features (AF) and append a language identification (LID) code, before feeding them into the ML-BNF network. The ML-BNFs are stacked as well and the LID code is again added. The second DNN computes the phone posteriors as part of the acoustic model.

ML-BNF and DNN Training

The training schedule for both networks is identical. First, greedy layer-wise pre-training is applied (see Section 2.2.6) to initialize the weights. Next, the network is fine-tuned. Newbob learning rate scheduling with an initial learning rate of 1.0 was applied, starting the exponential learning rate decay if the frame error rate of the validation set decreased by less than 0.005 between two epochs. The training

was stopped when the FER decreased by less than 0.0001 between two epochs in the exponential phase.

Both networks were trained multilingually, with hidden layers shared between languages. Depending on the setup, either multiple language dependent output layers were used, or a single language universal output layer was used. The ML-BNF network features 6 layers with 1,000 neurons each and the DNN for the acoustic model featured 5 layers with 1,600 neurons each.

6.1.2 Results

We first present results using a merged phone set, followed by language-dependent phone sets. This section concludes with a comparison to monolingual systems. For all experiments, we used English as the target language.

Merged Phoneme Sets

Using ML-BNFs improved the performance over the baseline, and providing the LID to the network improved the performance even further (see Table 6.1). Replacing GMMs by a DNN in a hybrid setup further lowers the WER. Applying LID in both early and late fusion resulted in the lowest WER of 17.7%, an improvement of 9% relative compared to non-adapted neural networks. The gains from adapting ML-BNFs and the DNN AM were additive. Supplying the LID code to the ML-BNFs enabled the extraction of more suitable features in this setting.

Table 6.1: Overview of results for multilingual systems with a merged phoneme set, showing WERs for English. Applying the LID code improved the performance.

System	Baseline	LID adapted	Rel. gain
Unadapted GMM/HMM	26.3%	-	-
ML-BNF	21.7%	**21.2%**	2.4%
+ Hybrid	19.3%	**17.7%**	9.0%

Language-Dependent Phoneme Sets

In a resource-rich scenario, the performance of speech recognition systems with a language-dependent or monolingual phone set is typically better compared to

their multilingual counterparts. As shown in Table 6.2, using the early fusion and adapting only the ML-BNF-network degrades the performance of the system. Using multiple language dependent output layers encodes the language information in itself. Applying the LID in addition shows no improvement. For the hybrid system, we therefore only applied the late fusion and observed a 3.5% relative gain over the unadapted baseline.

Table 6.2: Overview of results for systems using separate phoneme sets per language, showing WERs for English.

System	Baseline	LID adapted	Rel. gain
Unadapted GMM/HMM	18.9%	-	-
BNF	**17.5%**	18.7%	-6.4%
Hybrid	14.9%	**14.4%**	3.5%

Comparison to Monolingual Systems

As constrastive experiments, we trained monolingual systems on English and compared their performance to the best language-adapted systems with language-dependent phone sets. Table 6.3 shows the results. Training a multilingual system with language-dependent phone sets and applying LID by late fusion resulted in an improved WER compared to a system trained monolingually, although the gain is smaller compared to adding this feature to the multilingual system.

Table 6.3: Comparison of results for monolingual systems and multilingual setups using separate phoneme sets per language, showing WERs for English.

System	Monolingual	Multilingual	Rel. gain
GMM/HMM	18.9%	18.9%	-
BNF	18.6%	**17.5%**	6.3%
Hybrid	14.6%	**14.4%**	1.4%

6.1.3 Concluding Remarks

We have demonstrated that neural networks can be trained to adapt to languages in a multilingual scenario. Depending on the condition, early and/or late fusion of the

language information lowered the WERs. Providing LID explicitly to the network enables the network to capture language-specific features, which results in better multilingual performance.

6.2 Language Feature Vectors

Previous experiments have shown that neural networks benefit from supplying language information in a multilingual scenario. But providing only the language identity information does not account for language properties, relationships or similarities between languages or language families. Therefore, a better solution to encode language properties is required, and is expected to improve performance. In the regimen of speaker adaptation, a method similar to i-vectors but using neural networks was proposed. These so-called bottleneck speaker vectors (BSVs) [HS15] encode speaker properties, but are extracted entirely using a neural network. This network is trained to discriminate speakers and features a bottleneck layer. After training, all layers after the bottleneck are discarded and the output activations of this layer are taken as BSVs. We extracted language feature vectors (LFVs) in a similar manner [MSW16a], by training a bottleneck network for language identification.

6.2.1 LFV Network Architecture and Training

The network architecture is based on the architecture of hybrid ASR systems and composed of two networks: a ML-BNF net for feature extraction and a DNN with a bottleneck for language identification as shown in Figure 6.3. The ML-BNF is trained in the same manner as for an ASR system, using logMel and tonal acoustic features, similar to Section 6.1.1. It features 7 layers with 1,600 neurons each. The DNN for language classification has 6 layers and 1,000 neurons per layer, with the second-to-last layer being a bottleneck layer with only 42 neurons. Language information is expected to be longer-duration in nature compared to the length of phones. The pipeline therefore needs to be adjusted to capture language properties by using a larger context window of input frames. The same training strategy as for our ASR systems was used, with a pre-training and fine-tuning step. Newbob learning rate scheduling was applied, starting the exponential learning rate decay if the frame error rate in

the validation set decreased by less than 0.005 between two epochs. The training was stopped when the FER decreased by less than 0.0001 between two epochs in the exponential phase.

The ML-BNF net was trained first using data from 5 languages (French, German, Italian, Russian, Turkish) and then kept fixed. For training the LID network, we used data from all languages within the Euronews corpus, except for English. English, as our target language, was omitted since we wanted to assess the performance of LFVs on a language that was not seen during training. To extract language feature vectors (LFVs), we discarded all layers after the bottleneck layer and used the output activations of the bottleneck as feature vectors.

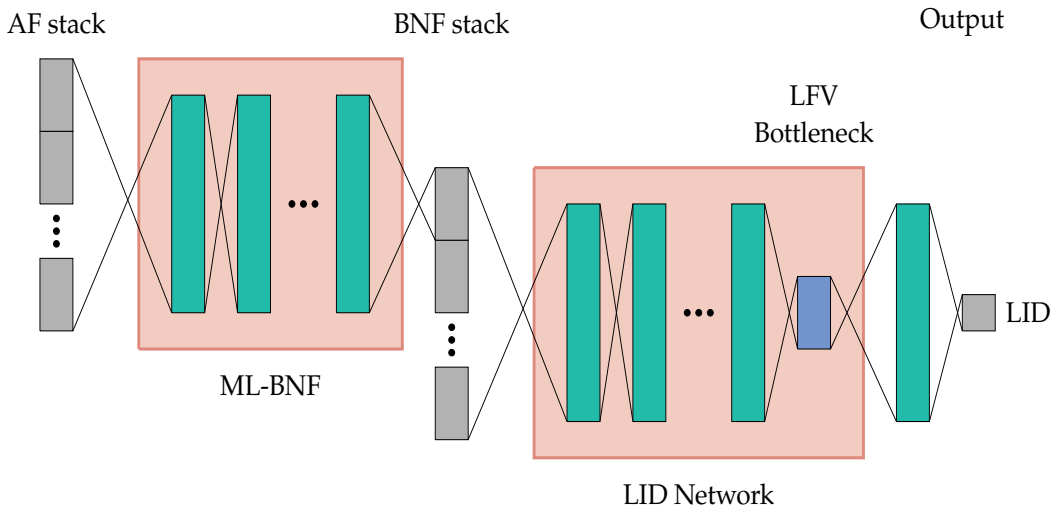

Figure 6.3: Overview of the network architecture used for LFV extraction. We first stack the acoustic features (AF) as input to the ML-BNF in order to extract BNFs. The BNFs are stacked and input into the LID network. This DNN is trained for language identification.

6.2.2 LFV Network Hyperparameter Optimization

We evaluated various parameters of the setup for LFV extraction [MSW16b]. We first studied using different kinds of acoustic features. Next, we varied the size of the hidden layers and the bottleneck. With the network geometry fixed, we determined the optimal context width. Based on the best network configuration, we built ASR systems using both multilingual and monolingual phone sets [MSW16a]. In a final

set of experiments, we evaluated the cross-lingual performance of LFVs on a phone recognition task.

Input Features and Network Topology

As input features, we evaluated using both logMel and tonal features, as well as ML-BNFs extracted based on these features. In addition, we varied both the network topology, as well as how the data is shuffled during training. We studied two kinds of topologies: using the default geometry where all hidden layers except the bottleneck layer have the same size, and a tree-shaped topology where each additional hidden layer has fewer neurons than the previous one. The expectation is that each hidden layer extracts higher order and more abstract features, which can potentially be represented by fewer neurons. Such an architecture uses fewer parameters and should therefore be less prone to overfitting and require less resources for training.

The second aspect explored concerns the shuffling of the data during training. When preparing data files for training, the data is divided on a per-speaker basis to multiple jobs which generate the data files. Those files will be merged afterwards, with the effect that the data is semi-randomly shuffled on a per-speaker basis. During training, the data is loaded in chunks of approx. 3 GB in size into memory, with a total number of chunks of approximately 100. Locally within each chunk, the data is then shuffled on-the-fly. Using an additional global shuffling step prior to training, the data within each chunk contains samples from a larger set of speakers.

The results are shown in Table 6.4. The column "tree" denotes whether results use a tree-shaped network geometry. With respect to shuffling, we indicated using local or global shuffling in the column "shuffle". As the results indicate, shuffling the data globally does reduce the frame error, as the network sees frames from a larger pool of speakers during each mini-batch. But given only minor improvements in the error rate, the increased computational effort required to train networks in such a way was not borne out. Using a tree-like structure resulted in a higher error rate, hence we kept a network architecture with large, equally sized hidden layers. Using a network with a non-tree shaped structure and local shuffling, we evaluated the use of ML-BNFs. In contrast to logMel and tonal features, this resulted in a lower FER.

Table 6.4: Different network configurations for LFV extraction, using local or global shuffling, and optionally a tree-like structure.

Type	Tree	Shuffle	FER
logMel + T	–	local	0.218
logMel + T	–	global	0.204
logMel + T	+	local	0.245
logMel + T	+	global	0.212
ML-BNFs	**–**	**local**	**0.172**

Hidden Layers and Bottleneck Size

Next, we varied the size of the hidden layers and the bottleneck. Starting with a configuration of 1,600 neurons per hidden layer and 42 neurons for the bottleneck like for an ASR setup. For speech recognition, the size of the bottleneck (42) is chosen to be much smaller than the number of output neurons (>1000). In the current case of language recognition, the network has a much lower number of output neurons (9). We therefore evaluated two different hidden layer sizes: 42 neurons, which we would use for the extraction of BNFs, and 5 neurons, to have a bottleneck layer which is smaller than the number of output neurons.

As shown in Table 6.5, using either smaller hidden layers or a smaller bottleneck decreases the performance of the network. The best configuration uses 1,600 neurons with a bottleneck of 42 neurons.

Table 6.5: Validation error for different hidden layer and bottleneck configurations for LFV extraction.

Hidden layer size	Bottleneck Size	FER
800	42	0.181
1,600	**42**	**0.172**
1,600	5	0.178

Context Width

Another parameter to be optimized is the amount of context which is fed into the LID network. Networks trained for speech recognition typically use a context window of 100 - 150ms, which is in the range of the average length of a phone. But language

information is supposedly longer-duration in nature and therefore requires a larger context. For varying the context width, we kept the number of input frames fixed, but used different spreads by omitting every n-th frame. We chose this method because preliminary experiments indicated that increasing the input dimensionality would degrade the performance. By increasing the spread, the context window increases to a multiple of 11 frames on each side with the same dimensionality. Since we extracted feature frames over a 32ms window with a 10ms frame-shift, there are some redundancies present in adjacent frames. We did not alter the context of the ML-BNFs network. This network uses a context of $+/-$ 7 frames in each direction, so the actual context which contributes to a single ML-BNF feature frame is much larger. We evaluated various context lengths, as shown in Table 6.6. It can be seen that using a spread of 3 results in the best performance. Increasing the context width further decreases the classification performance, presumably due to skipping too many frames in between. By using a spread of 3, there is still an overlap between adjacent frames by 2ms.

Table 6.6: Overview of different context widths for LFV extraction, showing FER for language classification.

Context width	Spread	FER
460ms	2	0.142
690ms	**3**	**0.136**
1380ms	6	0.139

6.2.3 LFV Analysis

Based on the previously determined optimal configuration, we now evaluate if the extracted LFVs do in fact encode language properties or if they correspond to other differences between recordings from different languages, e.g. a different subset of speakers, or a different TV studio with different acoustic conditions. Given that we chose the Euronews corpus, differences between languages related to acoustics should be minimal. As a first experiment, we extracted LFVs using data from all speakers in the training set. For better visualization, we applied the t-stochastic neighborhood embedding (t-SNE) [MH08] to reduce the dimensionality, so that the LFVs could be projected onto a 2D plane. t-SNE is known for being able to project high dimensional

data into a low dimensional space while preserving the structure. Figure 6.4 shows the projection. Each data point was colored according to the language identity. For the figure, we selected 10,000 samples across the training data randomly. As the figure indicates, t-SNE clustered the LFVs by language. This is an indication that LFVs represent language properties, as LFVs from the same language are projected onto the same region and appear to be clustered together.

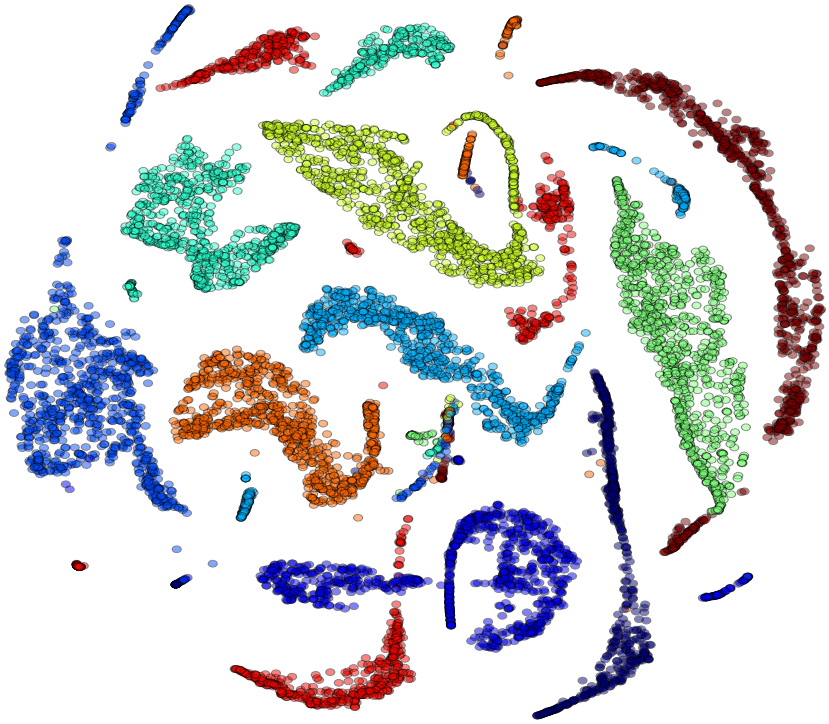

Figure 6.4: t-SNE projection of LFVs, colored by language identity

Language Identification Across Corpora

As a contrastive experiment, we attempted to detect languages across corpora to rule out that the network determines the language by acoustic cues other than the language spoken. Examples would be different types of air conditioning, jingles (as we are using TV broadcast news) or the microphones used. While such factors definitely have an impact on the system performance and the system should adapt to them, our goal

for adaptation using LFVs is to be transparent to those effects. For our experiment, we recorded two speakers in our lab reading English text as test data. Speaker 1 is a German native speaker and has a very strong German accent. The recording was performed in Karlsruhe at KIT. The mother tongue of speaker 2 is French. He speaks with an accent, although it is not as noticeable as the one from speaker 1. He recorded the data in Pittsburgh at CMU. Both recordings were done in a controlled environment, but the acoustic conditions differ from our training data.

In order be able to recognize the language based on LFVs, we derived so-called language prototype vectors by extracting LFVs for all speakers in the training database and averaging the LFVs per language. This resulted in 10 LFV representations, one for each language in the training set. For each speaker in our test set, we created a single LFV and computed the Euclidean distance between each of these vectors and the language prototype vectors.

As shown in Figure 6.5, the distance of the speaker 1's LFV to the English prototype vector is the lowest. The second lowest distance is to German, but only by a small margin compared to other languages like Italian or Arabic. Figure 6.6 shows a similar behavior for speaker 2. Here, the distance to English is the lowest as well, but the second lowest is Spanish, whereas the distance to French is the highest. These results indicate that the LFV network has learned to extract language features, as LFVs can be used to distinguish languages, independent of other factors like, e.g. acoustic conditions, which may be different between languages. In case of speaker 1, the LFVs also indicated his strong German accent, but this does not hold true for speaker 2. It may be related to the strength of the accent, but the available data is insufficient for a clear judgment. A systematic evaluation of accents is difficult as the degree to which someone speaks with an accent is difficult to measure. A single canonical accent does not exist, although people having the same mother tongue often pronounce foreign languages similarly. The language classification performance was studied in more detail [Dra17], evaluating different conditions. Trained on the Euronews corpus, we evaluated the systems on European Parliament speech, as well as on an internal data set based on lecture recordings. We showed that our approach enables language recognition across corpora, including the on-line recognition of the language with little delay and high accuracy.

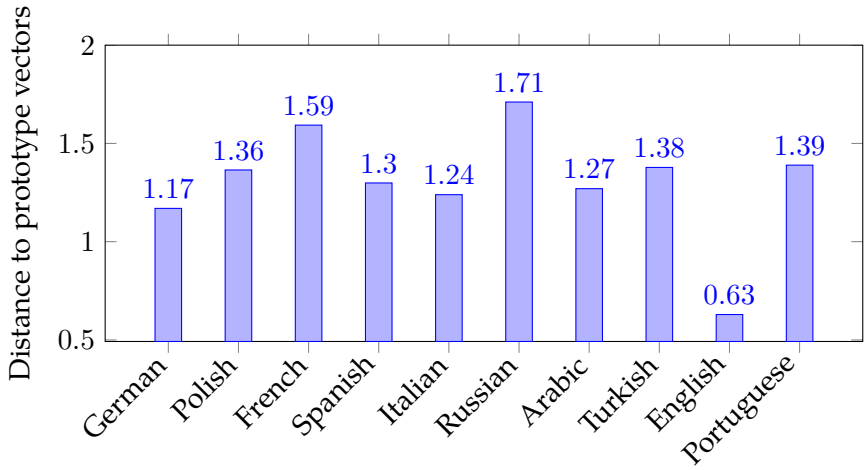

Figure 6.5: Comparison of distances from prototype vectors to LFVs from speaker 1 (German mother tongue).

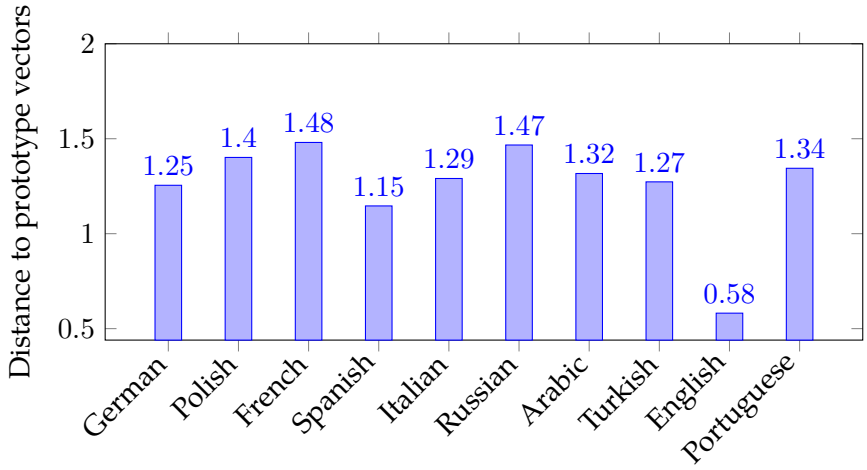

Figure 6.6: Comparison of distances from prototype vectors to LFVs from speaker 2 (French mother tongue).

6.2.4 Multilingual Speech Recognition

We chose the best configuration for LFV extraction and trained ASR systems with it [MSW16a]. For evaluation, we used a setup similar to the one described in Section 6.1 for the LID as language code. We again used data from the Euronews Corpus (see Section 3.1.2). In total, we used data from 6 languages (English, French, German, Italian, Russian, Turkish), limiting ourselves to 30h per language. Figure 6.7 shows the network architecture and where LFV language codes were added. The ML-BNF network featured 6 hidden layers with 1,000 neurons per layer, and the bottleneck had a size of 42 dimensions. Multiple input feature frames were stacked using a context of $+/-$ 6 frames. The network was trained multilingually with shared hidden layers and language-dependent output layers. The DNN acoustic model featured 6 hidden layers with 1,600 neurons per layer. As input features for this network, ML-BNFs with a context of $+/-$ 7 frames were used. Both networks were trained in a similar manner using a pre-training and fine-tuning step (see Section 6.1 for details). LFVs were added to both feature frame stacks, as indicated.

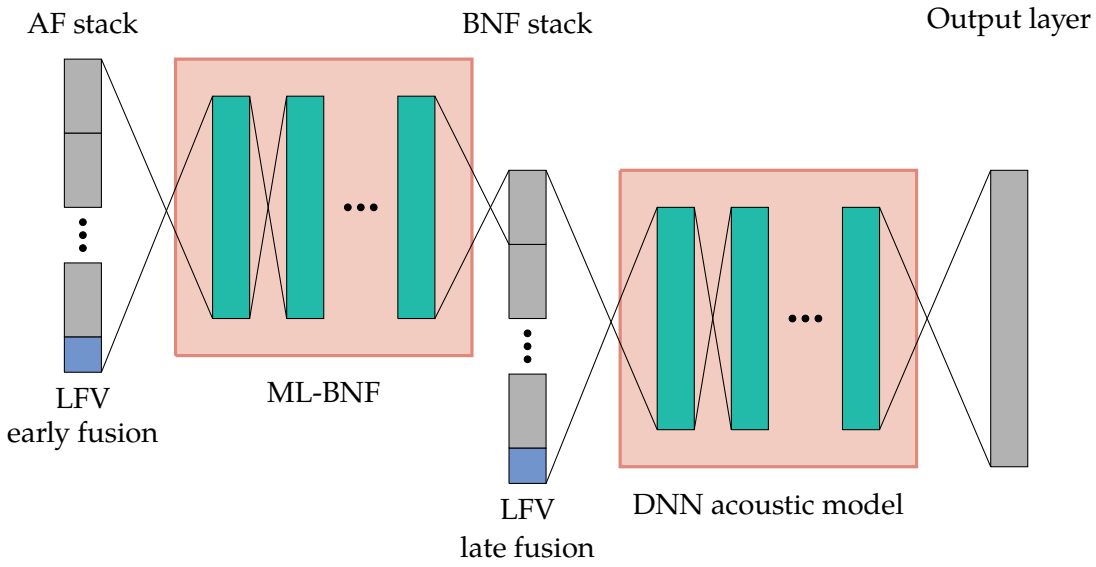

Figure 6.7: Overview of the network architecture used in our setup. We first stack the acoustic features (AF) and augment them with language feature vectors (LFV) before feeding them into the ML-BNF network in order to extract adapted ML-BNFs. The ML-BNFs are stacked as well and the LFV code is added again. The second DNN computes the phoneme posteriors.

Multilingual Phones Set

We first studied adding LFVs to a DNN/HMM hybrid system with a merged phone set. The GMM/HMM base system was trained multilingually using a joint set of phones on all 6 languages. Table 6.7 shows an overview of the results. To compare our results, we also included numbers from systems trained using LID (see Section 6.1). The WER of the setup trained using LFVs is lower compared to using the LID. This indicates that the networks are able to better adapt to multiple languages if LFVs are used instead of the LID. The extracted language properties allow for better adaptation compared to the language identity.

Table 6.7: Overview of WERs for multilingual systems, comparing LID and LFVs for adaptation.

System	BNF	Hybrid
w/o LI	21.4%	19.1%
LID	20.7%	17.7%
LFVs	20.7%	**16.2**%

Language-Dependent Phone Set

In addition to using a multilingual phone set, we trained systems with multilingual acoustic models and language-dependent phone sets. We first trained monolingual systems for each language, which were used to obtain training data for the DNN acoustic model. It was trained in such a way that the hidden layers were shared among languages, with language-specific output layers. The results in Table 6.8 indicate that there is still a gap between language-specific and multilingual systems. However, this gap decreases by using LID and even further by using LFVs.

Table 6.8: Comparison of WERs using mono- and multilingual phoneme sets in combination with LID and LFVs for language adaptation

System	w/o adaptation	with LID	with LFVs
Monolingual	16.7%	16.6%	15.3%
Multilingual	19.1%	17.7%	16.2%
Loss in perf.	14.4%	6.7%	5.8%

Cross-lingual Phone Recognition

In addition to building multilingual speech recognition systems, we also evaluated the use of LFVs in a cross-lingual scenario. In cases where no training data for the target language is available, one method is to use an existing speech recognition system and establish a mapping between the phone set of the source and the target language. For this experiment, we pretended English to be a language without available training data. We trained an ASR system with a multilingual phone set on 5 languages (French, German, Italian, Russian, Turkish). This system was configured for phone recognition and a manual mapping between the multilingual phone inventory of the system and the target language (English) was established. Table 6.9 shows the phoneme error rate (PER). Although the PERs are high (which is to some degree typical for phoneme recognition systems, especially for cross-lingual systems), the additional use of LFVs results in a decreased PER.

Table 6.9: Overview of results for cross-lingual phoneme recognition. The results show the phoneme error rate (PER).

System	PER	rel. improvement
Baseline	84.0%	—
with LFV	**81.4%**	**3.2%**

6.3 Conclusion

We studied two methods for adapting neural network based multilingual acoustic models to languages. Improvements were observed by supplying the language identity information as one-hot code to the network. But this method does not take language properties into account. We therefore used an ancillary neural network trained for language identification to extract language feature vectors, which encode language properties and not only the language identity alone. Using these features as language codes improved the performance further. We evaluated both setups using multilingual ASR systems. With a language universal phone set, the WER drops from 19.1% to 17.7% using the language identity and to 16.2% using language properties for adaptation. When training a multilingual acoustic model with a language dependent

phone set, the WER drops from 16.7% to 15.3% when using language properties for adaptation. While the performance of systems with a monolingual phone set is not yet met using a multilingual one, the use of additive language codes decreases the gap in recognition performance between the two setups.

For future work, an alternative to the extraction of language codes using neural networks is to explicitly model language properties, based on the World Atlas of Language Structures (WALS) [DH13]. It is a large database of structural language properties, which could be used instead of language features extracted entirely via a neural network.

Chapter 7

Language Adaptation by Multiplicative Language Codes

Using language codes which are added to the acoustic features improved the performance of multilingual ASR systems. But language properties are not as signal related as channel or speaker properties. In this chapter, we evaluate integrating language codes deeper into the network architecture. Traditional speech recognition systems feature many explicitly modeled components, like, e.g. context-dependent phones. In a multilingual setting, these components need to be adapted [SW98b, SW00] to the training on multiple languages. Recently, systems based on RNNs trained using connectionist temporal classification (CTC) [GFGS06] have gained in popularity. Instead of featuring many explicitly modeled components, such setups rely on a single neural network which models all aspects implicitly. As a consequence, the WER of such systems is typically higher compared to traditional systems because of less explicitly modelled knowledge. The networks have to learn everything by themselves.

Neural adaptation techniques become more important for all-neural setups, and adapting the neural network also adapts all implicitly learned features. Such systems are therefore better suited to study the effects of adaptation. We here present a method called "modulation" which uses language codes to gate the outputs of BiLSTM layers. This stimulates the network to learn features based on language properties.

7.1 Multilingual Systems Using Two Languages

We started by building systems using data from a pair of languages [MSW17b], English and German. Using a neural network architecture based on Baidu's Deep Speech 2 system [AAA+16], our setup featured two 2D convolutional / TDNN layers and 4 bi-directional LSTM layers, as it is shown in Figure 7.1. The output layer is a feed-forward layer which maps the output of the last BiLSTM layer to the output targets. To compute the loss, Baidu's CTC implementation called "warp-ctc" was used.

Deep Speech 2 uses a so-called end-to-end approach, where a single neural network is trained to recognize speech without explicitly modeling components like context-dependent acoustic models, pronunciation dictionaries or a language models. Using only the audio and the sequence of output characters, the network learns to generate transcripts from recordings. The Baidu setup uses the log spectrogram as input features. Instead of using a pre-processing pipeline with Mel-Scaled filterbanks, the 2D CNN layers implicitly learn filters to extract features relevant for speech recognition from the raw spectrogram. Traditional systems would use an additional neural network trained to extract bottleneck features.

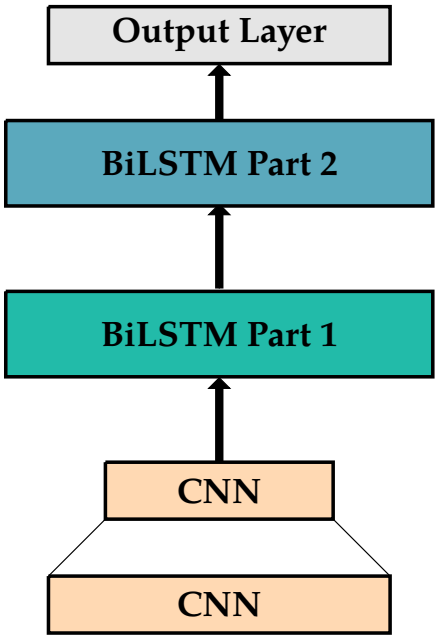

Figure 7.1: Network architecture, based on Baidu's Deepspeech2 configuration.

7.1.1 Experimental Setup

We used phones as acoustic modeling units. While the original Baidu setup used graphemes as acoustic modeling units, we chose phones as we opted for a simpler task in this first approach. By using phones, the networks do not need to learn pronunciation rules. To create a global phone set, we merged the phones sets of English and German, ensuring that the same sounds would map the symbols across languages. While the same sounds do share the same symbols independent of the language, the network has to learn phone contexts, which are language specific as the sounds do feature a language specific coloring.

We based our experiments on the Euronews corpus (see Section 3.1.2). Utterances shorter than 1s or having phonetic transcripts longer than 639 items were removed because of an internal limitation of the CUDA/warp-ctc implementation[1]. In total, we used 35h of data for training, and 5h for testing. The pronunciation dictionaries were created using MaryTTS. Each language has its own set of phone symbol. While most symbols represent the same phone across languages, some symbols differ between English and German. To ensure that the same symbols represent the same sounds across languages, we manually mapped symbols to create a global set of acoustic units. We used the definition of articulatory features for each phone in the language definition files of MaryTTS.

In addition to the mapping of sounds across languages, we also mapped symbols within languages. MaryTTS uses markers to indicate long vowels. In a series of preliminary experiments, we determined that our setup frequently confused long and short instances of the same vowel. Hence we discarded the length markers and use only one symbol per vowel. Table 7.1 shows the amount of phones before and after the mapping. The resulting multilingual phone set has a size of 56 phones, with 39 phones from English and 48 from German.

To extract acoustic features, we used the Janus Recognition Toolkit (JRTk) [ea94], which features the IBIS single-pass decoder [SMFW01]. We used our standard pre-processing pipeline consisting of 40 dimensional log Mel scaled coefficients and 14 dimensional tonal features (FFV [LHE08] and pitch [Sch99]). The features were extracted using a 32ms window with 10ms shift. For training the networks, we created

[1] see: https://github.com/baidu-research/warp-ctc, accessed 2018-04-13

Table 7.1: Size of different phone sets

Language	Phone Set	Size
English	MaryTTS	42
	Mapped	39
German	MaryTTS	59
	Mapped	48
Combined	Merged	56

a setup based on PyTorch. We applied stochastic gradient descent with Nesterov momentum of factor 0.9, and used a batch size of 20 with batch normalization and learning rate annealing with a factor of 1.1 after each epoch. In addition, we applied a max norm constraint of 400. The utterances were sorted ascending by length to stabilize the training, because the weights of the network are initially uninitialized. Shorter utterances feature less acoustic frames and target symbols and therefore account for less ambiguity.

7.1.2 Monolingual Baseline

We first trained monolingual systems, evaluating the different phone sets. The networks featured 2 2D CNN layers and 4 layers with 400 bi-directional LSTM cells each. As shown in Table 7.2, mapping phones based on articulatory features improved the performance. For English, the PER drops from 20.4%, while an improvement from 16.0% to 15.5% could be observed for German.

Table 7.2: Monolingual results on the test set showing the phone error rate (PER)

System	Phone Set	PER
English	MaryTTS	20.4%
English	Mapped	**19.0**%
German	MaryTTS	16.0%
German	Mapped	**15.5**%

7.1.3 Multilingual Experiments

Next, we built multilingual systems using the combined data from English and German. We varied the amount of layers and the number of BiLSTM cells per layer. As shown in Table 7.3, using 4 layers with 1,000 BiLSTM cells per layer results in the lowest PER. The "ML PER" shows the performance of the multilingual system evaluated on both English and German. For reference, we also trained an English monolingual system for some conditions.

Table 7.3: Multilingual results showing the phone error rate (PER) for different network configurations

BiLSTM layer size	# BiLSTM layers	ML PER	EN PER
350	5	19.6%	–
400	4	20.0%	19.0%
400	5	19.6%	–
600	4	17.3%	–
800	4	16.9%	17.8%
800	5	17.0%	–
1000	**4**	**16.3%**	**17.7%**

7.1.4 Language Adaptive Networks

Using the best network configuration as determined in the previous section, we applied LFVs to the network for multilingual adaption. Encoding language properties and being extracted using a neural network, adjacent dimensions in LFVs do not display a spacial relationship. We therefore appended them not to the input but to the output of the convolutional layers. The addition of LFVs did reduce the PER, see Table 7.4. Trained for only 7 epochs, the multilingual setups outperformed the monolingual ones. But training for more epochs, the PER of the monolingual systems drops below the multilingual one. One explanation for this behavior could be the amount of data used. The multilingual system uses twice as much data as the monolingual one. Hence, the network is trained with twice as much data during each epoch, therefore twice as much mini-batch updates are applied.

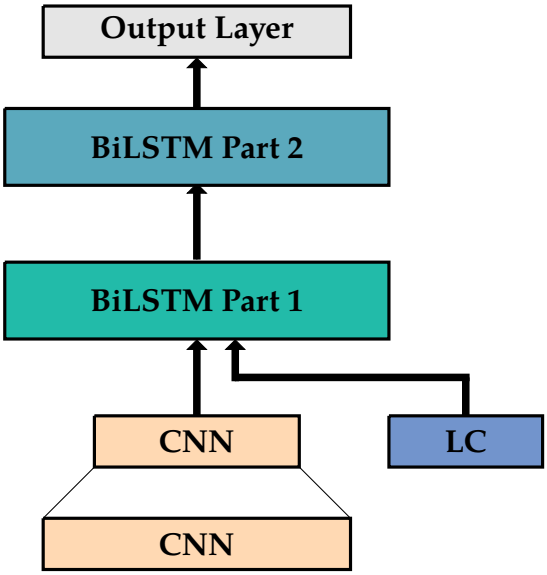

Figure 7.2: Network architecture with LFVs being added after the final convolution layer.

Table 7.4: Multilingual results showing the phone error rate (PER)

System	Type	LFV	PER (7 ep.)	PER (70 ep.)
English	Monolingual	–	17.7%	13.1%
	Multilingual	–	18.7%	14.8%
	Multilingual	+	**16.4**%	**13.5**%
German	Monolingual	–	14.6%	10.8%
	Multilingual	–	14.0%	11.8%
	Multilingual	+	**13.8**%	**11.0**%
Combined	Multilingual	–	16.3%	12.9%
	Multilingual	+	**15.7**%	**12.4**%

7.2 Multilingual Systems Using Multiple Source Languages

Based on initial results in the previous section, we extended our approach in three ways [MSW17c]: a) using data from 4 languages, b) using ML-BNFs over plain input features and c) omitting the pronunciation dictionary to train the systems on graphemes only. The data of the additional languages, French and Turkish, was processed in the same way as English and German. The additional phone symbols were mapped in the same manner to expand our global phone set. For evaluation, we used WER in addition to the PER/CER. For this, we decoded using a RNN based a basic character language

model.

Multilingual BNFs

We first evaluated using ML-BNFs instead of basic input features. In traditional systems, the use of (ML-)BNFs does improve the system performance. Similar gains should be observable for RNN/CTC based setups. We evaluated the performance by training monolingual systems for both English and German. The ML-BNFs were trained on 5 languages (French, German, Italian, Russian, Turkish) in the same manner as described in Section 6.1.1. As shown in Table 7.5, using ML-BNFs does improve the performance for both languages. Even though the ML-BNFs were not trained on English, the improvement from 13.0% to 10.2% indicates that ML-BNFs are able to extract generalized features across languages.

Table 7.5: Comparison of using ML-BNFs over log Mel + tone features

Condition	English CER	German CER
log Mel + Tone	13.0%	10.8%
ML-BNF	10.2%	7.8%

Multilingual Phoneme Based Systems

Next, we built systems using data from all 4 languages (English, French, German, Turkish). As contrastive experiments, we trained monolingual systems for each language. In the multilingual case, we also appended LFVs to the output of the CNN layers for language adaptation. As shown in Table 7.6, the monolingual systems do outperform the multilingual ones, but when using LFVs for adaptation, the gap between mono- and multilingual systems becomes smaller.

Table 7.6: Character Error Rate (**CER**) of multilingual (**ML**) phoneme CTC based systems, trained on 4 languages.

Condition	DE	EN	FR	TR
Monolingual	7.8%	10.2%	8.3%	7.1%
ML	9.9%	14.1%	12.8%	8.4%
ML + LFV	8.9%	12.9%	10.7%	7.6%

Multilingual Grapheme Based Systems

As the final experiment in this series, we evaluated the performance of our setup using graphemes as acoustic modeling units over phonemes. Using graphemes allows to omit the pronunciation dictionary, but the networks then have to infer the pronunciation rules in addition. Automatically learning pronunciation rules is a difficult task, e.g. the character sequence "ough" has eight different acoustic realizations[1] in English. In a multilingual scenario, inferring pronunciation rules becomes more difficult, as different acoustic inputs will result in the same grapheme sequence, e.g., "hat" or "die" in English or German.

As shown in Table 7.7, applying LFVs improves the performance throughout the different languages. In comparison to phoneme based systems, the CER of German and Turkish is lower for the grapheme case. A possible explanation would be that those languages feature a closer letter to sound mapping compared to English or French. Hence, the pronunciation rules could be learned more easily. On the other hand, the quality of the pronunciation dictionary might be not optimal as it was created fully automatically using a text to speech system.

Table 7.7: Character Error Rate (CER) of multilingual grapheme based systems, trained on 4 languages.

Condition	DE	EN	FR	TR
Monolingual	7.5%	12.9%	11.5%	6.6%
ML	9.1%	15.6%	13.4%	7.9%
ML + LFV	**7.9%**	**14.3%**	**12.5%**	**7.3%**

Decoding with RNN LM

In order to assess the performance of our setup by computing WERs, we used a neural network based language model to perform a greedy decoding [GFGS06]. The RNN LM was trained on the training utterances of the acoustic model only. While language models are typically trained on several million sentences, we here used only

[1]The following sentence of unknown origin contains all pronunciation variants of "ough": "A rough-coated, dough-faced, thoughtful ploughman strode through the streets of Scarborough; after falling into a slough, he coughed and hiccoughed."

110k sentences. But this being in-domain data, decoding with this model should still indicate if the gains observed in CER are also reflected in WER. The results in Table 7.8 show improvements by the use of LFVs after decoding.

Table 7.8: Word Error Rates (WERs) of English grapheme based CTC systems. Adding LFVs improves the multilingual performance.

Condition	Mono	ML	ML + LFV
English	25.2%	30.8%	**28.1%**

7.3 Neural Network Modulation

Appending LFVs to the acoustic features did show improvements for both feed-forward and recurrent neural network based setups. Given the recurrent nature of the latter, adding static features to the input is potentially not the best way for adapting this type of networks. In addition, language properties are not as signal related as, e.g. channel or speaker properties. Therefore adding language features deeper into the network architecture should improve the performance. We therefore evaluated a method called "modulation" in order to incorporate language codes in the network architecture [MSW18b]. Our goal is to force networks to learn features based on language properties.

Modulating Layers

First introduced as part of Meta-PI networks (see Section 2.7), the modulation is implemented as special connections which allow to multiply the outputs of a unit with a coefficient. In the original work [HW90, HW92], the modulation was used to combine the outputs of multiple source networks by a weighted sum, using one coefficient per network. We here applied the modulation in a different manner to the outputs of a hidden layer. One method to alter the way how networks learn features was proposed with "dropout training" [HSK$^+$12, SHK$^+$14]. By omitting connections between neurons randomly, dropout prevents co-adaptation as each training step would see a different network configuration. Instead of randomly omitting connections between neurons, the modulation emphasizes or attenuates the outputs, based on

language codes. It can therefore be considered as an "intelligent" way of dropout, where connections are systematically altered instead of on a random basis. The network will learn features modulated by language properties, as the outputs of units will be gated by language codes. In order to modulate the output of a layer, the same amount of coefficients as there are outputs are required.

In our scenario, we modulated the outputs of a BiLSTM layer with LFVs, while otherwise using the same architecture as in our previous experiments. The network architecture is shown in Figure 7.3. In a series of preliminary experiments, we determined that modulating the output of the second layer does result in higher performance. Possible reasons for this are that the lower two layers of the network are able to learn the relevant features and that the upper layers after the modulation learn language adaptive higher order features.

A related approach to modulation is stimulated training of DNNs [TSG15, WKGS16], where constraints are applied during training which influence the behavior of hidden units by arranging them in a 2D grid and enforcing spacial relationships.

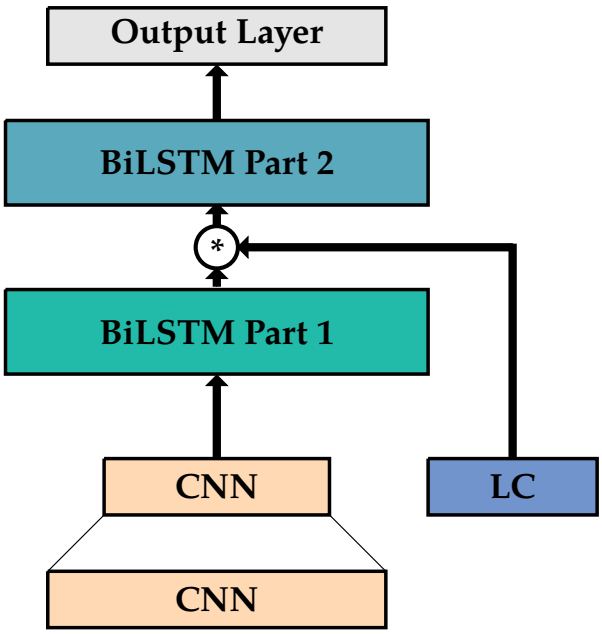

Figure 7.3: Network layout, based on Baidu's Deepspeech2 [AAA+16]. Modulating the output of the second LSTM layer improves the performance more than adding LFVs after the CNN / TDNN layers.

Network Architecture

For our setup, we chose the number of BiLSTM cells per layer to be a multiple of the dimensionality of the language codes. Given that c is the dimensionality of the language code and the number of BiLSTM cells per layer are $n \cdot c$, we then needed to stack the language codes n times to match the dimensionality. This way, we could match the dimensionality of both by stacking the language codes multiple times. The output layer could therefore be considered divided into n groups of c, with each group being modulated by the same coefficient.

7.3.1 Experimental Setup

To evaluate this approach, we used the same setup as in the previous Section 7.2. The systems were trained using a combination of 4 languages from the Euronews corpus. To study the effectiveness of the modulation in comparison to the addition, we trained systems of each type using identical conditions. We evaluated using graphemes and phonemes as acoustic units, as well as networks with both 420 and 840 bi-directional LSTM cells per layer. We evaluated our method also by varying the amount of data, using either 8h to simulate a resource constraint task or 45h for a resource-rich environment.

7.3.2 Results

The resource constraint condition was only applied to English in full, as we were using ML-BNFs as input to our setup and these were trained using 70h of data from French, German, Italian, Russian and Turkish, see Section 6.1.1. We present the results of our multilingual systems on the test set for each language individually, as well as averaged across all languages.

Grapheme Based Systems

We first evaluated the performance using graphemes as acoustic modeling units. As shown in Table 7.9, applying the modulation improves the performance in contrast to the simple addition of the language codes to the acoustic features. Improvements can be observed across languages. Using the full training set (45h), improvements across

Table 7.9: CER of grapheme based system trained on 8h per language, 420 BiLSTM cells per layer

Condition	AVG	DE	EN	FR	TR
ML Baseline	31.2%	30.8%	38.0%	29.4%	30.9%
Appending	25.9%	22.9%	33.3%	27.3%	21.3%
Modulation	**24.6%**	**20.7%**	**32.7%**	**25.4%**	**19.6%**

all conditions were observed (Table 7.10) as well. The relative improvements are within the same order of magnitude. As we were using more data, we evaluated increasing the

Table 7.10: CER of grapheme based system trained on 45h per language, 420 BiLSTM cells per layer

Condition	AVG	DE	EN	FR	TR
ML Baseline	14.4%	10.6%	18.2%	15.9%	9.1%
Appending	13.0%	9.5%	16.1%	14.3%	8.1%
Modulation	**12.4%**	**9.1%**	**15.5%**	**13.6%**	**8.0%**

size of the layers, as with more data more parameters can be estimated. With approx. 4 times as much data (8h vs. 45h), we doubled the number of BiLSTM cells per layer. As shown in Table 7.11, the CER decreases by the increase in layer size. Supplying LFVs still improves the performance, but the difference between modulation and addition becomes smaller, reaching parity for Turkish.

Table 7.11: PER of grapheme based system trained on 45h per language, 840 BiLSTM cells per layer

Condition	AVG	DE	EN	FR	TR
ML Baseline	12.2%	8.9%	15.0%	13.5%	7.9%
Appending	10.8%	7.9%	13.6%	11.8%	**7.1%**
Modulation	**10.7%**	**7.7%**	**13.3%**	**11.7%**	**7.1%**

Phoneme Based Systems

Next, we evaluated phoneme based systems in the same manner as grapheme based ones. Starting with only 8h of data (Table 7.12), improvements by modulation over addition can be observed for all languages. Increasing both the amount of training

Table 7.12: PER of phoneme based system trained on 8h per language, 420 LSTM cells per layer

Condition	AVG	DE	EN	FR	TR
ML Baseline	23.2%	21.7%	27.2%	23.9%	21.6%
Appending	21.9%	20.9%	26.4%	21.3%	19.5%
Modulation	**20.6%**	**19.0%**	**25.6%**	**19.8%**	**17.6%**

data and the size of the hidden layers improved the performance (see Table 7.13). In contrast to grapheme based systems, phoneme based ones benefited more from the modulation in this configuration. 45h of acoustic training data may not be enough for the networks to capture pronunciation rules needed for generalizing. Training on phonemes multilingually is an easier task, as the same sounds are represented by the same targets across languages. The phones appear in language specific contexts, which accounts for language dependent co-articulation artifacts. Using language codes enables a better adaptation to such scenarios, as the lower error rates indicate.

Table 7.13: CER of phoneme based system trained on 45h per language, 840 LSTM cells per layer

Condition	AVG	DE	EN	FR	TR
ML Baseline	12.0%	9.6%	14.6%	12.1%	8.5%
Appending	11.0%	9.3%	13.2%	10.8%	7.7%
Modulation	**10.5%**	**8.6%**	**12.5%**	**10.2%**	**7.3%**

Decoding with a RNN Language Model

As the final evaluation in this section, we used the grapheme based systems in combination with a RNN based language model to perform a greedy decoding for English. As shown in Table 7.14, the improvements in CER also propagate to WER for both 8h and 45h of training data per language.

Conclusion

We evaluated a novel approach for incorporating language codes into the network architecture. In comparison to previous methods, the error rates improved by the

Table 7.14: WER of English grapheme based systems, trained using 8h of data and 420 cells per BiLSTM layer (8h-420), or 45h and 840 cells per layer (45h-840)

Setup	Baseline	LFV add	LFV mod
8h-420	32.4%	30.6%	29.9%
45h-840	29.2%	27.7%	27.3%

modulation over the addition to the acoustic features. Language properties are not as signal related as speaker or channel properties. The modulation with language codes stimulates networks to learn features based on language properties. Depending on the language representation, the outputs of units will be emphasized or attenuated, similar to dropout training, but in a systematic manner. This allows for rapid (re-)configuration of networks to languages, as language characteristics modulate the signal flow inside the network. Different parts of the network become more or less active, depending on the language.

7.4 Optimizing the Network Architecture

After establishing modulation as the best method for performing language adaptation, we next optimized the network architecture, as well as the training strategy [MSW18a]. We originally started with Baidu's Deepspeech2 system. But it's network architecture and training strategy may not be ideal for our use case, so we optimized both aspects.

Removal of 2D CNN Layers

Baidu's original setup was developed to be trained end-to-end, including the extraction of acoustic features out of the spectrogram. For this task, the architecture uses 2 2D CNN layers to process the spectrogram and to learn filters for the extraction of features relevant for speech recognition. In our setup, these layers were no longer needed as we trained the system on ML-BNFs, which themselves already provide a well suited and robust feature representation for speech recognition. Applying 2D CNN layers to these features should not lead to improvements, as, in contrast to a spectrogram, no spacial relation between adjacent dimensions exist. Thus, removing these layers

should not decrease the performance and the architecture then features less parameters to be trained.

Combining Layer Outputs

In addition to removing the CNN layers, we also looked into possible optimizations in applying the modulation. As we were using bi-directional LSTM layers, each layer featured two outputs for each BiLSTM cell, one for each direction. These outputs could be passed along unchanged, or the outputs for each direction could be combined pairwise. There are multiple options for the combination. As default, the outputs for each direction were appended, resulting in the dimensionality of the outputs to be twice the number of BiLSTM cells. But other ways for combination are also possible. We evaluated using pairwise addition, multiplication and taking the maximum. The latter is inspired by maxout networks [GWFM+13] and maxpool layers. The new network architecture is shown in Figure 7.4.

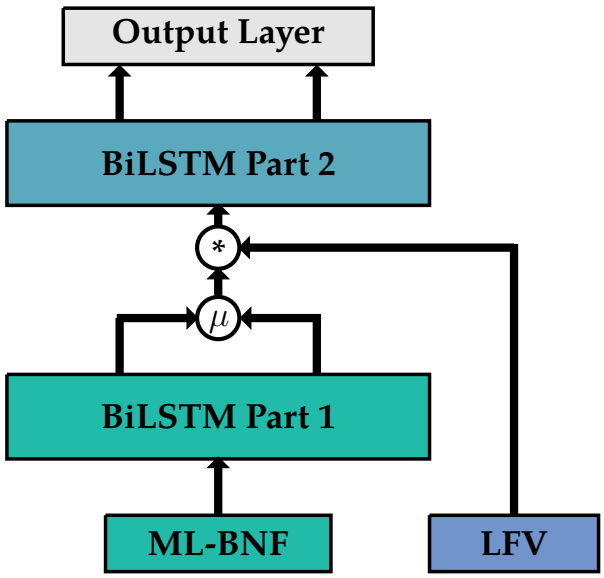

Figure 7.4: Network architecture with modulation, BiLSTM outputs for each direction are combined pairwise.

We evaluated the merging strategies using both graphemes and phonemes. Results for phoneme based systems are shown in Table 7.15. Appending the outputs for both directions can be considered to be the baseline. Pairwise multiplication lead to the

worst results. Improvements could be seen by the pairwise summation or taking the maximum. Across all languages, taking the maximum for each direction results in the lowest PER. Using graphemes as acoustic modeling units had a similar outcome, see

Table 7.15: Evaluation of merging strategies, PER on phoneme based systems

Strategy	DE	EN	FR	TR
Append	7.8%	11.2%	8.9%	6.1%
Sum	**7.7%**	**11.0%**	9.0%	6.2%
Multiply	7.9%	11.7%	9.2%	6.2%
Max	**7.7%**	**11.0%**	**8.8%**	**6.0%**

Table 7.16. Taking the maximum here resulted in the best performance for 3 languages. Multiplication again lead to the worst results.

Table 7.16: Evaluation of merging strategies, PER on grapheme based systems

Strategy	DE	EN	FR	TR
Append	**6.7%**	11.8%	**9.5%**	5.8%
Sum	7.0%	12.1%	10.1%	6.0%
Multiply	7.2%	12.8%	10.5%	6.0%
Max	**6.7%**	**11.7%**	9.8%	**5.7%**

In both conditions, improvements gained from different merging strategies are limited and in the case of the grapheme based French systems even worse. But as using an optimized strategy did slightly improve the performance in almost all conditions, we chose to take the pairwise maximum as default merging strategy.

7.5 Phonetic Pre-Training

We were building a speech recognition system based on a graphemic acoustic model. But incorporating phonetic information into the training process should improve the performance. Related to work presented in [RS17], we choose to pre-train part of the network architecture using phonemes as targets. The training of the network was divided into two steps. The network architecture is shown in Figure 7.5. First, the base network (4 BiLSTM layers, part 1 and 2) was trained on phonemic targets for a

fixed number of epochs. Next, two additional layers (part 3) were added and the whole network was trained again, this time using graphemes as targets.

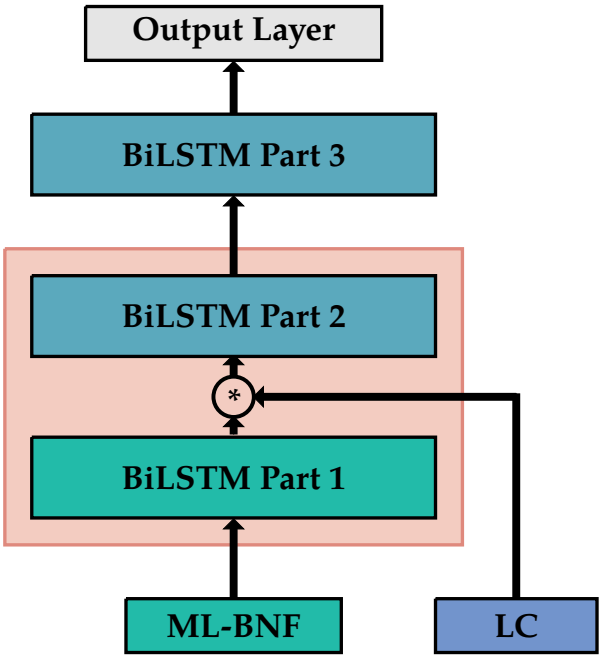

Figure 7.5: Network architecture used for pre-training. The red box indicates which layers were pre-trained. BiLSTM block 3 is added after pre-training.

We evaluated the performance of our approach by training a system in the described two stages. Table 7.17 shows the results. The CERs after applying the pre-training are lower for each language. As a contrastive experiment, we trained a network with 6 layers, but the numbers indicate that simply using more layers does not result in better performance.

Table 7.17: Evaluation of phonetic pre-training, CER on grapheme based systems

Strategy	DE	EN	FR	TR
Baseline (4L)	7.7%	11.0%	8.8%	6.0%
Baseline (6L)	9.0%	12.7%	10.3%	7.5%
Pre-training	**6.0%**	**10.0%**	**8.7%**	**5.3%**

As final evaluation in this section, we compared the WER of the resulting systems by decoding with a language model, identical to the setup used in previous experiments. As shown in Table 7.18, the improvements observed in CER are also

reflected in WER of the system.

Table 7.18: Evaluation of phonetic pre-training, WER

Strategy	EN
Baseline	26.3%
Pre-training	**25.4%**

7.6 Conclusion

We applied BiLSTM/CTC based setups to multilingual speech recognition. In traditional ASR systems, all explicitly modeled components need to be adapted to multiple languages. The all-neural architecture was chosen because it models all aspects implicitly. No explicit adaptation to languages is required, as neural adaptation methods adapt the neural network and all implicitly learned features. This enabled a better multilingual adaptation.

Using this new architecture, we first studied the effects of using additive language codes as proposed in the previous Chapter 6. While we first added the language codes to the acoustic features similar to features for speaker adaptation, language properties are not as signal related as channel or speaker properties. We therefore improved the adaptation by applying the language codes in a different manner. Inspired by Meta-PI networks, we used the language codes to gate the outputs of BiLSTM cells in hidden layers. We call this adaptation method "modulation". In comparison to simply adding language codes to the acoustic features, using modulation to integrate them as multiplicative codes deeper into the network architecture increased the recognition accuracy. Using a system with additive language codes as a baseline, the WER decreases from 28.1% to 25.4% using multiplicative codes in combination with additional optimizations.

Chapter 8

Adaptive Neural Language Codes

We have shown that choosing the right training strategy and adaptation methods can improve the recognition accuracy of multilingual systems. But reaching the monolingual baseline is still challenging when building systems with a multilingual acoustic model. In order to leverage the full potential of monolingual models in our multilingual setup, we explored how such models can be integrated into our network architecture [MSW18c]. One method is to use a network architecture based on Meta-PI [HW90, HW92]. In this approach, parts of the network are pre-trained and then combined into a single network superstructure. We adapt this method to our scenario by pre-training language dependent subnets. We also evaluate using adaptive neural language codes for modulation.

8.1 Relation to Meta-PI Networks

In the original publication, the Meta-PI combinational superstructure uses subnets which were pre-trained speaker dependent. An additional Meta-PI network was trained to provide mixture weights which were used to combine the classification outputs of the speaker dependent networks. One weight per network was used. We adapted this idea to the task of multilingual speech recognition. In our setup, we used language dependent subnets, each trained on a single language. The combined outputs of these networks were then fed into our main network. In contrast to the original approach, we discarded the output layer of these networks after training, as we aimed

for extracting language dependent features instead of the final classification results. This also allowed the use of language dependent targets without the need to map them to a global set of targets.

The language dependent BiLSTM based subnets were trained using CTC. This loss function trains the network to output spikes when it detects the occurrence of a target. With the actual point in time where this spike occurs being irrelevant, networks trained on the same task achieve the same performance, but the spikes appear at slightly different points in time, as shown in Figure 8.1. Multiple training runs lead to similar peaks, but they do not match exactly. Combining the outputs would therefore also

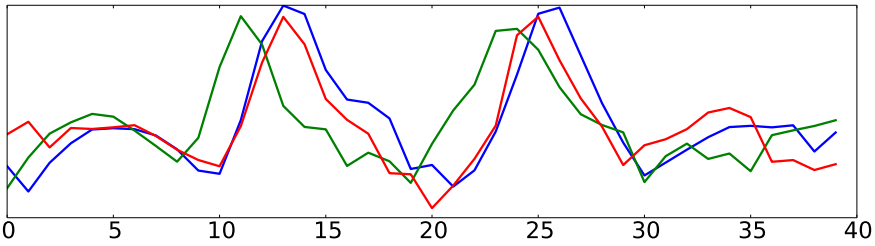

Figure 8.1: Example of output activations of the same network configuation, trained multiple times. The detection of word boundaries is shown.

have not been ideal, compared to a frame-based classification task presented in the original introduction of Meta-PI, where the networks are trained to output the same classification result at the points in time.

8.2 Neural Language Codes

In Section 6.2, we introduced LFVs and have shown that the language properties they encode carry more information than the language identity alone. We also proposed to apply them to neural networks using a method called modulation. As such, in order to modulate the outputs of a layer, the number of coefficients has to match the number of outputs. As described in Section 7.3, if the language code has a dimensionality of c, then the number of BiLSTM cells per layer are chosen to be $n \cdot c$. By stacking the language codes n times, the dimensionality of both could be matched. But this also means that n neurons are being modulated with the same coefficient. The output layer

could therefore be considered to be divided into n groups of size c, with each group being modulated with the same coefficient. This division of the layer's outputs into groups of equal size for modulation is arbitrary.

A better solution would be to break free of this fixed grouping imposed by the stacking of the language code multiple times. Also, the network for the extraction of these codes was trained for language identification. While these features lead to improvements, adapting the codes for speech recognition should improve the performance further. To address both matters, we trained a BiLSTM network to refine the extracted codes. This network was pre-trained to output stacked language codes, using both the language codes and acoustic features as input. During the training process, it will most likely learn to simply forward the language codes and output them in a stacked manner, while ignoring the acoustic features. But as this network will later be integrated into our network superstructure, it will potentially be able to adapt during the joint training and to extract language codes better suited for ASR.

While we were use this method to adapt neural networks to language properties, the architecture itself is agnostic to the kind of features used for adaptation. By using "hidden speaking modes" [OBB+96], adaptation to other properties, e.g. representing systematic variations in pronunciation, would also be possible. Human language carries multiple kinds of information. Extracting and adapting on features representing the emotional state of the speakers have proven to be useful [PW98] in ASR as well.

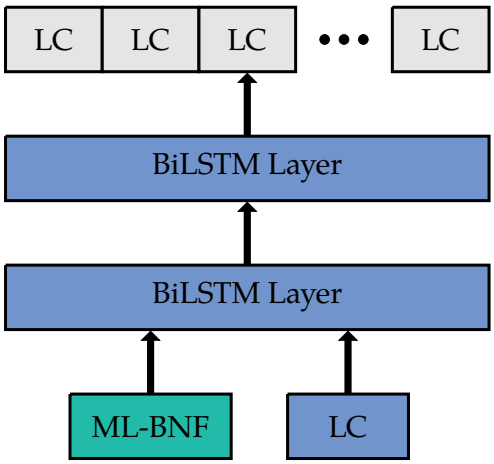

Figure 8.2: Neural Language Codes (NLC) network architecture, pre-trained to output stacked language codes (LC)

8.3 Network Architecture

We combined multiple subnets into a network superstructure. The network architecture used is shown in Figure 8.3. As input features, we used ML-BNFs (see Section 6.1.1), and LFVs (see Section 6.2). While they were each extracted using another neural network, we did not depict these networks here for clarity reasons.

We used 4 language dependent subnets. The merged outputs of these were fed into the first block of the main network, which consists of 2 BiLSTM layers, as described in Section 7.3. The outputs of the first BiLSTM block were modulated using NLCs. A feed-forward layer maps the outputs of the second BiLSTM block to the output targets.

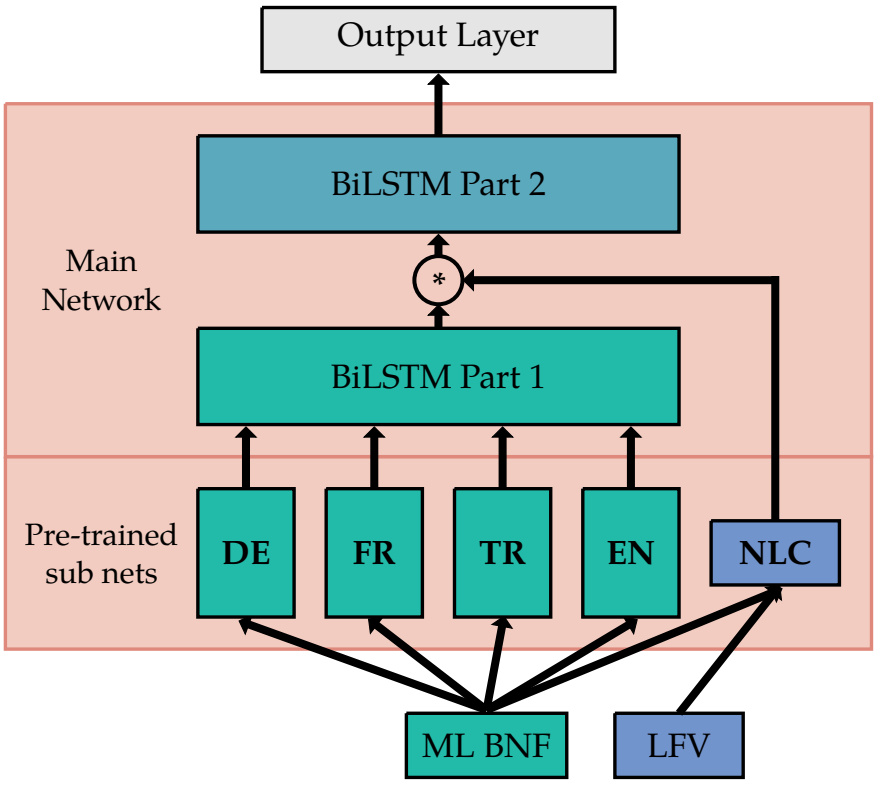

Figure 8.3: Network architecture, based on Meta-PI, using adaptive neural language codes (NLC) for network modulation.

Monolingual Sub Networks

The layers of the monolingual nets were smaller in size compared to the layers of the main network. We evaluated using networks having both only a quarter or half the amount of LSTM cells per layer in comparison to the main network. We also reduced the number of layers from 4 to 3. This network configuration was selected to limit the number of parameters to not only prevent over-fitting, but also due to memory constraints of the GPUs used for training. We trained multiple networks per language, using graphemic and phonemic targets.

Main Network

For our main network, we used the same architecture as in Section 7.4 with 4 BiLSTM layers and 420 neurons per layer. The network was divided into two blocks, with the modulation being applied between these parts. The modulation was applied as described in Section 7.4, with the outputs of the first block being merged pairwise by taking the maximum value prior to the modulation.

Network Training

In the original Meta-PI approach, all subnets were first trained separately, and then the Meta-PI net was trained to determine the mixture weights. Here, we followed a different training schedule. Initially, the parameters of the pre-trained subnets were loaded. This includes all the language specific nets, as well as the NLC network. The weights of the main network were initialized randomly. The entire network superstructure was then jointly trained on the combined graphemic targets of the 4 languages. This joint training allows the individual subnets to be optimized to the global task. Especially the language codes, which were originally trained for language classification will now be updated based on the new objective.

8.4 Experimental Setup

We used data from the Euronews corpus (see Section 3.1.2). For our experiments here, we used data from 4 languages (English, French, German, Turkish). The data was

filtered removing utterances shorter than 1s or having a transcript with more than 639 symbols[1]. The Euronews corpus featured only a very basic annotation of noises with a single noise marker representing any type of noise, e.g. music, human and non-human noises or unintelligible speech. Utterances containing only noise were therefore removed. In total, 50h of data per language remained after the cleaning process, and this dataset was split into two datasets: 45h of training and 5h of test data. We used English as our target language for the evaluation. As a contrastive experiment, we trained an English monolingual system by applying the same language independent optimizations to compare our multilingual results against a matching monolingual baseline.

As acoustic input features, we used multilingual bottleneck features (ML-BNFs). They were trained for previous experiments [MSW16a] on Euronews data, using a combination of 5 languages (French, German, Italian, Russian, Turkish) (see Section 6.2.4). The network used a combination of logMel and tonal features (FFV [LHE08] and pitch [Sch99]), extracted using a 32ms window with a 10ms frame-shift.

Acoustic Units

Phones and graphemes were used as acoustic modeling units. Similar to previous experiments, the pronunciation dictionaries were created automatically using MaryTTS (see Section 3.2). In order to create a global phone set, we mapped the phones of each language using the articulatory features embedded in MaryTTS' language definition files. In addition, we used a token representing word boundaries.

Language Dependent Subnets

We trained monolingual networks for each of the 4 languages, using both graphemic and phonetic targets. Each network featured 3 bi-directional layers with either 105 or 210 BiLSTM cells. The number of cells was chosen based on the size of the main network. We opted for using layers with half or a quarter the amount of BiLSTM cells as we intend to use these networks only for extracting intermediate features. As input features, we used ML-BNFs. A feed-forward layer was used as output layer to

[1]Internal limitation within the implementation of CUDA/warp-ctc, see: https://github.com/baidu-research/warp-ctc, accessed 2018-03-16

map the outputs of the last hidden layer to the output targets. The networks were trained using the CTC loss function, stochastic gradient descent (SGD) and Nesterov momentum [SMDH13] with a factor of 0.9. The utterances were sorted ascending by length to stabilize the training, as shorter utterances are easier to align. After the initial training, the feed-forward output layer was discarded and the outputs of the last BiLSTM layer were used as input to the main network. As we were using bi-directional LSTM layers, the output of each layer contains two coefficients per BiLSTM cell, one for each direction. Based on previous experiments [MSW18a], we opted for taking the pairwise maximum value for each direction, in the notion of a maxpool layer to reduce the dimensionality.

Neural Language Codes

For the modulation of our main network, we extracted NLCs based on a two layer, BiLSTM network with 420 BiLSTM cells per layer. As input features, we supplied ML-BNFs and LFVs. The network was trained to output stacked LFVs using the mean squared error as loss function. Given the bi-directional nature of the network, each cell outputs two coefficients. To apply a dimensionality reduction in the same manner as for the other networks, we chose to sum the outputs for each direction pairwise. Preliminary experiments indicated, that using the sum over the maximum yields better results.

Training Strategy

The whole network architecture was trained in multiple steps. First, the language dependent subnets and the NLC network were trained individually. These networks were then combined with the main network. The network superstructure was trained jointly, allowing updates to the parameters of all networks. Similar to training the subnets, SGD and Nesterov momentum [SMDH13] with a factor of 0.9 were used. As indicated, we also applied Dropout training with a rate of 0.2.

Grapheme Based RNN LM

We trained an RNN based LM using the same configuration as in Section 7.2 as a baseline configuration with a single layer and 1024 BiLSTM cells. Here, we further

refined it by optimizing the amount of BiLSTM cells used in a separate experiment.

8.5 Comparison of LFVs and NLCs

Using our setup as described in Section 7.4, we first evaluated using NLCs instead of LFVs. This setup used 4 BiLSTM layers, divided into two blocks with the modulation applied to the outputs of the first block. When applying the NLCs as language code, we allowed updates to the NLC network, training both networks jointly. As shown in Table 8.1, we got mixed results from using NLCs instead of LFVs. While performance for English improved marginally, there was no change for German and the CER of both French and Turkish increased.

Table 8.1: Comparison of adding LFVs and NLCs as language codes to our default RNN/CTC architecture, showing CERs.

Setup	DE	EN	FR	TR
No adaptation	7.58%	12.94%	11.23%	6.59%
Modulation LFV	**6.00%**	11.99%	**9.22%**	**5.46%**
Modulation NLC	**6.00%**	**11.92%**	9.52%	5.72%

8.6 Integration of Monolingual Subnets

Next, we added the pre-trained language dependent subnets, assembling the network superstructure. We first provide an overview of the pre-trained networks, and then studied their addition in different configurations to the combination of networks.

Monolingual Subnets

Table 8.2 shows the CERs of the monolingual subnets. Increasing the layer size did improve the performance of the source nets for all languages, using graphemes and phonemes. The networks trained on Graphemes for German and Turkish outperformed their phonemic counterparts. As outlined before, a possible reason for this could be that the quality of the automatically generated pronunciation dictionaries may not be optimal. Based on the experiments in Section 7.1.3, a drop in recognition

accuracy is expected as using fewer BiLSTM results prohibits the networks from learning better features. But as we do only want to use these networks to extract language dependent features, the overall recognition performance of the individual networks is not as important as the error rate of the final system. Even though networks with more BiLSTM cells result in a better recognition accuracy, the increased parameter count may decrease the overall performance of our network superstructure.

Table 8.2: CERs of monolingual subnets, using larger networks increases the performance.

Type	Size	DE	EN	FR	TR
Phone	105	9.0%	14.3%	11.7%	7.0%
Phone	210	7.2%	12.2%	8.6%	5.9%
Grapheme	105	8.3%	16.5%	13.3%	7.2%
Grapheme	210	6.4%	13.1%	9.6%	5.6%

Using Graphemic and Phonemic Pre-trained Subnets

Next, we added the pre-trained subnets to our architecture, evaluating the use of networks trained on graphemes, phonemes and different layer sizes. We first used subnets with 105 neurons per BiLSTM layer. We evaluated three conditions: a) using subnets in combination with LFVs, b) using NLCs instead of LFVs and c) using subnets trained on phonemic targets for French, German and Turkish, still using the subnet for English as target language trained on graphemes.

The results are shown in Table 8.3. As baseline, we compare against a setup without pre-trained subnets from Section 8.5. We first evaluated using subnets trained on graphemes. Using static LFVs for modulation, this setup shows a decrease in performance. Switching to adaptive NLCs improves the recognition accuracy over the baseline. Similar to results reported in Section 7.5, the use of phonetic features improves the performance. We therefore swapped the subnets trained on German, French and Turkish with their counterparts trained on phonemes. This further improved the performance. The phonetically pre-trained networks enable a better knowledge transfer between languages as they were trained on subsets of the same phoneme set. The targets for the graphemic networks are more language specific, as

each language has their own pronunciation rules. We applied dropout training to the best configuration which lowered the CER further.

Table 8.3: Comparison of Meta-PI configurations, showing CERs.

Setup	**DE**	**EN**	**FR**	**TR**
Baseline	6.00%	11.99%	9.22%	5.46%
Grapheme, LFV	6.07%	12.07%	10.36%	5.41%
Grapheme, NLC	5.47%	11.26%	8.79%	5.29%
Phoneme, NLC	5.27%	11.08%	8.32%	5.19%
Phoneme, NLC, dropout	**4.95%**	**10.39%**	**7.83%**	**4.69%**

Using Larger Subnets

As shown in Table 8.2, subnets with larger BiLSTM layers show an improved classification performance of the individual networks and languages. But as shown in Table 8.4, using larger subnets as part of the network superstructure decreases the performance. But while those networks displayed a better performance individually, they feature more parameters and this lead to a decrease in performance as the network superstructure then features more parameters and may be prone to overfitting. We applied dropout training, but while it improved the performance, the setup using the smaller subnets still outperformed the setup with the larger ones.

Table 8.4: Comparison of different subnet sizes, showing CERs.

Setup	**DE**	**EN**	**FR**	**TR**
Small subnets	5.27%	11.08%	8.32%	5.19%
Small subnets, dropout	**4.95%**	**10.39%**	**7.83%**	**4.69%**
Wide subnets	5.22%	10.80%	8.55%	5.22%
Wide subnets, dropout	5.26%	10.83%	8.47%	5.00%

8.7 RNN LM Optimization

We used a BiLSTM based language model trained on characters to compute WERs. Based on the baseline configuration with 1 layer and 1024 BiLSTM cells, we evaluated different amounts of BiLSTM cells to optimize the WER. The results on the

development set are shown in Table 8.5. Reducing the number of cells to 512 achieved the best WER. Organizing 1024 BiLSTM cells in 2 layers with 512 each did not improve the performance.

Table 8.5: RNN LM optimization, showing WERs.

BiLSTM Size	WER
1×256	22.9%
1×512	**22.6%**
1×768	23.3%
1×1024	23.3%
1×1536	23.1%
2×512	23.3%

8.8 Final Results

To obtain WERs, we applied greedy-decoding using the best RNN LM from the previous Section 8.7. We first present results using our baseline LM, including results from previous chapters for reference. Table 8.6 shows the current results and results from the phonetic pre-training (Section 7.5) for comparison. Incremental improvements are obtained by the modulation, phonetic pre-training and from using pre-trained language dependent subnets in a network superstructure. Combining the best multilingual setup with the optimized language model improves the WER further, as shown in Table 8.7.

Table 8.6: Final results on English, showing WERs.

Setup	WER
Monolingual baseline	25.3%
No adaptation	27.4%
LFV Modulation	26.3%
Phonetic pre-training	25.4%
Meta-PI + NLC	**24.2%**

Table 8.7: Final results on English, showing WERs from new LM.

Setup	WER
Monolingual baseline	24.2%
Best multilingual setup	23.5%

8.9 Conclusion

We developed an all-neural, multilingual open vocabulary system trained on graphemes. In contrast to traditional systems, such a system is not limited by a vocabulary, but at the same time is also not constrained to recognize only valid words. Multilingual acoustic models in general are inferior to monolingual ones. By using advanced neural adaptation techniques, we improved the performance of multilingual models not only to reach parity with monolingual ones, but even to surpass them. Key is a method called modulation, which stimulates the network to develop features based on language properties. Modulating neural networks with language codes allows for rapid re-configuration of the network's inner structure, depending on the properties of the input language. Moving to an approach inspired by Meta-PI networks by using pre-trained subnets improved the performance further. Pre-training parts of the network superstructure on different tasks allows the stimulation of the neural to learn feature detectors depending on language properties which have proven to be beneficial when training a network jointly on a combination of languages. In addition, adapting the neural language codes during the joint training further improved the performance.

Chapter 9

Conclusion

In this work, we addressed multilingual speech recognition. Multilingual speech recognition is a challenging problem, as each language and dialect requires a separate acoustic model and there are more than 7,000 living languages in the world. We approached this problem by building a large, language universal acoustic model. Many approaches have been proposed, but the performance of multilingual acoustic models is generally inferior to monolingually trained ones. We focused on neural network adaptation techniques to improve the performance of a language universal model.

We developed an all-neural speech recognition system with a large multilingual acoustic model based on a BiLSTM network architecture. It is based on a combination of several subnets, which were partly pre-trained on individual languages. For language adaptation, an ancillary network was trained to code useful language properties. This language code is used to gate the activity of BiLSTM cells in the main network. This method stimulates the main network to adapt the learned features based on language properties. Both networks were trained jointly, allowing the language code to be adaptive and to be optimized. The adapted multilingual acoustic model does not only reach parity with its monolingual counterpart (24.2% WER), but even surpasses its recognition performance (23.5% WER).

Acoustic Unit Discovery

In addition to working on speech recognition systems, we also attempted to build tools for supporting linguists in documenting (unwritten) languages. Out of the more than 7,000 living languages in the world, many are facing extinction. Language documentation is required to preserve the languages and the cultural heritage linked with them. But this is a very time-consuming and resource intense task. By supporting linguists with natural language processing (NLP) technology during this task, we aim at speeding up this process. One step in language documentation is the discovery of the acoustic units of a language. We developed tools which are able to derive a preliminary set of acoustic units and envision a workflow where linguists are able to iteratively refine this set to determine the acoustic inventory faster. The main focus is the extraction of articulatory features as part of our AUD pipeline. Starting with a baseline of an average frame error rate (FER) of 7.6%, we could decrease the error rate by additive codes to 6.2% and by multiplicative codes to 4.8%.

9.1 Language Selection

Building ASR systems for new languages is a challenging task because language resources are typically sparse. One common method to build ASR systems in such low-resource conditions is to use data from additional languages. The selection of these so-called source languages has an impact on the performance of the system on the target language. Language selection affects the phone coverage of the target language and seen phone contexts during training. If multiple source languages are available, data selection becomes important. We studied the effect of using different sets of languages during training given a fixed target language. A typical ASR pre-processing pipeline includes the extraction of bottleneck features (BNFs). As they are extracted using a neural network, the amount of data used for network training has an impact on the quality of the extracted features and ultimately on the system performance. If only limited resources from the target language are available, training the BNF network multilingually (ML-BNF) on a combination of languages improves the quality of the extracted features by better generalization. In a series of experiments, we studied the effects of choosing different language combinations given the target language.

We showed that selecting the right mix of languages does result in better recognition performance of the ASR system. Moreover, we also determined that adding more source languages to the mix of training data is more important than adding more data from a smaller set of languages. Training on more languages includes a wider variety of phones and other acoustic conditions (e.g. channels or noises), which enables the ML-BNF network to better generalize and the extracted features become more robust against such distortions. Starting with a baseline of 82.6% WER, using the combination of the 4 best fitting languages results in a WER of 79.7%, whereas using the 4 worst fitting languages a WER of 81.6% as achieved.

9.2 Language Adaptation by Additive Language Codes

Our main goal was to train a system with a multilingual acoustic model which is able to recognize speech from multiple languages in parallel. This model is based on neural networks. A well established method for speaker/channel adaptation is to use features representing those properties as low-dimensional vector. These so-called "i-vectors" are appended to the acoustic features and enable the network to adapt to those conditions. We used a similar approach for language adaptation by suppling low-dimensional language codes to the network. As first approach, we encoded the language identity (LID) in a naïve way using one-hot encoding. The system performance improved from 19.1% WER to 17.7% WER using a multilingual phoneme set by appending this code to the acoustic features, but the LID does not take language properties into account.

To further improve the performance, we opted to provide a language code which encodes language features. Similar to bottleneck speaker vectors (BSVs) or bottleneck features (BNFs), we trained a neural network to extract a low-dimensional representation of language properties. This network was trained for language identification and contained a bottleneck layer as second-to-last layer. After training, all layers after the bottleneck were discarded, and the output activations of the bottleneck were taken as language feature vectors (LFVs). We appended those features as language code to the acoustic input features of the acoustic model. Comparing the results of using the LID or LFVs as language code, the system improved from

17.7% WER (LID) to 16.2% WER (LFV). The performance gain by using LFVs is larger compared to the LID.

9.3 Language Adaptation by Multiplicative Language Codes

In addition to traditional speech recognition systems, we also chose a novel approach based entirely on neural networks. While traditional ASR systems feature many explicitly modeled components, all-neural setups learn to model all aspects implicitly. Language adaption of the neural network also updates all implicitly learned features. Recently, systems based on recurrent neural networks (RNNs) have become popular. RNNs are a powerful tool for sequence modeling and can be trained for speech recognition using the connectionist temporal classification (CTC) loss function. Networks trained this way model aspects implicitly, which were modeled explicitly in traditional ASR systems. Based on a single BiLSTM network, the language adaption by language codes is more effective as all the implicitly learned aspects are adapted as well. We first used additive language codes, which improved the system performance from 30.8% WER to 28.1% WER. The WERs of all-neural systems are in general higher compared to traditional systems.

But language adaptation differs from speaker/channel adaptation. The latter is more focused on the acoustic signal, because many aspects of speaker/channel variability are directly reflected in the signal. Although some language dependent aspects are signal related, e.g. language specific articulation of phones by small shifts in the tongue positions, the differences between languages are higher order concepts. Adding language codes at the signal level does improve the performance, but adding these features at a deeper level to the network enables better adaptation. Based on Meta-PI networks, we used Meta-PI connections which allow to modulate the output of neural units by multiplication with a coefficient. By modulating the outputs of BiLSTM layers with language codes, the performance improved over appending the codes to the acoustic features from 27.7%WER to 27.3% WER.

9.4 Adaptive Neural Language Codes

Using additive and multiplicative language codes improved the performance of multilingual systems. But the language features used are immutable, extracted via a network trained for language classification. To refine these codes, we trained an ancillary network which outputs Neural Language Codes (NLCs). This network is part of a network superstructure, inspired by Meta-PI. We trained multiple language dependent subnets and combined them together with the NLC network into a larger network. This larger network would then be trained for multilingual ASR, allowing updates to all pre-trained subnets. The features of the individual subnets which were pre-trained on different tasks would be updated thereby to maximize the performance for multilingual ASR. By this, we could improve the performance further and reach not only parity with monolingual acoustic models, but improve the recognition accuracy even beyond that.

9.5 Outlook

In the field of multilingual speech recognition, using and improving on adaptation techniques is very important. Unlike for speaker adaptation, where collecting data from 100+ speakers is feasible, collecting data from the same amount of languages is a very resource-intense, time-consuming process and next to impossible[1]. We here studied the proposed adaptation methods only in the context of multilingual speech recognition. Next steps in adaptation are to also consider dialects [LSS+17] and accents, or adaptation to speaking modes. Adapting to these domains poses interesting new challenges.

All-neural approaches are data-driven and very resource-intense methods. One of the big challenges will be to model features in networks explicitly by stimulate the network during training to elicit feature detectors which are beneficial for the task at hand. We have shown approaches to pre-training parts of the network to force the learning of certain features which then results in better performance. Especially for scenarios with only a limited amount of available training data, enabling networks to learn relevant features more efficient should prove beneficial. Using advanced

[1] Although some of the large companies did acquire this amount of data, but it is not publicly available.

methods to train networks and to analyze their weights is one step on the way to transition from artificial neural networks (ANNs) seen as black boxes to knowing their inner workings. Applied to the field of ASR, explicit neural modeling could be used to condition a network to model the building blocks of speech: articulatory feature detectors, or to pre-train a network on large text corpora to learn a language model. In the notion of Meta-PI, this language model could also be an external network which is then integrated into the superstructure. Pre-conditioned networks should make for better systems by using the data more efficiently.

Bibliography

[AAA+16] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016. xi, 92, 100

[Ald85] David J Aldous. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII—1983*, pages 1–198. Springer, 1985. 59

[ASAD+16] Gilles Adda, Sebastian Stüker, Martine Adda-Decker, Odette Ambouroue, Laurent Besacier, David Blachon, Hélène Bonneau-Maynard, Pierre Godard, Fatima Hamlaoui, Dmitri Idiatov, Guy-Noël Kouarata, Lori Lamel, Emmanuel-Moselly Makasso, Annie Rialland, Mark Van de Velde, François Yvon, and Sabine Zerbian. Breaking the unwritten language barrier: The Bulb project. In *Proceedings of SLTU (Spoken Language Technologies for Under-Resourced Languages)*, Yogyakarta, Indonesia, 2016. 33

[Ass99] International Phonetic Association. *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press, 1999. 7

[B+95] M Bishop, Christopher et al. Neural networks for pattern recognition. 1995. 13

[BBB+10] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David

Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation. 36

[BBdSM86] Lalit Bahl, Peter Brown, Peter V de Souza, and Robert Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11, pages 49–52. IEEE, 1986. 8

[BBKS14] Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. Automatic speech recognition for under-resourced languages: A survey. *Speech Communication*, 56:85–100, 2014. 35

[BHAL14] Steven Bird, Florian R Hanke, Oliver Adams, and Haejoong Lee. Aikuma: A mobile app for collaborative language documentation. *ACL 2014*, page 1, 2014. 33

[BL03] Alan W Black and Kevin A Lenzo. Building synthetic voices. *Language Technologies Institute, Carnegie Mellon University and Cepstral LLC*, 4:2, 2003. 62

[Bla06] Alan W Black. Clustergen: A statistical parametric synthesizer using trajectory modeling. In *Ninth International Conference on Spoken Language Processing*, 2006. 62

[BLP+07] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007. 16

[BLP+12] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 36

[BM94] Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 1994. 20

[Bri90] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990. 11

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 11

[BSMB15] Pallavi Baljekar, Sunayana Sitaram, Prasanna Kumar Muthukumar, and Alan W Black. Using articulatory features and inferred phonological segments in zero resource speech processing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015. 40, 59, 62

[CG96] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996. 8

[CJLV16] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE, 2016. 7, 27

[CKF11] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 36

[DCB+17] Ewan Dunbar, Xuan Nga Cao, Juan Benjumea, Julien Karadayi, Mathieu Bernard, Laurent Besacier, Xavier Anguera, and Emmanuel Dupoux. The zero resource speech challenge 2017. *arXiv preprint arXiv:1712.04313*, 2017. 40

[DDK+09] Najim Dehak, Reda Dehak, Patrick Kenny, Niko Brümmer, Pierre Ouellet, and Pierre Dumouchel. Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification. In *Tenth Annual conference of the international speech communication association*, 2009. 20

[DH13] Matthew S. Dryer and Martin Haspelmath, editors. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013. 89

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011. 15

[DKD+11] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011. 20

[DM80] Steven B. Davis and Paul Mermelstein. Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980. 6

[Dra17] Daniel Draper. Online neural network-based language identification. Master's thesis, Karlsruhe Institute of Technology, Germany, 2017. 84

[DSMB18] Siddharth Dalmia, Ramon Sanabria, Florian Metze, and Alan W Black. Sequence-based multi-lingual low resource speech recognition. *arXiv preprint arXiv:1802.07420*, 2018. 27, 72

[DSR+15] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, diogo149, Brian McFee, Hendrik Weideman, takacsg84, peterderivaz, Jon, instagibbs, Dr. Kashif Rasul, CongLiu, Britefury, and Jonas Degrave. Lasagne: First release., August 2015. 36

[DYDA12] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012. 20

[Dye13] Chris Dyer. Notes on adagrad. *School of Computer Science, Carnegie Mellon University*, 5000, 2013. 15

[ea94] Monika Woszczyna et al. JANUS 93: towards spontaneous speech translation. In *International Conference on Acoustics, Speech, and Signal Processing 1994*, Adelaide, Australia, 1994. 36, 93

[Elm90] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 11

[FAGD07] Jonathan G Fiscus, Jerome Ajot, John S Garofolo, and George Doddingtion. Results of the 2006 spoken term detection evaluation. In *Proc. sigir*, volume 7, pages 51–57, 2007. 9

[Fis97] Jonathan G Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 347–354. IEEE, 1997. 49

[FMSW16] Jörg Franke, Markus Müller, Sebastian Stüker, and Alex Waibel. Phoneme boundary detection using deep bidirectional LSTMs. In *Speech Communication; 12. ITG Symposium; Proceedings of*. VDE, 2016. 42

[Fou22] Joseph Fourier. *Theorie analytique de la chaleur, par M. Fourier*. Chez Firmin Didot, père et fils, 1822. 6

[GAAD+16] Pierre Godard, Gilles Adda, Martine Adda-Decker, Alexandre Allauzen, Laurent Besacier, Helene Bonneau-Maynard, Guy-Noël Kouarata, Kevin Löser, Annie Rialland, and François Yvon. Preliminary experiments on unsupervised word discovery in mboshi. In *Interspeech 2016*, 2016. 63

[GAAD+18] Pierre Godard, Gilles Adda, Martine Adda-Decker, J. Benjumea, Laurent Besacier, J. Cooper-Leavitt, Guy-Noel Kouarata, Lori Lamel, H. Maynard, Markus Müller, Annie Rialland, Sebastian Stüker, Francois Yvon, and M. Zanon-Boito. A very low resource language speech corpus for computational language documentation experiments. In *LREC 2018 (in press)*, Japan, 2018. 33, 61

[Gal99] M.J.F. Gales. Semi-tied covariance matrices for hidden markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, 1999. 66

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 15

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 17

[GBM+11] Ondřej Glembek, Lukáš Burget, Pavel Matějka, Martin Karafiát, and Patrick Kenny. Simplification and optimization of i-vector extraction. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4516–4519. IEEE, 2011. 20

[GFGS06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006. 7, 13, 14, 21, 22, 91, 98

[GKV14] Frantisek Grézl, Martin Karafiát, and Karel Vesely. Adaptation of multilingual stacked bottle-neck neural network structure for new language. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7654–7658. IEEE, 2014. 26

[GMMW13] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel. Extracting deep bottleneck features using stacked auto-encoders. In *Proceedings of the ICASSP*, Vancouver, Canada, May 2013. 16

[GMNM17] Abhinav Gupta, Yajie Miao, Leonardo Neves, and Florian Metze. Visual features for context-aware speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5020–5024. IEEE, 2017. 21

[Gra13] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. ix, 11, 12

[Gre14] Roberto Gretter. Euronews: A multilingual benchmark for ASR and LID. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014. 32

[GSR13] Arnab Ghoshal, Pawel Swietojanski, and Steve Renals. Multilingual training of deep-neural networks. In *Proceedings of the ICASSP*, Vancouver, Canada, 2013. 26

[GSS03] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*, 3:115–143, 2003. 11

[GW96] Mark JF Gales and Philip C Woodland. Mean and variance adaptation within the mllr framework. *Computer Speech & Language*, 10(4):249–264, 1996. 8

[GWFM+13] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013. 105

[HAH01] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken language processing : a guide to theory, algorithm, and system development*. Voice/speech recognition technology. Prentice Hall, Upper Saddle River, NJ, 2001. Includes bibliographical references and index. 5, 9, 23

[Har75] John A Hartigan. Clustering algorithms. 1975. 59

[HBFS01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001. 11

[HHL89] XD Huang, Hsiao-Wuen Hon, and Kai-Fu Lee. Large-vocabulary speaker-independent continuous speech recognition with semi-continuous hidden markov models. In *Proceedings of the*

workshop on Speech and Natural Language, pages 276–279. Association for Computational Linguistics, 1989. 8

[HIW90] John B Hampshire II and Alex Waibel. Connectionist architectures for multi-speaker phoneme recognition. In *Advances in neural information processing systems*, pages 203–210, 1990. 28

[HMM+18] Fatima Hamlaoui, Emmanuel-Moselly Makasso, Markus Müller, Jonas Engelmann, Gilles Adda, Alex Waibel, and Sebastian Stüker. BULBasaa: A bilingual Bàsàá-French speech corpus for the evaluation of language documentation tools. In *LREC 2018 (in press)*, Japan, 2018. 34, 42

[Hoc98] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998. 11

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006. 16

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 11

[HS15] Hengguan Huang and Khe Chai Sim. An investigation of augmenting speaker representations to improve speaker normalisation for DNN-based speech recognition. In *ICASSP*, pages 4610–4613. IEEE, 2015. 78

[HSK+12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 17, 99

[HSN16a] Michael Heck, Sakriani Sakti, and Satoshi Nakamura. Iterative training of a DPGMM-HMM acoustic unit recognizer in a zero resource scenario. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 57–63. IEEE, 2016. 59

[HSN16b] Michael Heck, Sakriani Sakti, and Satoshi Nakamura. Supervised learning of acoustic models in a zero resource setting to improve DPGMM clustering. In *INTERSPEECH*, pages 1310–1314, 2016. 59

[HSN16c] Michael Heck, Sakriani Sakti, and Satoshi Nakamura. Unsupervised linear discriminant analysis for supporting DPGMM clustering in the zero resource scenario. *Procedia Computer Science*, 81:73–79, 2016. 59

[HVS+13] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean. Multilingual acoustic models using distributed deep neural networks. In *Proceedings of the ICASSP*, Vancouver, Canada, May 2013. 26

[HW90] John B Hampshire and Alex H Waibel. The Meta-Pi network: Connectionist rapid adaptation for high-performance multi-speaker phoneme recognition. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 165–168. IEEE, 1990. 27, 99, 109

[HW92] John B Hampshire and Alex Waibel. The Meta-Pi network: Building distributed knowledge representations for robust multisource pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):751–769, 1992. ix, 27, 28, 57, 99, 109

[JE05] R. G. Gordon Jr. and B. F. Grimes (Eds.). *Ethnologue: Languages of the World*. SIL International, Dallas, Texas, USA, 2005. 34

[JJNH91] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991. 29

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 15

[KFN98] T. Kaukoranta, P. Fränti, and O. Nevalainen. Iterative split-and-merge algorithm for VQ codebook generation. *Optical Engineering*, 37(10):2726–2732, 1998. 66

[Kil15] Kevin Kilgour. *Modularity and neural integration in large-vocabulary continuous speech recognition*. Karlsruhe, [2015]. 19

[KM14] Timothy Kempton and Roger K Moore. Discovering the phoneme inventory of an unwritten language: A machine-assisted approach. *Speech Communication*, 56:152–166, 2014. 39

[KN95] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE, 1995. 8

[KS17] Suyoun Kim and Michael L Seltzer. Towards language-universal end-to-end speech recognition. *arXiv preprint arXiv:1711.02207*, 2017. 26

[LHE08] Kornel Laskowski, Mattias Heldner, and Jens Edlund. The Fundamental Frequency Variation Spectrum. In *Proceedings of the 21st Swedish Phonetics Conference (Fonetik 2008)*, pages 29–32, Gothenburg, Sweden, June 2008. 6, 48, 93, 114

[LSS+17] Bo Li, Tara N Sainath, Khe Chai Sim, Michiel Bacchiani, Eugene Weinstein, Patrick Nguyen, Zhifeng Chen, Yonghui Wu, and Kanishka Rao. Multi-dialect speech recognition with a single sequence-to-sequence model. *arXiv preprint arXiv:1712.01541*, 2017. 125

[MB90] Nelson Morgan and Hervé Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in neural information processing systems*, pages 630–637, 1990. 18

[MB14] Prasanna Kumar Muthukumar and Alan W Black. Automatic discovery of a phonetic inventory for unwritten languages for statistical speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 2594–2598. IEEE, 2014. 40, 59

[MBS00] Lidia Mangu, Eric Brill, and Andreas Stolcke. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400, 2000. 49

[Met05] Florian Metze. *Articulatory Features for Conversational Speech Recognition*. PhD thesis, Karlsruhe, Univ., Diss., 2005. 44

[MFSW17a] Markus Müller, Jörg Franke, Sebastian Stüker, and Alex Waibel. Improving phoneme set discovery for documenting unwritten languages. *Elektronische Sprachsignalverarbeitung (ESSV) 2017*, 2017. 51

[MFSW17b] Markus Müller, Jörg Franke, Sebastian Stüker, and Alex Waibel. Towards phoneme inventory discovery for documentation of unwritten languages. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017. 57

[MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 82

[MKC+11] L. Mangu, Hong-Kwang Kuo, S. Chu, B. Kingsbury, G. Saon, Hagen Soltau, and F. Biadsy. The ibm 2011 gale arabic speech transcription system. In *Proceedings of the ASRU*, Waikoloa, HI, USA, December 2011. 19

[MM13] Yajie Miao and Florian Metze. Improving low-resource CD-DNN-HMM using dropout and multilingual DNN training. In *Interspeech*, volume 13, pages 2237–2241, 2013. 26

[MM15] Yajie Miao and Florian Metze. Distance-aware DNNs for robust speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015. 21

[MR97] Manohar N Murthi and Bhaskar D Rao. Minimum variance distortionless response (mvdr) modeling of voiced speech. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 3, pages 1687–1690. IEEE, 1997. 6

[MRTD17] Paul Michel, Okko Räsänen, Roland Thiolliere, and Emmanuel Dupoux. Blind phoneme segmentation with temporal prediction errors. In *Proceedings of ACL 2017, Student Research Workshop*, pages 62–68, 2017. 40

[MS69] Marvin Minsky and Papert Seymour. Perceptrons. 1969. 10

[MSN+14] Vikramjit Mitra, Ganesh Sivaraman, Hosung Nam, Carol Espy-Wilson, and Elliot Saltzman. Articulatory features from deep neural networks and their role in speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3017–3021. IEEE, 2014. 23

[MSS+14] Markus Müller, Sebastian Stüker, Zaid Sheik, Florian Metze, and Alex Waibel. Multilingual deep bottle neck features - a study on language selection and training techniques. *Proceedings of the 11th International Workshop on Spoken Language Translation (IWSLT)*, 2014. 27, 65

[MSW16a] Markus Müller, Sebastian Stüker, and Alex Waibel. Language adaptive DNNs for improved low resource speech recognition. In *Interspeech*, 2016. 41, 42, 78, 79, 86, 114

[MSW16b] Markus Müller, Sebastian Stüker, and Alex Waibel. Language feature vectors for resource constraint speech recognition. In *Speech Communication; 12. ITG Symposium; Proceedings of*. VDE, 2016. 79

[MSW16c] Markus Müller, Sebastian Stüker, and Alex Waibel. Towards improving low-resource speech recognition using articulatory and language features. In *Proceedings of the 11th International Workshop on Spoken Language Translation (IWSLT)*, Seattle, U.S.A., 2016. 45

[MSW17a] Markus Müller, Sebastian Stüker, and Alex Waibel. DBLSTM based multilingual articulatory feature extraction for language documentation. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 417–423. IEEE, 2017. 57

[MSW17b] Markus Müller, Sebastian Stüker, and Alex Waibel. Language adaptive multilingual CTC speech recognition. In *International Conference on Speech and Computer*, pages 473–482. Springer, 2017. 26, 92

[MSW17c] Markus Müller, Sebastian Stüker, and Alex Waibel. Phonemic and graphemic multilingual ctc based speech recognition. *arXiv preprint arXiv:1711.04564*, 2017. 26, 96

[MSW18a] Markus Müller, Sebastian Stüker, and Alex Waibel. Enhancing multilingual graphemic rnn based asr systems using phone information. *Elektronische Sprachsignalverarbeitung (ESSV) 2018*, 2018. 104, 115

[MSW18b] Markus Müller, Sebastian Stüker, and Alex Waibel. Multilingual adaptation of rnn based asr systems. In *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018. 26, 99

[MSW18c] Markus Müller, Sebastian Stüker, and Alex Waibel. Neural language codes for multilingual acoustic models. In *Interspeech*, 2018. submitted to. 26, 109

[MTSC01] Mikiko Mashimo, Tomoki Toda, Kiyohiro Shikano, and Nick Campbell. Evaluation of cross-language voice conversion based on gmm and straight. 2001. 62

[MW02] Florian Metze and Alex Waibel. A flexible stream architecture for asr using articulatory features. In *INTERSPEECH*, 2002. 23, 44

[MW15] Markus Müller and Alex Waibel. Using language adaptive deep neural networks for improved multilingual speech recognition. *IWSLT*, 2015. 74

[MZM14a] Yajie Miao, Hao Zhang, and Florian Metze. Distributed learning of multilingual dnn feature extractors using GPUs. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014. 26

[MZM14b] Yajie Miao, Hao Zhang, and Florian Metze. Towards speaker adaptive training of deep neural network acoustic models. 2014. 21

[NDG+17] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017. 37

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983. 15

[Nes13] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013. 15

[NGM+14] Quoc Bao Nguyen, Jonas Gehring, Markus Müller, Sebastian Stüker, and Alex Waibel. Multilingual shifting deep bottleneck features for low-resource ASR. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5607–5611. IEEE, 2014. 19

[NSM15] Tasha Nagamine, Michael L Seltzer, and Nima Mesgarani. Exploring how deep neural networks form phonemic categories. In *INTERSPEECH*, pages 1912–1916, 2015. 23

[OBB+96] M Ostendorf, B Byrne, M Bacchiani, M Finke, A Gunawardana, K Ross, S Roweis, E Shriberg, D Talkin, A Waibel, et al. Modeling systematic variations in pronunciation via a language-dependent hidden speaking mode. In *1996 LVCSR Summer Research Workshop Technical Reports*, volume 8, 1996. 111

[OBČ16] Lucas Ondel, Lukáš Burget, and Jan Černockỳ. Variational inference for acoustic unit discovery. *Procedia Computer Science*, 81:80–86, 2016. 40

[OGB+18] Lucas Ondel, Pierre Godard, Laurent Besacier, Elin Larsen, Mark Hasegawa-Johnson, Odette Scharenborg, Emmanuel Dupoux, Lukas Burget, François Yvon, and Sanjeev Khudanpur. Bayesian models for unit discovery on a very low resource language. *arXiv preprint arXiv:1802.06053*, 2018. 63

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017. 36

[PKK+08] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah. Boosted

mmi for model and feature-space discriminative training. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4057–4060. IEEE, 2008. 8

[PW98] Thomas S Polzin and Alex Waibel. Detecting emotions in speech. In *Proceedings of the CMC*, volume 16. Citeseer, 1998. 111

[QSM08] Yu Qiao, Naoya Shimomura, and Nobuaki Minematsu. Unsupervised optimal phoneme segmentation: objectives, algorithm and comparisons. In *Acoustics, Speech and Signal Processing (ICASSP), 2008 IEEE International Conference on*, pages 3989–3992. IEEE, 2008. 41

[Rab89] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 7, 14

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 10

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. 14

[Ros57] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957. 10

[RS17] Kanishka Rao and Haşim Sak. Multi-accent speech recognition with hierarchical grapheme based models. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4815–4819. IEEE, 2017. 106

[SAAD+16] Sebastian Stüker, Gilles Adda, Martine Adda-Decker, Odette Ambouroue, Laurent Besacier, David Blachon, Hélène Bonneau-Maynard, Pierre Godard, Fatima Hamlaoui, Dmitri Idiatov, Guy-Noël Kouarata, Lori Lamel, Emmanuel-Moselly Makasso, Markus Müller, Annie Rialland, Mark Van de Velde, François Yvon, and Sabine Zerbian. Innovative technologies for under-resourced language

documentation: The Bulb project. In *Proceedings of CCURL (Collaboration and Computing for Under-Resourced Languages : toward an Alliance for Digital Language Diversity)*, Portorož Slovenia, 2016. 33

[Sch99] Kjell Schubert. Grundfrequenzverfolgung und deren anwendung in der spracherkennung. Master's thesis, Universität Karlsruhe (TH), Germany, 1999. In German. 6, 48, 93, 114

[SF17] Gary F. Simons and Charles D. Fennig, editors. *Ethnologue: Languages of the World, Twentieth edition.* SIL International, Dallas, Texas, 2017. Online version: `http://www.ethnologue.com`. 34

[SGR12] Pawel Swietojanski, Arnab Ghoshal, and Steve Renals. Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR. In *SLT*, pages 246–251. IEEE, IEEE, 2012. 26

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 99

[SKR12] T.N. Sainath, B. Kingsbury, and B. Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *Proceedings of the ICASSP*, Kyoto, Japan, March 2012. 19

[SLCY11] Frank Seide, Gang Li, Xie Chen, and Dong Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 24–29. IEEE, 2011. 20

[SLF+08] Stefano Scanzio, Pietro Laface, Luciano Fissore, Roberto Gemello, and Franco Mana. On the use of a multilingual neural network front-end. In *Proceedings of the Interspeech*, pages 2711–2714, 2008. 26

[SLS16] Hagen Soltau, Hank Liao, and Hasim Sak. Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. *arXiv preprint arXiv:1610.09975*, 2016. 7

[SLY11] Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*, pages 437–440, 2011. 20

[SMDH13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1139–1147, 2013. 15, 115

[SMFW01] Hagen Soltau, Florian Metze, Christian Fugen, and Alex Waibel. A one-pass decoder based on polymorphic linguistic context assignment. In *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*, pages 214–217. IEEE, 2001. 36, 93

[SMNW14] Sebastian Stüker, Markus Müller, Quoc Bao Nguyen, and Alex Waibel. Training time reduction and performance improvements from multilingual techniques on the babel ASR task. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014. 66

[SMSW03] Sebastian Stüker, Florian Metze, Tanja Schultz, and Alex Waibel. Integrating multilingual articulatory features into speech recognition. In *Eighth European Conference on Speech Communication and Technology*, 2003. 23

[SP97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997. 11

[SSMW03] Sebastian Stüker, Tanja Schultz, Florian Metze, and Alex Waibel. Multilingual articulatory features. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2003. 23

[SSNP13] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *ASRU*, pages 55–59. IEEE, 2013. 20, 73

[ST03] M. Schröder and J. Trouvain. The german text-to-speech synthesis system mary: A tool for research, development and teaching. *International Journal of Speech Technology*, 6(4):365–377, 2003. 34

[Stü08a] Sebastian Stüker. Modified polyphone decision tree specialization for porting multilingual grapheme based asr systems to new languages. In *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 4249–4252, Las Vegas, NV, USA, April 2008. IEEE. 25

[Stü08b] Sebastian Stüker. Multilingual acoustic features for porting speech recognition systems to new languages. *ESSV, Frankfurt, Germany*, 2008. 25

[Stü09] Sebastian Stüker. *Acoustic modelling for under-resourced languages*. PhD thesis, Karlsruhe, Univ., Diss., 2009. 25, 63

[SVN37] Stanley Smith Stevens, John Volkmann, and Edwin B Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937. 6

[SW97] Tanja Schultz and Alex Waibel. Fast bootstrapping of lvcsr systems with multilingual phoneme sets. In *Eurospeech*, 1997. 25

[SW98a] Tanja Schultz and Alex Waibel. Language independent and language adaptive large vocabulary speech recognition. In *Fifth International Conference on Spoken Language Processing*, 1998. 25

[SW98b] Tanja Schultz and Alex Waibel. Multilingual and crosslingual speech recognition. In *Proc. DARPA Workshop on Broadcast News Transcription and Understanding*, pages 259–262. Citeseer, 1998. 25, 91

[SW00] Tanja Schultz and Alex Waibel. Polyphone decision tree specialization for language adaptation. In *Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1707–1710. IEEE, 2000. 25, 91

[SW01] Tanja Schultz and Alex Waibel. Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, 35(1):31–51, 2001. 25

[SWE09] Odette Scharenborg, Vincent Wan, and Mirjam Ernestus. Unsupervised speech segmentation: An analysis of the hypothesized phone boundaries. *Acoustical Society of America, Journal of*, 127(2):1084–1095, 2009. 40, 41

[SWH+18] H Seki, S Watanabe, T Hori, J Le Roux, and JR Hershey. An end-to-end language-tracking speech recognizer for mixed-language speech. 2018. 27

[TBC98] Paul Taylor, Alan W Black, and Richard Caley. The architecture of the festival speech synthesis system. 1998. 62

[TSCH13] Samuel Thomas, Michael L Seltzer, Kenneth Church, and Hynek Hermansky. Deep neural network features and semi-supervised training for low resource speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6704–6708. IEEE, 2013. 26

[TSG15] Shawn Tan, Khe Chai Sim, and Mark Gales. Improving the interpretability of deep neural networks with stimulated learning. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 617–623. IEEE, 2015. 100

[TSW+17] Shubham Toshniwal, Tara N Sainath, Ron J Weiss, Bo Li, Pedro Moreno, Eugene Weinstein, and Kanishka Rao. Multilingual speech recognition with a single end-to-end model. *arXiv preprint arXiv:1711.01694*, 2017. 27

[VAJD16] Maarten Versteegh, Xavier Anguera, Aren Jansen, and Emmanuel Dupoux. The zero resource speech challenge 2015: Proposed approaches and results. *Procedia Computer Science*, 81:67–72, 2016. 40

[VBMS12] Ngoc Thang Vu, Wojtek Breiter, Florian Metze, and Tanja Schultz. Initialization schemes for multilayer perceptron training and their

impact on ASR performance using multilingual data. In *Proceedings of the INTERSPEECH*, Portland, Oregon, September 2012. 26

[VEB10] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010. 60

[VKG+12] Karel Vesely, Martin Karafiat, Frantisek Grezl, Milos Janda, and Ekaterina Egorova. The language-independent bottleneck features. In *Proceedings of the Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 336–341. IEEE, 2012. 26

[VLBM08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008. 16

[VMH+16] Marco Vetter, Markus Müller, Fatima Hamlaoui, Graham Neubig, Satoshi Nakamura, Sebastian Stüker, and Alex Waibel. Unsupervised phoneme segmentation of previously unseen languages. In *Proceedings of the Interspeech*, 2016. 40

[VTS+15] Maarten Versteegh, Roland Thiolliere, Thomas Schatz, Xuan Nga Cao, Xavier Anguera, Aren Jansen, and Emmanuel Dupoux. The zero resource speech challenge 2015. In *Proceedings of Interspeech*, 2015. 40

[W+97] John C Wells et al. Sampa computer readable phonetic alphabet. *Handbook of standards and resources for spoken language systems*, 4, 1997. 7

[WCL17] Yu-Hsuan Wang, Cheng-Tao Chung, and Hung-Yi Lee. Gate activation signal analysis for gated recurrent neural networks and its correlation with phoneme boundaries. *Proc. Interspeech 2017*, pages 3822–3826, 2017. 40

[Wel95] John C Wells. Computer-coding the ipa: a proposed extension of sampa. *Revised draft*, 4(28):1995, 1995. 7

[Wer90] Paul J. Werbos. Backpropagation through time: what it does and how to do it. *Proc. IEEE*, 78(10):1550–1560, 1990. 11

[WGT+00] Alex Waibel, Petra Geutner, L Mayfield Tomokiyo, Tanja Schultz, and Monika Woszczyna. Multilinguality in speech and spoken language systems. *Proceedings of the IEEE*, 88(8):1297–1313, 2000. 25

[WHH+87a] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. In *Meeting of the Institute of Electrical, Information and Communication Engineers (IEICE)*, pages SP87–100. IEICE, 1987. 11, 19

[WHH+87b] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. Technical report, ATR, 1987. ATR Technical Report, TR-I-0006. 11, 19

[WHH+88] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and K Lang. Phoneme recognition: neural networks vs. hidden markov models vs. hidden markov models. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 107–110. IEEE, 1988. 11, 19

[WHH17] Shinji Watanabe, Takaaki Hori, and John R Hershey. Language independent end-to-end architecture for joint language identification and speech recognition. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 265–271. IEEE, 2017. 27

[WKGS16] Chunyang Wu, Penny Karanasou, Mark JF Gales, and Khe Chai Sim. Stimulated deep neural network for speech recognition. Technical report, University of Cambridge Cambridge, 2016. 100

[yLG12] Chia ying Lee and James Glass. A nonparametric bayesian approach to acoustic model discovery. In *50th Annual Meeting of the Association for Computational Linguistics*, pages 40–49. Association for Computational Linguistics, 2012. 40, 41

[YS11] Dong Yu and Michael L. Seltzer. Improved bottleneck features using pretrained deep neural networks. In *INTERSPEECH*, pages 237–240, 2011. 16, 19

[YW93] Steve J Young and Phil C Woodland. The use of state tying in continuous speech recognition. In *Third European Conference on Speech Communication and Technology*, 1993. 8

[ZDV+16] Albert Zeyer, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, and Hermann Ney. A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition. *arXiv preprint arXiv:1606.06871*, 2016. 21

[Zei12] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 15

[ZSN16] Albert Zeyer, Ralf Schlüter, and Hermann Ney. Towards online-recognition with deep bidirectional lstm acoustic models. In *Proceedings of the Interspeech*, San Francisco, CA, USA, 2016. 21, 51

Appendices

Appendix A

Appendix A

A.1 Acknowledgments

Data from IARPA BABEL Program

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD / ARL) contract number W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government. This effort uses the following IARPA Babel Program language collection releases:

- Assamese: IARPA-babel102b-v0.4
- Bengali: IARPA-babel103b-v0.3
- Cantonese: IARPA-babel101b-v0.4c
- Haitian Creole: IARPA-babel201b-v0.2b
- Lao: IARPA-babel203b-v3.1a
- Pashto: IARPA-babel104b-v0.4bY

- Tagalog: IARPA-babel106-v0.2f
- Tamil: IARPA-babel204b-v1.1b
- Turkish: IARPA-babel105b-v0.4
- Vietnamese: IARPA-babel107b-v0.7
- Zulu: IARPA-babel206b-v0.1d

BULB

This work was in part realized in the framework of the ANR-DFG project BULB (STU 593/2-1 and ANR-14-CE35-002) and also supported by the French Investissements d'Avenir - Labex EFL program (ANR-10-LABX-0083).

JSALT Workshop 2017

This work used in part the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation award number ACI-1053575. Specifically, it used the Blacklight system, which is supported by NSF award number ACI-1041726, at the Pittsburgh Supercomputing Center (PSC). Part of the work reported here was done at the Jelinek Speech and Language Technology Workshop JSALT 2017, in Pittsburgh, and was supported by JHU and CMU via grants from Amazon, Apple, Google and Microsoft.