Karlsruher Institut für Technologie

Master Thesis

# Multilingual Sequence-To-Sequence Speech Recognition

Helen Gremmelmaier

21.05.2021

Reviewer: PD Dr. Gudrun Thäter
(Institute for Applied and Numerical Mathematics)
Second Reviewer: Prof. Dr. Alexander Waibel
(Institute for Anthropomatics and Robotics)
Advisor: Juan Hussain, M.Sc.
(Institute for Anthropomatics and Robotics)
Second advisor: Dr. Sebastian Stüker
(Institute for Anthropomatics and Robotics)

Department of Mathematics

Karlsruhe Institute of Technology

# Abstract

There are a multitude of languages and dialects in the world. To develop a speech recognition system for all of them is an expensive and time-consuming task. Multilingual systems can be advantageous here, but don't work as good as monolingual ones currently. In this thesis we investigate sequence-to-sequence systems as the Transformer and an encoder-decoder architecture consisting of Long Short-Term Memory (LSTM) layers. By inserting the language identity those systems are supposed to learn language dependent features.
Therefore, we compare different methods like a gating algorithm, based on modulation, and language specific Multi-Head Attention layers.
Especially gating worsens the recognition rate of the Transformer and also language specific Multi-Head Attention layers did not lead to any improvements. The Transformer without language adaption, only trained with mixed language data, outperforms both methods and provides results that are comparable to the monolingual Transformer.
In the LSTM based encoder-decoder architecture gating and language specific Multi-Head Attention performed well. In particular the combination of both methods works best and outperforms monolingual and multilingual systems without language adaption.

# Zusammenfassung

Die Vielzahl an existierenden Sprachen und Dialekten macht es schwierig, für alle unterschiedliche Spracherkenunngssysteme zu entwickeln. Multilinguale Spracherkenner können hier helfen, funktionieren aber oft nicht so gut wie monolinguale Systeme.
Wir untersuchen Sequence-To-Sequence Modelle, wie den Transformer und eine Encoder-Decoder Architektur, die aus Long Short-Term Memory (LSTM) Layern besteht. Durch Angabe der Sprache sollen die Netzarchitekturen sprachabhängige Eigenschaften lernen. Dabei vergleichen wir verschiedene Methoden, wie sprachspezifische Multi-Head Attention Layer und einen Gating-Algorithmus, der auf Modulation basiert.
Insbesondere der Gating-Algorithmus hat negative Auswirkungen auf die Erkennungsrate des Transformers und auch mit Sprachspezifischen Attention Layern konnten wir keine Verbesserung erzielen. Der Transformer, nur trainiert mit gemischten Sprachdaten, liefert hierbei in der Regel bessere Ergebnisse, die zudem vergleichbar sind mit monolingualen Systemen.
Beim LSTM zeigte sich, dass insbesondere die Kombination aus Gating und sprachspezifischem Multi-Head Attention zu einer geringeren Wortfehlerrate, verglichen mit den monolingualen und dem multilingualen System ohne Sprachinformation, führt.

# Acknowledgement

First of all I would like to thank my advisor Juan Hussain, who familiarized me with the topic and therefore spent a lot of time. Furthermore I want to tank him for the idea of gating of the output embedding in Chapter 4.3.4.
Moreover I would like to thank PD Dr. Gudrun Thäter, Dr. Sebastian Stüker and Prof. Dr. Alexander Waibel, who also provided advise and the opportunity to write my thesis at the Interactive System Labs.
I would like to thank Matthias Schmitt, who recorded Denglish data, which I was allowed to use for experiments.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Automatic Speech Recognition (ASR) deals with the problem of translating human spoken speech into text automatically by machines. In everyday life ASR already has a lot of use cases, we control devices free-handed, e.g. mobile devices or program the navigation system in cars. Lawyers or doctors can use ASR systems for dictations. But even if such systems are adapted to a specific domain and for example only used for doctor's letters the recognition is not perfect and wrong predictions are sometimes made. Strong accents or dialects may be a problem. Furthermore the system should to be able t differentiate between words with similar pronunciation of words, that have different transcriptions.
Usually such systems are adapted to a specific task, for example one language. Considering the high number of existing languages and dialects in the world this is a very expensive and time-consuming challenge.
Traditional systems model dependencies of the language, its acoustics and pronunciation explicitly and separately, nowadays using neural networks this is not necessary anymore. The network gets audio and corresponding target text as input and learns dependencies by itself.
To train an ASR system a lot of data is necessary. The network processes this data and makes a prediction. By comparing this prediction with the correct outcome and adjusting it's weights by minimizing the error, the network learns and improves the prediction. For some languages like German or English, which are both spoken by many people, a lot of data is available. But for languages with fewer speaker this might be a problem, too.

Multilingual ASR is difficult because usually the acoustics and also the modeling of language vary for different languages. Nevertheless these are not complete various and independent tasks. The idea behind multilingual ASR is to use such similarities and benefit from them like people do speaking more than one language. This would remove the need of different systems for each language and especially low-resource languages could profit. Maybe it is possible to do the recognition for languages that have not been seen during training some day.

This thesis looks at sequence-to-sequence models, a special kind of neural networks, which transform one sequence to another. Speech recognition can be considered as such a problem since the audio input and its transcription are both sequences. Two architectures are considered, an attention based encoder-decoder architecture consisting of Long Short-Term Memory (LSTM) layers and the Transformer, also an encoder-decoder model.
We investigate several methods for language adaption in multilingual ASR to study their effect compared to our baseline without language information.

For the experiments with the Transformer we use two languages English and German. In the case of the LSTM based system we also experiment with French as third language.

Most of the evaluated methods are based on the insertion of the language identity into different stages of the architecture. We especially investigate a gating mechanism [6], that was originally proposed for an LSTM based architecture. As in the underlying paper it performs very well in our LSTM encoder-decoder architecture. Comparison of its application only in the decoder or in encoder and decoder shows that the model benefits most when being applied after each layer of the encoder and decoder. In addition we experimented with gating of the output embedding in the decoder.
For the Transformer we couldn't achieve any improvements, instead gating yields even worse results the more gating layers were inserted into the architecture.
Another technique applied to both Transformer and LSTM was language adaption in Multi-Head Attention layers [7]. The first approach here was adding a language specific bias to key, query and value. In addition a combination of language specific and shared heads was evaluated. The results were comparable to the baseline, trained with mixed language data only.
To further improve results, gating and language adaption in the Multi-Head Attention of the LSTM architecture were combined. This leads to a model with language adaption in almost every part of the architecture. In total we could achieve an average of 9.9% reduced word error rate compared to the based multilingual systems in a bilingual setup. When adding a third language the word error rate decreased even more by 10.9%.
Furthermore we applied modulation to the hidden states of LSTM layers to force the network to learn language properties, which also outperforms the recognition rate of the baseline.

The thesis begins with a chapter about the basics of ASR and neural networks. The LSTM and the Transformer are described in Chapter 2.7.
The subsequent Chapter 3 gives an overview of related work and investigated methods of other researchers in multilingual ASR. Especially the gating mechanism and language dependent attention later applied to the Transformer and the LSTM based encoder-decoder model are described in detail.
Chapter 4 presents all experiments and their results
Finally Chapter 5 proposes ideas of how to further investigate this topic to improve multilingual ASR.

# 2 Basics

## 2.1 Basics of Automatic Speech Recognition

This chapter briefly explains the basic structure of a statistical ASR system. The description is based on [8] and further details can be found there.

Figure 1 shows a diagram of a traditional statistical ASR system.

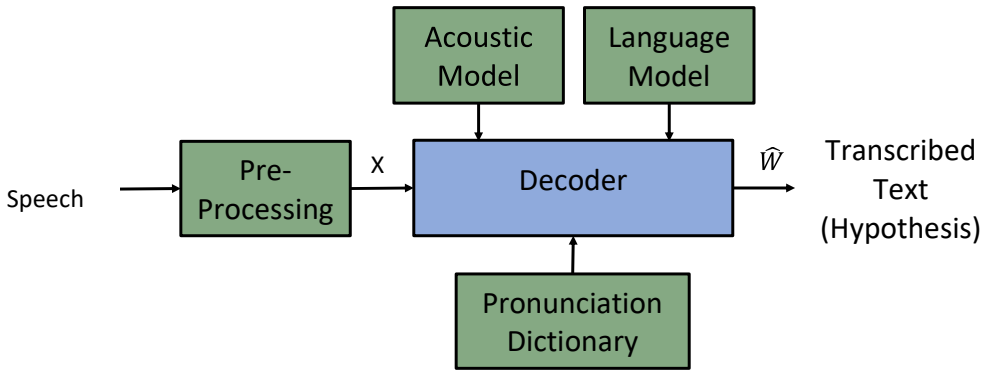


Figure 1: ASR system

The first step is the pre-processing of the recorded audio, which is explained in Chapter 2.2.
The actual recognition, i.e. the transcription of the spoken speech is done by the decoder. The decoder receives the pre-processed speech as input and computes the most probable word sequence and outputs it as transcribed text. This output is called the hypothesis.
The other components shown in the diagram are the acoustic model, language model and a pronunciation dictionary. The language model is independent from the acoustics of spoken speech and models the probability of a word sequence.The language model should be able to differentiate between words that sound similar, but have diverse transcriptions.
The pronunciation dictionary usually includes phonemes (small acoustic units) and their pronunciations, so that any word can be created by composition, even if it was not featured during training.
The acoustic model says how probable a signal corresponds to a word sequence and models the relation of a speech signal and words. This is traditionally achieved by Hidden Markov Models (HMMs). Alternatively neural network approaches as time delay neural networks (TDNNs) [9] can be used.

Such an ASR system can be mathematically described by the fundamental equation of ASR. The most probable word sequence of all possible word sequences is $\hat{W}$. Given

the pre-processed audio signal $X$, the feature vectors, we need to find the word sequence $W$, so that the probability $P$ becomes maximal

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

With Bayes rule we can write

$$\hat{W} = \arg\max_{W} P(W|X) = \arg\max_{W} \frac{P(X|W) \cdot P(W)}{P(X)} = \arg\max_{W} P(X|W) \cdot P(W).$$

The last equality is given since $P(X)$ remains constant for all possible word sequences and the division does not make a difference for arg max and thus for the resulting word sequence.
In the formula $P(X|W)$ represents the acoustic model and $P(W)$ the language model.

## 2.2 Audio Pre-processing

After recording speech it needs to be converted into a machine readable form and prepared for ASR. This includes a reduction of dimensionality of the features. Next to the spoken words there is a lot of information contained within the signal, which are not important for our task. For example speaker related information, like pitch, or microphone specific noises.
The descriptions in this chapter are based on [8].

The pre-processing of the audio signal is the first step in an ASR system and can be done by different techniques. We use logarithmic Mel scaled spectrum [10] and describe the process shortly in the following.
The digitization of the signal is usually done by sampling it with a rate of 16kHz and a resolution of 16 bit.
This digital signal is then split into small windows, that are called frames. Since we are interested in frequency domain this windowing is done by convolution with a Hamming window. The frames have a length of e.g. 25ms, in which we assume the signal to be stationary. Choosing a smaller window size would increase time resolution, but frequency resolution would suffer. Vice versa time resolution would decrease when choosing a larger window size.
The windows do overlap with 10ms to improve time resolution again and catch changes at the borders coming from windowing. Discrete Fourier transform (DFT) is used to transform the signal into frequency domain.
As a next step the signal gets Mel scaled, which reduces the number of coefficients. Therefore the fact that humans can hear finer differences in lower frequencies than in higher ones is used.
The result for two example sequences is shown in Figure 2 and Figure 3. As we see there are only 40 coefficients for each point in time.

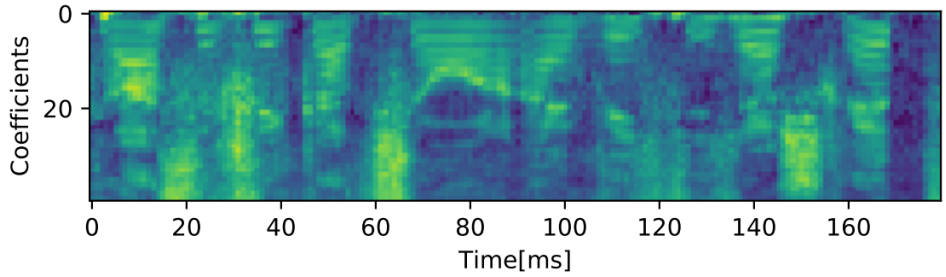


Figure 2: Acoustic features for the utterance: "Klassifikation ein wichtiger Schritt"

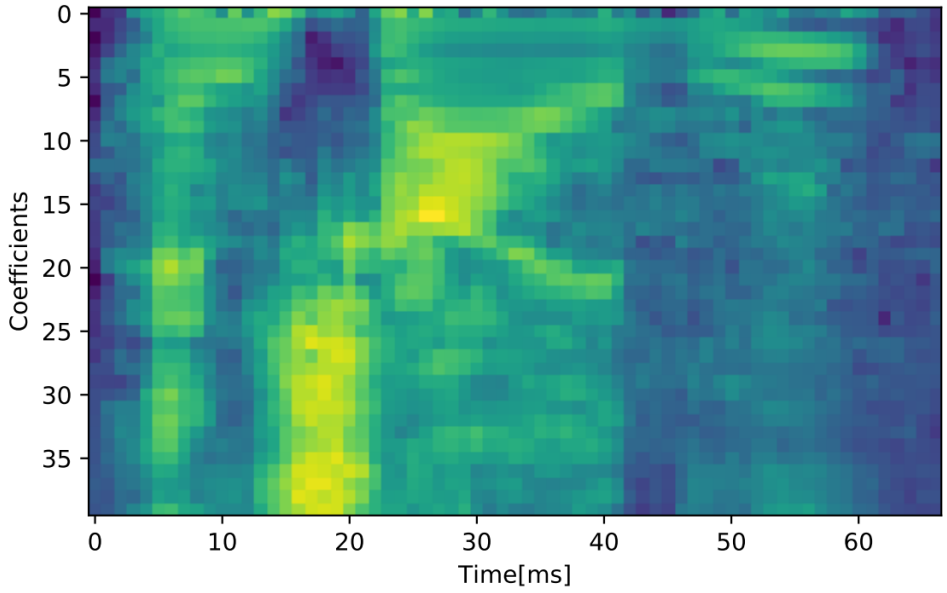


Figure 3: Acoustic features for the utterance: "entscheiden"

## 2.3 Data Preparation

As the audio is pre-processed the output text has to be prepared for training. Each training sequence has a unique Id to assign audio feature and correct transcription during training.
Out of the transcriptions of the sequences a vocabulary is created by a variation of byte pair encoding (bpe) [11]. Byte pair encoding compresses data by replacing the most common pair of bytes with another symbol, that is not in the data. This process is repeated until no further pairs can be replaced.
SentencePiece [12] is a tool, based on bpe, that automatically splits a text into subwords, which create the vocabulary. The size of the vocabulary can be chosen. Depending on this choice the resulting word pieces vary in length. Using word pieces allows us to build every word of the underlying text by combining the single pieces and create words that

have not been seen during training. SentencePiece computes the vocabulary by grouping more common letter combinations into one subword, while rare combinations are split into smaller units. Each of these pieces gets a unique id between zero and the vocabulary size, which are called the tokens. The audio transcription is then represented by these tokens.
In addition there are tokens for unknown words $< unk >$ and those which declare start $< sos >$ and end of sequence $< eos >$ included in the vocabulary.

## 2.4 Evaluation metrics

To compare different systems with each other evaluation metrics are used. Those measure how well a system is performing.

### 2.4.1 Word Error Rate (WER)

One possible metric is the Word Error Rate (WER). It can be looked up in [13], for example.
It compares the hypothesis with the correct transcription of the utterance (the reference) and computes how many words need to be changed to obtain the reference from the hypothesis. This is done by counting the minimal number of necessary deletions $\#del$, word insertions $\#ins$ and substitutions $\#sub$. They are added and divided by the number of words $\#words$ in the reference and multiplied by 100%

$$WER = \frac{\#del + \#ins + \#sub}{\#words} * 100\%$$

The same can be done with the characters, giving the Character Error Rate (CER), or other units.
The calculation of the WER always requires a trained system and is computed on a test set.

### 2.4.2 Word accuracy

The word accuracy $WA$ is the ratio of the correctly predicted words $\#correct$ and the number of words in the reference $\#words$, multiplied by 100%

$$WA = \frac{\#correct}{\#words} * 100\%.$$

## 2.5 T-distributed stochastic neighborhood embedding (t-SNE)

T-SNE can be used to visualize high dimensional data by mapping this data into a two or three-dimensional space. The high dimensional points are modeled in such a way that similar points are close to each other and dissimilar points far away in the low dimensional space. Based on the explanations in [14] we briefly describe t-SNE in the following.

The Euclidean distance between two high-dimensional points $x_i$ and $x_j$ is converted into a conditional probability $p_{i|j}$ by

$$p_{i|j} = \frac{exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} exp\left(\frac{-\|x_i - x_k\|^2}{2\sigma_i^2}\right)}.$$

The variance $\sigma_i$ of the Gaussian that is centered around data point $x_i$ and computed by a binary search such that the user-defined perplexity of the probability distribution is kept fix. Since only pairwise similarities between points are relevant we set $p_{i|i} = 0$.
The low dimensional points are modeled by a student t-distribution with one degree of freedom as

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}.$$

As before we set $q_{ij} = 0$. Having $n \in \mathbb{N}$ high-dimensional points in total we set $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$ and minimize the cost function $C$ (Kullback-Leibler divergence)

$$C = \sum_i \sum_j p_{ji} log\left(\frac{p_{ji}}{q_{ji}}\right)$$

by using a gradient descent method to compute low dimensional points $y_1, ..., y_n$.

## 2.6 Basics of Artificial Neural Networks

Artificial Neural Networks (ANNs) are becoming more and more important and are already frequently used. They help solving problems such as classification or prediction tasks. Next to speech recognition typical applications are image processing, machine translation or data mining, but they can be found in many other fields, as well.
While the single-layer perceptron [15], a very simple network, can only be used for linear classification, ANNs can become more or less arbitrarily complex and solve more complicated problems.
This chapter describes its basics, important types and how they are used in machine learning to train a model to solve a specific task.

ANNs are inspired by natural neural networks even though the functionality differs in some ways and they are not exact reproductions. The descriptions in this section are based on [16].
Natural neural networks consist of neurons, also called nerve cells. These cells receive, process and send information and are connected among each other to form a whole net. The connections between neurons are called synapses.
A neuron usually consists of the cell body, dendrites and an axon. Dendrites receive

signals from other neurons and transmit them to the cell body (soma), where all signals are joined and then processed. If the stimulus is strong enough, an impulse will be fired and transmitted by the axon to other neurons or muscles for example. Depending on the activated neurons we can perform certain tasks, for example process impressions or use one of our muscles. Performing a task more often, the corresponding synapses in the network get stronger, we "learn" and become better in what we do. Learning something can even create new connections, just like connections can get weaker or disappear. This is one key difference to ANNs, where it is not possible to create new connections. Figure 4 schematically shows a biological neuron. Directly after the activation of a neuron, dur-

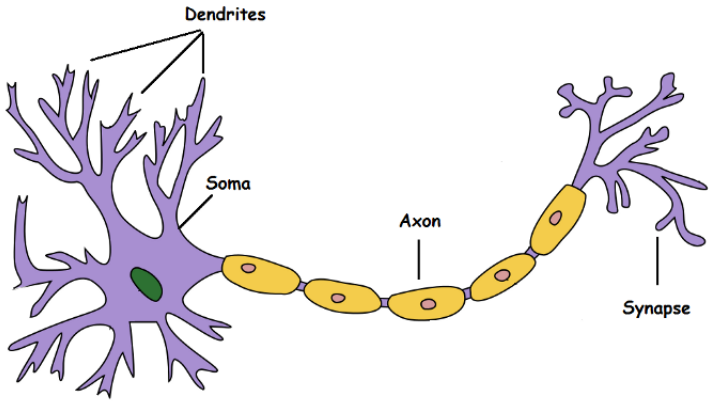


Figure 4: biological neuron [1]

ing the refractory phase, it can not react to incoming signals, which is another difference to an artificial neuron.

### 2.6.1 Artificial Neuron

The equivalent of nerve cells in biological neural networks are artificial neurons in ANNs, which are connected in a specific way. Chapter 2.6.2 presents a selection of important structures.
Basically each neuron computes an output $y \in \mathbb{R}$ based on the $n \in \mathbb{N}$ incoming signals, the inputs $x_1, ..., x_n \in \mathbb{R}$. This is done by multiplying each input $x_i$ by a coefficient $w_i \in \mathbb{R}$ first and summing up the weighted inputs afterwards. The weighting allows us to take greater account of parameters, that are of more significance to the result, while others with less significance can be dampened. This is comparable to the different strengths of the connections in natural neural networks.
Sometimes a bias $b \in \mathbb{R}$ is added to the weighted sum. The sum is then processed by a transfer or activation function, which is explained in Chapter 2.6.3 in more detail. Figure 5 shows an artificial neuron schematically.

Definition 2.1, based on [17], summarizes the description given above.

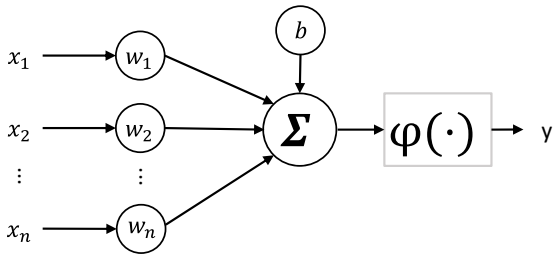


Figure 5: artificial neuron

**Definition 2.1** (Neuron). *A neuron with $n \in \mathbb{N}$ inputs $x_1, ..., x_n \in \mathbb{R}$ can be defined by $n$ weights $w_1, ..., w_n \in \mathbb{R}$, a bias $b \in \mathbb{R}$ and an activation function*

$$\varphi : \mathbb{R} \to Y,$$

*with $Y \subseteq \mathbb{R}$. Then the output $y$ of the neuron is given by*

$$y = \varphi\left(\sum_{i=1}^{n} x_i w_i + b\right).$$

### 2.6.2 Artifial Neural Network (ANN)

In an artificial neural network usually multiple layers of connected neurons exist. The number of layers and neurons depends on the individual task of the network just as the kind of connections does. Since connections can only become stronger or weaker and not emerge during training it is important to choose a network topology that fits the problem. Important kinds of artificial neural networks are described below.
Definition 2.2 summarizes the main characteristics of ANNs. It's leaned on [18], but strongly shortened and not completely formalized. It's purpose is to give an overview of important architectural properties to be able to differentiate between different kinds of ANNs.

**Definition 2.2** (Artificial Neural Network topology)**.** *The topology of an ANN can be described by its*

- *number of layers* $l \in \mathbb{N}$
- *number of neurons per layer* $n_i \in \mathbb{N}$, $i \in 1, ..., l$
- *connection scheme*

*The connection scheme is defined by*

- *Type of connections, i.e. which layers are connected*
- *connectivity, i.e. which neurons are connected*
- *symmetry vs. asymmetry, i.e. if connections are unidirectional or bidirectional*
- *order of connections, i.e. if a connection combines inputs from several neurons.*

*The first layer is called the input layer, the last the output layer and all layers in between are called hidden layers.*

These can be considered as the static properties of ANNs. Just like people can improve their abilities when practicing and benefit from experiences, neural networks need training with data to improve their performance on their specific task.
At the beginning the weights of a network are initialized, for example randomly and during the training stage these weights adapt to the problem that is to be solved.
Next to the description of the networks architecture some parameters like

- initial values of all parameters
- constraints (value ranges)
- transition functions (learning rule, activation function)

need to be defined for training. The learning rule defines how the weights are later updated, which is explained in Chapter 2.6.4.

ANNs can be also defined using graph theory as for example done in [19].

### Deep neural networks (DNN)

A deep neural network is a network with at least one hidden layer. Together with the input and the output layer it has at least three layers in total.

### Feed Forward neural networks (FFNN)

In a feed forward neural network (FFNN) information can only move forward, it is unidirectional. A neuron can be connected to every other neuron in the subsequent layer, but not to any neurons in the same or preceding layer. Thus it has no memory of previous input and is usually used for classification and not for prediction.

#### Recurrent neural networks (RNN)

The recurrent neural network (RNN), based on [20], can be seen as a generalization of FFNN. Connections between all neurons, including self-connections are possible. RNNs are often used for prediction tasks, since information of previous time steps can be taken into account.
Unfortunately RNNs have problems with long sequences, they "forget", due to the vanishing gradient problem [21]. A special kind of RNNs, the Long Short-Term Memory (LSTM), described in Chapter 2.7.1 helps in solving this problem.

### 2.6.3 Activation Functions

The activation function transfers the output to a specific range, often $[0, 1]$, or $[-1, 1]$, but every subset of $\mathbb{R}$ is possible. Different types of activation functions exist. In the single layer perceptron a binary function is traditionally used. Based on a threshold, the outcome is zero or one.
Most activation functions are continuous and nonlinear. One of them is the Sigmoid function $\sigma : \mathbb{R} \to (0, 1)$. It is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Further description and other activation functions and can be found in [17].

Another important activation function is the Softmax function [22]. Instead of being applied to each output $x_i \in \mathbb{R}$ independently, it also takes the other outputs of a layer into account. The Softmax function $Softmax : \mathbb{R}^K \to (0, 1)^K$, $K \in \mathbb{N}$, is defined as

$$Softmax(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \forall i \in \{1, ..., K\}.$$

When $K$ is the number of neurons in a layer with outputs $x_i$, $i \in 1, ..., K$, by applying the Softmax function the outputs get normalized in such a way, that they sum up to 1 and can thus be interpreted as a probability distribution. Therefore it is often used as a last layer to predict the probabilities of the outcomes.

### 2.6.4 Back-propagation, error functions and learning

During training the weights of a network need to be adjusted to optimize the outcomes. After a specific number $N \in \mathbb{N}$ of data points the error between actual and expected outcomes is computed and back-propagated through the network to update the weights [23].
For each datapoint $p$ the actual output $y_{pj} \in \mathbb{R}$, of the $j$th neuron is compared to the expected output $d_{pj} \in \mathbb{R}$. The total error of the net is defined by a loss function, e.g

mean squared error

$$E = \sum_{p=1}^{N} \frac{1}{N} \sum_{j} (y_{pj} - d_{pj})^2 .$$

The gradients of the error are computed and back-propagated through the network to update the weights with the goal to minimize the error. One possible rule to update a weight $w$ is

$$\tilde{w} = w - \alpha \cdot \frac{\delta E}{\delta w},$$

depending on the gradient $\frac{\delta E}{\delta w}$ and learning rate $\alpha > 0$.
Several other possible loss functions and methods to update the weights are possible.

The network is trained in epochs. In each epoch the whole training data is fed into the network and processed one time. After each epoch the model's accuracy on the training and the cross-validation data is computed. The cross-validation data examines the model's performance on data, that was not seen during training.

### 2.6.5 Dropout

One problem of ANNs is overfitting: the networks adaption to the training data is too good and it is not able to handle unseen data in a proper way anymore.
Dropout [24] helps solving this problem by randomly omitting connections between neurons during training. This forces the network to not train the same connections all the time.

## 2.7 Sequence-To-Sequence model (seq2seq)

A sequence-to-sequence (seq2seq) model is an ANN that processes sequences. Many problems like speech recognition or machine translation have a sequence as input and another sequence as output. In general the sizes of in- and output are neither known nor fixed, which makes the encoder-decoder architecture, a special seq2seq model, a good choice. In contrast to simple DNNs it is able to handle variable length of input and output sequences.
The seq2seq encoder-decoder architecture was introduced in 2014 in [25],[26] and the recognition is done in two steps. First the input sequence is processed item by item by the encoder into some representation, which is then fed into the decoder, that transfers this representation into the output sequence.
Encoder and decoder both consist for instance of Long Short-term Memory (LSTM) or RNN layers. LSTMs have the great advantage that they can handle long sequences because they don't have the vanishing gradient problem. The LSTM is explained in Chapter 2.7.1.
In speech recognition the encoder can be seen as the acoustic model, since it processes

the acoustic features.
The decoder on the other side operates the output tokens and thus corresponds to the language model.

Two model architectures are explained in detail: a LSTM based encoder-decoder architecture and the Transformer.Both architectures use an attention mechanism to align audio features and output tokens. Attention is explained in Chapter 2.7.3. Another common method of alignment is CTC [27]. It is also possible to use a combination of both methods, for example realized in [28], [29].

### 2.7.1 Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) cell [30] was first introduced in 1997 and was later modified for example in [31]. This chapter is based on these papers.

In a LSTM cell an input, an output and a forget gate are used to update the currently stored information. Figure 6 shows such a memory cell with the three gates.

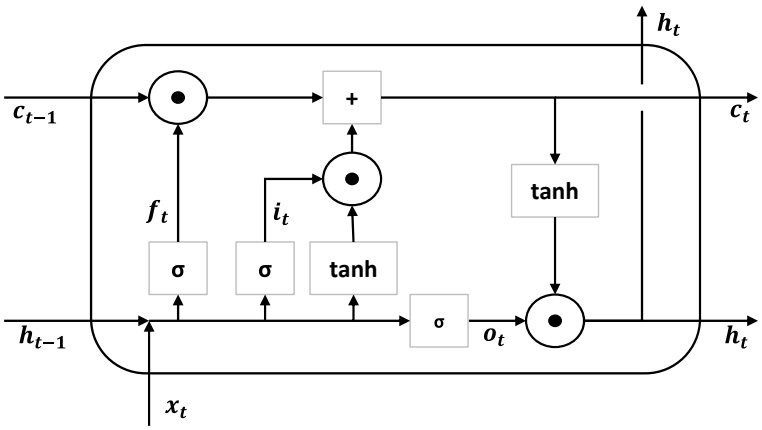


Figure 6: LSTM cell with input, output and forget gate inspired by [2]

Let $x_t \in \mathbb{R}^{d_{input}}$ be the input at time $t$ and $h_{t-1} \in \mathbb{R}^{d_{model}}$ the hidden state one time-step $t-1$ before or the initial state. For each of the three additional gates $h_{t-1}$ and $x_t$ are concatenated and first multiplied by weight matrices $W_i, W_f, W_o \in \mathbb{R}^{d_{model} \times (d_{model}+d_{input})}$. Biases $b_i, b_f, b_o \in \mathbb{R}^{d_{model}}$ are added and the Sigmoid function (see Chapter 2.6.3) is applied. Thus the input gate $i_t$, forget gate $f_t$ and output gate $o_t$ are computed as

$$
\begin{aligned}
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right), \\
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \\
o_t &= \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right),
\end{aligned}
$$

where $\sigma$ denotes the Sigmoid function.
The input gate $i_t$ is multiplied element-wise (see 2.1) with

$$\tilde{c}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right),$$

which is also based on the last hidden state $h_{t-1}$ and the actual input $x_t$. Using tanh allows $\tilde{c}_t$ to have negative values, too. Here $W_C \in \mathbb{R}^{d_{model} \times (d_{model}+d_{input})}$ and $b_C \in \mathbb{R}^{d_{model}}$ are weight matrix and bias again.
The old cell state $c_{t-1}$ is multiplied element-wise by the forget gate to calculate how much to forget and added to the modulated input gate to receive the new cell state

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t. \tag{2.1}$$

Here $\odot$ denotes the element-wise multiplication.
The output gate multiplied by $\tanh(c_t)$ yields the hidden state for time $t$

$$h_t = o_t \odot \tanh\left(c_t\right),$$

which is outputted by the memory cell together with the cell state $c_t$.

#### Multilayer LSTM

A multilayer LSTM is composed of multiple stacked LSTM cells. The output $h_t$ of one layer is the input $x_t$ of the next layer, so $d_{input}$ is equal to $d_{model}$ after layer 1.

#### Bidirectional LSTM (BiLSTM)

In a bidirectional LSTM (biLSTM) the information does not only travel from past to future, but also from future to past. The hidden states of the corresponding cells of the two directions need to be combined, before being sent to the next cell. This can be done by concatenation, element-wise addition or multiplication for example.

### 2.7.2 LSTM based encoder-decoder seq2seq model architecture

Our encoder-decoder architecture with LSTM layers [32] is shown in Figure 8.

**Encoder:**
The encoder is composed of $N_{Enc} \in \mathbb{N}$ stacked bidirectional LSTM layers. The encoder processes the audio feature vectors. The acoustics are not only depending on the features of one point in time, but also on subsequent and previous ones. Thus it can be useful to take the whole sequence of features into account.

**Decoder:**
The decoder is composed of $N_{Dec} \in \mathbb{N}$ LSTM layers. It's inputs are the already known output tokens, that are used to predict future ones. It can be seen as the language model. The output tokens are embedded before being processed by the LSTM layers.

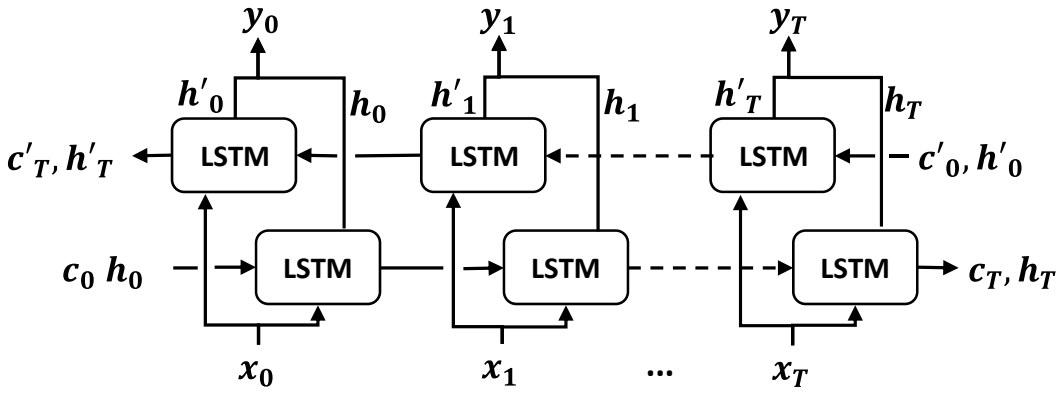


Figure 7: bidirectional LSTM inspired by [3]

In the Multi-Head Attention (MHA) layer the acoustic features and tokens are aligned. MHA is explained in detail in Chapter 2.7.3. The dashed line shown in Figure 8 constitutes a residual connection, that adds the scaled output of the last LSTM layer to the output of the MHA layer.
When $x_{enc} \in \mathbb{R}^{d_{model}}$ is the output of the encoder, $x_{dec} \in \mathbb{R}^{d_{model}}$ the decoder's output and $MHA$ the attention layer function, this can be written as

$$y = x_{dec} \cdot s + MHA(x_{enc}, x_{dec})$$

with some scaling factor $s \in [0, 1]$. It's output $y \in \mathbb{R}^{d_{model}}$ is then inputted to the Linear layer, where the projection to the new output token takes place. The Softmax layer is used to give a probability estimation of the recognized output tokens.

### 2.7.3 Attention

Attention [33],[34] is a mechanism that enables the model to focus on important parts of a sequence. As described before it can be used to align acoustic features and tokens. When modeling dependencies between the single positions of one sequence, we say self-attention. The following description explains self-attention and is based on [35]. Other attention functions than presented here are possible, as well.

For each position in a sequence a similarity score is computed for all parts of the sequence. Let $x_i \in \mathbb{R}^{d_{model}}$ be the input feature vector of the attention layer for position $i \in \{1, ..., d_X\}$ in the sequence with length $d_X \in \mathbb{N}$. By linear transformation of $x_i$ the query $q_i$ is computed

$$q_i = W_Q x_i,$$

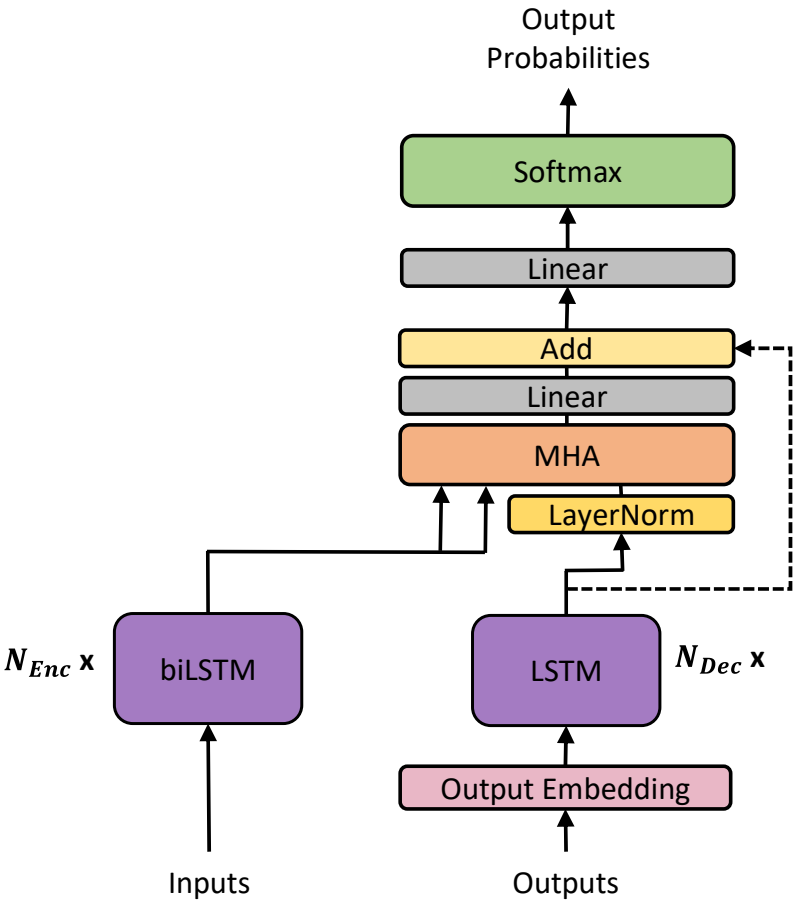


Figure 8: LSTM based encoder-decoder architecture

with $W_Q \in \mathbb{R}^{d_k \times d_{model}}$. For position $j$ of the same sequence the key $k_j$ and value $v_j$, $j \in \{1, ..., d_X\}$ are computed as

$$k_j = W_K x_j,$$
$$v_j = W_V x_j$$

with $W_K, W_V \in \mathbb{R}^{d_k \times d_{model}}$. The inner product of $q_i$ and $k_j$ is used to measure the similarity score

$$s_{ij} = \frac{q_i^T k_j}{\sqrt{d_k}}.$$

Without scaling by $\frac{1}{\sqrt{d_k}}$ the dot product could become very large for large $d_k$, which would result in unstable gradients.
The score $s_{ij}$ of the $i$th position is computed for every $j \in \{1, ..., d_X\}$ of the sequence and the Softmax function normalizes the resulting vector

$$o_i = Softmax\left([s_{i1}, s_{i2}, ..., s_{id_X}]\right).$$

Each value $v_j$ is then weighted by the corresponding normalized score value $o_{ij}$. The attention vector for $q_i$ is then given by

$$Attention(q_i, k_j, v_j) = \sum_{j=1}^{d_X} o_{ij} \cdot v_j.$$

This can be done using matrix multiplication allowing to do the calculation in parallel for all positions.
Let $X \in R^{d_{model} \times d_X}$ be the input matrix, each column containing the input vector for the corresponding position in the sequence. Then

$$Q = W_Q X, K = W_K X, V = W_V X$$

are the key $K$, query $Q$ and value $V$ matrices and

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_K}}\right) V.$$

denotes the attention function.

In encoder-decoder attention the query vector would come from the decoder, while key and value are based on the output of the encoder.

#### Multi-Head-Attention (MHA)

The attention mechanism can be seen as a projection of the input into one space by the embedding matrices. We define this to be one attention head. Instead of using only one head it is possible to use multiple attention heads. This means multiple embedding matrices for key, value and query. Each of them projects query, key and value matrices into a different representation subspace, which allows to always focus on different information.
The computation of the output of each head is as described before. The resulting matrices of the single heads are concatenated and down-projected. In Figure 8 this step is done in the Linear layer after the MHA layer.
Figure 9 shows an example of multiple attention heads for self-attention of the acoustic features of one utterance before multiplication with the value matrix. The focus lies on or around the diagonal in most cases, which could mean that features close to each other are most important. Only two heads seem to attend to completely other positions in the utterance.

### 2.7.4 Transformer

Another seq2seq model is the Transformer. It was published 2017 in [35] for translation and uses the attention mechanism in encoder and decoder. The explanations in this chapter are based on [35] and [4], where the Transformer was adapted for ASR.

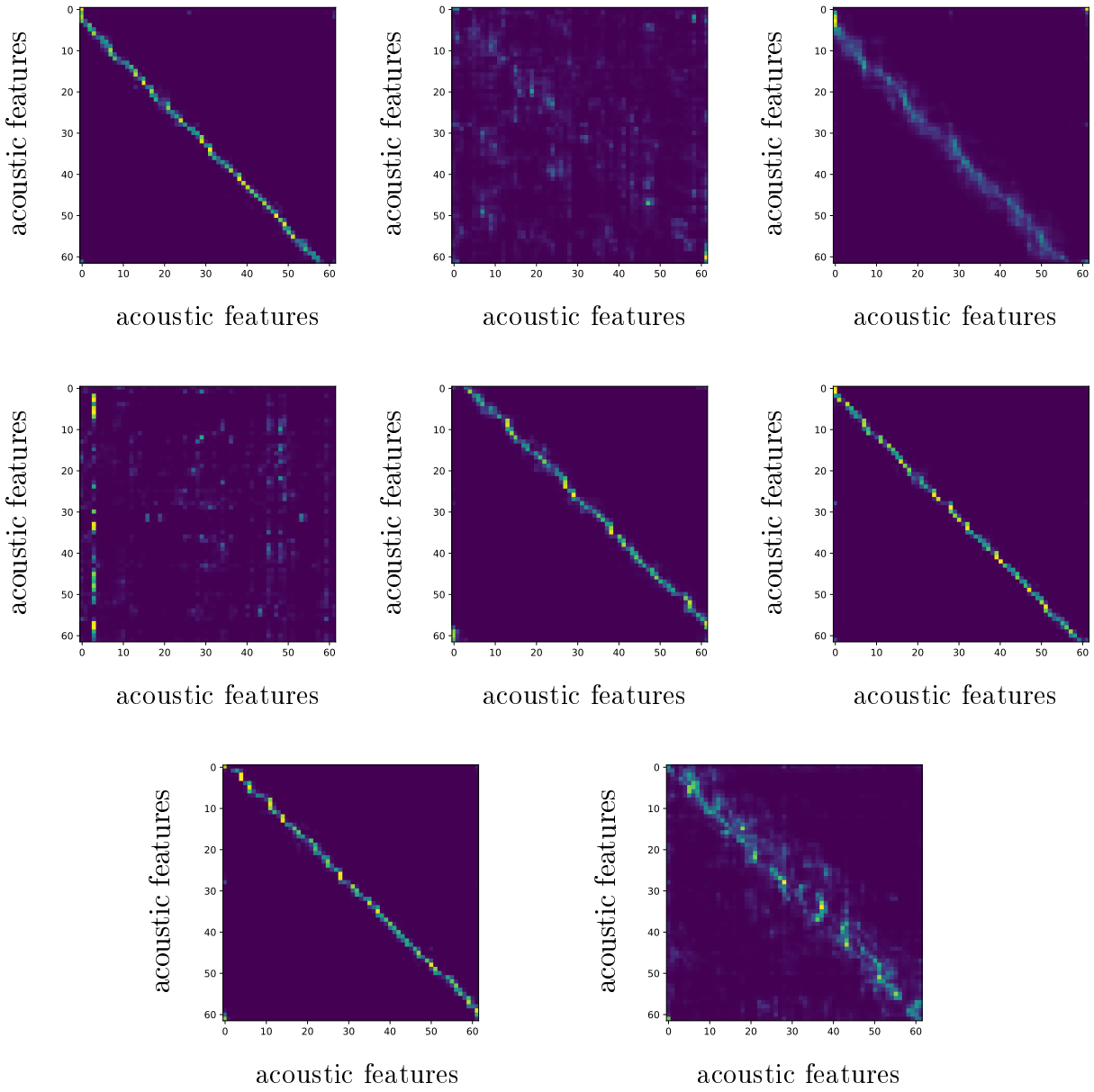


Figure 9: Eight self-attention heads for acoustic features of "keine Ereignisse suchen Bewegungen analysieren"

**Encoder:**
The encoder is stacked of $N_{Enc} \in \mathbb{N}$ encoder layers, each consisting of a self-attention and a feed forward layer. The feed forward layers are fully connected FFNs, which are applied position-wise as

$$FFN(x) = W_2 \cdot \max(0, W_1 x + b_1) + b_2$$

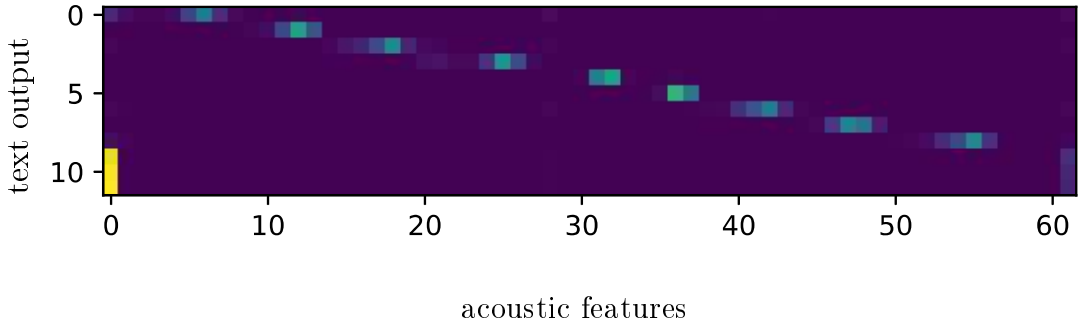


Figure 10: one encoder-decoder attention head

with weigh matrices $W_2 \in \mathbb{R}^{d_{model} \times d_{inner}}$ and $W_1 \in \mathbb{R}^{d_{inner} \times d_{model}}$ and biases $b_1 \in \mathbb{R}^{d_{inner}}$ and $b_2 \in \mathbb{R}^{d_{model}}$.
The encoder outputs a representation of the audio features.

**Decoder:**
The $N_{Dec} \in \mathbb{N}$ decoder layers are each composed of a masked self-attention, an encoder-decoder attention and a feed forward layer. The masking of the self-attention layer forces the network to focus on the already recognized tokens only.
The encoder-decoder attention layer key and value are computed based on the encoder's output, while the query is based on the preceding masked self-attention layer.

Around all single layers of encoder and decoder there are residual connections, that add the input of the single layers to their outputs, similar to the residual connection around the MHA layer in 2.7.2 for the LSTM encoder-decoder architecture.

**Positional Encoding:**
Inputs and outputs are embedded to vectors of size $d_{model} \in \mathbb{N}$, before being fed into encoder or decoder and the positional encoding

$$pe[pos, i] = \sin(pos/(10000^{(2 \cdot i/d_{model})})), \text{i even}$$
$$pe[pos, i] = \cos(pos/(10000^{(2 \cdot i/d_{model})})), \text{i odd}$$

is added. The positional encoding provides information about the relative positions in the sequence during, which is not explicitly present in the Transformer. This is different to the LSTM encoder-decoder architecture where information flow through in chronological order. The positional encoding is based on the position $pos$ in the sequence and the index $i \in \{1, ..., d_{model}\}$ of vectors of size $d_{model}$ .

Figure 10 shows one example head for the encoder-decoder attention. The lighter the higher is the score for theses parts. The $y$-axis corresponds to the embedded output

tokens and the $x$-axis to the acoustic features.
The utterance "keine ereignisse suchen bewegungen analysieren" is tokenized as

keine-ereig-nisse-suchen-be-weg-ungen-anal-ys-ieren

in this example.

As in the encoder-decoder architecture with LSTM layers the Softmax layer as output layer predicts probabilities for each target. Figure 11 shows the architecture of the Transformer.

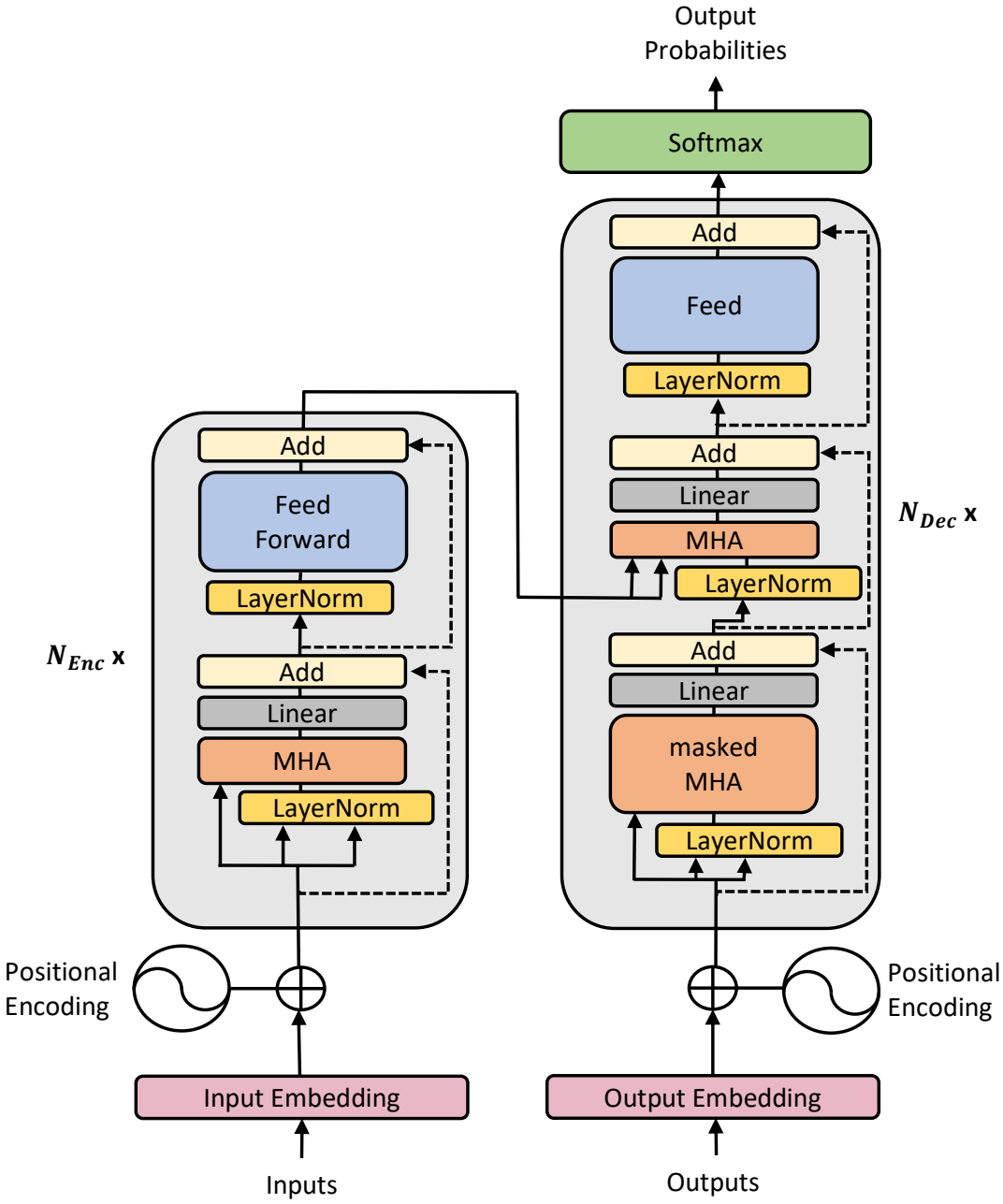


Figure 11: Transformer architecture based on [4]

# 3 Related Work: end-to-end approaches for multilingual speech recognition

In this chapter we give an overview of different approaches to multilingual ASR. We concentrate on research in the field of multilingual ASR with neural networks.
Simply training a network with mixed language data can lead to improvements as well in some cases [36], [37]. However this is not the general behavior. The following methods all try to reach language adaption by explicitly adding language information. By doing so the model is supposed to learn language dependent features.

To model the language identity (LID) a one-hot vector can be used. It has the same size as there are languages available. Each of the vector's indices correspond to a language and the entry at this index is set to one, while all other values are zero.
Another possibility is the use of a language embedding vector, which can be for example obtained by linear transformation of the one-hot language vector.

## 3.1 Modulation

Modulation denotes basically the multiplication of the outputs of a layer with weights.

In [38] multilingual ASR with modulation was investigated. The concept is based on Meta-Pi [39] networks, which was introduced for speaker adaption. The architecture

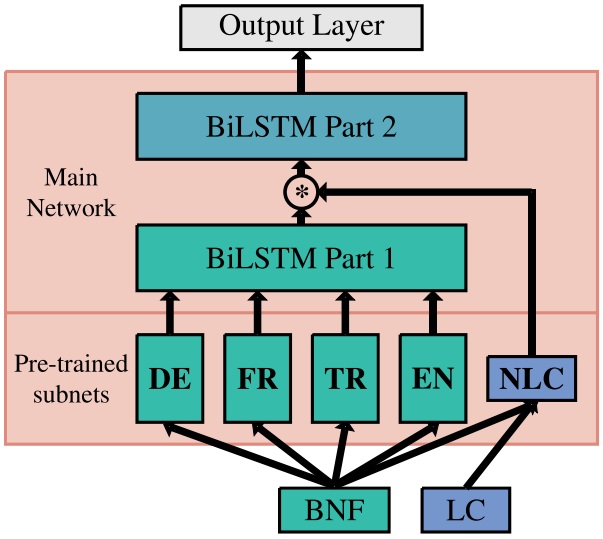


Figure 12: Model architecture with modulation [5]

shown in Figure 12 consists of a main network and pre-trained subnets. The main network is composed of two parts each consisting of two biLSTM layers. After Part 1 the outputs are modulated with neural language codes (NLCs) obtained from one of the

pre-trained subnets. These NLCs are learned language properties, that were proved to carry more language information than the LID alone.
Modulation can be seen as an intelligent way of dropout. By strengthening and damping connections depending on the language properties the network learns features based on the language properties.

## 3.2 Language Token

Language adaption can also be achieved by including language tokens into the vocabulary. In [40] a Transformer was trained using language tokens integrated into the labels. At the beginning or end of each utterance the LID token was inserted in addition to the start $< sos >$ or end $< eos >$ token. When inserting it at the beginning the label of an utterance is of the form

$$< sos >< lid > \text{utterance} < eos > .$$

This approach can also be used for multitask learning to recognizing speech and predict the language at once.
Similar approaches were chosen in [41],[42],[43] and [36].
It was shown that inserting an extra token at the end is slightly better than inserting it at the beginning. Compared to a jointly trained model without any language adaption the improvements in WER were between 1% and 2% for most languages.
Alternatively the LID can also be integrated into the start token as

$$< sos - lid > \text{utterance} < eos > .$$

However this approach can only be used for experiments when the LID is known during inference. In [40] the WER decreased up to 6.6% when using language tokens inserted into the start token.

## 3.3 Language Vector

A common method to adapt to a specific language is the insertion of language information as additional input to the model. To represent the language a one-hot vector or a learned language embedding can be used. For an encoder-decoder architecture with attention mechanism the addition of a learned embedding to the first layer of encoder, decoder or both was compared in [41]. Best results were obtained when adding it to either the encoder or to encoder and decoder. In average the model, that was trained with nine Indian languages, outperformed the jointly trained baseline by a reduced WER of 6.9%. When the language vector was only added as input to the encoder the WER decreased by 6.7%.
In [36] a one-hot vector or alternatively an embedding of different dialects was added as input to different layers of an encoder-decoder model. Tests showed that feeding the one-hot vector to both the encoder and decoder gave best results and outperformed the

multi-dialect model without adaption by 12.5% improved WER.
For most dialects adding the one-hot vector instead of an embedding led to better performance.

To add the language vector $d_l$, describing language $l \in \{0, ..., L-1\}$, having $L \in \mathbb{N}$ languages, as input to a layer, one common way is the concatenation of the feature vector $x \in \mathbb{R}^{d_{input}}$ and $d_l$

$$W \cdot [x, d_l] = W_x x + W_l d_l + b$$

with $W = [W_x, W_l]$, $W_x \in \mathbb{R}^{d_{model} \times d_{input}}$, $W_l \in \mathbb{R}^{d_{model} \times L}$ and $b \in \mathbb{R}^{d_{model}}$.
This is equivalent to add a language dependent bias $b_{lang} \in \mathbb{R}^{d_{model}}$

$$W_x x + W_l d_l + b = W_x x + b_{lang}$$

since $W_l d_l$ only depends on the language and not on the feature vectors.

## 3.4 Fine-tuning/ Adapter modules

A pre-trained network can be adapted to a new task by fine-tuning. In the context of multilingual ASR pre-training with data of other languages and then fine-tuning by data of the target language can improve the results. This is particularly useful when not enough data of the target language is available. Another advantage is the reduction in training time when using a pre-trained model.

It is also possible not to re-train the whole network, but only specific parts or single layers. In [37] language dependent output layers were trained.
The authors of [44] investigated linear hidden unit contribution (LHUC) in multilingual ASR. The outputs of a hidden layer are re-scaled by language dependent parameters. When adding a new language the output layer is extended and the parameters are updated by additional training.
Instead of only having language dependent layers, also complete language specific subnets can be included. In [45] a conventional Transformer was trained with preceding encoders for each language. The output of the - in this case - two encoders were concatenated and then down-projected and fed into the classic Transformer, which gave an average improvement in WER of 4.6%.
In [46] first a global model with imbalanced data of nine Indian languages was trained jointly. Then the model parameter were kept fix and for every language an adapter module was introduced after each layer of the encoder. They achieved an average improvement of 17.5% in WER compared to the monolingual baselines and 53.2% compared to the multilingual baseline.

## 3.5 Gating

In [6] a gating mechanism was applied to the hidden states of each biLSTM layer of a CTC-based end-to-end model.

For the gating mechanism the language identity of language $l \in \{0, ..., L-1\}$, having $L \in \mathbb{N}$ languages, is represented by an one-hot vector $d_l$. Based on $d_l$ and the hidden state $h_i \in \mathbb{R}^{d_{model}}$ of a layer $i$ a gate-value is computed as

$$g\left(h_i, d_l\right) = \sigma\left(Uh_i + Vd_l + b\right)$$

with $\sigma$ as the sigmoid function.
The hidden state $h_i$ and $d_l$ are linearly mapped with matrices $U \in \mathbb{R}^{d_{model} \times d_{model}}$, $V \in \mathbb{R}^{d_{model} \times L}$ and a bias $b \in R^{d_{model}}$ is added. The hidden state is then modulated by the gate value

$$\hat{h}_i = g\left(h_i, d_l\right) \odot h_i.$$

Here $\odot$ denotes the element-wise multiplication. The resulting vector $\hat{h}_i$ and $d_l$ are concatenated

$$\tilde{h}_i = \left[\hat{h}_i : d_l\right]$$

and down-projected by $W \in \mathbb{R}^{d_{model} \times (d_{model}+L)}$

$$h'_i = W\tilde{h}_i.$$

The new hidden state $h'_i$ is inputted into the next layer. The above described gating mechanism was compared to the following other methods

- concatenation of hidden state and LID $[h_i : d_l]$ (see Chapter 3.3)
- modulation of the embedded hidden state with the hidden state $g\left(h_i\right) \odot h_i$
- modulation of the embedded LID with the hidden state $g\left(d_l\right) \odot h_i$
- modulation of the embedded LID and hidden state with the hidden state $g\left(h_i, d_l\right) \odot h_i$ (without concatenation with LID and down-projection as last steps)

The methods are named in the order of descending WER obtained using them. Gating outperformed the bilingual models by an average decrease in WER of 6.5%. In a trilingual model the improvement was even higher with 9.3% relative decrease in WER.
In [47] gating after the first encoder layer and after all encoder layers in a RNN-Transducer [48], consisting of LSTM layers, was researched.

## 3.6 Language Embedding Concatenation (LEC)

In [7] language adaption in the attention layer was investigated for a RNN-Transducer model with audio encoder replaced by the encoder of the Transformer. Instead of the

attention mechanism described in Chapter 2.7.3 a language specific bias term is added to key, value and query. For each layer $i$ key $K_i$, value $V_i$ and query $Q_i$ are computed as

$$Q_i = Q_x x_i + Q_l d_l + b_q$$
$$K_i = K_x x_i + K_l d_l + b_k$$
$$V_i = V_x x_i + V_l d_l + b_v$$

where $x_i$ is the feature vector and $d_l$ is the one-hot vector describing the language $l \in \{0, ..., L-1\}$, having $L \in \mathbb{N}$ languages. Here $Q_x$, $K_x$ and $V_x$ are the "standard" weight matrices (see Chapter 2.7.3), while $Q_l, K_l, V_l \in \mathbb{R}^{d_k \times L}$ are language dependent weight matrices for query, key and value. In addition biases $b_q, b_k, b_v \in \mathbb{R}^{d_k}$ are added to the query, key and value. During inference $Q_l$, $K_l$, $V_l$ do not depend on the input such that this method can be considered as adding language dependent biases. The method is called Language Embedding Concatenation (LEC).
When applying LEC to every layer in the audio encoder of the RNN-Transducer model the WER decreased by 7.1% compared to the multilingual baseline without language adaption.

## 3.7 Language Specific Attention Heads (LSAH)

In [7], where LEC was proposed, another method, Language Specific Attention Heads (LSAH), was presented. The encoder of the Transformer was used as audio encoder in a RNN-Transducer model. Instead of adapting to a language by adding language specific biases as done in LEC, a combination of shared and language specific attention heads is used in the MHA layers.

Usually in Multi-Head Attention $H \in \mathbb{N}$ heads are used in each layer, that are shared by all languages in multilingual ASR. Now $K \in \mathbb{N}$ language-specific heads per language are added and at the same time the number of shared heads is reduced by $K \cdot L$ heads, $L \in \mathbb{N}$ being the number of languages.

When $H$ being the total number of heads, the $i$th head

$$head_i = Attention(Q, K, V), i \in \{1, ..., H\}$$

is computed as described in Chapter 2.7.3.
For each language $l \in \{0, ..., L-1\}$ the $K$ language specific attention heads are trained using data of the specific language only and concatenated

$$LanguageSpecificHeads_l = concat_{k \in \{1,...,K\}}(head_{l \cdot K + k}).$$

The shared heads

$$SharedHeads = concat_{c \in \{L \cdot K, ..., H\}}(head_c)$$

are trained commonly and independent from language. For language $l$ then language specific and shared heads are concatenated

$$MultiHead_l = concat(LanguageSpecificHeads_l, SharedHeads)$$

and proceeded as usual afterwards.
When maintaining the total number of attention heads as used in MHA without language adaption this leads to a reduction of learnable parameters, because the number of heads used for each language is reduced by $K \cdot (L-1)$.
In experiments with five languages best results were obtained using $K = 1$ and $H = 12$ heads in total. Overall a relative improvement of 7.6% in WER was obtained compared to the baseline with eight heads.

By combination of LEC and LSAH the WER decreased further by 8.6%.

# 4 Experiments

This chapter describes our experiments in detail. The first section is about the data used. In the subsequent Chapter 4.2 we explain our experimental setup and present the results for the Transformer. Chapter 4.3 addresses the encoder-decoder architecture consisting of LSTM layers.

## 4.1 Data

In the first step we trained the systems with the two languages, English and German.
The English training data set consists of ted talks of different people, German was trained with recordings from parliament talks and lectures.
Table 1 gives an overview of the total length of the data and the number of available utterances for both languages.

| Language | Length of data | Number of utterances |
|---|---|---|
| English | $425h$ | $251,201$ |
| German | $425h$ | $317,975$ |

Table 1: Training data (mixed-case) for English and German

We tried to use a similar amount of data for each language in training. This was about 425 hours for English and German, the number of utterances used for German was slightly higher.
The audio features are always extracted as described in Chapter 2.2 and the transcriptions are tokenized as explained in Chapter 2.3.
The total length and the number of utterances used for cross-validation are summarized in Table 2.

| Language | Length of data | Number of utterances |
|---|---|---|
| English | $22.5h$ | $13,221$ |
| German | $22.1h$ | $16,735$ |

Table 2: Cross-Validation (mixed-case) data for English and German

A specificity of German is the capitalization of words like nouns. Creating a vocabulary in multilingual scenario with mixed-case data for German will result in tokens that can not be used to recognize English. Furthermore words like "sie"/ "Sie" only differ in the capital letter at the beginning, but are represented by different tokens.
The splitting of the transcriptions into tokens is based on the frequency of letter combinations. This results in larger tokens for more common subwords, while rare subwords are split into smaller tokens.
Using only lower-case data might give a more realistic reflection which subwords are common and the number of shared tokens might increase, as well.

For this reason we later used lower-case data only with the expectation of improving

recognition further. As before for the mixed-case data, parliament and lecture recordings are used for German in training. In addition we also use common voice data. For English the data set remains the same. As a third language we added French, with data consisting of common voice utterances.
The lower-case data is summarized in Table 3 and Table 4 for training and cross-validation.

| Language | Length of data | Number of utterances |
|---|---|---|
| English | $425h$ | $251,201$ |
| German | $425h$ | $258,149$ |
| French | $419h$ | $292,539$ |

Table 3: Training data (lower-case) for English, German and French

| Language | Length of data | Number of utterances |
|---|---|---|
| English | $22.5h$ | $13,221$ |
| German | $22.6h$ | $13,587$ |
| French | $9.3h$ | $6,443$ |

Table 4: Cross-Validation (lower-case) data for English, German and French

After training we measure the performance of each model by computing the WER. The English test data consists of recordings of ted talks. For German we use lecture and parliament talks and for French common voice data is used for testing.

We also have a small data set consisting of Denglish utterances. These are German sentences containing English words, e.g.

"schreib mir dazu mal einen reminder fürs nächste meeting".

In German, English words are used in a germanized form in daily speech. Those words like "gecrasht", "outgesourced" or "googeln" are also contained in the Denglish data set.

Table 5 shows the length and number of utterances for each test set. The tests itself are not case-sensitive.

| Language | Length of data | Number of utterances |
|---|---|---|
| English | $2.6h$ | $1,155$ |
| German | $6.4h$ | $8,869$ |
| French | $23.3h$ | $14,504$ |
| Denglish | $350s$ | $90$ |

Table 5: Test data for English, German, French and Denglish

## 4.2 Transformer

In this chapter the experiments for the Transformer and their results are presented. The Transformer model architecture is explained in Chapter 2.7.4.
In Chapter 4.2.1 we first explain the parameters of the model and in the subsequent sections different approaches for the multilingual Transformer are evaluated.

### 4.2.1 Parameter Transformer

Table 6 summarizes the parameters of the Transformer. We use $N_{Enc}$ encoder layers and $N_{Dec}$ layers in the decoder. The input feature vectors have a length of $d_{input}$ at the beginning. By the input embedding they are transformed to vectors of size $d_{model}$. The feed forward network consists of 2 layers, the first projects the vectors into a higher dimensional space of size $d_{inner}$.
The number of heads used in Multi-Head Attention is identical in encoder and decoder and set to $n_{head}$. The size of key, query and value is chosen to be $d_k$ for each MHA layer. We use a vocabulary size of $n_{vocab}$, independent of monolingual or multilingual systems.

| | |
|---|---|
| $N_{Enc}$ | 8 |
| $N_{Dec}$ | 4 |
| dropout | 0.2 |
| $d_{input}$ | 160 |
| $d_{model}$ | 512 |
| $d_{inner}$ | 1024 |
| $d_k$ | 64 |
| $n_{head}$ | 8 |
| $n_{vocab}$ | 4003 |

Table 6: Parameters in the Transformer architecture

Table 7 shows the total number of trainable parameters in the Transformer.

| Model | Number of parameters |
|---|---|
| Transformer | 31,526,307 |

Table 7: Number of learnable parameters in the Transfromer

### 4.2.2 Baseline Transformer

As baseline we trained monolingual Transformers for English and German. In addition a bilingual Transformer was jointly trained with mixed language data.
In Table 8 the results for German and English are shown for the monolingual and bilingual setups. The column "Bpe" says which language data was used during training. Here we differentiate between monolingual German (ge), monolingual English (en) and bilingual (mix) with German and English mixed data.
We trained the models twice, with mixed-case and lower-case data. For this reason

the result table contains different columns for the word error rate (WER). "WER mc" denotes the setup with mixed-case training data and "WER lc" is used fo the WER of the model trained with lower-case data.

| Language | Model | Bpe | WER mc | WER lc |
|---|---|---|---|---|
| English | Transformer | en | 12.3% | 12.3% |
| German | Transformer | ge | 20.6% | 20.7% |
| English | Transformer | mix | 12.7% | 11.7% |
| German | Transformer | mix | 20.7% | 21.1% |

Table 8: Baseline Transformer

Comparing monolingual and multilingual systems both yield similar results. This is surprising, since we expected the performance to decrease in the multilingual setup. Maybe the fact that German and English are related languages and a lot of words are similar helps the bilingual model in handling the different data sets. Possibly the doubled amount of training data compared to monolingual training also plays a role and improves generalization.

In the bilingual model trained with mixed-case data the WERs increase for both languages even though the differences are small. In the lower-case model the recognition of English improves, while for German it's worse. We assume that the vocabulary now contains less German specific tokens and tokenization is more suitable for English. In addition the model doesn't need to learn capitalization that is irrelevant for English. However WER mc and WER lc are not completely comparable due to the slightly different data taken for German.

One observation made for German was that often the recognition is either really good for one utterance with only minor mistakes or nothing is recognized at all. Instead specific words are hypothesized like "axiallasten" for the spoken utterance "gucken wir mal" or "wälzkörpersatz" instead of "dass sie das verinnerlichen". Those words are only used in very specific domains and do not have any acoustic similarity to the spoken sentences. One possible explanation is that they showed up in training very often and the Transformer learns to predict them, when nothing else was understood. This kind of confusion seems to be a Transformer specific problem and showed up during all of our experiments for German.

We also tested the monolingual systems with the respective other language, which results in very high WERs as expected. The results are shown in Table 9.

| Language | Model | Bpe | WER mc | WER lc |
|---|---|---|---|---|
| English | Transformer | ge | 106.5% | 97.6% |
| German | Transformer | en | 129.7% | 129.9% |

Table 9: Baseline Tranformer tested with wrong language

Comparing hypotheses and references for the German system tested with English little acoustic similarity can be observed. The predicted words are very often neither correct German nor English words, although they are primarily close to German. This is as expected, since the vocabulary consists of German subwords. For example the utterance "the possibility of an individual to see themselves as capable" is recognized as "ja beparsteten büro die aber in ende wildjor testie dem seros ist kate geborgt" by the model trained with lower-case data. Furthermore the hypotheses don't build sequences that would be possibly seen in German. If so, they are very short like "das ist" or "es war ein".
The English system tested with German often predicts existing English words. As for the German system tested with English only low acoustic similarity can be observed. However the sequences of words, at least partly, in the hypotheses are more often plausible. The utterance "dafür haben wir natürlich jetzt" is recognized as "if you have one or two leaktes" by the lower-case monolingual German system, for example.

The Denglish data clearly benefits from the bilingual model. As to see in Table 10 the WER decreases compared to the monolingual German model and the word accuracy increases. The word accuracy is computed considering only the English and germanized English words.

| Language | Model | Bpe | WER lc | Word accuracy |
|---|---|---|---|---|
| Denglish | Transformer | ge | 30.5% | 25.1% |
| Denglish | Transformer | mix | 25.8% | 44.4% |

Table 10: Baseline Transformer for Denglish

As expected especially the Denglish words are recognized better, but the word accuracy is still low. Even if many words have a wrong transcription, there is more acoustic similarity. For example "awarness", by the monolingual system recognized as "während ist", is recognized as "werness" by the bilingual system. The monolingual system only recognizes commonly used english words as "system", "research", "feedback" or "statement". Both systems have problems with English words with changed form due to germanization as "gecovert", "gecheckt" or "abgelost".

### 4.2.3 Gating Transformer

As a first language adaption method we evaluate gating as explained in Chapter 3.5. In [6] the gating method applied after every layer of an architecture consisting of stacked LSTM layers achieved good results.

In the original paper the gating algorithm takes the hidden state of all time steps, concatenates the language identity one-hot vector and multiplies the resulting vector by a weight matrix. The weight matrix has one language dependent column per language. The columns corresponding to other languages are not taken into account, because of

the multiplication by zero values in the one-hot vector.
The values of the resulting vector are then transferred to $(0, 1)$ by the sigmoid function. As an additional step we added a layer normalization before applying the sigmoid function. Afterwards the gated hidden state is again concatenated with the LID vector, down-projected and inputted into the next layer.

Since we have two languages, English and German, our language identity (LID) one-hot vector is of size two. LID 0 is allocated to German, the corresponding LID vector is given by $d_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$. For English with LID 1 we analogously get $d_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$.

We experimented with the Transformer using gating after every layer in encoder and decoder (all layer), since this worked best in the original paper. This was compared to the effect of gating only in the decoder after every layer (all dec layer). Since the decoder processes the output tokens, we hope the network learns language modeling dependent on LID. Here we once used shared key and value weight matrices in both MHA layers (all dec layer (shared kv)). The latter model has almost the same number of learnable parameters as the baseline Transformer.
As a last experiment we only applied gating after the last layer of the encoder and after the first layer of decoder (last enc + 1. dec).
The number of parameters increases due to gating as shown in Table 11. The column "Model" states which method was applied to Transformer, e.g. "all layer" denotes the Transformer with gating after each layer. See Table 6 for comparison with the number of parameters in the Transformer without language adaption.

| Model | Number of languages | Number of parameters | Parameter increase |
|---|---|---|---|
| all layer | 2 | 37,860,771 | 20.09% |
| all dec layer | 2 | 33,637,795 | 6.7% |
| all dec layer (shared kv) | 2 | 31,540,643 | 0.05% |
| last enc + 1. dec | 2 | 32,582,051 | 3.35% |

Table 11: Parameter increase due to gating in the Transformer

Table 12 shows the WERs for German and English for the Transformer with gating. The language identity used during testing is denoted by "LID". The relative decrease of WER compared to the multilingual baseline model (see Table 8) is shown in the last column. A negative sign actually states an increase, a worsening of the recognition.

| Language | Model | Bpe | LID | WER mc | WER mc decrease | WER lc | WER lc decrease |
|---|---|---|---|---|---|---|---|
| English | all layer | mix | en | 16.1% | −26.8% | 15.8% | −35.0% |
| German | all layer | mix | ge | 21.1% | −1.9% | 23.4% | −9.5% |
| English | all dec layer | mix | en | 12.7% | 0.0% | 13.8% | −17.9% |
| German | all dec layer | mix | ge | 18.8% | 9.2% | 21.4% | −1.4% |
| English | all dec layer (shared kv) | mix | en | 12.8% | −0.8% | - | - |
| German | all dec layer (shared kv) | mix | ge | 18.4% | 11.1% | - | - |
| English | last enc + 1. dec | mix | en | 12.3% | 3.1% | - | - |
| German | last enc + 1. dec | mix | ge | 20.5% | 1.0% | - | - |

Table 12: Results for the Transformer with gating

Obviously gating has no positive effect on the recognition rate. Especially gating after each layer is clearly worse than the baseline for both mixed- and lower-case data.
Instead of enabling adaptation to one language and thus improving results, gating seems to make recognition harder. Maybe the gating algorithm, developed for LSTM and suitable to the gates in a LSTM cell, is not compatible to the Transformer's architecture.
Another observation we could make is that the learned features of the monolingual system range in a more or less constant scope among all layers. In the system with gating the value ranges get wider for subsequent layers. After the last layer of encoder, for example, the values are often ten times higher than for outputs of the first layer. Maybe these high differences also lead to problems.

Gating after every decoder layer doesn't seem to impact the results negatively for the mixed-case model. For German the WER even decreased. In case of the lower-case model on the other hand the WERs increase for both tested languages.
Eventually this is only due to the difference in data, but since the baseline results are similar for mixed- and lower-case data this seems unlikely. Another explanation might be that the language modeling differs, because there is no capitalization for German anymore. Since the results of both models with gating after every decoder layer differ a lot and capitalization is probably learned in the decoder this might influence each other. Without capitalized words, nouns can not always be clearly identified in German, this could complicate the prediction of subsequent words. Especially because German is very flexible in sentence formation. However, the effect could also be conditioned by training.

The WERs for gating after the last encoder layer and the first decoder layer are comparable to the bilingual baseline and the model was not trained with the lower-case data for this reason. Although it could be interesting for further comparison of lower-case and mixed-case data and could be investigated in the future.

When testing the Transformer with gating using the wrong LID we achieve the WERs shown in Table 13. For comparison with the monolingual systems tested with the respective other language see Table 9.

| Language | Model | Bpe | LID | WER mc | WER lc |
|---|---|---|---|---|---|
| English | all layer | mix | ge | - | 79.7% |
| German | all layer | mix | en | - | 126.0% |
| English | all dec layer | mix | ge | 55.3% | 24.8% |
| German | all dec layer | mix | en | 99.2% | 94.4% |
| English | last enc + 1. dec | mix | ge | 26.6% | - |
| German | last enc + 1. dec | mix | en | 46.6% | - |

Table 13: Results for the Tranformer with gating with wrong LID

The monolingual system tested with the wrong language and the multilingual system tested with wrong LID yield both high WERs. This indicates that gating in the Transformer actually leads to language adaption of the given LID.

The system with gating after each layer for English with German LID, recognizes a lot of words correctly in English, partly even complete phrases. In other cases not even existing German words are predicted. For example the utterance "but what he effectively did for me was reshape an awful daily occurrence into a new and promising experience for me" is recognized as "bei wochee effekte did for me was rechee in orthor derry aquas into a new and promising experience for me".
Decoding German with English LID on the other hand also results in English hypotheses for almost all utterances. However nonexistent words are predicted, as well.
Even if the WERs are similar in monolingual and bilingual systems, the predictions appear to be better when using gating, at least considering acoustic similarity. In Table 14 the hypotheses, made by the monolingual English system and the Transformer with gating after every layer with English LID, are compared for a German utterance.

| Model | Utterance |
|---|---|
| Transformer | on in this examined and darkened and i lay in tactuan |
| all layer | in in diesen gesammothendaten syndon aletent action |

Table 14: Hypotheses made by the Transformer without language adaption and the Transformer with gating after every layer with English LID for the utterance "und in diesen gesammelten daten sind nun alle interaktionen"

The mixed-case model with gating after each decoder layer and German id doesn't predict anything for many English utterances. Comparing the results of mixed- and lower-case models, this is probably the main difference. In cases of available predictions, those are correct in large parts and furthermore comparable to the lower-case model.
When using the English id to recognize German we also obtain English predictions. Besides the systems predict many nonexistent, and only few German words. An example is shown in Table 15.

| Data | Utterance |
|---|---|
| mc | they've been in the next eye and googlees noise yeah wünschen |
| lc | it stuff indones and goodes neues ya wünschen |

Table 15: Hypotheses made by the Transformer with gating after every decoder layer with English LID, trained with mixed-case or lower-case data, for the utterance "ich darf ihnen zunächst ein gutes neues jahr wünschen"

The example utterance shown in Table 14 is recognized almost completely correct by the Transformer with gating in the decoder.

We also tested the model with gating after the last encoder layer and after the first decoder layer. For English with German LID no prediction is made for complete utterances. In summary the recognition rate with wrong LID is worse than with correct LID, but the differences are small. Furthermore language adaption seems to work, eg. the English reference "hurt" is recognized as "hört" or the German reference "ist ja auch" is predicted as "this is our". Again an increased number of nonexistent words can be recognized.

At last we tested the Transformer with gating after each layer using the Denglish data set. The results are consistent with the results obtained for German regarding the increase in WER. Especially, but not exclusively, the recognition of English or Denglish words is worse compared to the bilingual baseline. The word accuracy of the English words in the test set is almost as low as for the monolingual German system. For comparison see Table 10.
The Denglish words, predicted correctly, are also to a large extent the same. The hypotheses of English words often have a certain similarity to existing German words or are transcribed as pronounced in German. For example "gecrasht" becomes "gegräscht", "managen" becomes "menschen". The bilingual baseline model predicts "ge crashed" and "manage".
The WERs are compared to the result of the bilingual baseline model, shown in Table 10.

| Language | Model | Bpe | LID | WER lc | WER lc decrease | Word accuracy |
|---|---|---|---|---|---|---|
| Denglish | all layer | mix | ge | 35.5% | −37.6% | 25.7% |
| Denglish | all layer | mix | ge + en | 52.4% | −103.1% | 35.7% |

Table 16: Results for the Transformer with gating for Denglish

Instead of using a one-hot vector representing only one language, we also inserted the mixed language vector $d_{0,1} = \begin{bmatrix} 1 & 1 \end{bmatrix}$ representing English and German. This results in hypotheses mixed with English and German words. For the reference "wie ist das nightlife in hamburg", the system predicts "these does neid life in hamburg". Another utterance "ich war beim zahnarzt und habe mir die zähne bleachen lassen" is recognized as "ich war beim arzt und habe mir die center beach in lesson". However, the Denglish

words benefit and the word accuracy increases, but is stil lower compared to the bilingual baseline.

In general we do not obtain improvements with gating in the Transformer. Roughly speaking the more gating layers the worse the results. Nevertheless an adaption to the indicated language seems to be happening. The WERs, when using the wrong LID, are clearly higher compared to the correct LID. Furthermore the word accuracy of the Denglish words obtained by the Transformer with gating using the mixed language vector is clearly higher than for the model adapted to German.

### 4.2.4 Language Adaption in MHA for Transformer

Another approach we evaluated is language adaption in the Multi-Head Attention layer. In [7] two techniques were applied to the audio encoder of a RNN-Transducer architecture.

Language embedding concatenation (LEC), was explained in Chapter 3.6. The one-hot vector, describing the LID, is linearly transformed and added to the key, query and value in the MHA layer. Attention was explained in Chapter 2.7.3. We experimented with LEC in every encoder layer (Enc LEC) and in every MHA layer of the encoder and decoder (T LEC).

Language Specific Attention Heads (LSAH), explained in Chapter 3.7, was applied to every encoder layer (Enc LSAH) and to every encoder and decoder layer (T LSAH). We use one language specific head per language and eight heads in total as before. For each language we thus use a combination of seven shared and one language specific head.

The total numbers of learnable parameters of the networks are summarized in Table 17. Overall the impact on the numbers of parameters of both methods is quite small. See Table 6 for comparison with the number of parameters in the Transformer without language adaption techniques.

| Model | Number of languages | Number of parameters | Parameter increase |
|---|---|---|---|
| T LEC | 2 | 31,600,035 | 0.23% |
| Enc LEC | 2 | 31,563,171 | 0.12% |
| T LSAH | 2 | 31,002,019 | −1.66% |
| Enc LSAH | 2 | 31,264,163 | −0.83% |

Table 17: Parameter increase due to language specific attention layers in Transfromer

Table 18 shows the WERs obtained for English and German. The WERs are compared to the WERs obtained by the bilingual baseline, shown in Table 8.

| Language | Model | Bpe | LID | WER lc | WER lc decrese |
|---|---|---|---|---|---|
| English | T LEC | mix | en | 12.8% | −9.4% |
| German | T LEC | mix | ge | 21.4% | −1.4% |
| English | Enc LEC | mix | en | 12.3% | −5.1% |
| German | Enc LEC | mix | ge | 20.6% | 2.4% |
| English | T LSAH | mix | en | 13.2% | −12.8% |
| German | T LSAH | mix | ge | 22.7% | −7.6% |
| English | Enc LSAH | mix | en | 13.1% | −11.9% |
| German | Enc LSAH | mix | ge | 22.7% | −7.6% |

Table 18: Results for the Transformer with language adaption in the MHA layers

All methods yield worse results compared to the bilingual baseline. Interestingly applying LEC and LSAH only in the encoder results in lower WERs than applying them in encoder and decoder. However especially for LSAH the differences are very small and could also be conditioned by training issues.

Since both methods yield best results when applying them in the encoder only, we restrict the tests with wrong LID on these two methods. The results are shown in Table 19.

| Language | Model | Bpe | LID | WER lc |
|---|---|---|---|---|
| English | Enc LEC | mix | ge | 48.4% |
| German | Enc LEC | mix | en | 100.9% |
| English | Enc LSAH | mix | ge | 55.3% |
| German | Enc LSAH | mix | en | 110.0% |

Table 19: Results for the Transformer with language adaption in MHA layer with wrong LID

Both systems predict hypotheses mixed with English and German words. Even though the main parts are in English. An example utterance is shown in Table 20 for English and in Table 21 for German.

| Model | Utterance |
|---|---|
| Enc LSAH | weil wochee effectively did for me was reshape in oreford derry accounts [...] |
| Enc LEC | but what he effectively did for me was reshape in ohr vor daily accounts [...] |

Table 20: Hypotheses made by "Enc LEC" and "Enc LSAH" with German LID for the utterance "but what he effectively did for me was reshape an awful daily occurrence [...] "

| Model | Utterance |
|---|---|
| Enc LSAH | in einer besprechung by the idea the underneath gestured werden zone |
| Enc LEC | in einer besprechung by the idea underneath gestured we hadn't owned |

Table 21: Hypotheses made by "Enc LEC" and "Enc LSAH" with English LID for the utterance "in einer besprechung bei der die anderen nicht gestört werden sollen"

For Denglish the results are shown in Table 22. The WERs are higher than obtained for the bilingual baseline. The column "WER lc decrease" shows the relative increase in WER, compared to the results in Table 10. Not only the overall WER is higher, the word accuracy of the Denglish words is lower compared to the bilingual Transformer, although the differences are small. However, all four models with LEC and LSAH outperform the monolingual German Transformer.

| Language | Model | Bpe | LID | WER lc | WER lc decrese | Word accuracy |
|---|---|---|---|---|---|---|
| Denglish | T LEC | mix | ge | 28.1% | −8.9% | 40.4% |
| Denglish | Enc LEC | mix | ge | 28.4% | −10.1% | 39.8% |
| Denglish | T LSAH | mix | ge | 27.5% | −6.6% | 39.2% |
| Denglish | Enc LSAH | mix | ge | 26.8% | −3.9% | 40.9% |
| Denglish | T LEC | mix | en | 87.5% | - | 39.2% |
| Denglish | Enc LEC | mix | en | 90.0% | - | 38.6% |
| Denglish | T LSAH | mix | en | 101.6% | - | 32.7% |
| Denglish | Enc LSAH | mix | en | 92.0% | - | 37.4% |

Table 22: Results for the Transformer with language adaption in MHA layer for Denglish

Compared to the bilingual baseline the Transformer with LEC in the encoder adapted to German has more problems to recognize English and Denglish words. For example, "style sheets", recognized correctly by the bilingual baseline, becomes "stahl shieds". Also German words are confused, e.g. "fürs nächste" is predicted as "für snakeste" by the "Enc LEC"-model.
Using LEC in encoder and decoder with German id seems to enable the model to be more flexible regarding acoustic similarity. For example "fischen" is predicted instead of "fashion". The bilingual baseline predicts "feschen". In contrast to the example given above, the Transformer with LEC in encoder and decoder does not always predict existent words.

In Table 23 hypotheses, made by the "Enc LEC"-model and the "T LEC"-model adapted to English, are shown. Interestingly the language itself is better when using language adaption in the encoder only.

| Model | Utterance |
|---|---|
| Enc LEC | i've made out to male einen remind office next to meeting |
| T LEC | tried madarzumal and reminded for snakes to meeting |

Table 23: Hypotheses made by "Enc LEC" and "T LEC" with English LID for the Denglish utterance "schreib mir dazu mal einen reminder fürs nächste meeting"

The hypotheses made by the Transformer with LSAHs in the encoder using German id are in general comparable to the bilingual baseline. However, more nonexistent words are predicted, in some cases with less acoustic similarity, as "sportelneck" instead of "bottleneck" or "bischofs" for "visuals". Comparing the results of LSAH in encoder with LSAH in encoder and decoder we don't notice general differences. Even if the WER of the "T LSAH"-model is higher overall, the predictions are better in some cases. For example "visuals" is recognized correctly and "bottleneck" becomes "bordelneck". In both models the occurring errors are concentrated on the Denglish and English words.

The Transformer with LSAHs in the encoder and decoder with English id yields a decreased word accuracy compared to the "Enc LSAH"-model. Even English words as "editor", "scoring" or "production", that are recognized by the other models adapted to English, are not predicted correctly.
However the English head in the MHA layers actually seems to lead to an adaption to English. The hypotheses made by the "T LSAH" and "Enc LSAH"- systems are in English in large parts.

In general LSAHs and LEC worsen the performance of the Transformer. When using LSAHs an increased number of heads could be investigated to increase the number of learnable parameters again.
In [7] both methods were applied to the Transformer's encoder in a RNN-Transducer model, which can be used for streaming ASR. To enable streaming relative positional embedding [49] was used instead of adding positional encoding as we do. This could also be investigated in the future.

### 4.2.5 Conclusion

We evaluated language adaption in the Transformer by a gating mechanism and language specific MHA layers. Both of these approaches did not lead to improvements over the baseline without language adaption.
The gating layers were applied after every layer of the encoder and decoder, only after the decoder layers or after the last encoder layer and the first decoder layer. The latter model was only trained with mixed-case data, but outperformed the bilingual baseline slightly. Due to the small differences in WER over the baseline, we did not train the model with lower-case data.
For the Transformer with gating in the decoder we obtained different results when training with mixed-case or lower-case data. One possible explanation could be the different

German data for training. However large parts of the data were equal and the baseline results are comparable for both setups. The differences could also be conditioned by training. Further investigation, if this behavior can be lead back to the missing capitalization in the German data or is due to the experimental setup, is necessary.
For language adaption in the MHA layers two methods, LEC and LSAHs, were investigated. We compared using both methods only in the encoder and in both, encoder and decoder. The WERs obtained for all methods are similar and couldn't improve the recognition rate for English, German or Denglish.
However, all investigated methods seem to lead to language adaption, as the high WERs suggest, when testing with the wrong LID. The changes in the word accuracy of the Denglish words support this impression.
Nevertheless the Transformer, simply trained with mixed language data yields the best results in the bilingual setups. It even achieves similar WERs as the monolingual systems. English and German are related languages, multiple words exist in both languages or at least have a similar pronunciation and transcription. Those similarities could help the network when it's trained jointly.
A general observation we could make when testing with the wrong LID is that the WERs are lower for English than for German. English systems or English adapted systems predict more words in general as we noticed. This is probably one reason for the high WERs obtained for German. However, also the hypotheses are better. One explanation could be that in German English words are used in daily speech, which could help the model predicting English even when German LID is given.

For all systems the recognition of Denglish is a problem. Words that are commonly used in both languages as "computer" or "research" are often recognized, but also included in the German training data. English words, that are rarely used in daily speech or that are germanized, have a low recognition rate. Especially verbs with changed form as "gecovert" or "agbelost" are problematic.

## 4.3 LSTM based encoder-decoder seq2seq model

The second evaluated seq2seq model is an encoder-decoder architecture with stacked LSTM layers as explained in Chapter 2.7.2. We first outline the parameters of the architecture in Chapter 4.3.1. In Chapters 4.3.2 - 4.3.7 the results for the baseline and for different language adaption methods are presented.

### 4.3.1 Parameter LSTM

Table 24 summarizes the parameters used for the LSTM based encoder-decoder model. The number of encoder layers is denoted by $N_{Enc}$, the number of decoder layers is $N_{Dec}$. The input feature vectors have a length of $d_{input}$ and the number of features in the hidden states is equal to $d_{model}$.
We use $n_{head}$ heads for the MHA layer with key, query and value of size $d_k$.

The number of tokens in our vocabulary is $n_{vocab}$ for monolingual and multilingual models, if nothing else is stated.

| | |
|---|---|
| $N_{Enc}$ | 6 |
| $N_{Dec}$ | 2 |
| dropout | 0.2 |
| $d_{input}$ | 40 |
| $d_{model}$ | 1024 |
| $d_k$ | 128 |
| $n_{head}$ | 8 |
| $n_{vocab}$ | 4003 |

Table 24: Parameters in the LSTM encoder-decoder architecture

Table 25 shows the number of parameters for our baseline. When speaking of "LSTM" as model we always mean the encoder-decoder architecture consisting of LSTM layers and not a single LSTM cell. We used two different methods of combining the outputs of the biLSTM layers in the encoder in the experiments. The first is simply concatenating the outputs. In some cases we add the outputs of the two directions element-wise. To distinguish between both of them we use "LSTM" for the model with concatenated outputs and "LSTM with addition" to refer to the model with element-wise addition of the bidirectional outputs.

| Model | Combination of biLSTM output | Number of parameters |
|---|---|---|
| LSTM | concatenation | 161,777,923 |
| LSTM with addition | addition | 119,834,883 |

Table 25: Number of learnable parameter in the LSTM ecoder-decoder architecture

### 4.3.2 Baseline LSTM

We first trained monolingual systems for English and German. Then we mixed the language data and trained a bilingual system. The results are presented in Table 26. The structure of the tables used in this section is equivalent to those in Chapter 4.2.2 for the Transformer. "LSTM" as model denotes the LSTM based encoder-decoder model with concatenation of the outputs of the biLSTM cells in the encoder.

| Language | Model | Bpe | WER mc | WER lc |
|---|---|---|---|---|
| English | LSTM | en | 10.4% | 10.4% |
| German | LSTM | ge | 19.6% | 17.7% |
| English | LSTM | mix | 9.7% | 9.0% |
| German | LSTM | mix | 18.2% | 18.5% |

Table 26: Baseline LSTM with concatenation of the bidirectional outputs of biLSTM cells in encoder

For German the monolingual system trained with lower-case data yields better results than the model trained with mixed-case data. Here we see very big differences comparing the WERs of the different speakers of our test set. While the recognition rate improves for three of eight speakers using lower-case data, it decreases for the other five of them.
In the multilingual setup using lower-case data, results for English improve, while for German they are worse. This is the same behavior as already seen for the Transformer. When training the multilingual model with mixed-case data both languages benefit.

Testing the monolingual systems with the respective other language gives very high WERs as shown in Table 27. In general the monolingual systems predict words in the language they were trained with. However both systems also create nonexistent words. For instance the German system predicts "ja bepasste bürger die auch ein ende video auch das ziel im servus ist key gebraucht" for the utterance "the possibility of an individual to see themselves as capable".

| Language | Model | Bpe | WER lc |
|---|---|---|---|
| English | LSTM | ge | 98.4% |
| German | LSTM | en | 122.6% |

Table 27: Baseline LSTM tested with wrong language

We added French as third language, to see if the multilingual models benefit using more languages. Table 28 shows the WERs of the baseline for the trilingual model. To distinguish between the bi- and trilingual models we use "mix+" to denote the model trained with English, German and French data.

| Language | Model | Bpe | WER lc |
|---|---|---|---|
| French | LSTM | fr | 21.9% |
| English | LSTM | mix+ | 9.0% |
| German | LSTM | mix+ | 18.8% |
| French | LSTM | mix+ | 21.3% |

Table 28: Baseline LSTM for English, German and French

One general problem for French is that apostrophes are often counted as error, even when they were recognized correctly. Comparing the monolingual system with the multilingual, English and French benefit, the WER for German is even slightly worse compared to the bilingual model and clearly worse than the WER obtained using the monolingual German system.

Table 29 shows the WERs for the monolingual and multilingual baseline models for Denglish. As expected the monolingual German system has problems to recognize the English and Denglish words correctly. In the mixed-case model they are usually replaced by German words, e.g. "outsourcen" becomes "Autobahn" or "higlighten" becomes "heiraten". Even if these were not the spoken words, they do exist.

In the lower-case model on the other hand the predictions have a higher acoustic similarity as in the mixed-case model. For example "outsourcen" becomes "autsourcen" and "highleiten" becomes "heileiten", which both don't exist in German, but sound as the spoken words. Words, that are used commonly in German as "meeting", "office", "googeln" or "joggen" are recognized correctly. Most of these words also exist in the German training data.
Eventually training with the lower-case data complicates learning of language modeling, which would make predictions of subsequent words in a sequence harder. German language is very flexible. To learn the relation between words in a sentence, capital letters denoting nouns, might be useful. On the other hand transcription and pronunciation are closely related. Maybe the network learns to rely on acoustic more when training with lower-case data.
The word accuracy is computed for the English and germanized words in the utterances for the model trained with lower-case data. Here the multilingual models clearly outperform the monolingual German and English models. Furthermore the trilingual model yields a better WER and word accuracy than the bilingual model.

| Language | Model | Bpe | WER mc | WER lc | Word accuracy |
|---|---|---|---|---|---|
| Denglish | LSTM | ge | 37.5% | 27.2% | 29.8% |
| Denglish | LSTM | mix | 32.9% | 22.9% | 44.4% |
| Denglish | LSTM | mix+ | - | 21.8% | 46.2% |
| Denglish | LSTM | en | - | 109.1% | 31.6% |

Table 29: Baseline LSTM for Denglish

In the bilingual lower-case model the recognition is much better compared to the German monolingual system. Especially English, but also germanized English words, benefit. In the examples given above "outsourcen" is recognized correctly and "highlighten" becomes "highleiten". Many mistakes are only small or caused by a wrong separation of compound words. However, the model doesn't recognize germanized words as "approven" or "gecovert" correctly. We can not observe general differences in the hypotheses of the bilingual and trilingual models.
The bilingual mixed-case system also has problems with English words, but often hypotheses and references have a similar pronunciation, for example "outsourcen" becomes "audsausen" and "highlighten" becomes "heileiten" in the multilingual system.

Testing the English model with Denglish naturally yields a high WER, however, at least English words are often predicted correctly. The word accuracy among them is higher compared to the German monolingual system. Germanized words as "googlen" or "twittern" are not recognized.

When using element-wise addition of the outputs of the biLSTM cells we obtain the WERs shown in Table 30 for the bilingual model. Table 31 shows the results of the trilingual model. This model is denoted by "LSTM with addition". Overall the WERs of both models are similar to those obtained when concatenating the outputs of the

biLSTM cells, but for the single languages they differ. The word accuracy is again only computed for the English and Denglish words in the Denglish data set.

| Language | Model | Bpe | WER lc | Word accuracy |
|---|---|---|---|---|
| English | LSTM with addition | mix | 10.3% | - |
| German | LSTM with addition | mix | 16.9% | - |
| Denglish | LSTM with addition | mix | 23.7% | 41.5% |

Table 30: Baseline LSTM with element-wise addition of bidirectional outputs of biLSTM cells in encoder

| Language | model | bpe | WER lc | word accuracy |
|---|---|---|---|---|
| English | LSTM with addition | mix+ | 9.3% | - |
| German | LSTM with addition | mix+ | 18.3% | - |
| French | LSTM with addition | mix+ | 20.3% | - |
| Denglish | LSTM with addition | mix+ | 22.9% | 46.2% |

Table 31: Baseline trilingual LSTM with element-wise addition of bidirectional outputs of biLSTM cells in encoder

### 4.3.3 Gating LSTM

As a first language adaption method gating [6], as explained in Chapter 3.5, was applied to the LSTM based encoder-decoder model. We compared gating of the hidden states after each layer (all layer) and only after each decoder layer (all dec layer).
As for the Transformer we added a layer normalization before applying the sigmoid function.
The encoder in our architecture is composed of bidirectional LSTM layers, that always output the hidden states for both directions. In the encoder of the "all layer"-model we apply gating before combining the outputs of both directions by element-wise addition. When using gating only in the decoder we concatenate the bidirectional outputs of the LSTM layers in the encoder.
The total number of learnable parameters of the networks are shown in Table 32. For the "all dec layer"-model we compare the numbers of parameters to the baseline model "LSTM". The "all layer"-model is compared to the baseline "LSTM with addition". The number of parameters of the baseline models can be found in Table 25.

| Model | Number of languages | Number of parameters | Parameter increase |
|---|---|---|---|
| all dec layer | 2 | 165,986,563 | 2.6% |
| all layer | 2 | 174,461,187 | 45.6% |
| all layer | 3 | 174,489,859 | 45.6% |

Table 32: Parameter increase due to gating in the LSTM based encoder-decoder architecture

Table 33 shows the WERs for the bilingual LSTM encoder-decoder architecture with gating for English and German. The WER decrease using gating after every layer is computed for the training setup with lower-case data. We compare it to the WER in Table 30, obtained by the bilingual model without language adaption. For the model with gating only in the decoder we used mixed-case data in training and the WER was compared to the baseline results (WER mc) in Table 26.

| Language | Model | Bpe | LID | WER mc | WER lc | WER decrease |
|---|---|---|---|---|---|---|
| English | all layer | mix | en | 8.4% | 8.4% | 18.4% |
| German | all layer | mix | ge | 17.3% | 17.3% | −2.4% |
| English | all dec layer | mix | en | 9.5% | - | 2.1% |
| German | all dec layer | mix | ge | 18.9% | - | −3.8% |

Table 33: Results for the LSTM based encoder-decoder architecture with gating

In contrast to the Transformer, gating has a positive effect on the WERs of the LSTM based encoder-decoder architecture. The gating algorithm has a similar structure as the LSTM cells, probably gating integrates better into the architecture.
The highest improvements are achieved with gating after each layer. As for the Transformer we first experimented with mixed-case data. On average the WER decreases by 5.8% over the bilingual systems without language adaption and by 15.5% compared to the monolingual systems in Table 26. However, since we did not train a baseline model with element-wise addition with mixed-case data, the results are not totally comparable.
On the system trained with lower-case data, the WER improve by an average of 8% compared to the bilingual baseline and by 10.7% compared to the monolingual system with concatenation of the bidirectional outputs in the biLSTM cells.

Compared to the English monolingual system the "all layer"-model testing English with English LID more often predicts small words, e.g. "a"'s, that were missing in the hypotheses of the baseline. Besides, mistakes like omitting single letters, e.g. "d" in "and", occur less frequently. Many hypotheses with a high acoustic similarity to the reference, but with wrong transcription, without language adaption, are recognized correctly using gating.

Gating of the hidden states of the decoder layers only, didn't lead to improvements. Training a model with gating after each encoder layer could be investigated in future for better comparison. Since gating after each decoder layer gave no improvements, it was not repeated with lower-case data.

Testing with the wrong language ID results in WERs, that are almost as high as in the monolingual systems tested with a different language. The results are shown in Table 34. This supports the impression that language adaption works well.

| Language | Model | Bpe | LID | WER lc |
|---|---|---|---|---|
| English | all layer | mix | ge | 89.7% |
| German | all layer | mix | en | 115.1% |

Table 34: Results for the LSTM based encoder-decoder architecture with gating with wrong LID

The LSTM based encoder-decoder model with gating tested with wrong LID often predicts English words independent of LID. In German often English words are used in daily speech. As a result this could eventually lead to the observed behavior.

The results for Denglish are shown in Table 35. Compared to the results obtained by the German monolingual system, see Table 30 for comparison, the WER decreases and the word accuracy increases by 11.3%.

| Language | Model | Bpe | LID | WER lc | Word accuracy |
|---|---|---|---|---|---|
| Denglish | all layer | mix | ge | 23.1% | 46.2% |
| Denglish | all layer | mix | en | 102.4% | 41.5% |
| Denglish | all layer | mix | ge + en | 29.0% | 48.5% |

Table 35: Results for the LSTM encoder-decoder architecture with gating for Denglisch

Especially the Denglish expressions benefit using gating compared to the monolingual system. In some cases, mostly English or Denglish words, are replaced by nonexistent words, e.g "online" becomes "oilline", a word that was recognized correctly by the bilingual baseline model.
Although the model is adapted to German, the system seems to take advantage of the bilingual data. For example "highlighten" is recognized as "hileiten" by the model with gating. It's neither a correct German nor English word, but "heileiten" as recognized by the monolingual German system is closer to the acoustic of "highlighten" than German pronunciation of "hileiten". Adapting to German enables the system to also recognize germanized words as "recyceln" or "abgelost", even when this kind of words is still most problematic.
Using the English LID during inference the made predictions are in English mostly. However, in contrast to the monolingual English system, nonexistent words are predicted, too. For example "der unternehmer" is recognized as "the undernema". Especially germanized words are not recognized.
With mixed English and German LID many utterances are recognized correctly. In some cases English phrases are predicted. For example for "hast du die best practice[...]" the system predicts "as to the best practice[...]". Alternatively the system hypothesizes nonexistent words as "imphirm" for "dem film". While the recognition of English words is better, the Denglish expressions are often predicted wrongly. Here adapting to German is more beneficial.

Table 36 shows the WERs for the trilingual model with gating after each layer, trained

with English, German and French data. The WER is compared to the results of the trilingual model without language adaption in Table 31.

| Language | Model | Bpe | LID | WER lc | WER lc decrease |
|---|---|---|---|---|---|
| English | all layer | mix+ | en | 8.9% | 4.3% |
| German | all layer | mix+ | ge | 17.1% | 6.6% |
| French | all layer | mix+ | fr | 17.7% | 12.8% |

Table 36: Results for the trilingual LSTM encoder-decoder architecture with gating

The WER obtained for English is higher compared to the bilingual model with gating. However, the system outperforms the multilingual baseline models (see Table 26 and Table 28 for comparison).
German, on the contrary, benefits from the additional language, also compared to the bilingual model with gating the WER decreases.
Compared to the monolingual models we achieve an average decrease in WER of 12.3% and compared to the trilingual baseline model we gain 7.9% relative improvement.

We also tested the trilingual model with gating after every layer with the wrong LIDs. Table 37 shows the WERs.

| Language | Model | Bpe | LID | WER lc |
|---|---|---|---|---|
| English | all layer | mix+ | ge | 77.4% |
| English | all layer | mix+ | fr | 85.1% |
| German | all layer | mix+ | en | 116.5% |
| German | all layer | mix+ | fr | 104.2% |
| French | all layer | mix+ | en | 111.3% |
| French | all layer | mix+ | ge | 98.2% |

Table 37: Results for the trilingual LSTM encoder-decoder architecture with gating with wrong LID

In general the hypotheses are most often in the language of the given LID. However, especially when testing with German LID, often not existing words are predicted. Using the English id the system often predicts a lot more words than available in the references, i.e. 7.5% additional words for French and even 14.3% for German. This explains the high WERs for English LID. Nevertheless, even when taking this into consideration the recognition seems to be better for German with French id and vice versa. For German decoded with French id we achieve 5.5% word accuracy, compared to 4.7% with English id. For French the difference is even higher with 5.4% word accuracy obtained for German id, compared to 3.0% word accuracy with English id. Since German and English are related languages, we expected the performance for German to be better with English LID than with French LID.

For Denglish the results are shown in Table 38. Testing with German id outperforms

the bilingual model with gating. We achieve 19.9% relative improvement compared to the monolingual German system and 4.8% compared to the trilingual baseline model. See Table 31 for comparison. The word accuracy of the system adapted to German and the trilingual baseline are equal, but using the mixed language vector the word accuracy increases further.

| Language | Model | Bpe | LID | WER lc | Word accuracy |
|---|---|---|---|---|---|
| Denglish | all layer | mix+ | ge | 21.8% | 46.2% |
| Denglish | all layer | mix+ | en | 103.5% | 40.3% |
| Denglish | all layer | mix+ | fr | 96.2% | 22.2% |
| Denglish | all layer | mix+ | ge + en | 32.7% | 47.4% |

Table 38: Results for the trilingual LSTM encoder-decoder architecture with gating for Denglish

Adapting the system to English leads to mostly English predictions. However, the system also creates nonexistent words. With French, on the contrary, a lot of nonexistent words are predicted. An example is shown in Table 39.

| LID | Utterance |
|---|---|
| en | [...] emback and to stay in integration on donefully automatising |
| fr | [...] une beckhinthestine integrienne on dan fallautomatisienne |

Table 39: Hypotheses made by the "all layer"-model with English and French id for the utterance "[...] die backend systeme integrieren und dann voll automatisieren"

The word accuracy considering only English and Denglish words is clearly higher using the English LID than inserting the French id during inference. This is as expected, since the Denglish data consists of English or English-based words in a large part. The French system only recognizes English, but no Denglish words. Nevertheless the system adapted to French seems to benefit of the German and English data, since not all of the correctly predicted words showed up in the French training data.

Independent of LID, the trilingual model has again, in particular, problems recognizing the Denglish words. Using the mixed LID vector representing English and German, words as "stylische", abgelost" or "committen" are recognized, in contrast to the system adapted to only one language. The overall WER increases, but using the mixed LID vector leads to a better recognition of the Denglish words.

All in all the results suggest that gating based on the LID enables the model to adapt to the given language and leads to a reduced WER. In addition the recognition is further improved by adding another language. This was also observed in [6].
Also using gating, the recognition rate of Denglish words is low and furthermore comparable to the multilingual baseline models. However, the mixed language vector, representing English and German, provides improvements in the word accuracy.

### 4.3.4 Gating of Output Embedding

We applied gating after the embedding of the decoder's input before it is fed into the first LSTM layer. In this way we hope the model learns language specific representation of the text output. To refer to the LSTM based encoder-decoder model with gating of the decoder's embedding we use "dec emb".
We experiment with English and German and use a vocabulary size of 4000 as before in a first step. In this vocabulary many tokens can be uniquely assigned to one language. Later we reduced the vocabulary size to create a vocabulary with more shared tokens. A vocabulary size of 300 was chosen, because no tokens could be clearly associated with a language.

We concatenate the bidirectional outputs of biLSTM cells in the encoder. The numbers of trainable parameters are shown in Table 40 for the model with gating of the output's embedding and compared to the baseline "LSTM" in Table 25.

| Model | Bpe size | Number of languages | Number of parameters | Parameter increase |
|---|---|---|---|---|
| dec emb | 4,000 | 2 | 163,882,243 | 1.3% |
| dec emb | 300 | 2 | 160,089,743 | −1.0% |

Table 40: Parameter increase due to gating of decoder's embedding in the LSTM encoder-decoder architecture

Table 41 shows the WERs for the model trained with the large vocabulary of size 4,000. The WERs are compared to the WERs of the bilingual baseline in Table 26. The negative sign means that the WER increases, the performance worsens.

| Language | Model | Bpe | LID | WER lc | WER lc decrease |
|---|---|---|---|---|---|
| English | dec emb | mix | en | 11.0% | −22.2% |
| German | dec emb | mix | ge | 19.9% | −7.6% |

Table 41: Results for the LSTM encoder-decoder architecture with gating of the output embedding for a vocabulary of size 4,000

Table 42 shows the results for the model trained using the smaller vocabulary size of 300. It contains tokens of short lengths, mostly of only one ore two letters. In the column "Bpe" this setup is denoted by "mix 300". We also trained a model without gating for comparison. The WERs are similar to our bilingual baseline with larger vocabulary.

| Language | Model | Bpe | LID | WER lc | WER lc decrease |
|---|---|---|---|---|---|
| English | LSTM | mix 300 | en | 9.4% | - |
| German | LSTM | mix 300 | ge | 18.3% | - |
| English | dec emb | mix 300 | en | 9.4% | 0.0% |
| German | dec emb | mix 300 | ge | 18.1% | 1.1% |

Table 42: Results for the LSTM encoder-decoder architecture with gating of the output embedding for a vocabulary of size 300

The gating of the output embedding works clearly better for the smaller vocabulary. The model trained with 300 tokens even outperforms the baseline model with small vocabulary slightly. For the large vocabulary the WERs obtained for the "dec emb"-model increase for both languages.

We visualize the effect of gating the embedding of the outputs by plotting the embedded and gated tokens using t-SNE, explained in Chapter 2.5.
Figure 13 shows the projection into the two-dimensional space of all tokens, included in the large vocabulary, after gating of their embedding for English and German. Analog Figure 14 shows the projection of the tokens in the small vocabulary.

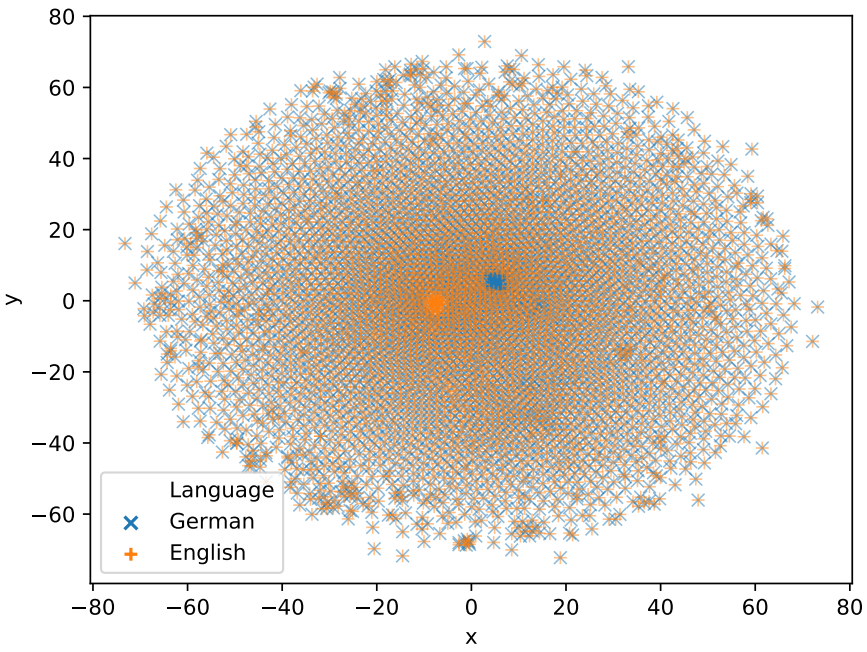


Figure 13: t-TSNE projection of token's embedding after gating using a vocabulary of size $4,000$

In both plots projection centers can be located for each language. However, especially

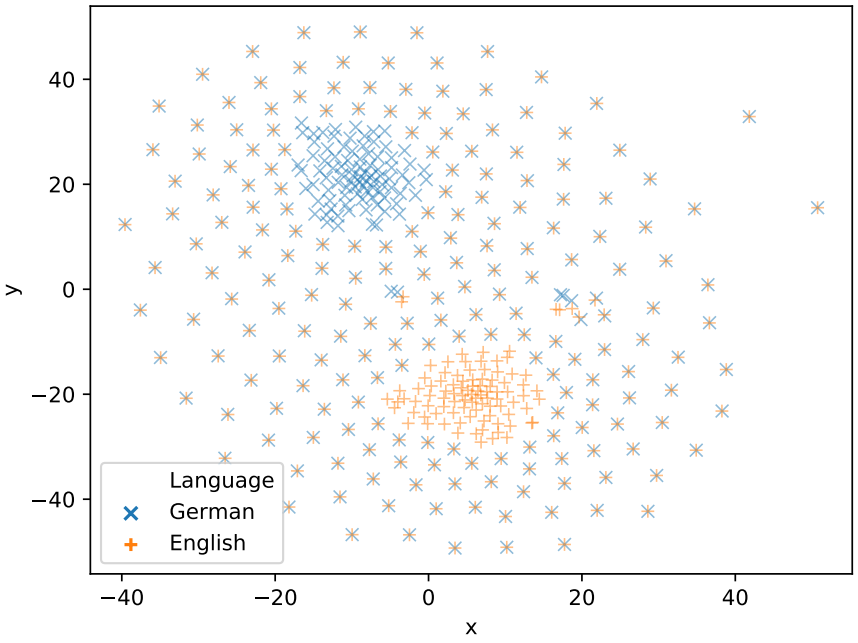


Figure 14: t-TSNE projection of token's embedding after gating using a vocabulary of size 300

for the large vocabulary the projections can not be separated by LID, which indicates that the features of the embedded tokens are not language specific. Outside the centers most token get projected to similar points with English and German id. The coordinates are not exactly equal, but very close to each-other.
For the tokens of the small vocabulary with more shared tokens, the separation is more clear. Eventually reducing the vocabulary size further would lead to further improvements.
We conclude that gating of the output's embedding is more helpful for a smaller vocabulary. If the vocabulary can be split in almost disjunct sets of language specific tokens anyway, the network probably learns to not predict them for the respective other language, either way. The weights in the gating mechanism corresponding to one language are only trained using tokens from this language. Thus we probably don't generate added value using gating after the output embedding.

Testing with the wrong LID worsens the results for both models, as Table 43 shows. However the WERs are clearly higher for the model trained with the larger vocabulary. For English with German LID most predictions are in English with a high recognition rate. By the model trained with the large vocabulary, single German words are predicted, e.g."hört" instead of "hurt", "liest" instead of "list" or "sie" instead of "see".
The main errors of the model, trained with the small vocabulary, are typing errors, e.g.

"someon" instead of "someone".
For German with English id both models make English predictions for some German references. The model trained with the large vocabulary for example predicts "rome" instead of "räume" or "we're none" instead of "wir nennen". One error that occurs very often for both models when decoding German with englisch id, is the prediction of "under" instead of "und".

| Language | Model | Bpe | LID | WER lc |
|---|---|---|---|---|
| English | dec emb | mix | ge | 17.0% |
| German | dec emb | mix | en | 45.3% |
| English | dec emb | mix 300 | ge | 12.1% |
| German | dec emb | mix 300 | en | 30.3% |

Table 43: Results for the LSTM encoder-decoder architecture with gating of output embedding with wrong LID

Larger tokens are embedded using vectors of the same size as used for shorter tokens. This could mean that more language information needs to be represented by the same number of features. Maybe tokens of the given LID are preferred during alignment and small differences in the pronunciation, as e.g. in "hört" and "hurt" are ignored.

As shown in Table 44, the performance using a smaller vocabulary improves for Denglish compared to the results obtained by the bilingual baseline using the larger vocabulary (Table 29).
Gating of the output embedding does not lead to further improvements, on the contrary it's clearly worse.
However the differences using English or German id during inference are only small, the word accuracy even increases using the English id. All models have again mostly problems with Denglish words.

| Language | Model | Bpe | LID | WER lc | WER lc decrease | Word accuracy |
|---|---|---|---|---|---|---|
| Denglish | LSTM | mix 300 | - | 19.3% | - | 50.9% |
| Denglish | dec emb | mix 300 | ge | 22.2% | −15.0% | 45.0% |
| Denglish | dec emb | mix 300 | en | 23.7% | - | 49.7% |
| Denglish | dec emb | mix 300 | ge + en | 22.2% | −15.0% | 48.0% |

Table 44: Results for the LSTM encoder-decoder architecture with gating for Denglisch

The hypotheses created by the systems with German, English LID or mixed language vector, are very similar. Table 45 shows an selection of words predicted by all systems, but containing errors, made by at least one model.
Words that are recognized correctly by all systems are, for instance, "brainstormen", "twittern" and words that exist equally in both languages or are commonly used, e.g. "computer", "party" or "meeting".

| Ref/ Hyp | LID | | | | | |
|---|---|---|---|---|---|---|
| Reference | - | assets | gecrasht | lohngedumt | bottleneck | fashion |
| Hypothesis | ge | essetz | gecrash | lohn gedammt | botte neck | fasshion |
| Hypothesis | en | essitz | gecrashed | lohn gedankt | bottleneck | fashion |
| Hypothesis | ge + en | essitz | gecrashed | lohn gedankt | bottelneck | fashion |

Table 45: Comparison of Denglish words decoded with German, English id or mixed LID

### 4.3.5 Modulation

Modulation with neural language codes was described in Chapter 3.1. We experiment with a much more simple form and modulated the hidden states of the LSTM layers with the embedded one-hot vector, describing the LID. Since gating after each layer performed well in previous experiments, we also apply modulation after each layer. The one-hot vector $d_l$ representing LID is embedded by $V \in \mathbb{R}^{d_{model} \times L}$, when $L$ is the number of languages, and a bias $b \in R^{d_{model}}$ is added. The sigmoid function is applied to the embedded LID vector as

$$d_{lang} = \sigma(V d_l + b).$$

The hidden states $h \in \mathbb{R}^{d_{model}}$ are multiplied element-wise with $d_{lang}$. This should enable the model to learn features depending on language id.
In contrast to gating treated in Chapter 4.3.3, modulation is only based on the LID. For gating we modulate the hidden state with a vector based on the hidden state itself and LID. Furthermore for gating we later concatenate hidden state and LID before the new hidden state is inputted into the next layer.
We use element-wise addition of the bidirectional hidden states in the encoder. Table 46 shows the total number of trainable parameters in the LSTM architecture with modulation. Compared to the number of parameters in our baseline "LSTM with addition" in Table 25 the increase is negligible.

| Model | Number of languages | Number of parameters | Parameter increase |
|---|---|---|---|
| Modulation | 2 | 119,877,891 | 0.04% |

Table 46: Parameter increase due to modulation in the LSTM based encoder-decoder architecture

Table 47 shows the results obtained for English and German using modulation. The WERs are compared to the results of the bilingual baseline in Table 30.

| Language | Model | Bpe | LID | WER lc | WER lc decrease |
|---|---|---|---|---|---|
| English | Modulation | mix | en | 9.6% | 7.7% |
| German | Modulation | mix | ge | 17.1% | −1.1% |

Table 47: Results for the LSTM encoder-decoder architecture with modulation

While the model performs well for English, the WER increases slightly for German.

Table 48 shows the WERs obtained for the LSTM based encoder-decoder model with modulation using the wrong LID. Even when inserting German LID when recognizing English, large parts are recognized correctly. Others parts, with German predictions, even have only low acoustic similarity. For example the utterance "[...]outside the familiy unit and as[...]" is recognized as "[...]hauptsahl bisher molynit andreas[...]". Only low acoustic similarity can be observed for German references and English LID, as well. However the majority of hypotheses is made in English in this case, which might explain the high WER.

| Language | Model | Bpe | LID | WER lc |
|---|---|---|---|---|
| English | Modulation | mix | ge | 62.5% |
| German | Modulation | mix | en | 118.5% |

Table 48: Results for the LSTM encoder-decoder architecture with modulation with wrong LID

For the Denglish data set the results are shown in Table 49. We obtain a relative worsening of 0.8% compared to the bilingual baseline model in Table 30. The word accuracy is computed for the English and Denglish words in the utterances.

| Language | Model | Bpe | LID | WER lc | Word accuracy |
|---|---|---|---|---|---|
| Denglish | Modulation | mix | ge | 23.9% | 43.9% |
| Denglish | Modulation | mix | en | 105.1% | 34.5% |
| Denglish | Modulation | mix | ge + en | 33.0% | 48.0 |

Table 49: Results for the LSTM encoder-decoder architecture with modulation with wrong LID

Words that exist in both languages as "computer", "manager", "event" or "service" are mostly recognized correctly independent of LID.
When decoding with German LID, Denglish words are often replaced by German ones, e.g. "followen" becomes "verloren" or "performt" is recognized as "befreundet". However, in some cases Denglish words as "brainstormen" or "outsourcen" are also recognized correctly with the German LID.
Using English LID gives English predictions for almost each utterance, as already observed for German decoded with English LID. English words are recognized correctly very often. However, the model has problems with the germanized words.
The predictions made by the system for the mixed language vector with German and English id are also correct in large parts. Especially the Denglish words seem to benefit from the combination of both LIDs, since English and Denglish words are both recognized better. For wrongly recognized Denglish words the confusion is often small. For example "produktion" is predicted instead of "production" or "gerated" instead of "geratet". The word accuracy is higher than using only one language id, but overall the

recognition is worse, as to see in the higher WER. In general the predictions are either German, English or neither of them in the same extent.

Modulation by vectors, that carry more information than simply language identity, as done in [38] might lead to further improvements. The results of Denglish show, that adaption might even work too good in some cases, and the network cannot switch to English if necessary.

### 4.3.6 Language Adaption in MHA for LSTM

In [7] the authors integrated Language Embedding Concatenation (LEC) and Language Specific Attention Heads (LSAH) into the MHA layers of the audio encoder in a RNN-T architecture. We apply both, LEC and LSAH, to the MHA layer in our LSTM based encoder-decoder architecture. The underlying attention mechanism is as explained in 2.7.3. For an introduction of LEC see Chapter 3.6, LSAH was explained in Chapter 3.7. We use one language specific attention head for each language and eight heads in total, as in the baseline.
We trained bilingual models with English and German. For two languages the network has the following numbers of learnable parameters, shown in Table 50. In the encoder the bidirectional outputs are concatenated, thus the numbers are compared to the number of parameters in the "LSTM" baseline in Table 25. The differences to the number of parameters in the baseline are small.

| Model | Number of languages | Number of parameters | Parameter increase |
|---|---|---|---|
| LEC | 2 | 161,787,139 | 0.01% |
| LSAH | 2 | 161,646,851 | −0.08% |

Table 50: Parameter increase due to language specific MHA in the LSTM based encoder-decoder architecture

The WERs obtained for English and German are presented in Table 51. For both methods and their combination, the results are comparable to the baseline bilingual LSTM encoder-decoder model, shown in Table 26.

| Language | Model | Bpe | LID | WER lc | WER lc decrease |
|---|---|---|---|---|---|
| English | LEC | mix | en | 8.6% | 4.4% |
| German | LEC | mix | ge | 18.5% | 0.0% |
| English | LSAH | mix | en | 8.8% | 2.2% |
| German | LSAH | mix | ge | 18.8% | −1.6% |
| English | LSAH + LEC | mix | en | 8.9% | 1.1% |
| German | LSAH + LEC | mix | ge | 18.7% | −1.1% |

Table 51: Results for the LSTM based encoder-decoder architecture with language adaption in MHA layer

LEC yields the best results and improves the recognition rate of English slightly, while for German it's equal.
For LSAH the difference in WERs to the bilingual baseline model are even smaller. Instead of eight heads as in the baseline, we only use seven heads for each language with LSAH. Further investigations could show if an increased number of heads might give additional improvements.

The results of the three investigated methods obtained for Denglish are shown in Table 52 and compared to the baseline results, shown in Table 29. The word accuracy is computed separately for the English and germanized words.

| Language | Model | Bpe | LID | WER lc | WER lc decrease | Word accuracy |
|---|---|---|---|---|---|---|
| Denglish | LEC | mix | ge | 22.9% | 0.0% | 43.9% |
| Denglish | LSAH | mix | ge | 22.1% | 3.5% | 49.7% |
| Denglish | LSAH + LEC | mix | ge | 22.1% | 3.5% | 49.7% |
| Denglish | LEC | mix | en | 22.6% | 1.3% | 45.6% |
| Denglish | LSAH | mix | en | 22.3% | 2.6% | 49.7% |
| Denglish | LSAH + LEC | mix | en | 21.6% | 5.7% | 50.3% |

Table 52: Results for the LSTM based encoder-decoder architecture with language adaption in MHA layer for Denglish

Except for LEC with German id, all methods, independent of inserted LID, outperform the monolingual and bilingual baseline regarding WER and word accuracy of the Denglish words. In contrast to the recognition rates of English and German, the combination of LEC and LSAH, yields the highest improvements for Denglish.
However, we don't observe general differences between the correctly recognized words, comparing the different methods. The systems all have problems with the same - mostly Denglish - words. No system recognizes, e.g. "stalken", "outgesourced" or "getaggt". Words that are recognized correctly by all systems are for example "brainstormen" or "outsourcen" and English words as "resources", "manager" or "global".
The combination of LEC and LSAH with English id results in the best recognition rate. Denglisch words are often either recognized correctly or replaced by German words. The same system adapted to German often predicts German or nonexistent words.

Although the obtained improvements for English and German are small, the methods might be worth further investigation. Especially Denglisch seems to benefit. The WER of 21.6 obtained for the LSTM based encoder-decoder model with LSAH and LEC is the best WER we obtain with bilingual data and a vocabulary size of $4,000$.

### 4.3.7 Combining Methods

In a last step we combined gating after each layer (see Chapter 4.3.3) and the language adaption methods in the MHA layer (see Chapter 4.3.6). This results in architectures

with language adaption in or after almost every part of the network. Solely output embedding and the Linear and Softmax layers as last layers are not language specific. We compare the combination of gating after each layer with LEC (LEC + all layer) and gating after each layer together with LSAH (LSAH + all layer). Furthermore we trained a bilingual model with combination of all three methods (LEC + LSAH + all layer).
All three models were trained with English and German data. Additionally the combined model with gating , LEC and LSAH was trained with trilingual data of English, German and French.
The outputs of the biLSTM layers are element-wise added as done before in the network with gating after every layer. The numbers of parameters are shown in Table 53 and compared to the number of parameters in our baseline "LSTM with addition" in Table 25.

| Model | Number of languages | Number of parameters | Parameter increase |
|---|---|---|---|
| LEC + all layer | 2 | 174,470,403 | 45.6% |
| LSAH + all layer | 2 | 174,330,115 | 45.5% |
| LEC + LSAH + all layer | 2 | 174,338,179 | 45.5% |
| LEC + LSAH + all layer | 3 | 174,236,931 | 45.4% |

Table 53: Parameter increase due to language adaption in the LSTM based encoder-decoder architecture

The WERs of the bilingual models for English and German are presented in Table 54. The decrease in WER shown, is relative to our bilingual baseline in Table 30.

| Language | Model | Bpe | LID | WER lc | WER lc decrease |
|---|---|---|---|---|---|
| English | LEC + all layer | mix | en | 8.6% | 16.5% |
| German | LEC + all layer | mix | ge | 17.0% | −0.6% |
| English | LSAH + all layer | mix | en | 8.2% | 20.4% |
| German | LSAH + all layer | mix | ge | 17.6% | −4.1% |
| English | LEC + LSAH + all layer | mix | en | 8.2% | 20.4% |
| German | LEC + LSAH + all layer | mix | ge | 17.0% | −0.6% |

Table 54: Results for the LSTM based encoder-decoder architecture with language adaption in MHA layer and gating

All methods outperform the bilingual baseline for English. For German the WERs are comparable to the bilingual baseline, except for "LSAH + all layer", which yields an increased WER. The combination of LEC, LSAH and gating after each layer yields the highest improvement overall. In average we achieve a relative decrease in WER of 9.9%. Compared to the monolingual models the decrease in WER is even higher with 12.6% in average, see Table 26 for comparison. The combination of the bidirectional outputs in the encoder is different in the monolingual models and the combined model, which makes an exact comparison difficult.

Especially the combination of all three methods gives even slightly better results than gating after each layer with an average improvement of 2.1%.

Table 55 shows the WERs for English and German for the "LEC + LSAH + all layer"-model.

| Language | Model | Bpe | LID | WER lc |
|---|---|---|---|---|
| English | LEC + LSAH + all layer | mix | ge | 90.2% |
| German | LEC + LSAH + all layer | mix | en | 115.4% |

Table 55: Results for the LSTM based encoder-decoder architecture with language adaption and gating with wrong LID

The WERs are very similar to the "all layer"-gating bilingual LSTM architecture evaluated in Chapter 4.3.3. Furthermore, also the hypotheses and errors are comparable.

For Denglisch the results, using German LID and English LID during inference, are shown in Table 56. Compared to the bilingual baseline (see Table 30 for comparison) we achieve a relative improvement of 8.4% for the model adapted to German. Compared to the monolingual German model the WER decreases by 20.2%, see Table 29 for the results of the monolingual baseline.

| Language | Model | Bpe | LID | WER lc | Word accuracy |
|---|---|---|---|---|---|
| Denglish | LEC + LSAH + all layer | mix | en | 102.8% | 42.1% |
| Denglish | LEC + LSAH + all layer | mix | ge | 21.7% | 46.2% |

Table 56: Results for the LSTM based encoder-decoder architecture with language adaption and gating for Denglish

For both LIDs the results, also regarding hypotheses, are similar to those of the LSTM based encoder-decoder model with gating after each layer. The models also have problems with the recognition of the same words. The WER of the system adapted to German gives a lower WER as the system using only gating after each layer. This, at least partly, might also come from predicting more compound words. The "all layer"-gating bilingual model splits many words, which results in a higher WER.
The system adapted to English recognizes English words, but is not able to predict Denglish words correctly. When adapting to German most Denglish words are not recognized either, only few as "relaxen", "twittern", or "abgelost" are predicted correctly.

To see if the combined model yields further imrovements with an additional language we repeated the training using data of English, German and French. We concentrated on the model with gating after each layer, LEC and LSAH, since this performed best in the bilingual setup. The results are shown in Table 57 and compared to the trilingual baseline results in Table 31.

| Language | Model | Bpe | LID | WER lc | WER lc decrease |
|---|---|---|---|---|---|
| English | LEC + LSAH + all layer | mix+ | en | 9.0% | 3.2% |
| German | LEC + LSAH + all layer | mix+ | ge | 15.9% | 13.1% |
| French | LEC + LSAH + all layer | mix+ | fr | 17.0% | 16.3% |

Table 57: Results for the trilingual LSTM based encoder-decoder architecture with language adaption in MHA layer and gating

Compared to the trilingual baseline we achieve an average decrease in WER of 10.9%. Compared to the monolingual baseline we achieve 15.3% relative improvement in average and the trilingual model with combined language adaption method also outperforms the "all layer"-gating trilingual system by 6.4%.
Especially German and French benefit of the additional language data, but also the WER for English decreases.

For Denglish the combination of language adaption methods outperforms the trilingual baseline by 8.3% relative decrease in WER, see Table 31 for comparison. The results of the monolingual systems are shown in Table 31. With German LID the WER is reduced by 22.8% for the combined model.

| Language | Model | Bpe | LID | WER lc | Word accuracy |
|---|---|---|---|---|---|
| Denglish | LEC + LSAH + all layer | mix+ | ge | 21.0% | 52.0% |
| Denglish | LEC + LSAH + all layer | mix+ | en | 100.3% | 41.5% |

Table 58: Results for the trilingual LSTM based encoder-decoder architecture with language adaption in MHA layer and gating for Denglish

The occurring errors are very similar to those done by the LSTM encoder-decoder network with gating after each layer, as we could already observe for the bilingual models. However, in particular the word accuracy of the English and Denglish words increases for the trilingual combined model adapted to German. With 52% word accuracy it outperforms the bilingual combined model and the trilingual model with gating adapted to German by 12.5%.

In general the combined model yields similar results as the models with gating in Chapter 4.3.3 regarding hypotheses and recognition rate. However the WERs decrease further, when using language adaption in the MHA layer additionally to gating. We also showed that the performance can improve by adding another language.

### 4.3.8 Conclusion

For the LSTM based encoder-decoder architecture language adaption leads to an improved recognition rate. The highest improvement was obtained when using a gating algorithm after every layer of the encoder and decoder together with language specific attention heads.

Applying all before discussed methods solely, gating, applied after every layer in encoder and decoder, yields the highest improvement. The language adaption methods in the MHA layer, LEC and LSAH, only give a minor decrease in the WER, but perform well when being combined with gating. Applying language adaption to every part of model seems to be most efficient for the LSTM based encoder-decoder architecture.
Language specific attention layers might be especially useful for Denglish or in general utterances with mixed languages. The WER decreased, while the word accuracy increased. Although the model seems to adapt to the language, given by LID, it is able to handle code-switching if necessary and benefits from the different language data.
Modulation, which is based on a similar principle as gating, also leads to a reduced WER. We modulated the outputs of the layers with the embedded language identity vectors. Using a vector that carries more language properties than the identity alone, might enable the model to learn more language property related features. This could help the model in using similarities and differences of languages and lead to further improvements. Besides, modulation is easier than gating and adds less trainable parameters to the architecture. In our bilingual setup the recognition rate was not as good as achieved by the model with gating, but also outperformed the bilingual baseline.
The idea of gating of the output embedding was to learn language dependent representations of the output tokens. We trained two models with different vocabulary sizes. The baseline models for the large and small vocabulary are comparable, for Denglish using a reduced vocabulary size even outperforms the model trained with the larger vocabulary. We obtained clearly better results using gating of the output embedding with the small vocabulary. Nevertheless the improvements are small.

The exact comparison of the evaluated methods is difficult, since we used two different ways of combining the bidirectional outputs of the biLSTM layers. Comparing the results of the multilingual baselines, element-wise addition of the bidirectional outputs performs better slightly, and furthermore leads to a decreased number of learnable parameters.
The results of the baseline also show that the multilingual models without language adaption yield comparable results as the monolingual models. Also when adding French as third language, the WERs decrease only slightly compared to the bilingual baseline.

# 5 Future Work

We investigated multiple language adaption methods for the Transformer and a LSTM based encoder-decoder architecture. For most of our experiments we used English and German data, for the LSTM based model we also experimented with a trilingual setup and added French data. Experimenting with more than two or three languages could be very interesting with and without language adaption. Especially if adding languages, that have less shared properties, as e.g. English and German.
We obtained good results for the model consisting of LSTM layers, but for the Transformer the performance decreased compared to the baseline without adaption techniques.

The results obtained for the Transformer suggest that the gating method and language specific attention layers lead to language adaption, but the WERs increase. Additional investigation is needed to find out why those methods do not improve the results. Eventually adjusting the gating algorithm to be better suited for the Transformer leads to improvements.

Since the evaluated methods improve the results for the LSTM based encoder-decoder model, further investigation and refinement of these methods, might lead to an additional decrease in WER.
Instead of using a one-hot vector to represent LID, a language feature vector can be inserted. Thus the model might learn features depending on properties of languages. Furthermore, instead of applying gating or modulation after every layer of the encoder and decoder, their application after specific layers could be investigated. For instance a comparison of gating in the encoder with gating in the decoder could show if language specific features of the acoustics or language lead to a higher improvement.
For LSAH a different number of shared and language specific heads can be investigated in the future.

In our experiments we experimented with three languages English, German and French and observed that in the multilingual model German and French seem to be highly beneficial for each-other. For better comparison testing the monolingual models with the respective other language is necessary, as well as training a bilingual model for French and German.

Even if we could also improve the recognition of Denglish, in particular the germanized English words are not recognized. The highest word accuracy of the English and Denglish words we could achieve is 52%. The word accuracy of the words that are neither correct English nor German words, is significantly lower.
We therefore want to investigate the combination of language vectors. The experiments with mixed language vector during inference improved the word accuracy of the Denglish words towards inserting only German or English id. In Training a combination of mixed and single language vectors could be tried.
Regarding LSAH the combination of German and English heads might also lead to

improvements for Denglish. At the moment we can only use one LID during inference.

# References

[1] https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1.

[2] Christopher Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

[3] Christopher Olah. Neural networks, types, and functional programming. `http://colah.github.io/posts/2015-09-NN-Types-FP/`, 2015.

[4] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, Sebastian Stüker, and Alexander Waibel. Very deep self-attention networks for end-to-end speech recognition. *arXiv preprint arXiv:1904.13377*, 2019.

[5] Markus Müller, Sebastian Stüker, and Alex Waibel. Neural codes to factor language in multilingual speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8638–8642. IEEE, 2019.

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

[7] Anshuman Tripathi Bhuvana Ramabhadran Brian Farris Hainan Xu Han Lu Hasim Sak Isabel Leal Neeraj Gaur Pedro J. Moreno Qian Zhang Yun Zhu, Parisa Haghani. Multilingual speech recognition with self-attention structuredparameterization, 2020. INTERSPEECH.

[8] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. Spoken language processing: A guide to theory, algorithm, and system development. 01 2001.

[9] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

[10] S. S. Stevens, J. Volkmann, and E. 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.

[11] Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994.

[12] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, 2018. arXiv:1808.06226.

[13] Ahmed Ali and Steve Renals. Word error rate estimation for speech recognition: e-wer. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 20–24, 2018.

[14] L. V. D. Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[15] Frank Rosenblatt. The perceptron, a perceiving and recognizing automaton project para. 1957.

[16] Jürgen Markl. *Markl Biologie Oberstufe*. Klett Schulbuchverlag, 2010.

[17] Sun-Chong Wang. Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer, 2003.

[18] E. Fiesler. Neural network formalization. 1992.

[19] Rudolf Kruse, Christian Borgelt, Christian Braune, Sanaz Mostaghim, Matthias Steinbrecher, Frank Klawonn, and Christian Moewes. *Computational intelligence*. Springer, 2011.

[20] 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.

[21] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.

[22] John S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Françoise Fogelman Soulié and Jeanny Hérault, editors, *Neurocomputing*, pages 227–236, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[23] D. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[24] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[25] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. arXiv:1409.0473.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014. arXiv:1409.3215.

[27] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. volume 2006, pages 369–376, 01 2006.

[28] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning, 2017. arXiv:1609.06773.

[29] Julian Salazar, Katrin Kirchhoff, and Zhiheng Huang. Self-attention networks for connectionist temporal classification in speech recognition. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7115–7119, 2019.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[31] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999.

[32] Thai-Son Nguyen, Ngoc-Quan Pham, Sebastian Stüker, and Alex Waibel. High performance sequence-to-sequence model for streaming speech recognition. *arXiv preprint arXiv:2003.10022*, 2020.

[33] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[34] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. arXiv:1706.03762.

[36] 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, 2017. arXiv:1712.01541.

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

[38] Markus Müller. *Multilingual Modulation by Neural Language Codes*. PhD thesis, Karlsruher Institut für Technologie (KIT), 2018.

[39] J. B. Hampshire and A. H. Waibel. The meta-pi network: connectionist rapid adaptation for high-performance multi-speaker phoneme recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 165–168 vol.1, 1990.

[40] Shiyu Zhou, Shuang Xu, and Bo Xu. Multilingual end-to-end speech recognition with a single transformer on low-resource languages, 2018. arXiv:1806.05059.

[41] 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, 2018. arXiv:1711.01694.

[42] Ne Luo, Dongwei Jiang, Shuaijiang Zhao, Caixia Gong, Wei Zou, and Xiangang Li. Towards end-to-end code-switching speech recognition, 2018. arXiv:1810.13091.

[43] K. Soky, S. Li, T. Kawahara, and S. Seng. Multi-lingual transformer training for khmer automatic speech recognition. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1893–1896, 2019.

[44] Sibo Tong, Philip N. Garner, and Hervé Bourlard. Multilingual training and cross-lingual adaptation on ctc-based acoustic model, 2018. arXiv:1711.10025.

[45] M. S. Mary N J, V. M. Shetty, and S. Umesh. Investigation of methods to improve the recognition performance of tamil-english code-switched data in transformer framework. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7889–7893, 2020.

[46] Anjuli Kannan, Arindrima Datta, Tara N. Sainath, Eugene Weinstein, Bhuvana Ramabhadran, Yonghui Wu, Ankur Bapna, Zhifeng Chen, and Seungji Lee. Large-scale multilingual speech recognition with a streaming end-to-end model, 2019. arXiv:1909.05330.

[47] A. Waters, N. Gaur, P. Haghani, P. Moreno, and Z. Qu. Leveraging language id in multilingual end-to-end speech recognition. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 928–935, 2019.

[48] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer, 2018.

[49] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

## Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Ort, den Datum