FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Peter Polák

Spoken Language Translation via Phoneme Representation of the Source Language

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	doc. RNDr. Ondřej Bojar, Ph.D.
Study programme:	Computer Science
Study branch:	Artificial Intelligence

Prague 2020

Dedication.

Title: Spoken Language Translation via Phoneme Representation of the Source Language

Author: Peter Polák

Institute: Institute of Formal and Applied Linguistics

Supervisor: doc. RNDr. Ondřej Bojar, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Abstract.

Keywords: key words

Contents

Introduction

Work organization

We settled on a less traditional strategy for the organization of this work: we serve every chapter as a standalone small paper. Notably, each chapter includes a section about related work devoted to the chapter's subject. We review mutual related work in Chapter 1 (such as neural network architectures or corpora).

Chapters progressively deal with problematic starting with Automatic Speech Recognition in Chapter 2, further enhancing ASR in Chapter 3, continuing with Spoken Language Translation in Chapter 4. Adaptation to speaker is described in Chapter 6. We also deal with "bringing our work to life" in Chapter 7. Finally, we conclude the work in Chapter 7.

1. Tasks, architectures, and data sets

In this chapter, we give a brief overview of the theoretical foundations necessitated for this thesis. First, we define the main tasks that are part of the solution. Moreover, we describe the utilized neural network architectures. Lastly, we describe used data sets and introduce evaluation metrics.

1.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is one of the most popular tasks in NLP, and without a doubt, it is one of the most important ones. With ever-growing technology that becomes more and more integrated with our day-to-day life, it is clear that ASR will take an essential role in this process.

In this section, we first briefly examine the history and then describe the most popular contemporaneous approaches for ASR.

1.1.1 Brief history

The first attempts for ASR systems stem from the 1950s. At that time, researchers focused their interest on acoustics-phonetics, which describes phonetic elements of speech, including phonemes [Juang and Rabiner, 2005]. The first ASR system worked with syllables, vowels, and phonemes. An example can be taken spoken digit recognizer from Bell laboratories. Their system estimated formant frequencies (as a vowel is pronounced, vocal tract sounds with natural modes of resonance called formant) of vowel regions of an isolated digit.

A big leap in ASR systems was the introduction and popularization of Hidden Markov models (HMMs) in the 1980s. HMMs caused architecture to shift from pattern recognition to statistical modeling. The success of HMM-based ASR systems continues even today in the form of hybrid models consisting of HMMs and either Gaussian Mixture Models (GMMs) or Artificial Neural Networks (ANNs).

In the last few years, deep neural networks gained much popularity. In many tasks, ranging from image processing to natural language processing, DNNs outperformed other known methods. Besides their performance, they tend to require less expertise and engineering skills for a particular task than other methods. This makes them available for researchers in many areas. DNNs are becoming a standard in ASR currently, but the first attempts were already made briefly after the introduction of the backpropagation algorithm [Rumelhart et al., 1986]. They were used for recognition of phonemes [Waibel et al., 1989] or few words [Lubensky, 1988]. Further important milestones for ASR in the 2010s were recurrent neural networks and attention (applied, for example, in Chorowski et al. [2014]) in the 2010s.

1.1.2 Models

In this section, we introduce contemporary models utilized for ASR. There are two such approaches: (1) HMM-based models, (2) End-to-End neural models.

HMM-based models

HMMs (Hidden Markov Models) are probably the most used technique in ASR [Padmanabhan and Johnson Premkumar, 2015, Yu and Deng, 2016]. Hidden Markov Models capture the the speech information sequence. HMMs utilize in the ASR context either GMMs (Gaussian Mixture Models) or ANNs (Artificial Neural Networks) in order to model frame-based speech features (features with no temporal information). Practitioners often refer to the HMM-based models combined with neural networks as hybrid models.

A typical pipeline of an HMM-based model works as follows: the input is first pre-processed and converted to features, typically the MFCCs (see Section 1.4.1). These featured are further passed to the estimator. Common, estimated units in HMM-based pipelines are phones. The decoder employs *acoustic model*, *dictionary* and *language model* to decode the speech. The acoustic model estimates the probability of the acoustic sequence given word sequence. ANNs or GMMs represents the acoustic model. Dictionary maps phone sequences to words. Finally, the language model estimates the apriori probability of word sequence, independent of observed sound. Usual are n-gram language models.

End-to-End models

Opposed to the HMM-based models are End-to-End ASR pipelines. The common trait for these models is their ability to produce the final, human-readable text given acoustic features using one end-to-end deep neural network. Recent advancements (for example, Amodei et al. [2016], Li et al. [2019a]) in the ASR field clearly show the future direction towards this kind of ASR. The advantage of the E2E model is that they require less engineering than HMM models. Open problem remains the higher requirement of training data.

The input of the model are MFCC features extracted from speech data. E2E model then outputs probabilities of graphemes at each time step employing CTC loss. Applications extend conventional beam search with language model (again, n-gram LM is used) that re-scores the beams during the decoding. The use of LM in the beam search further enhances the transcription quality.

1.2 XXX Spoken Language Translation or Speech translation

Spoken Language Translation is another NLP task dealing with the translation of the speech. XXX In literature, the SLT task describes translation into a target language text. Commonly, SLT can also be a speech-to-speech translation. Traditional layout of speech translation, before the emergence of end-to-end systems, consisting of a speech recognition unit followed by machine translation. With the upraise of deep learning, E2E frameworks that do not need intermediate transcription step are gaining popularity [Bérard et al., 2016, 2018, Jia et al., 2019].

Direct comparison of both methods seems inconclusive [Sperber et al., 2019]. Also, the actual tasks of E2E and cascaded SLT differ: E2E SLT must be trained on E2E corpora, while the cascading can be trained on independent corpora for ASR and MT. This makes the latter suitable in cases where is no corpus for given language pair available (which is mostly the case). On the other hand, cascading speech recognition and machine translation for SLT could introduce errors in source language transcription that might be further propagated to final translation.

1.3 Models/architectures

In this section, we introduce neural network models/architectures we engage in our work. The first two models, Jasper and QuartzNet, are used as acoustic models. Finally, we present the Transformer, which serves as a translation and correction model.

1.3.1 Jasper

Jasper [Li et al., 2019a] is a family of end-to-end, deep convolutional neural network ASR architectures. We incorporate the architecture in our ASR and SLT pipelines.

Architecture Overview

The input of the model are Mel Frequency Cepstrum Coefficients (see Section 1.4.1) obtained from 20 ms frames with 10 ms stride. We use 64 features. The model outputs probability over given vocabulary for every processed frame. In our pipeline, the vocabulary is IPA phonemes.

Input is passed through one pre-processing layer, followed by the central part of the network. Finally, tree post-processing layers are applied. The central part of the model consists of so-called “blocks”.

Jasper model consists of B blocks and R sub-blocks. Jasper authors introduce a naming convention where such model is described as “Jasper BxR”. In our work, we use Jasper 10x5.

Sub-blocks applying operations as follows: 1D convolution, ReLU activation, and dropout. All sub-blocks of a block have the same number of output channels.

The input of a block is connected to the last sub-block via residual connection. Because the number of channels differs, 1x1 convolution is applied to account this. After this projection, batch normalization is applied. The output is then added to the output of the batch normalization layer in the last sub-block. Afterward, activation function and dropout are used, producing the output of the current block.

Further, Jasper’s authors observed that models deeper than Jasper 5x3 require residual connections to converge. Residual connections inspired by DenseNet [Huang et al., 2017] and DenseRNet [Tang et al., 2018] are employed.

Schema of the Jasper with residual connections is pictured in Figure 1.1. The biggest used configuration for English (graphemes) has 333 million parameters.

1.3.2 QuartzNet

QuartzNet [Kriman et al., 2019] is another end-to-end ASR architecture used in our work. QuartzNet is a convolutional neural network based on Jasper [Li et al., 2019a] architecture having a fraction of parameters (18.9 million versus 333 million) while still achieving near state-of-the-art accuracy.

Same as the Jasper model, the network’s input is 64 MFCC features computed from windows of length 20 ms and overlap 10 ms. The network outputs probability over the given alphabet for each time frame. For training is used CTC loss and for decoding beam search.

The main difference between the model and Jasper is the application of 1D time-channel separable convolutions. Such convolutions can be separated into 1D depthwise convolutional layers with kernel K and a pointwise convolution operating on each time

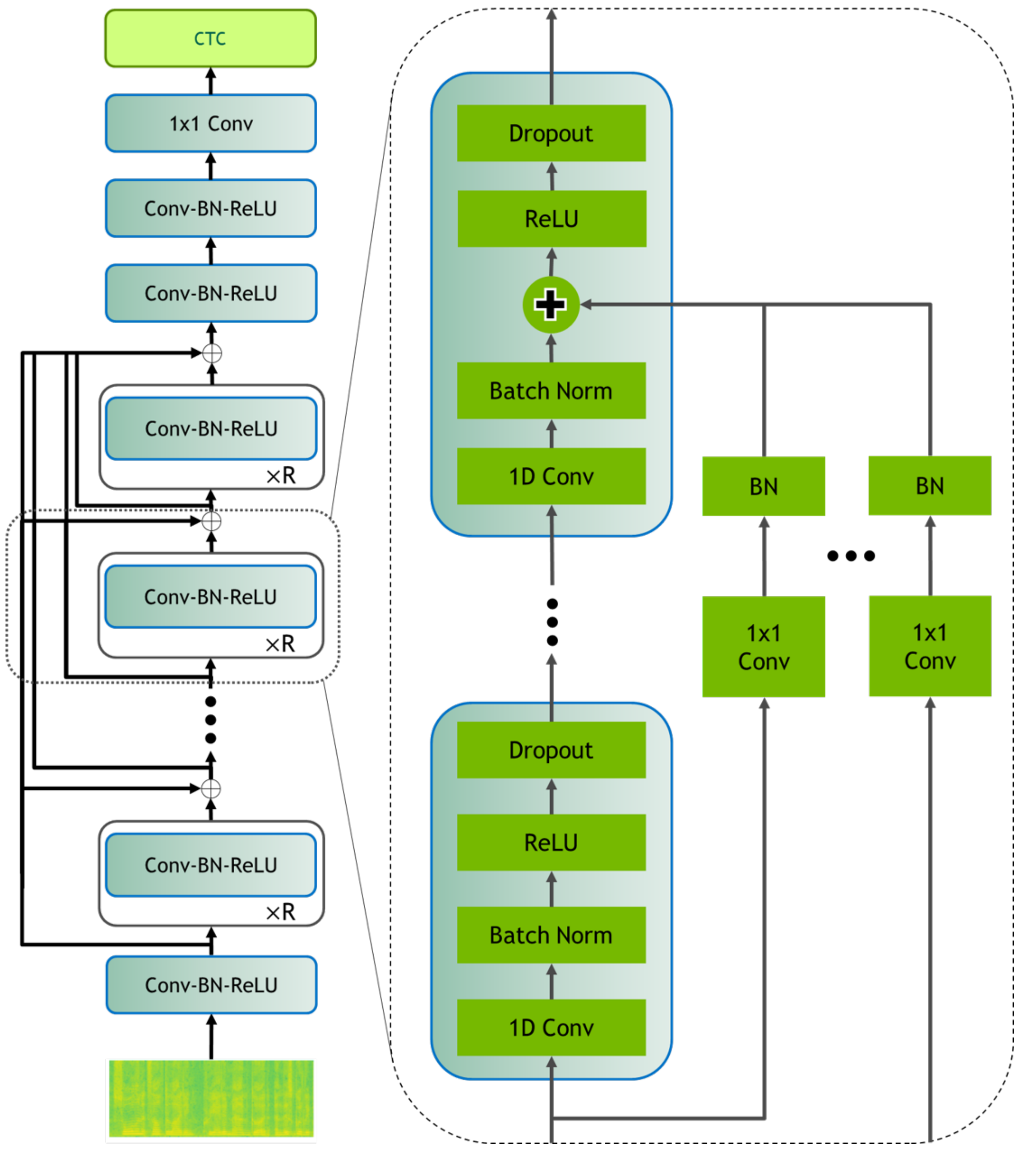

Figure 1.1: Jasper Dense Residual, taken from Li et al. [2019b].

frame independently. Because of this separation, the model has fewer parameters and can even have 3× larger kernel than the bigger Jasper model.

A further reduction of weights can be achieved by using grouped pointwise convolution instead of the pointwise convolution layer (see Figure 1.3). When using four groups, the number of parameters is halved (for QuartzNet-15x5 from 18.9M to 8.7M) with slightly worse performance (WER 3.98 increases to 4.29 for LibriSpeech dev clean and 11.58 increases to 13.48).

As the first described weights reduction is sufficient for our purposes, we work with QuertzNet without the grouped pointwise convolution.

1.3.3 Transformer

Transformer [Vaswani et al., 2017] has become a very good established architecture in Neural Machine Translation [Bojar et al., 2018, Barrault et al., 2019]. The main idea behind the architecture is to get rid of recurrence and convolutions and rather base the model solely on attention. As the attention mechanism is implemented as matrix multiplications, contemporary GPUs can better parallelize the computations leading to faster training.

A Transformer model is composed of encoder and decoder (see Figure 1.4). Both encoder and decoder are stacked identical layers. Encoder layer has two sub-layers: a multi-head self-attention layer and a fully connected feed-forward network. Decoder layer has extra multi-head attention sub-layer, which performs attention over the output of the encoder. This additional layer is between the self-attention and the feed-forward layers. Self-attention in the decoder is modified so that the auto-regressive property holds, i.e., the encoder cannot look to the right side ("future").

The advantage of self-attention is that it can access arbitrary position in a constant number of sequentially executed operations while recurrent networks need $O(n)$ sequentially executed operations. If the sequence length is less than the representation dimension, which is often the case, the total computational complexity is lower than of the recurrent models. Another advantage presented by the authors is that self-attention leads to better interpretable models. They claim the individual attention heads seem to learn some specific functions that may be related to the syntactic and semantic structure of a sentence.

We employ this model in our enhanced ASR pipeline as a correction and language model and as a translation model in our SLT pipeline.

1.4 Data representation

This section introduces to the reader the data representations. The way how we encode and feed the neural network can significantly influence performance. In this work, we transcribe recordings, i.e., we work with voice and text data. First, we introduce MFCC — the voice presentation, and then we discuss text encoding.

1.4.1 MFCC

Mel frequency cepstral coefficiets (MFCC) is the most commonly used representation of speech for ASR and SLT. This method exploits the way how the human auditory system perceives voice. Filter in the MFCC pipeline is linearly spaced for frequencies up to 1000 Hz and logarithmically above.

MFCC pipeline as described in Muda et al. [2010] and Kamath et al. [2019]:

1. **Pre-emphasis** Application of filter that emphasises higher frequencies:

$$Y[n] = X[n] - \alpha X[n-1] \tag{1.1}$$

.

This filter makes the signal less dependent on strong signals from previous time steps.

2. **Framing** Raw audio is segmented into small windows. Signal in small windows can be then treated as stationary. Typically, the length of the window is about 20 ms, and windows have an overlap of 10 ms.

3. **Windowing** To avoid potential abrupt changes caused by framing windowing is applied. Windowing is a multiplication of samples in a window with a scaling function. Most commonly used in ASR is Hann and Hamming windowing:

$$w(n) = \sin^2\left(\frac{\pi n}{N-1}\right) \tag{1.2}$$

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{1.3}$$

where N is window length and $0 \leq n \leq N-1$.

4. **Fast Fourier Transform** FFT converts one-dimensional signal from time to frequency domain.

5. **Mel Filter Bank** The Mel Filter Bank is a set of filters that mimic the human auditory system. Usually, 40 filters are used. Each filter is of a triangular shape. These are used to compute a weighted sum of spectral filter components approximating the Mel scale. Each filter output is then the sum of its filtered spectral components.

6. **Discrete Cosine Transform** This process converts the log Mel spectrum into a time domain. The result is called the Mel Frequency Cepstrum Coefficient (the MFCC), and the set of coefficients is called acoustic vectors.

An example of a mel-spectrogram is in the Figure 1.5.

1.4.2 Text representation in NMT

There are many approaches to text representation in NMT. Each of these representations has its advantages and drawbacks. We differentiate the following representations:

- character,
- word,
- sub-word level representation.

The first one is a very simple and straightforward approach. Character representation permits one to encode any word (written in given characters). Its downside is that it produces longer sequences compared with other methods. Less output classes lead to the reduction of computational complexity. However, the model needs to attend more positions, which substantially increases time complexity during decoding.

Word level representation, on the other hand, produces shorted sequences. It may be better in some applications as the encoded string is shorter compared with the character-level approach. Generally, though, neural machine translation is an open-vocabulary problem. Word-level representation is undesirable, as it cannot handle unknown words. Several techniques have been proposed, such as NMT with post-processing step [Luong et al., 2014, Luong and Manning, 2016].

The most versatile method seems to be a sub-word representation. It addresses both problems, as it produces shorter sequences compared, and is also capable of handling unknown and rare words. The number of contemporary NMTs that use sub-word level representation demonstrates its utility. The most prominent sub-word level tokenizers are BPEs [Sennrich et al., 2016] and subword regularization Kudo [2018]. Both methods are based on similar ideas — they produce more compact text representations. The former is based on the "merge" operation that joins the most frequent character sequences together, while the latter is based on unigram language model. A particular benefit of the subword regularization is that it is also able to produce different segmentation.

In our work, we use BPE implementation YouTokenToMe[1]. We chosen this particular implementation as it supports multithreading is considerably faster than other implementations and has Python and command-line interface. Further, it comes with BPE-dropout [Provilkov et al., 2019]. BPE-dropout is an enhancement of traditional BPE, which addresses the deterministic nature of the method. BPE-dropout randomly drops some merges from the BPE merge table, which results in different segmentation. Introducing a noise to the data helps to regularize an NMT model training.

Byte Pair Encoding

We would like to point out inconsistency of reporting BPE size in literature. Some authors use terms "*BPE size*" and "*number of merge operations*" as synonyms, although the actual *BPE size* equals *number of merge operations* plus *characters*. In this work, we use term "*BPE size*" as absolute vocabulary size — including characters.

[1]`https://github.com/VKCOM/YouTokenToMe`

1.5 Data sets

We dedicate this section to data sets used for the training of models in our work. First, we introduce speech corpora LibriSpeech and Common Voice, and then we introduce translation corpus CzEng.

1.5.1 LibriSpeech

LibriSpeech [Panayotov et al., 2015] is a large corpus of read English speech. The corpus contains 1000 hours of transcribed speech based on audiobooks from project VoxForge[2].

The data set is structured into three parts that have approximately 100, 360, and 500 hours. Using a trained model on the Wall Street Journal corpus [Paul and Baker, 1992], authors divided the speakers by WER into two pools: "clean" and "other". From the "clean" pool, 20 male and female speakers were randomly selected to development and test sets. The rest was assigned to 100 and 360 hours of "clean" sets. For the "other" pool (500 hours), authors selected more challenging data for development and test sets.

1.5.2 Common Voice

Common Voice Ardila et al. [2019] is a multi-lingual, crowdfunded speech corpus. At the time of the writing, 29 languages were available. English data set contained 1118 hours of validated, transcribed recordings. Unfortunately, the Czech data set was not available.

All utterances are collected and validated by volunteers. The data collection runs solely online through a web form. Speech utterances are stored in MPEG-3 format with a 48 kHz sampling rate. After at least two out of three volunteers up-votes an utterance, it is considered to be valid.

The number of clips is divided among the three datasets according to statistical power analyses. Given the total number of validated clips in a language, the number of clips in the test set is equal to the amount needed to achieve a confidence level of 99% with a margin of error of 1% relative to the number of clips in the training set. The same is true of the development set.

1.5.3 Large Corpus of Czech Parliament Plenary Hearings

Large Corpus of Czech Parliament Plenary Hearings Kratochvíl et al. [2019] is a corpus of Czech transcribed speech. In our work, we use this dataset for the training of Czech ASR. The corpus has approximately 400 hours. As usual, the dataset contains training, evaluation, and test sets with no overlap. Training and development sets may have some common speakers. The test set should have no speaker overlap with the rest of the corpus.

A notable contrast with other speech corpora is the segmentation. The authors segmented the recordings into short utterances (the longest one has 44s), disregarding the sentences.

[2] `http://www.voxforge.org/`

1.5.4 CzEng 1.7

Czeng 1.7 is a Czech-English parallel corpus containing about half a billion words. Czeng 1.7 is a filtered version of Czeng 1.6 [Bojar et al., 2016]. The advantage of the Czeng corpus is its rich origins such as subtitles, EU legislation, fiction, web pages, technical documents, or medical data.

Filtered CzEng

For our purposes, we distilled out all sentence pairs that "probably" do not occur in spoken language. We assume the following: characters, numbers, apostrophe, punctuation, currency sign, dash, and quotation marks on the English side mark the sentence pair as a "probable" spoken utterance. Moreover, we filtered out too long and too short sentences (sentences must have at least two characters, at most 511 characters).

We intend to filter out sentences as for example

E-006961/11 (PL) Marek Henryk Migalski (ECR) to the Commission (15 July 2011)

Such utterances do not have straightforward pronunciation, could break `phonemizer`, and could degrade the transcript and translation quality.

1.6 Error Metrics

In this work, we use two error metrics, one for measuring the quality of speech recognition systems — word error rate (WER) — and a metric for evaluation of spoken language translation — BLEU.

1.6.1 Word Error Rate

Word error rate is one of the most commonly used metrics. This metric measures edit distance (insertions, deletions, and substitutions) between target and hypothesis:

$$WER = 100 \times \frac{I + D + S}{N} \tag{1.4}$$

where

- I is number of insertions,
- D is number of deletions,
- S is number of substitutions,
- N is number of words in target.

Similarly is defined character error rate where, instead of words, are characters considered.

1.6.2 BLEU

For the evaluation of translation tasks, we use BLEU (Bilingual Evaluation Understudy) score introduced by Papineni et al. [2002]. This metric assigns scores in the interval $[0, 1]$, where 1 is the perfect score. The method counts occurrences of matching n-grams in candidate and reference (there can be more than one reference translations). It computes a modified n-gram precision (number of occurrences of an n-gram in candidate sentence must not exceed the maximum number of occurrences in any reference, otherwise is clipped) and does a weighted geometric mean. In addition to the implicit penalization of length, the metric introduces the brevity penalty.

Usage of this metric used to lead to confusion previously because its actual implementation could vary. We therefore use in our work `SacreBLEU` proposed by Post [2018].

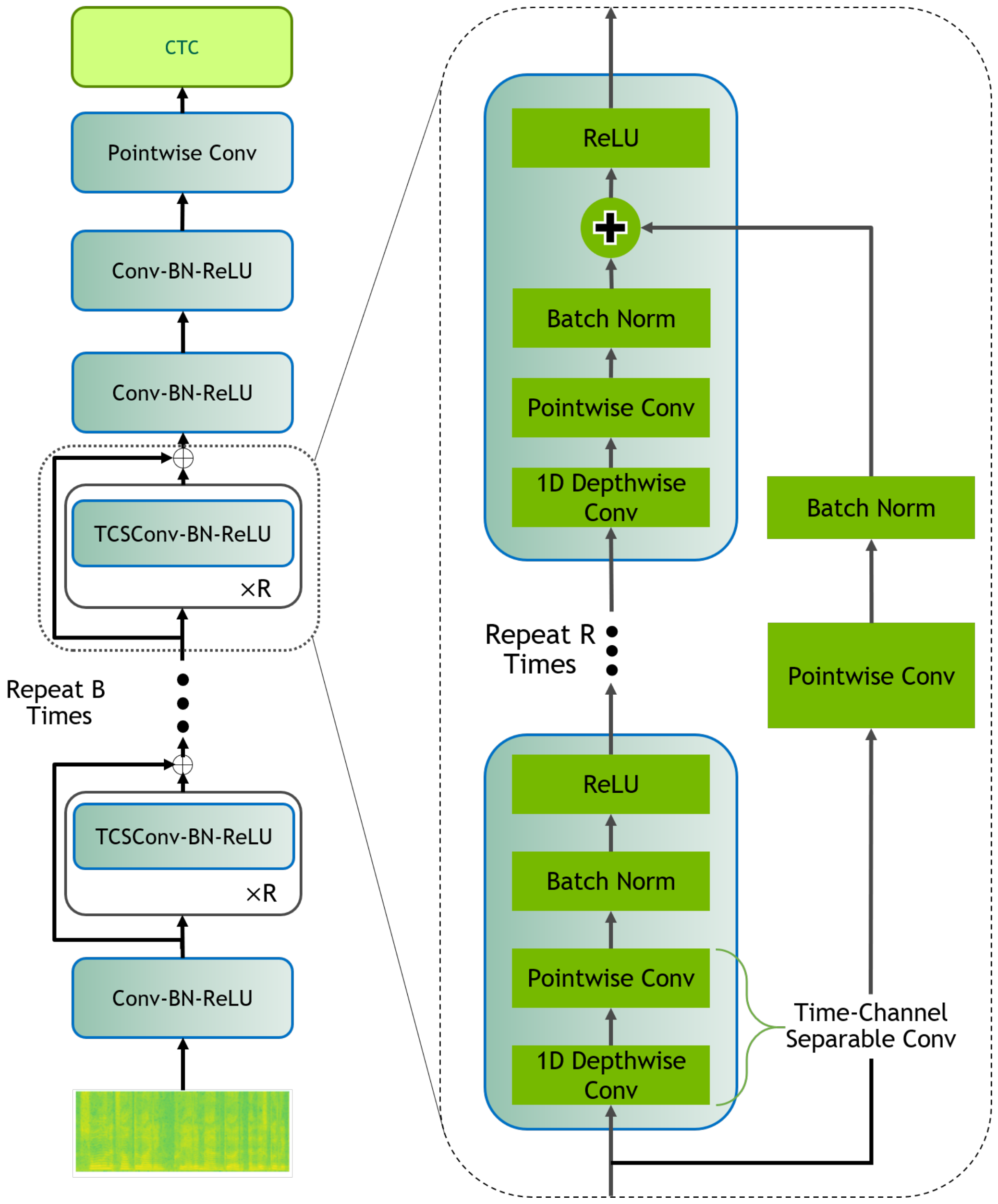

Figure 1.2: QuartzNet BxR architecture. Taken from Kriman et al. [2019].

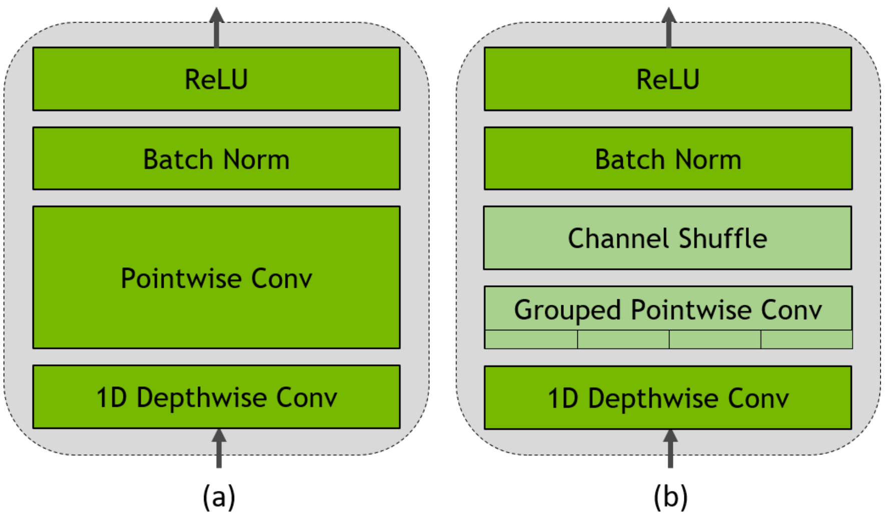

Figure 1.3: (a) Time-channel separable 1D convolutional module (b) Time-channel separable 1D convolutional module with groups and shuffle. Taken from Kriman et al. [2019].

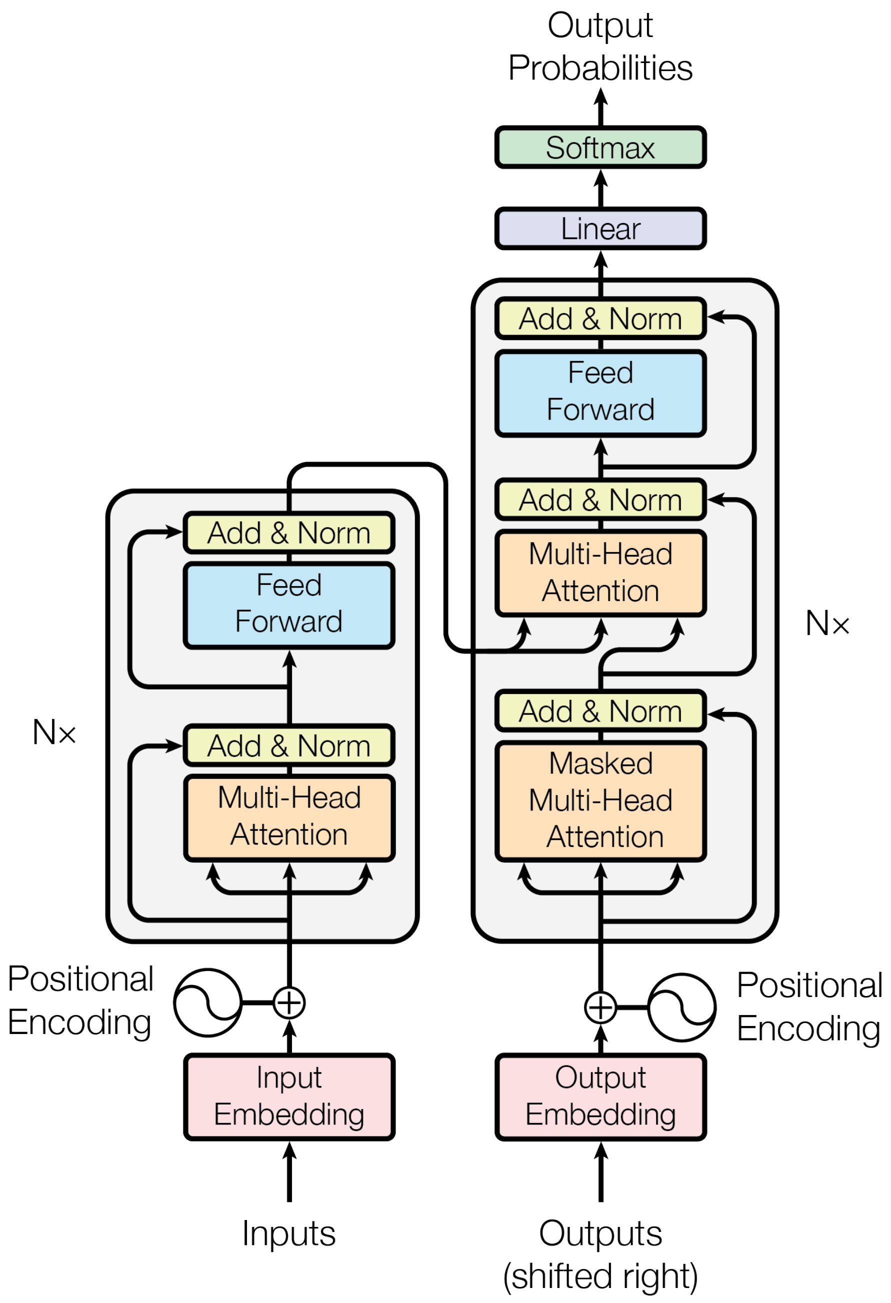

Figure 1.4: Transformer model architecture with detailed encoder (left) and decoder (right). Taken from Vaswani et al. [2017]

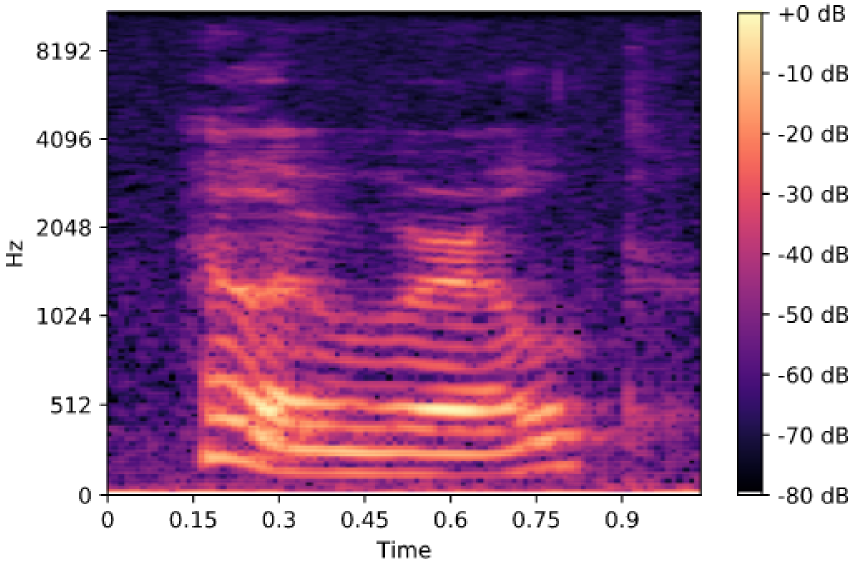

Figure 1.5: Mel spectogram of “Hello World!”.

2. Automatic Speech Recognition

This chapter is devoted to Automatic Speech Recognition (ASR). Throughout this chapter, we describe in detail the process of training an ASR pipeline. One of the central challenges of this work is the scarcity of Czech training data. We propose a technique to overcome this problem.

This chapter is organized as follows: Section 2.1 presents and compares two techniques for overcoming the shortage of Czech ASR data. Section 2.2 describes transfer learning from graphemes to phonemes. the training of acoustic models for the final ASR/SLT pipeline.

2.1 Coarse-to-Fine Intermediate Step and Cross-Lingual ASR Transfer

In this section, we review techniques for overcoming training data shortage. Note, this section was submitted to the Interspeech conference. The differences between this section and the paper are only subtle: we added a description of QuartzNet architecture (see Section 1.3.2), made the tables and figures a bit smaller and detailed, and shortened the text a bit (to adhere to the 4-page limit).

2.1.1 Introduction

Contemporary end-to-end, deep-learning automatic speech recognition systems achieved state-of-the-art results on many public speech corpora, see e.g. Han et al. [2019] on test-clean set from LibriSpeech [Panayotov et al., 2015].

To outperform traditional hybrid models, deep-learning ASR systems, however, must be trained on vast amounts of training data, on the order of a thousand of hours. Currently, there is only a limited number of public datasets that meet these quantity criteria. The variety of covered languages is also minimal. In fact, most of these large datasets contain only English. Although new speech datasets are continually emerging, producing them is a tedious and expensive task.

Another downside of new end-to-end speech recognition systems is their requirement of an extensive computation on many GPUs, taking several days to converge, see e.g., Karita et al. [2019].

These obstacles are often mitigated with the technique of transfer learning [Tan et al., 2018] when a trained model or a model part is reused in a more or less related task. Furthermore, it became customary to publish checkpoints alongside with the neural network implementations and there emerge repositories with pre-trained neural networks such as *TensorFlow Hub*[1] or *PyTorch Hub*.[2] This gives us an opportunity to use pre-trained models, but similarly, most of the published checkpoints are trained for English speech.

In our work, we propose a Coarse-to-Fine Intermediate Step and experiment with transfer learning [**?**], i.e., the reuse of pre-trained models for other tasks. Specifically,

[1]`https://tfhub.dev/`
[2]`https://pytorch.org/hub/`

we reuse the available English ASR checkpoint of QuartzNet [Kriman et al., 2019] and train it to recognize Czech speech instead.

This section is organized as follows. In Section 2.1.2, we give an overview of related work. In Section 2.1.3 we describe used models and data. Our proposed method is described in Section 2.1.4, and the results are presented and discussed in Section 2.1.5. Finally, in Section 2.1.6 we summarize the work, and we give a brief outlook of our future plans.

2.1.2 Related Work

Transfer learning [Tan et al., 2018] is an established method in machine learning because many tasks do not have enough training data available, or they are too computationally demanding. In transfer learning, the model of interest is trained with the help of a more or less related "parent" task, reusing its data, fully or partially trained model, or its parts.

Transfer learning is step by step becoming popular in various areas of NLP. For example, transferring some of the parameters from parent models of high-resource languages to low-resource ones seems very helpful in machine translation [Zoph et al., 2016, Kocmi and Bojar, 2018].

Transfer learning in end-to-end ASR is studied by Kunze et al. [2017]. They show that (partial) cross-lingual model adaptation is sufficient for obtaining good results. Their method exploits the layering of the architecture. In essence, they take an English model and freeze weights in the upper part of the network (closer to the input). Then they adapt the lower part for German speech recognition yielding very good results while reducing training time and the amount of needed transcribed German speech data.

Other works concerning end-to-end ASR are Tong et al. [2018] and Kim and Seltzer [2018]. The former proposes unified IPA-based phoneme vocabulary while the latter suggests a universal character set. The first demonstrates that the model with such alphabet is robust to multilingual setup and transfer to other languages is possible. The latter proposes language-specific gating enabling language switching that can increase the network's power.

Multilingual transfer learning in ASR is studied by Cho et al. [2018]. First, they jointly train one model (encoder and decoder) on ten languages (approximately 600 hours in total). Second, they adapt the model for a particular target language (4 languages, not included in the previous 10, with 40 to 60 hours of training data). They show that adapting both encoder and decoder, boosts performance in terms of character error rate.

Coarse-to-fine processing [Raphael, 2001] has a long history in NLP. It is best known in the parsing domain, originally applied for the surface syntax [Charniak et al., 2006] and recently for neural-based semantic parsing [Dong and Lapata, 2018]. The idea is to train a system on a simpler version of the task first and then gradually refine the task up to the desired complexity. With neural networks, coarse-to-fine training can lead to better internal representation, as e.g., Zhang et al. [2018] observe for neural machine translation.

The term coarse-to-fine is also used in the context of hyperparameter optimization, see e.g., Moshkelgosha et al. [2017] or the corresponding DataCamp class,[3] to cut down

[3] `https://campus.datacamp.com/courses/hyperparameter-tuning-in-python/informed-search?ex=1`

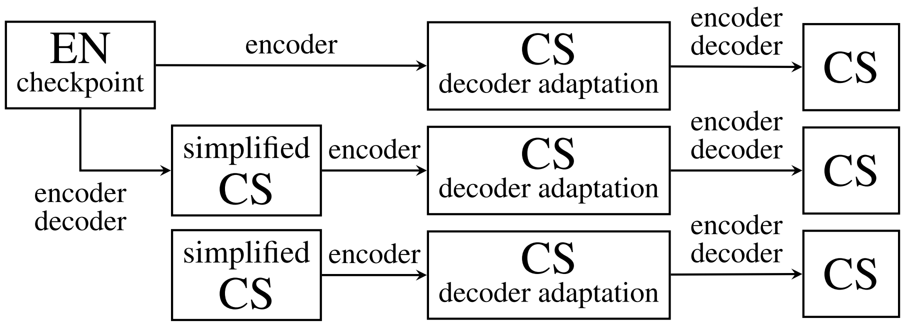

Figure 2.1: Examined setups of transfer learning. The labels on the arrows indicate which model parts are transferred, i.e., used to initialize the subsequent model. No parameter freezing is involved except for the encoder weights in the "CS decoder adaptation" phase.

the space of possible hyperparameter settings quickly.

Our work is novel and differs from the abovementioned twofold: first, we reuse existing models and checkpoints to improve the speed and accuracy of an unrelated language. Second, in the Coarse-to-Fine Step method, we simplify, instead of unifying, Czech character set in order to improve cross-lingual transfer, but also to enhance monolingual training significantly.

2.1.3 Data and Models Used

Pre-Trained English ASR. We use teh checkpoint available at the *NVIDIA GPU Cloud*.[4] It is trained for 100 epochs with batch size 512 on 8 NVIDIA V100 GPUs and achieves 3.98 % WER on LibriSpeech [Panayotov et al., 2015] test-clean.

During the experiments, the model configuration provided by the NeMo authors is used with minor changes (we used 1000 warm-up steps and for decoder adaptation learning rate 10^{-3}). We note that we use $O1$ optimization setting. That is mixed-precision training (weights are stored in single precision, gradient updates are computed in double precision). We perform training on 10 NVIDIA GeForce GTX 1080 Ti GPUs with 11 GB VRAM.

Czech Speech Data. In our experiments, we use Large Corpus of Czech Parliament Plenary Hearings [Kratochvíl et al., 2019]. At the time of writing, it is the most extensive available speech corpus for the Czech language, consisting of approximately 400 hours.

The corpus includes two held out sets: the development set extracted from the training data and reflecting the distribution of speakers and topics, and the test set which comes from a different period of hearings. We choose the latter for our experiments because we prefer the more realistic setting with a lower chance of speaker and topic overlap.

A baseline, end-to-end Jasper model, trained on this corpus for 70 epochs, has an accuracy of 14.24 % WER on the test set.

[4]`https://ngc.nvidia.com/catalog/models/nvidia:quartznet15x5`

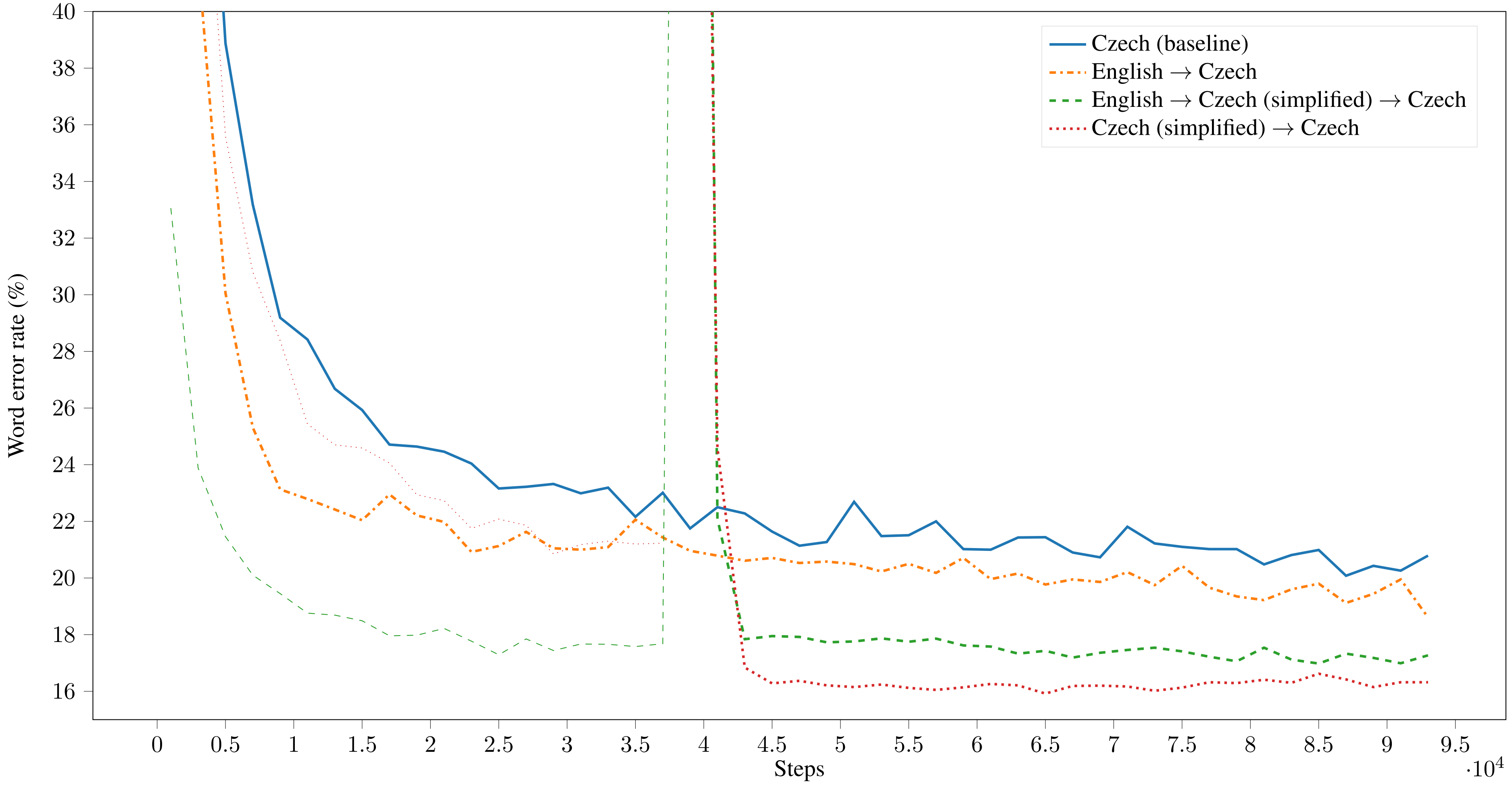

Figure 2.2: Evaluation on test set during training. Note, that WER curves for experiments with simplified vocabulary (thin lines) are not directly comparable with other curves until step 40,000 as the test set is on different (simplified) vocabulary. 10,000 steps takes approximately 5 hours of time.

2.1.4 Examined Configurations

Figure 2.1 presents the examined setups. In all cases, we aim at Czech ASR. The baseline (not in the diagram) is to train the network from scratch on the whole Czech dataset, converting the speech signal directly to Czech graphemes, i.e., words in fully correct orthography, except punctuation and casing which are missing in both the training and evaluation data.

Basic Transfer Learning

The first method is very similar to Kunze et al. [2017]. We use the English checkpoint with the (English) WER of 3.98 % on LibriSpeech test-clean, and continue the training on Czech data.

Czech language uses an extended Latin alphabet, with diacritic marks (acute, caron, and ring) added to some letters. This extended alphabet has 42 letters, including the digraph “ch”. Ignoring this digraph (it is always written using the letters “c” and “h”), we arrive at 41 letters. Only 26 of them are known to the initial English decoder.

To handle this difference, we use a rapid decoder adaptation. For the first 1500 steps, we keep the encoder frozen and train the decoder only (randomly initialized; Glorot uniform).

Subsequently, we unfreeze the encoder and train the whole network on the Czech dataset.

Transfer Learning with Vocabulary Simplification

In this experiment, we try to make the adaptation easier by first keeping the original English alphabet and extending it to the full Czech alphabet only once it is trained.

To coerce Czech into the English alphabet, it is sufficient to strip diacritics (e.g. convert “čárka” to “carka”). This simplification is quite common in Internet communication but it always conflates two sounds ([ts] written as “c” and [tʃ] written as “č”) or their duration ([aː] for “á” and [a] for “a”).

In this experiment, we first initialize both encoder and decoder weights from the English checkpoint (English and simplified Czech vocabularies are identical so the decoder dimensions match), and we train the network on the simplified Czech dataset for 40 thousand steps.

The rest (adaptation and training on the full Czech alphabet) is the same as in Section 2.1.4. Overall, this can be seen as a simple version of coarse-to-fine training where a single intermediate model is constructed with a reduced output alphabet.

Vocabulary Simplification Only

In the last experiment, we disregard the English pre-trained model and use only our vocabulary simplification trick. We first train the randomly initialized model on Czech without diacritics (26 letters) for 38 thousand steps. We then adapt the pre-trained, simplified Czech model to the full Czech alphabet with diacritics (41 letters), again via the short decoder adaptation. Note that the original decoder for simplified Czech is discarded and trained from random initialization in this adaptation phase.

Experiment	Simplified	Adaptation phase	Full training
Baseline CS	-	-	20.19
EN $\rightarrow$ CS	-	97.35	19.06
EN $\rightarrow$ sim. CS $\rightarrow$ CS	17.11	22.01	16.88
sim. CS $\rightarrow$ CS	20.56	24.59	16.57

Table 2.1: Results in % of word error rate on the Czech (CS) test set. Note, all results are obtained on the same test set with following difference: "Simplified" reflects WER on Czech without accents (both hypothesis and reference stripped, e.g., "čárka" as "carka"). "Adaptation phase" and "Full training" already use the original, unmodified test set.

2.1.5 Results and Discussion

Table 2.1 presents our final WER scores and Figure 2.2 shows their development through the training. For simplicity, we use greedy decoding and no language model. We do not use a separate development. We simply take the model from the last reached training iteration.[5]

Transfer across Unrelated Languages

We observe that initialization with an unrelated language helps to speed-up training. This is best apparent in "English $\rightarrow$ Czech simplified" where the unchanged vocabulary allows us to reuse all the weights. WER drops under 30 % after only 2000 steps (1 hour). This can be particularly useful if the final task does not require the lowest possible WER, such as sound segmentation.

If the target alphabet is altered ("English $\rightarrow$ Czech"), we observe a speed-up at the beginning of the training. Our setting with QuartzNet and as well as these results are similar to Kunze et al. [2017] with a high $k = 18$. However, this advantage diminishes with longer training, gaining only 1 to 2 % points of WER over the baseline in the end.

It seems, though, that in the experiments with different vocabularies, after the change to the full Czech vocabulary, the results are better for a randomly initialized network, rather than the one with transfer from English.

Transfer across Target Vocabularies

In the course of two experiments, we altered the target vocabulary: the training starts with simplified Czech, and after about 40,000 steps, we switch to the full target vocabulary. This sudden change can be seen as spikes in Figure 2.2. Note that WER curves prior to the peak use the simplified (the same test set, but with stripped diacritics) Czech reference, so they are not directly comparable to the rest.

The intermediate simplified vocabulary always brings a considerable improvement. In essence, the final WER is lower by 2.18 (16.88 vs. 19.06 in Table 2.1) for the models transferred from English and by 3.62 (16.57 vs. 20.19) for Czech-only runs. One possible reason for this improvement is the "easier" intermediate task of simpler Czech. Although the exact difficulty is hard to compare as the target alphabet is smaller,

[5]Little signs of overfitting are apparent for the "Simplified CS $\rightarrow$ CS" setup so that an earlier iteration might have worked better, but we do not have another test set with unseen speakers to validate it.

Set	Greedy	Beam search with LM
Development	5.19	4.91
Test	7.64	6.42

Table 2.2: Our best Czech model. Results in % WER measured after fixing errors in development and test set transcriptions.

but more spelling ambiguities may arise. This task helps the network to find a better-generalizing region in the parameter space. Another possible reason that this sudden change and reset of the last few layers allows the model to reassess and escape a local optimum in which the "English $\rightarrow$ Czech" setup could be trapped.

2.1.6 Conclusion and Future Work

We presented our experiments with transfer learning for automated speech recognition between unrelated languages. In all our experiments, we outperformed the baseline in terms of speed of convergence and accuracy.

We gain a substantial speed-up when training Czech ASR while reusing weights from a pre-trained English ASR model. The final word error rate is better only marginally in the basic transfer learning setup.

We are able to achieve a substantial improvement in WER by introducing an intermediate step in the style of coarse-to-fine training, first training the models to produce Czech without accents, and then refining the model to the full Czech. Our final model for Czech is better by over 3.5 WER absolute over the baseline, reaching WER of 16.57%. Further gains are expected from beam-search with language model or better iteration choice to avoid overfitting.

We see further potential in the coarse-to-fine training. We would like to explore this area more thoroughly, e.g., by introducing multiple simplification stages or testing the technique on more languages.

2.1.7 Note on the results

After reviewing the Large Corpus of Czech Parliament Plenary Hearings, we found out that the transcripts contained systematic errors (SIL tag for silence was accidentally left in at the beginnings and ends of some transcripts). This resulted in systematic errors during testing.

Unfortunately, we made this finding after discarding some of the models, so we were unable to re-run the test. We kept only the best model. After fixing the transcript errors in development and test set — no model retraining — the model obtained marvelous results: WER of 4.91 % on the development and 6.42 % on the test set using beam search with language model (see Table 2.2). These results are thus ten % points better than initially stated in Table 2.1.

2.2 Speech recognition to phonemes

In this section, we build ASR systems for English and Czech with phonemes as target vocabulary. During the training, we build upon our findings from the previous section.

This section is organized as follows: first, in Section 2.2.1 we review related work and discuss the motivation for the transition from graphemes to phonemes as target vocabulary. Next, in Section 2.2.2 we present the experiment set-up. Section 2.2.3 gives details about training. Finally, we evaluate results in Section 2.2.4.

2.2.1 Related work

Phonemes are well-established modeling units in speech recognition. From the beginnings of the ASR technology in the 1950s, researchers employed phonemes [Juang and Rabiner, 2005].

For the GMM-HMM models, a state typically represents part of a phone [Yu and Deng, 2016]. Phone is a linguistic unit representing a sound regardless of the meaning in a particular language. On the other hand, we have phonemes that are also linguistic units representing a sound. However, opposed to the phones, if a phoneme is switched for another, it could change the meaning of a word in a particular language. A common trick for improving quality is to combine phones into diphones (two consecutive phones) or triphones (three successive phones) [Kamath et al., 2019].

Riley and Ljolje [1992] attend the comparison of different linguistic units for sound representation. Namely, they compare phonemic (using phones), phonetic (using phonemes), and multiple phonetic realizations with associated likelihoods. All experiments use the GMM-HMM ASR model. Using the phonemic representation, the authors were able to achieve 93.4 % word accuracy on DARPA Resource Management Task. Using the most probable phonetic spelling, the authors improved the result to 94.1 %. Authors also propose using more phonetic realizations, which are obtained automatically from the TIMIT phonetic database. In their best setup, which achieves an accuracy of 96.3 %, they report the average 17 representations per word.

Important work popularizing neural networks in ASR is Waibel et al. [1989] published at the end of the 1980s. Work proposes to use a time-delayed neural network to model acoustic-phonetic features and to model the temporal relationship between them. Authors demonstrate that the proposed NN can learn shift-invariant internal abstraction of speech and use this to make a robust final decision. Authors choose phonemes as a modeling unit.

More recent studies that still use hybrid models (mainly, HMM and DNN) employ multi-task learning to improve phone recognition task significantly.

For example, Seltzer and Droppo [2013] improves the primary task of predicting acoustic states (61 phonemes, three states each) using a different secondary assignment. They propose three additional problems: (1) Phone Label Task, where the system predicts phone labels for each acoustic state. (2) State Context Task, in which the model predicts previous and following acoustic state. (3) Phone Context Task, where the model predicts previous and next phone.

Another example of multi-task learning in ASR is Chen et al. [2014]. The authors use DNN for the triphone recognition task. To improve the performance in this task, they suggest using the additional assignment of trigrapheme recognition, which they claim to be highly related learning tasks. In their setup, both problems share the internal representation (hidden layers of DNN). Authors successfully demonstrate that their contribution outperforms other single task methods, even the ROVER system [Fiscus, 1997], that combines both the abovementioned tasks.

Finally, though not directly from the ASR field, an excellent utilization for phoneme ASR in Spoken Language Translation suggests Salesky et al. [2019]. For their End-to-End SLT pipeline, they first obtain phonemes alignment using the DNN-HMM system. They average feature vectors with the same phoneme for consecutive frames. Phonemes then serve as input features for End-to-End SLT.

2.2.2 Experiment set-up

Based on the abovementioned review of related work, we design our experiment. The main goal of our work is to build an SLT system. This has a significant impact on our decision-making about the rough outline.

We target our ASR to phonemes rather than graphemes. We presume they are more suitable linguistic modeling units for the recognition task, while they keep the problem similar to the grapheme recognition task (mainly supported by Riley and Ljolje [1992], Juang and Rabiner [2005], Chen et al. [2014]). XXX This especially holds for Czech, as the phonetical and graphical transcriptions are relatively alike. We specifically decided for IPA phonemes and not for other phonetic units like phones. We believe that they are adequate and we can quickly get phonetical transcription for most languages using `phonemizer`[6] tool.

We find the End-to-End system promising for the future. Therefore, unlike the DNN-HMM architecture used by the Salesky et al. [2019], we bet on E2E architecture. Notably, for English, we employ bigger Jasper [Li et al., 2019a], and for Czech, we utilize smaller QuartzNet [Kriman et al., 2019]. Our decision to use different architectures for English and Czech is motivated by the volume of the training data. For English, we have almost 2000 hours and for Czech only about 400 hours of transcribed speech.

We take advantage of the fact that the grapheme and phoneme recognition tasks are similar. We employ transfer learning from pre-trained grapheme models. Inspired by our previously proposed "Adaptation method" (see Section 2.1.4), we first strip the "grapheme" decoder and adapt the model (only a randomly initialized decoder) for new modeling units — phonemes. After the adaptation, we train the whole model.

2.2.3 Training

In this section, we review the training of the experiment, as described in Section 2.2.2. Note, we have different training plans for English and Czech. Common are the schemes: first, the Adaptation phase, and second, the Full training.

We use mixed-precision training, which means that the weights are stored in `float16` (making the model smaller), and weight updates are computed in `float32` precision (reducing the quality loss).

When it comes to phoneme segmentation, technical question arises. Both trained architectures output character labels. But, some phonemes are multi-character (for example "aː" or "dʒ"). We already dealt with this problem in the previous section. Czech alphabet has digraph "ch". We decided to disregard this character in the ASR to

[6] `https://github.com/bootphon/phonemizer`

Czech graphemes as it can be easily rewritten using “c” and “h”. Here is the problem more complicated. Many phonemes in both languages are multi-characters (Czech has 13 and English 27). Hence, disregarding these phonemes might influence model performance. Bearing this in mind, we conduct two additional experiments. The first experiment obeys the phoneme segmentation, and the second brakes all phonemes to single characters. As the English model has much greater hardware requirements, we train only Czech variants and generalize the results to English.

We first attempt to follow a phoneme segmentation given by the `phonemizer`. `phonemizer` can output phonetic transcriptions including separators between phonemes and words. We use “|” as phoneme separator.

In the second attempt, we break all multi-character phonemes. Additionally, we break single-character phonemes that consists of combining letters and we replace these special letters with a placeholder. For English we substitute “ʲ” with “J”, combining tilde (like “ɔ̃”) with “I”, and vertical line below (e.g., “n̩”) with “L”. For Czech, we introduce placeholders for vertical line below (e.g., “l̩”) with “L”, for up tack below (e.g., “r̝”) with “T” and ring above (e.g., “r̊”) with “R”. So, for example Czech word “případně” has phonetical transcription “pr̝̊iːpadɲe”, which is rewritten to “prTRiːpadɲe”.

Czech ASR For training and testing, we use the “phonemized” corpus of the Czech Parliament Plenary Hearings. As the dataset is five times smaller than English corpora, we utilize the smaller QuartzNet.

We train off the best model from the previous experiment described in Section 2.1. This model yields on the Parliament corpus WER of 4.91 % on development and 6.42 % test set using beam search with decoding.

The Adaptation Step takes 2000 steps. As the model’s memory footprint is smaller during this phase, we increase the batch size to 256. Two thousand steps are warm-up, the maximal learning rate is 4×10^{-3}.

Full training takes 30000 steps. The model memory requirements increases, therefore we reduce the batch size to 32. We also reduce the learning rate to 10^{-3}.

Both experiments with phonemes alphabets use the same configuration.

English ASR For training and evaluation, we translate to phonemes (using `phonemizer` tool) corpora LibriSpeech and Mozilla Common Voice. We use both (shuffled) datasets for the Adaptation and Full training.

The training is executed on 10 NVIDIA GTX 1080 Ti GPUs with 11 GB VRAM.

We train off the Jasper ASR model available online[7]. This model was trained on LibriSpeech, Mozilla Common Voice, WSJ, Fisher, and Switchboard corpora. The model yields on LibriSpeech 3.69 % on test-clean, and 10.49 % on test-other using greedy decoding.

The Adaptation Step takes 2000 steps. As the model’s memory footprint is smaller during this phase, we increase the batch size to 64 (global batch size is 640). One thousand steps are warm-up; the maximal learning rate is 4×10^{-3}.

The Full training takes ten epochs. The model memory requirements increases, therefore we reduce the batch size to 16 (global batch size is 160). We also reduce the learning rate to 10^{-3}.

[7]`https://ngc.nvidia.com/catalog/models/nvidia:multidataset_jasper10x5dr`

2.2.4 Evaluation

In this section we review the course of training and evaluate model's performance.

Note, the results are not directly comparable with graphemes. The mapping to phonemes is not a bijection. For example in American English *I am* maps to only one "phoneme word" [aɪæm].

Czech Course of training is displayed in Figure 2.3 and final evaluation are in Table 2.3.

As expected, rapid PWER fall at the beginning of the training (see Figure 2.4) proves that the grapheme and phoneme recognition tasks are indeed similar.

We made a surprising observation: beam search decoding is always worse than greedy decoding. We tested beam sizes of power two from 1 to 64. We use two different beam search implementations: TensorFlow[8] and Baidu's CTC decoder[9] with KenLM[10] language model.

Using TensorFlow, we got the best result 9.14 %, about 0.2 % percent point higher than with the greedy search for multi-character ASR. For single-character ASR, the difference is even more significant: 9.09 and 9.53 % PWER.

We also tried adding a language model to the second CTC decoder[11]. Using KenLM, we trained a 3-gram language model on phonemized Czeng. For both single- and multi-character phonemes, using the language model helps. The result is for the single-character setup a bit better. We suspect the difference stems from our trick to overcome Baidu's CTC decoder limitation. Training data for language model data does not contain phoneme boundaries. Some multi-character phonemes consists of two other valid single-character ones. Thus, some neighbors might be considered as one phoneme and incorrectly substituted. The ASR model is trained on data with known phoneme segmentation. So, during decoding, the model outputs with higher probability two phonemes instead of one, while the language model gives to these phonemes lower likelihood.

To test this assumption, we train 3-gram language models (for single- and multi-character ASR) on ASR training data (including phoneme segmentation). Note, we do not re-phonemize Czeng, as phonemization is time and power-consuming process. The best results are 8.30 and 8.45 % PWER for multi- and single-character ASR. These results are consistent with the greedy and beam search without LM decoding. Therefore, we conclude that using native phoneme segmentation is slightly better. Hence, for English, we use multi-character phonemes.

English The course of the training was controlled on two development sets and is pictured in Figure 2.4. The final results on development and test sets are in Table 2.4.

We again test beam search with and without a language model. We train 3-gram LM and use ASR training data, as these contain separated phonemes. Obtained results could be, therefore, better if we took more training data.

[8]`https://www.tensorflow.org/`

[9]`https://github.com/PaddlePaddle/DeepSpeech`

[10]`https://github.com/kpu/kenlm`

[11]When setting LM weight to higher than 0, we found that this implementation does not support labels containing more that one character, which causes a problem, as phonetic alphabet contains several multi-character phonemes. To overcome this complication, we substituted these multi-character labels with unused single-character labels (such as capital letters and numbers).

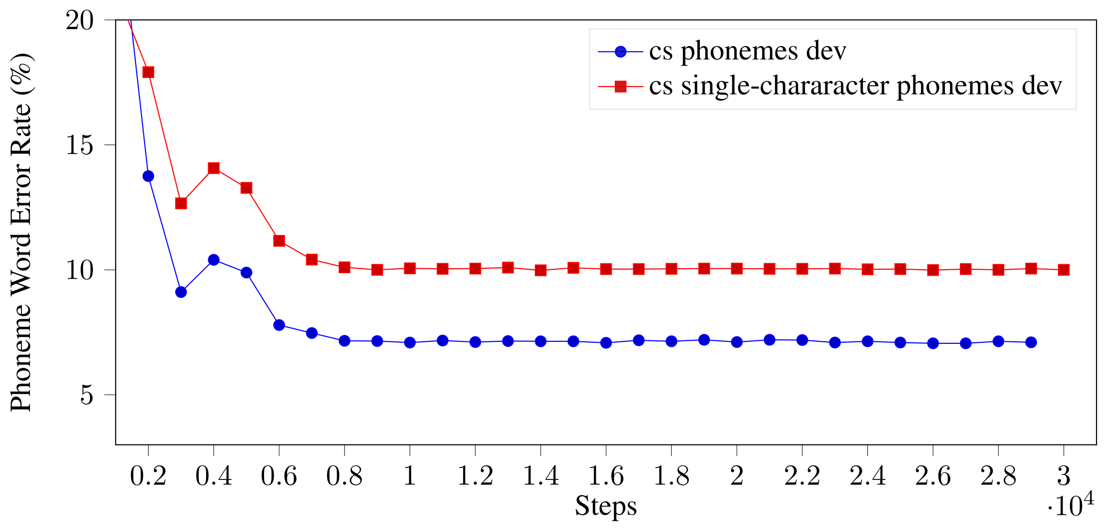

Figure 2.3: Evaluations on phonemized Czech Parliament Hearings development set.

Type	Method	Adaptation phase	Full training
dev	multi-char. phonemes	13.75	7.09
	single-char. phonemes	17.91	10.04
test	multi-char. phonemes	-	8.94 / 9.14† / 8.57‡
	single-char. phonemes	-	9.09 / 9.53† / 7.84‡

Table 2.3: Results in % of *Phoneme* Word Error Rate (PWER) using greedy decoding, except for † and ‡, using beam search with and without language model respectively. The language model is trained on Czeng. Note, PWER is not directly comparable to WER.

Again, beam search decoding performs worse than greedy decoding. Adding a language model to re-score beams helps reduce PWER. There is a slight reduction in the Libri Speech test clean. Test other gets better about three % points PWER. For the Common Voice test set, the PWER reduction is considerable — almost 40 % relative to the greedy decoding. As possible explanation why LM helps Common Voice so much is that nearly two-thirds of the LM training data come from this data set.

2.2.5 Conclusion and future work

We proposed an alternative text representation for ASR — phonemes — and trained models for Czech and English language. Further, we examined two different approaches for phonemes encoding: respecting multi-character phonemes and single-character ones. We concluded that using native, multi-character phonemes is slightly better. We also successfully trained language models for use with phonetic transcriptions.

If the phoneme-based ASR better than grapheme-based ASR, cannot be answered simply. The mapping between graphemes and phonemes is not straightforward. Therefore, we leave it to the next chapter.

In the future, we would like to review coarse-to-fine methods for improving phoneme-based ASR. We believe that similarly to grapheme-based ASR, this could reduce training time and lower final PWER.

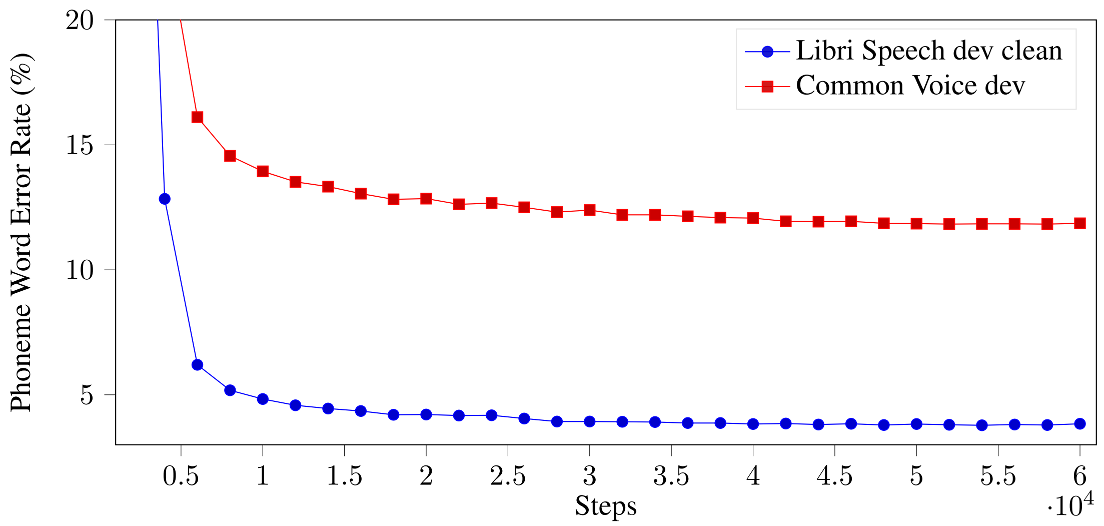

Figure 2.4: Evaluations on phonemized Libri Speech dev clean and Common Voice dev.

Type	**Corpus**	**Adaptation phase**	**Full training**
dev	Libri Speech Clean	46.07	3.84
	Common Voice	54.69	11.86
test	Libri Speech Clean	-	4.18 / 4.48† / 3.58‡
	Libri Speech Other	-	11.48 / 11.67† / 8.57‡
	Common Voice	-	10.21 / 10.47† / 6.46‡

Table 2.4: Results in % of *Phoneme* Word Error Rate (PWER) using greedy decoding (no mark), beam search (†) and beam search with language model (‡). The language model is trained on phonemized ASR training data. Note, PWER is not directly comparable to WER.

Another technique, multi-task learning, proposed and applied in HMM-based ASR by Chen et al. [2014], could source of a further performance boost.

3. Enhanced ASR

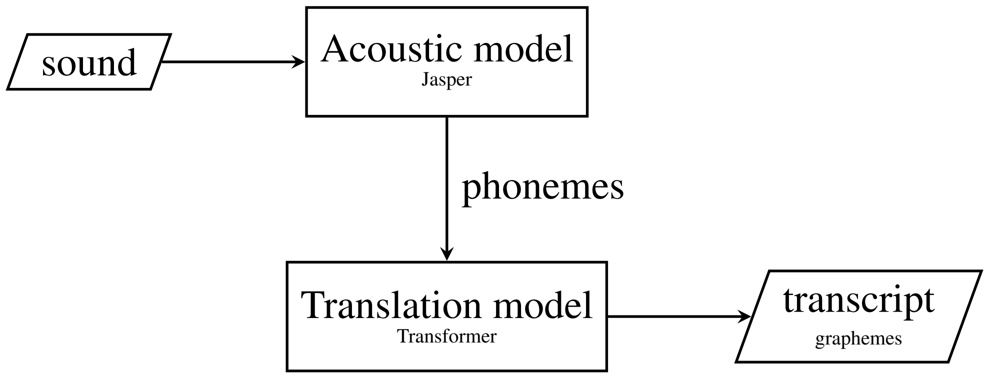

Figure 3.1: Enhanced ASR pipeline. The input sound is first transcribed to phonemes using Jasper/QuartzNet "acoustic" model. Phonemes are fed to the "translation" model. Translation model not only translates, but also fix some errors in the ASR output and produces orthographic transcriptions.

In this chapter we describe and build enhanced ASR. We propose to split a conventional end-to-end ASR into to two successive models:

1. an acoustic model that outputs phonemes instead of graphemes,
2. "translation" model that consumes previously outputted phonemes and translates them into the graphemes.

Illustration of proposed enhanced ASR pipeline is in Figure 3.1.

The main idea is, that the translation model that comes right after acoustic model in our setup not only "blindly" translates phonemes into graphemes, but also corrects errors. Errors can occur for example due to bad conditions during voice recording (e.g. background noise), speaker's dialect or pronunciation errors. Some of these errors may be obscure for a person, as we are naturally able to communicate in noisy environment. The motivation for introduction of such translation step into our pipeline is that such model better understands language and can take longer context into account when compared with plain end-to-end Jasper model. Furthermore, we can utilize other non-speech corpora, e.g. easy obtainable monolingual data, to train and/or finetune part of our pipeline.

We decided to use phonemes as intermediate representation. We believe that conventional grapheme representation is too complicated and constrained for some languages with complicated rules of mapping speech to transcript. This issue becomes more immense when dealing with dialects and non-native speakers.

We step by step describe selection of hyperparameters and training of translation model for the enhanced ASR pipeline. First, we discuss and experiment with source encoding and afterwards we train the model.

The chapter is organized as follows: we first review related work in Section 3.1. In Section 3.3 we experiment with various approaches to text encoding. In Section 3.4 and **??** we present the main objective of this chapter — translation model for ASR.

3.1 Related work

In this section, we review related work. We examine ASR-related work in Section 3.1.1, and in Section 3.1.2 we take a closer look at text encoding.

3.1.1 ASR

We already reviewed usage of phonemes in ASR in Section 2.2.1. At this point, we further expand corresponding work.

One of the main challenges of this chapter is to build phoneme-to-grapheme (P2G) translation model. In the most studies, the P2G translation is utilized for enhancing ASR. More precisely, for out-of-vocabulary (OOV) words, the ASR system outputs phonemes instead of phonemes. P2G then tries to find proper orthographic representation. Examples of studies employing P2G in this manner are Decadt et al. [2001], Horndasch et al. [2006], Basson and Davel [2013].

One of the first attempts on P2G translation is Reddy and Robinson [1968]. Authors propose to use a tree-structure mechanism to keep track of the possibilities at various stages combined with phoneme-to-word dictionary and the structure of the English language. In order to speed-up the translation process, authors divide the dictionary into 10 sections determined by parts of speech. Authors also consider erroneous input. They propose to have more dictionary entries (for example to account for dialects) and human aid.

Another study in P2G translations conduct Decadt et al. [2001]. They apply P2G to enhance readability of out-of-vocabulary (OOV) output in speech recognition. In their setup, ASR outputs standard (orthographic) text for known words. For OOVs, phonemes are outputted. Because the phonemes are hard to read for most users, authors propose to translate phonemes using a memory-based learning. Specifically, they propose two approaches. Fist, to use similarity metric to find closest examples in lexicon (features — phonemes — are weighted using gain ratio) and extrapolate result from them. Second, they propose to use IGTREE algorithm. Authors, surprisingly, report that actual word error rate in their setup (Dutch ASR) is higher. But on the other hand, the output should be better readable for a person. They report that 41 % words are transcribed with most 1 error and 62 % have only two errors. Furthermore, most of the incorrectly transcribed words do not exists in Dutch.

Horndasch et al. [2006] introduce data-driven approach MASSIVE. Their main objective is to find appropriate orthographic representations for dictated internet search. Their system iteratively refines sequences of symbol pairs in different alphabets. In the first step, they find best phoneme-grapheme alignment using expectation-maximization algorithm. In the second step, they cluster neighboring symbols together to account for the insertions. Finally, from n-gram probabilities of symbol pairs are learned. During the inference, input string is split into single symbols. For each symbol is generated all possible symbol pairs. According to beam with, best sequences are taken. On German CELEX lexical database, they obtain 96.1 % word accuracy. For English CMU, they obtain accuracy of 87.2 % for 10-fold cross-validation.

3.1.2 Text encoding

In this subsection we give a brief overview of text encoding related work. High-level review of text representation in NMT can be found in Section 1.4.2. In this part, we study work on various techniques for enhancing translation quality.

Note that some authors use terms "*BPE size*" and "*number of merge operations*" as synonyms. But, the actual *BPE size* equals *number of merge operations* plus *characters*. In most cases, the number of characters is small relative compared to the merge operations. Then is the discrepancy negligible. In our work, we use the term "*BPE size*" as the total number of unique entries in BPE dictionary.

First attempt to study impact of BPE vocabulary size in Neural Machine Translation was made by Denkowski and Neubig [2017]. Specifically, they compare full-word systems with 16k and 32k BPEs. In their setups they use *shared vocabularies* — BPE learning is done on concatenation of source and target data sets. They conclude that using BPE is definitely better than using full-word vocabularies. For BPE, they suggest to use larger vocabulary over smaller one in high-resource setups (in their case over 1M parallel sentences). Reviewing they results, we observe only little performance degradation using 16k (smaller) BPE in high-resource setups (in DE-EN translation task by 0.4 BLEU and no difference for other tasks) and slightly better performance in low-resource tasks (0.3 and 0.4 BLEU for EN-FR and CS-EN respectively).

In different direction — towards character encoding went Cherry et al. [2018]. In their study, the authors compare BPE and character encoding in combination with LSTM NMT. They claim that artificial representations such as BPE are sub-optimal leading to e.g. (linguistically) improbable word fragmentations. Although, they outperform BPE, they acknowledge the problem of much higher computational requirements for both, training and inference.

A deeper study of different setups (architectures, Joint vs Separate BPE, languages) and impact of vocabulary sizes on NMT performance offer Ding et al. [2019]. Authors review several setups and a broad range of BPE sizes ranging from character-level to 32K. They show that using appropriate setting can help gain 3 to 4 BLEU points. Most experiments with smaller vocabularies (sizes up to 4K) performed better for low-resource setting. Although, for high-resource setting the larger BPE sizes are better. Authors also study joint and separate BPE. They conclude that the difference is negligible.

Another authors [Gupta et al., 2019] study character-based and BPE NMT with Transformers under various conditions. They conclude that the BPE with 30K vocabulary is a standard choice in high resource setting. They also experiment with noisy data: when training on clean data, BPE performs slightly worse, however, when training on corrupted data, BPE with large vocabulary (30000) performed better as character level or BPE with smaller vocabulary. In low-resource setting, character lever models perform better. In high-resource setting however, large BPE models outperform other settings. Only exception is WMT Biomedical test set, which contains large proportion of unseen words.

For our specific use case, Hrinchuk et al. [2019] use Bert [Devlin et al., 2018] original 30K WordPieces vocabulary and does not examine other sizes or other training data.

3.2 Experiment motivation and outline

As reviewed in previous section, we see that utilization of intermediate phonetic transcription is an established method in ASR. We therefor propose a pipeline described at the beginning of the chapter (see Figure 3.1).

Unlike the most studies reviewed in Section 3.1.1, we propose to use Transformer architecture for phoneme-to-grapheme translation. We believe that Transformer is best option for this tasks. Transformer has shown its potential in many NLP tasks. The most important we consider its ability to learn a structure of a sentence. We are convinced, this could help reduce errors in transcripts. Our other motivation to apply this architecture is its task versatility. Just by swapping training data, we can easily train spoken language translation system (precisely, translation of phoneme in source language to graphemes in target language).

We describe and evaluate training of Transformer phoneme-to-grapheme translation in Section 3.4 and **??**.

Our survey in Section 3.1.2 clearly demonstrates an important role of text encoding on the translation model performance. Hence, preceding the model training we first study the impact of tokenizer selection on P2G translation in Section 3.3.

3.3 Tokenizer selection

In this section we review alternatives for input and output tokenization. Because of resource scarcity, we conduct the experiment only on English and generalize the conclusion to Czech.

Throughout this experiment we use Byte Pair Encoding for tokenization of source and target sentences. We rule out word representation. It creates models with worse performance (in terms of speed and quality) [Denkowski and Neubig, 2017]. Also, it cannot deal with out-of-vocabulary items.

We decided for separate source and target vocabularies. We are motivated by findings of Ding et al. [2019]. In general, they did not observe difference when using joint and separate vocabularies. They encourage the practitioner to consider particular use case. In our task, the input and output alphabets — phonemes and graphemes — are different. In this experiment we choose 8k BPE for target tokenization. We follow authors of NeMo toolkit[1]. They claim this BPE size should help lower memory footprint and hence increase throughput leading to faster convergence. As training data, we use phonemized Czeng corpus.

As we previously discussed, selection of training data and size of vocabulary for target BPE is relatively straightforward. On the other hand, considering the source may be corrupted as it is produced by ASR system in our setup, there arise two questions:

- is better to use smaller or larger vocabulary size?
- which training data use to train the tokenizer (uncorrupted data translated from CzEng using `phonemizer`, data obtained from from ASR or mixture of both)?

[1]`https://nvidia.github.io/NeMo/nlp/neural_machine_translation.html`

3.3.1 Experiment outline

Our task differs from situations described in previous work. Hence there is no clear answer for our previously stated questions.

In order to resolve these questions we conduct a series of experiments. We train 16 Transformers, each with different source vocabulary (sizes with step of multiple of four: character-level, 128, 512, 2k, 8k and 32k. Each BPE size except for character-level is trained on clean, corrupted and mixture (procedure generating corrupted data is described in Section 5.2). Target vocabulary remains same for all configurations — 8k trained on clean graphemes, phonemized filtered CzEng 1.7. Afterwards we evaluate their performance on “corrupted” dev sets that were obtained in Section 5.2.

Taking into account the time and hardware complexity of training Transformer `big` configuration, we choose the `base` configuration for these experiments. We believe the behavior of the model will still be reasonable alike.

3.3.2 Data preparation

Source BPE vocabularies are trained on: clean filtered Czeng for *clean* setup and corrupted data from ASR ensemble setup (see Section 5.2). We have approximately 7M corrupted sentences. For *mixture* setup, we taken a random subset of 7M from clean filtered Czeng. We do not take whole Czeng as it has 57M sentence pairs.

As training data for Transformers we use corrupted data from our ASR ensemble setup. We selected two development sets: first the *dev clean* set from LibriSpeech and second the *dev* set from Common Voice. The reason is that LibriSpeech contains longer utterances than Common Voice, but on the other hand the former has lower WER tha the latter. It is also worth noting that LibriSpeech *dev clean*’s utterances are twice that long on overage (107 characters versus 52 characters).

3.3.3 Training

As mentioned previously, we use Transformer `base` configuration. We alter maximum sequence length to 1024 because for character-level, 128 and 512 BPE configurations many sentences do not fit into model. We train all models for 70000 steps on one GPU using same batch size for all configurations: 12000 tokens.

Overview of runs is pictured in Figures 3.2 and 3.3. Final results on development sets of all experiments are in Tables 3.1 and 3.2.

3.3.4 Results and analysis

Final results of all experiments are in Tables 3.1 and 3.2 for development sets and in Tables 3.3 to 3.5.

Graphical comparison is in Figures 3.4 to 3.6.

First of all, we can observe a drastic reduction of WER for Common Voice test set compared to Libri Speech test other. Acoustic model performs similarly on both test sets (measured in PWER — see Table 2.4). We take a closer look at the training data. Common Voice filtered has 611990 recordings, but 81334 unique transcripts. This means that same text has been recorded multiple times or by many speaker. Common Voice “corrupted” has 3262524 unique ASR-provided transcritions (meaning that a sentence has a unique error) and 80710 different true transcripts. On the other hand,

Size	**Clean**	**Corrupt**	**Mixed**
character	5.82	-	-
128	6.03	5.69	6.69
512	5.54	5.50	5.49
2k	5.48	5.27	5.46
8k	5.44	5.39	5.47
32k	5.18	5.24	5.25

Table 3.1: Results in % of word error rate on the Libri Speech dev set.

Size	**Clean**	**Corrupt**	**Mixed**
character	7.55	-	-
128	7.38	7.32	7.40
512	7.21	7.27	7.19
2k	7.12	7.20	7.22
8k	7.10	7.05	7.10
32k	6.98	7.03	6.93

Table 3.2: Results in % of word error rate on the Common Voice dev set.

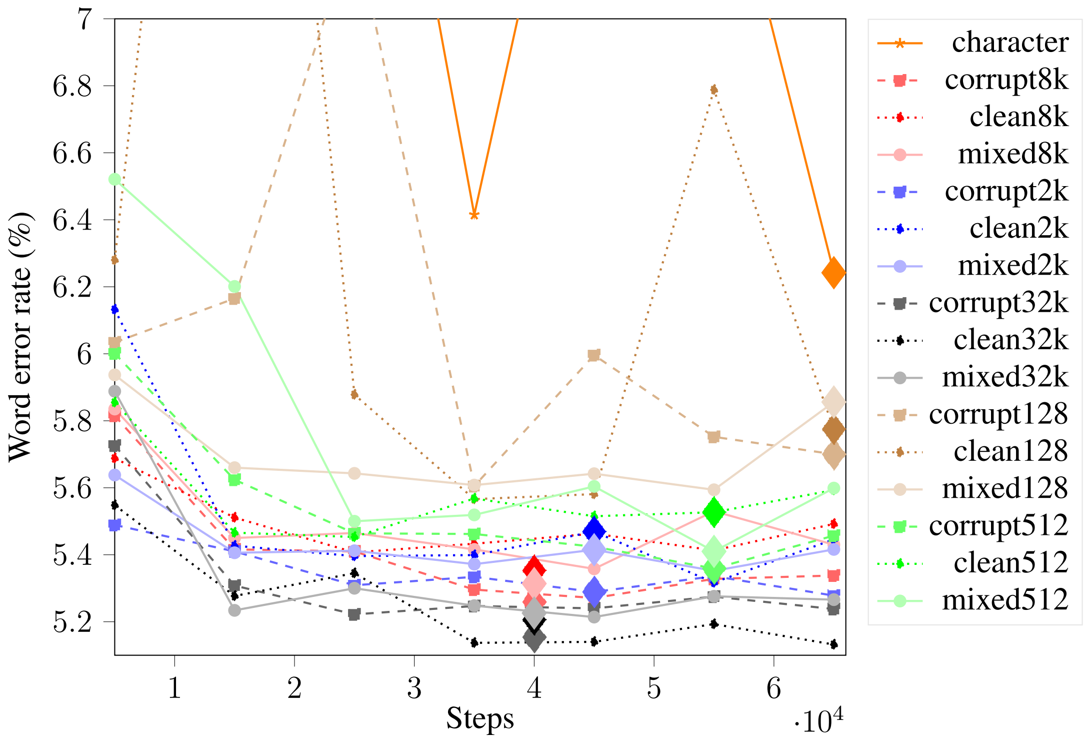

Figure 3.2: Each model is evaluated on LibriSpeech corrupted dev set (see Section 5.2) every 5000 steps. Bigger diamond marks shows where each trained model reached 10th epoch.

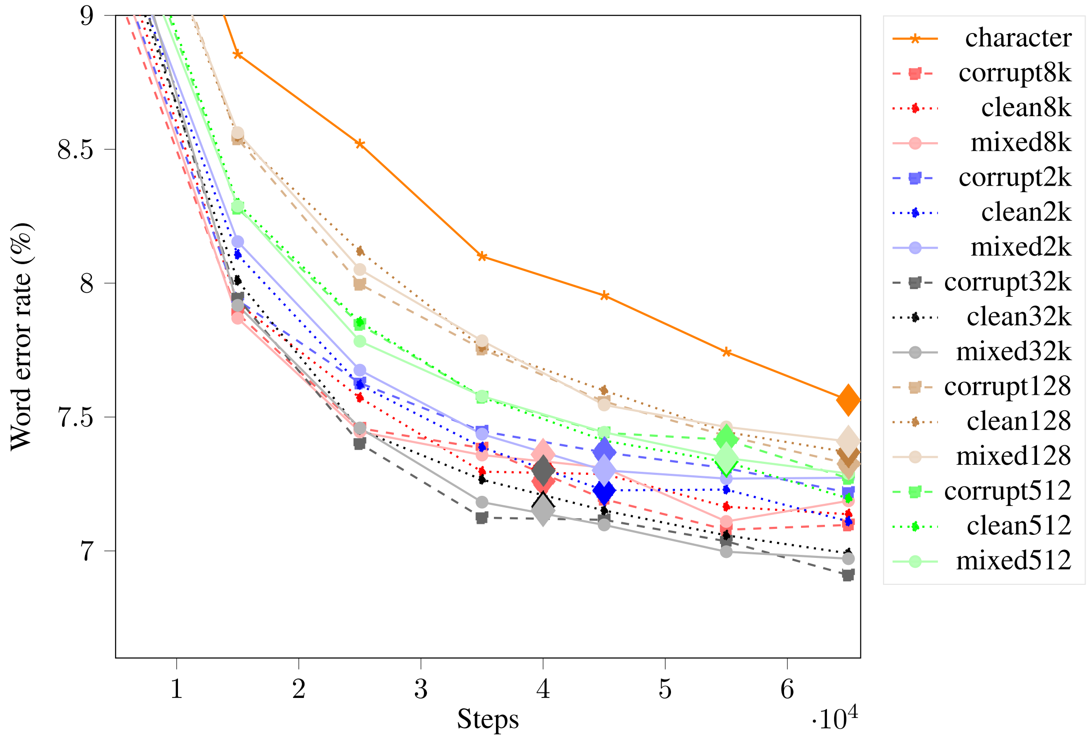

Figure 3.3: Each model is evaluated on Common Voice corrupted dev set (see Section 5.2) every 5000 steps. Bigger diamond marks shows where each trained model reached 10th epoch.

Size	**Clean**	**Corrupt**	**Mixed**
character	5.53	-	-
128	5.06	4.95	5.05
512	5.05	4.79	4.87
2k	4.86	5.09	4.97
8k	4.92	4.99	4.96
32k	4.81	4.65	4.55

Table 3.3: Results in % of word error rate on the Common Voice test set.

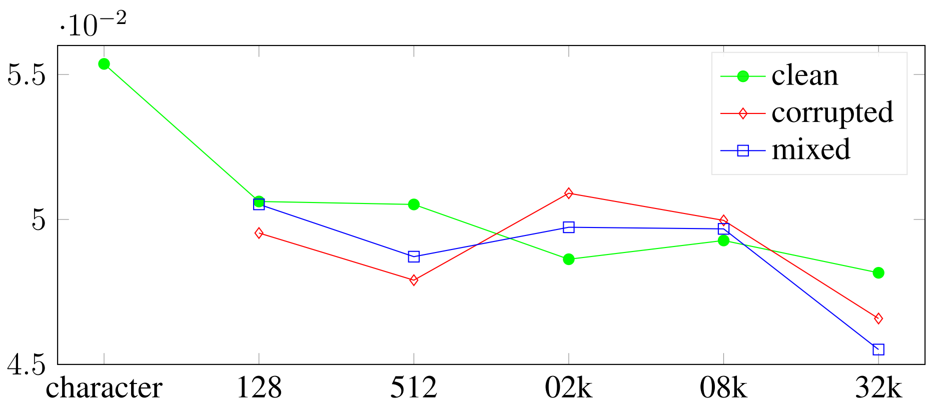

Figure 3.4: Results in % of word error rate on the Common Voice test set.

Size	**Clean**	**Corrupt**	**Mixed**
character	5.64	-	-
128	5.93	5.97	6.04
512	5.40	5.48	5.64
2k	5.34	5.30	5.34
8k	5.30	5.28	5.34
32k	5.19	5.25	5.18

Table 3.4: Results in % of word error rate on the Libri Speech test clean.

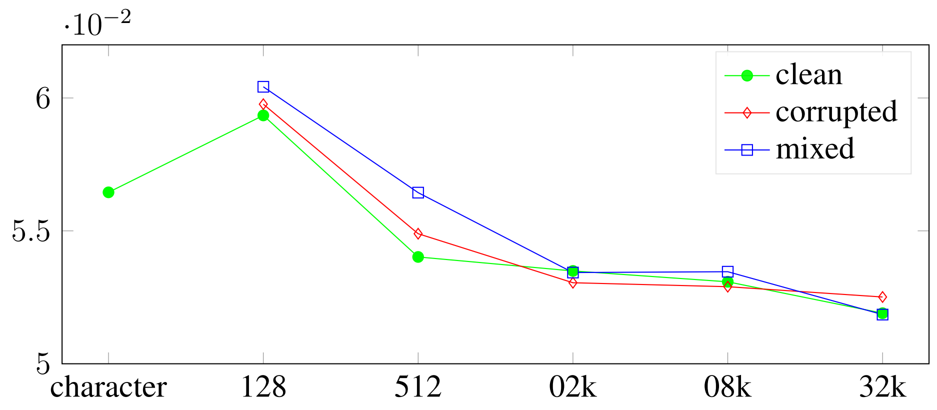

Figure 3.5: Results in % of word error rate on the Libri Speech test clean.

Size	**Clean**	**Corrupt**	**Mixed**
character	11.79	-	-
128	11.98	11.79	12.54
512	11.45	11.59	11.74
2k	11.60	11.45	11.47
8k	11.69	11.56	11.57
32k	11.36	11.43	11.37

Table 3.5: Results in % of word error rate on the Libri Speech test other.

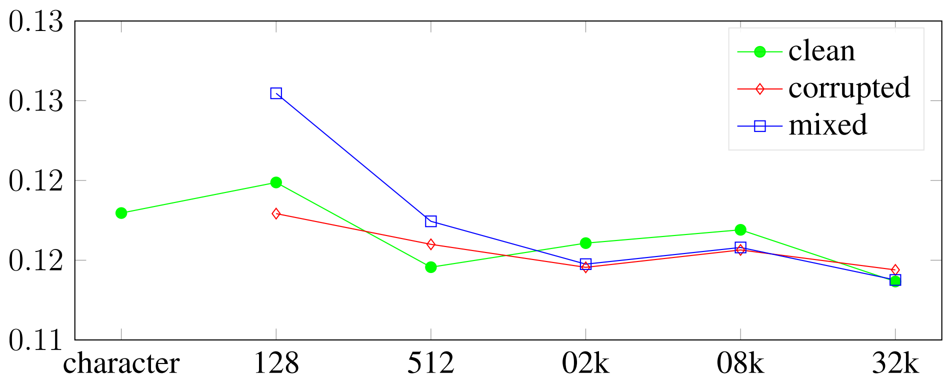

Figure 3.6: Results in % of word error rate on the Libri Speech test other.

Libri Speech has 281241 recordings and 281071 unique transcripts. Hence, there are almost none different recordings for the same text. “Corrupted” Libri Speech has 3844983 unique erroneous ASR transcripts with 280850 different true transcripts. Hence, on average, each Common Voice sentence has 7 different recordings and 40 unique faulty ASR transcriptions. Libri Speech on the contrary does not have more recordings per text and has about 13 ASR-corrupted transcript per one original text. Hence, we are strongly convinced, that the trained Transformer models are over-fitting the Common Voice dataset.

BPE size Character-level encoding seems to be the worst or second worst possible representation. For Common Voice test set, it scores almost one percent point of WER more compared to the best result (5.53 vs. 4.55). Also, all other encodings performed almost half a percent point better. For both Libri Speech test sets it performed a bit better than BPE 128.

Generally, the figures suggest a clear trend: the larger the vocabulary the better. Among the different BPE sizes we can recognize the 32k vocabulary size has systematically best results on all test sets.

Finally, we consider the following: a model can better learn from larger vocabulary sizes. XXX First, a model does not have to learn so much low-level orthography. Rather than memorizing characters (or other smaller units) it can focus on whole sentence and how individual words interacts. Second, a larger model has ability to detect errors because of anomalies in input encoding. Larger vocabularies produce a shorter representation. Corrupted word is more likely to be broken down to smaller peaces. When a model detects such situation, it can for example decide the right target word based on context, rather than the suspicious word. Such anomaly will most likely not occur in text encoded with small BPE.

Source of BPE training data For Common Voice, we can witness some variation in performance. Best seems to be “mixed” configuration. Somewhat worse is “corrupted” and the worst is “clean” version. In this case, we think the “mixed” is best as it has enough frequent “corrupted” words. This enables a model to learn translate these corrupted words to correct ones. Also, it knows enough other words, so it can properly work with correct phonemes.

For other test sets, we observe almost none differences. Only “corrupted” configuration has slightly less performance.

XXX We conclude therefore, that the source of training data for BPE has almost none impact on final result. One should consider to sweep various option for a more specific tasks. For example, we thing it could help as sort of domain adaptation and it could also help for dictated speech task.

3.3.5 Conclusion

We carried out extensive study on impact of BPE vocabulary size and BPE training data source. Based on the empirical evidence, we assume it is better to use bigger vocabularies. This is consistent with Gupta et al. [2019]: in high-resource setting (as ours: we train on 7M sentence pairs) when trained on corrupted data, the bigger vocabularies are better. We do not see much difference between clean, corrupt and mix. But the selection of particular source could be interesting for a specific task.

Therefore, in further experiments and setups we will use 32k BPE vocabulary trained on clean training data.

3.4 English and Czech Enhanced ASR

In this section is described training of the translation model of proposed enhanced ASR pipeline (see Figure 3.1).

3.4.1 Training overview

First, the translation model is trained on clean (no ASR errors) phonemized CzEng data set. Second, we finetune model on ASR corrupted data. As discussed in previos section, we use 32k BPEs for source and target encoding.

3.4.2 Data preparation

For initial training, filtered and phonemized CzEng data set is used. This data set contains approximately 57M parallel sentences. As validation data sets we use following: small portion of phonemized CzEng original test set (3000 sentence pairs), ASR corrupted LibriSpeech `dev clean` and Common Voice `dev` sets.

Finetuning is done on ASR corrupted training data acquired in Section 5.2 while development sets remain same as in the initial training.

3.4.3 First training phase

In this phase we train the model on clean phonemes. We use Transformer `big` architecture. Our configurations:

- GPUs: 8 with 15 GB video RAM,
- batch size: max 9000 tokens,
- learning rate: 0.04,
- warm-up steps: 4000,
- steps: 40000.

We prematurely interrupted the training after 30000 steps, as deallocation of hardware was required and we saw no further improvement on development sets. Note, this differs from training planned for 30000 steps as the learning rate is dependent on maximum steps.

4. Spoken Language Translation

5. Fine-tuning Enhanced ASR

We devote this chapter to fine-tuning ASR pipeline described in Chapter 3.

In order to build the translation part of the pipeline (see Figure 3.1) as robust as possible, we train the translation model to be capable of correcting some errors that are created by ASR. Therefore, in Section 5.2 we describe procedure of obtaining suitable "corrupted" training data which we use later on for making the translation model prone to the ASR-sourced errors.

This chapter is organized as follows: in Section 5.1 we review related work. Sections 5.2 and 5.3 describe process in which we obtain training data with "ASR errors". Finally, **??** reviews setups for final part of proposer Enhanced ASR pipeline and trains it.

5.1 Related work

5.2 Obtaining "ASR corrupted" training data

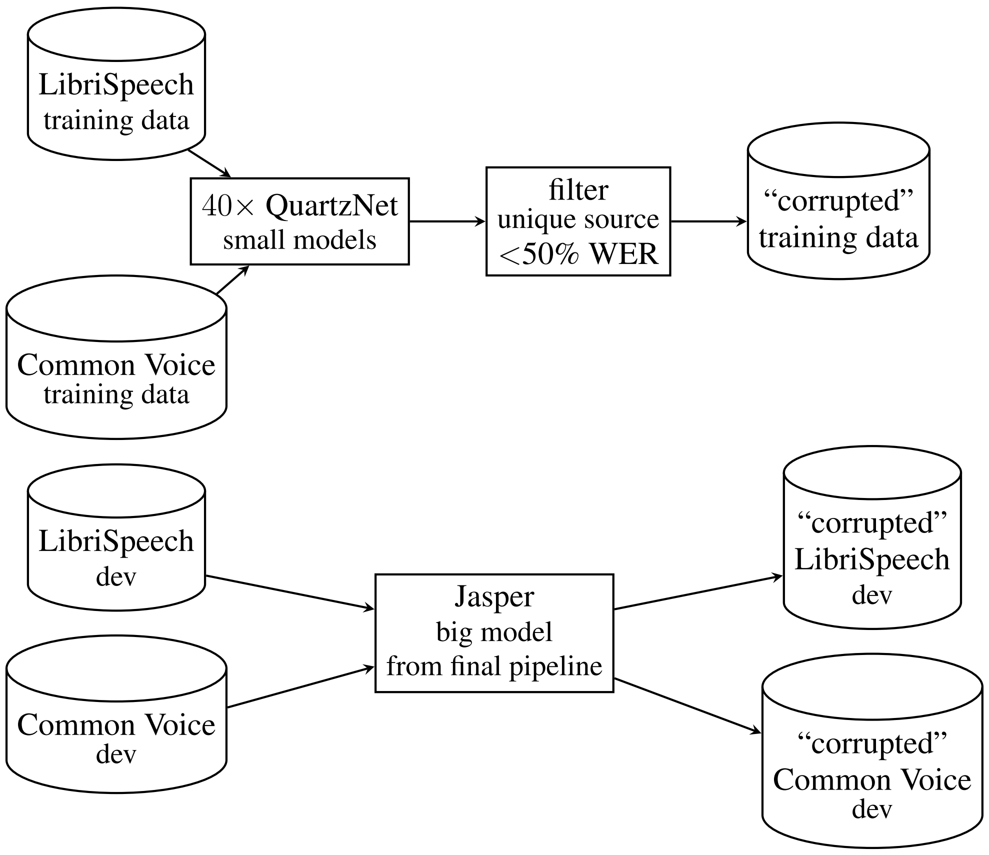

Figure 5.1: After training of 10 "smaller" QuartzNet models with 4 chechpoints made along the way (hence $40\times$), training data are transcribed and filtered. Similarly, Development sets are transcribed using "bigger" Jasper model that will be in the final ASR pipeline (see Figure 3.1).

In this section we describe process in which we gather "corrupted" data from speech recognition model. We will use these data later to improve robustness of translation model.

We design the setup similarly to that of Hrinchuk et al. [2019]. First, ten ASR models are trained. Additionally, we store checkpoints during training and keep last 4 of them, yielding 40 models. Second, we transcribe all available training data using the set of models obtained in previous step. Subsequently, we pair the "corrupted" transcriptions obtained with true transcriptions. We obtain parallel corpus of "corrupted" and "clean" sentences. These data will be then used in further training as source of "natural" noise that occurs in speech recognition.

In the later stage, we will train translation model, the second part of the proposed pipeline (see Figure 3.1). One of the requested properties is the ability to correct errors introduced by acoustic model. To be able to test this property during and after training, we prepare a special development sets. We will build on existing dev sets from LibriSpeech and Common Voice and "corrupt" them in the same manner as training data with a distinction: the final big Jasper model (that will be in the final ASR pipeline) is used instead of the ten smaller models.

Overview of the setup is pictured in Figure 5.1.

5.2.1 Data preparation

Speech corpora LibriSpeech and Common Voice are used. We concatenate these two and divide them into ten folds of same size. We intentionally do not shuffle the concatenated data set prior to splitting it into folds, so that the difference among the trained models is as much as possible (proportion of training data from LibriSpeech and from Common Voice will vary more). The models are trained on these folds in cross-validation manner: i-th model skips i-th fold during training.

5.2.2 Training

Similarly to Hrinchuk et al. [2019] we train ten models and also store checkpoints every 5000 steps. Instead of bigger Jasper we choose QuartzNet. Jasper and QuertzNet are two distinct architectures, nevertheless, they are similar enough and we assume they behave likewise. The main reason of our choise are reduced hardware requirements and hence faster convergence. In contrast with bigger ASR Jasper model that we train on 10 GPUs, each QuertzNet model for data collection is trained only on 1 GPU. After less than a day of training, the models perform almost as good as the bigger model.

Transfer learning technique is again employed to reduce training time and improve model's performance. For English, the QuartzNet encoders are initialized with checkpoint available at NVIDIA NGC[1] which is trained on LibriSpeech and Common Voice. As the target vocabulary differs for our setup (phonemes instead of graphemes), we apply the method from chapter 2 and so the training is divided into phases: (1) Decoder adaptation phase: encoder is initialized with pretrained weights and is freezed while decoder is randomly initialized. Only decoder is then trained, (2) Full training: encoder is unfreezed and trained together with decoder. Adaptation phase is set to take 2000 steps and full training then continues for another 30000 steps.

[1]`https://ngc.nvidia.com/catalog/models/nvidia:quartznet15x5`

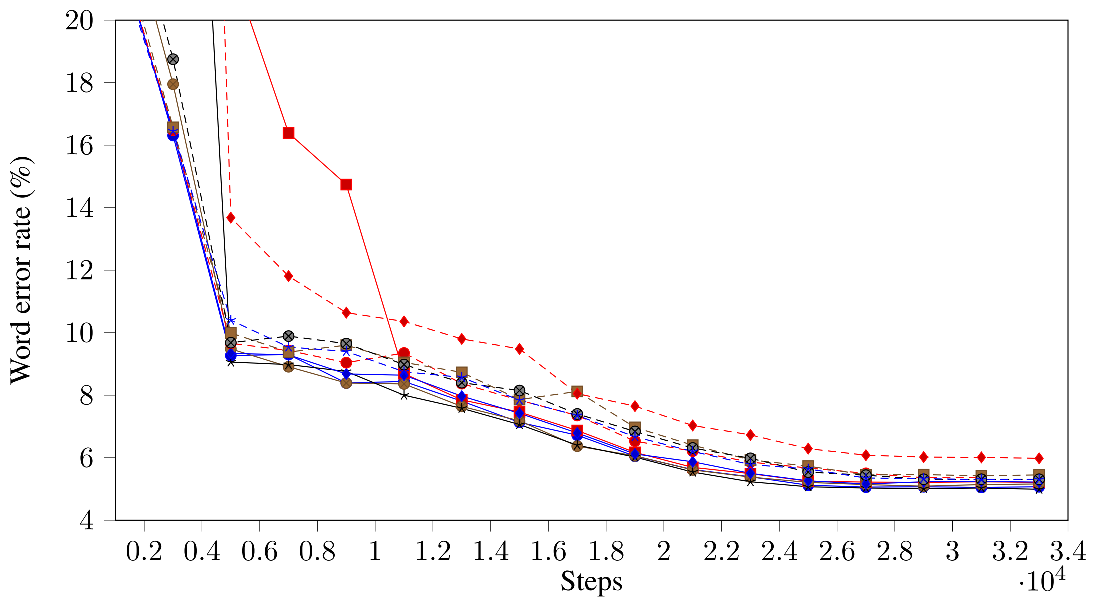

Figure 5.2: Evaluation on dev set during training of 10 models (using greedy decoding). One epoch takes approximately 24400 steps.

Model	**1**	**2**	**3**	4	**5**	**6**	**7**	**8**	**9**	**10**
Adapt. phase	15.17	15.02	16.66	22.15	15.07	15.39	15.24	17.44	15.38	33.05
Full training	5.07	5.21	5.16	4.99	5.22	5.34	5.45	5.31	5.30	5.98

Table 5.1: Results in % of word error rate (using greedy decoding) on LibriSpeech `dev clean` for all trained models.

5.2.3 Results

On average, after the first adaptation phase the word error rate of most models on LibriSpeech `dev clean` dropped under 16 % after less than 5 hours. One model had WER two times worse than others (33.05 %) and one did not converge at all. After full training which took about 15 hours, average WER is 5.3 % with very small variance. Compared with big Jasper it is only about 1.5 percent points more. Evaluations during training can be seen in Figure 5.2 and final results are shown in Table 5.1.

5.2.4 "Corrupted" data collection

Unlike Hrinchuk et al. [2019], in our experiment we do not employ cutout and dropout during data collection. In order to generate more data, we make checkpoints during training every 5000 steps and keep last 4 for every model. This give us 40 unique models.

Concatenated LibriSpeech and Common Voice training data are used for inference on all 40 models yielding 36M sentence pairs. We filter pairs with unique source sentence and keep pairs where word error rate is under 50 %. From the 36 millions sentence pairs we get 7M filtered sentence pairs, particulary 3.7M from LibriSpeech and 3.3M from Common Voice.

Distribution of WER in training and dev sets is visible in histogram 5.3. Following Hrinchuk et al. [2019] we filter out all pairs with WER greater than 50 %. As we can see in the histogram 5.3, only 4 % of training data are left out. We can observe that the

Set	**AVG WER**	**Median WER**	**STD**
Training	10.24 %	2.70 %	16.64 %
LibriSpeech `dev clean`	4.33 %	0.0 %	8.37 %
Common Voice `dev`	11.98 %	0.0 %	17.71 %

Table 5.2: Results in % of word error rate on LibriSpeech `dev clean` for all trained models.

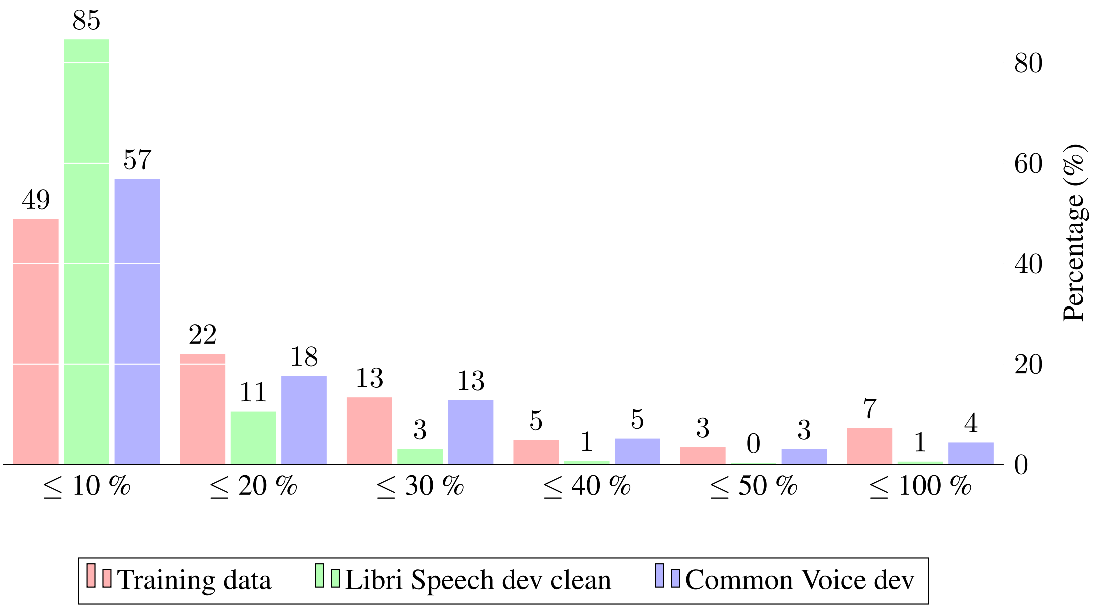

Figure 5.3: Evaluation on test set during training of 10 models.

distribution of WER for training data almost copy the distribution of Common Voice dev with training data having slightly more pairs with smaller WER. On the other hand, LibriSpeech dev clean has significantly more examples with small WER.

5.2.5 Error analysis

XXX description of errors in corrupted data

5.3 Czech ASR corrupted data

In this section we reproduce previously described task for Czech language. Most challenging is to overcome scarcity of speech data — Czech corpus has approximately 400h and we have two English corpora that yield together almost 2000h.

5.3.1 Task setup

Similarly to the setup described in Section 5.2: we split the available data into "folds". Each fold then serves as a training corpus for one model. Next, we take all training data (the Parliament corpus) and put it through the previously obtained models. The output are pairs of ground-truth and "corrupted" data. Finally, we keep pairs with unique corrupted side and with word error rate under 50 %.

5.3.2 Training

Following the receipt from English, we employ QuartzNet architecture, train all models on one GPU and follow the same transfer learning technique. We train-off from our best performing, fully trained, Czech ASR model (see Chapter 2). Because of the Czech training data scarcity, we train only 5 folds. For more details regarding the training see Section 5.2.

5.3.3 Training results

Table 5.3 and Figure 5.4 offer detailed training results for all models. We observe a dramatic WER decline at the beginning (until step 12k), followed by no change until the end. There are probably two reasons for this behavior: (1) data scarcity; (2) contrast in orthography of English and Czech — Czech language has high grapheme-to-phoneme correspondence. We assume the latter to have a greater impact, as the model converged after 7 thousand steps, which is roughly right after seeing all examples once (2000 steps adaptation phase + one epoch of 4775 steps of the full training).

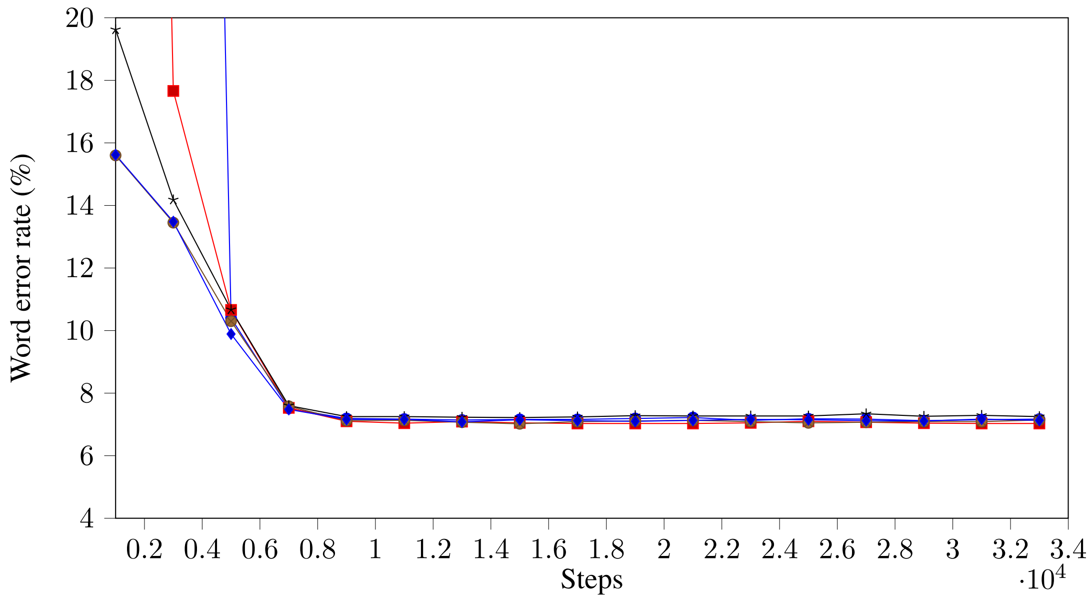

Figure 5.4: Evaluation on dev set during training of 5 models (using greedy decoding). One epoch is approximately 4750 steps.

Model	**1**	**2**	**3**	4	**5**
Adapt. phase	97.91	17.63	13.53	14.09	13.37
Full training	7.17	7.03	7.14	7.25	7.13

Table 5.3: Results in % of word error rate (greedy decoding) on LibriSpeech `dev clean` for all trained models.

5.3.4 Czech corrupted data collection

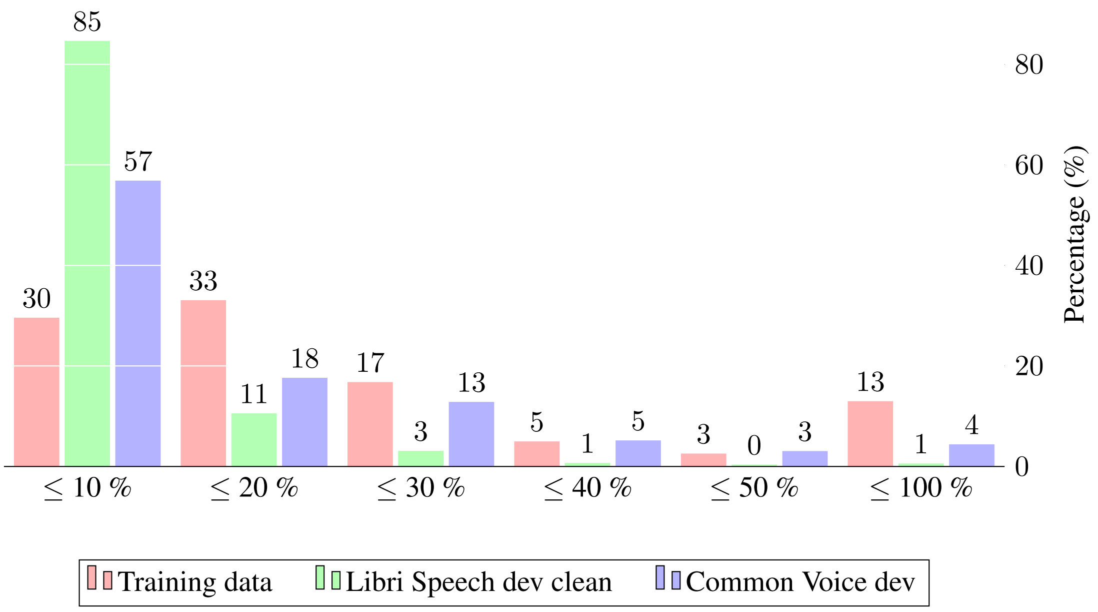

Figure 5.5: Evaluation on test set during training of 10 models.

6. Adaptation

`https://github.com/clab/fast_align`

7. ASR/SLT onlinezation

Conclusion

Bibliography

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

Loïc Barrault, Ondřej Bojar, Marta R Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, et al. Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, 2019.

Willem D Basson and Marelie H Davel. Category-based phoneme-to-grapheme transliteration. 2013.

Alexandre Bérard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. Listen and translate: A proof of concept for end-to-end speech-to-text translation. *arXiv preprint arXiv:1612.01744*, 2016.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228. IEEE, 2018.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovickỳ, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. Czeng 1.6: enlarged czech-english parallel corpus with processing tools dockered. In *International Conference on Text, Speech, and Dialogue*, pages 231–238. Springer, 2016.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, et al. Proceedings of the third conference on machine translation: Shared task papers. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, 2018.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. Multilevel coarse-to-fine PCFG parsing. In Robert C. Moore, Jeff A. Bilmes, Jennifer Chu-Carroll, and Mark Sanderson, editors, *HLT-NAACL*. The Association for Computational Linguistics, 2006. URL `http://acl.ldc.upenn.edu/N/N06/N06-1022.pdf`.

Dongpeng Chen, Brian Mak, Cheung-Chi Leung, and Sunil Sivadas. Joint acoustic modeling of triphones and trigraphemes by multi-task learning deep neural networks for low-resource speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5592–5596. IEEE, 2014.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, 2018.

Jaejin Cho, Murali Karthick Baskar, Ruizhi Li, Matthew Wiesner, Sri Harish Mallidi, Nelson Yalta, Martin Karafiat, Shinji Watanabe, and Takaaki Hori. Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 521–527. IEEE, 2018.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. *arXiv preprint arXiv:1412.1602*, 2014.

B. Decadt, J. Duchateau, W. Daelemans, and P. Wambacq. Phoneme-to-grapheme conversion for out-of-vocabulary words in large vocabulary speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU '01.*, pages 413–416, 2001.

Michael Denkowski and Graham Neubig. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Shuoyang Ding, Adithya Renduchintala, and Kevin Duh. A call for prudent choice of subword merge operations in neural machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 204–213, 2019.

Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1068. URL `https://www.aclweb.org/anthology/P18-1068`.

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

Rohit Gupta, Laurent Besacier, Marc Dymetman, and Matthias Gallé. Character-based nmt with transformer. *arXiv preprint arXiv:1911.04997*, 2019.

Kyu J Han, Ramon Prieto, Kaixing Wu, and Tao Ma. State-of-the-art speech recognition using multi-stream self-attention with dilated 1d convolutions. *arXiv preprint arXiv:1910.00716*, 2019.

Axel Horndasch, Elmar Nöth, Anton Batliner, and Volker Warnke. Phoneme-to-grapheme mapping for spoken inquiries to the semantic web. In *Ninth International Conference on Spoken Language Processing*, 2006.

Oleksii Hrinchuk, Mariya Popova, and Boris Ginsburg. Correction of automatic speech recognition with transformer sequence-to-sequence model. *arXiv preprint arXiv:1910.10697*, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7180–7184. IEEE, 2019.

Biing-Hwang Juang and Lawrence R Rabiner. Automatic speech recognition–a brief history of the technology development. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, 1:67, 2005.

Uday Kamath, John Liu, and James Whitaker. *Deep learning for nlp and speech recognition*. Springer, 2019.

Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. A comparative study on transformer vs rnn in speech applications. In *Proceedings of the ASRU 2019 IEEE Automatic Speech Recognition and Understanding Workshop*, 2019. (in print).

S. Kim and M. L. Seltzer. Towards language-universal end-to-end speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4914–4918, April 2018. doi: 10.1109/ICASSP.2018.8462201.

Tom Kocmi and Ondřej Bojar. Trivial transfer learning for low-resource neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6325. URL `https://www.aclweb.org/anthology/W18-6325`.

Jonáš Kratochvíl, Peter Polák, and Ondřej Bojar. Large Corpus of Czech Parliament Plenary Hearings. `http://hdl.handle.net/11234/1-3126`, 2019.

Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. *arXiv preprint arXiv:1910.10261*, 2019.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, 2018.

Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and Sebastian Stober. Transfer learning for speech recognition on a budget. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 168–177, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2620. URL `https://www.aclweb.org/anthology/W17-2620`.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An End-to-End Convolutional Neural Acoustic Model. In *Proc. Interspeech 2019*, pages 71–75, 2019a. doi: 10.21437/Interspeech.2019-1819. URL `http://dx.doi.org/10.21437/Interspeech.2019-1819`.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An end-to-end convolutional neural acoustic model. *arXiv preprint arXiv:1904.03288*, 2019b.

David Lubensky. Learning spectral-temporal dependencies using connectionist networks. In *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pages 418–421. IEEE, 1988.

Minh-Thang Luong and Christopher D Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, 2016.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

V. Moshkelgosha, H. Behzadi-Khormouji, and M. Yazdian-Dehkordi. Coarse-to-fine parameter tuning for content-based object categorization. In *2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA)*, pages 160–165, April 2017. doi: 10.1109/PRIA.2017.7983038.

Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

Jayashree Padmanabhan and Melvin Jose Johnson Premkumar. Machine learning in automatic speech recognition: A survey. *IETE Technical Review*, 32(4):240–251, 2015.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Douglas B Paul and Janet M Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

Matt Post. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, 2018.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. Bpe-dropout: Simple and effective subword regularization. *arXiv preprint arXiv:1910.13267*, 2019.

C. Raphael. Coarse-to-fine dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1379–1390, Dec 2001. ISSN 1939-3539. doi: 10.1109/34.977562.

D. Reddy and A. Robinson. Phoneme-to-grapheme translation of english. *IEEE Transactions on Audio and Electroacoustics*, 16(2):240–246, 1968.

Michael D Riley and Andrej Ljolje. Recognizing phonemes vs. recognizing phones: a comparison. In *Second International Conference on Spoken Language Processing*, 1992.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Elizabeth Salesky, Matthias Sperber, and Alan W Black. Exploring phoneme-level speech representations for end-to-end speech translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1835–1841, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1179. URL `https://www.aclweb.org/anthology/P19-1179`.

Michael L Seltzer and Jasha Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6965–6969. IEEE, 2013.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. Attention-passing models for robust and data-efficient end-to-end speech translation. *Transactions of the Association for Computational Linguistics*, 7:313–325, 2019.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer, 2018.

Jian Tang, Yan Song, Lirong Dai, and Ian McLoughlin. Acoustic modeling with densely connected residual network for multichannel speech recognition. *Proc. Interspeech 2018*, pages 1783–1787, 2018.

Sibo Tong, Philip N. Garner, and Hervé Bourlard. Cross-lingual adaptation of a ctc-based multilingual acoustic model. *Speech Communication*, 104:39 – 46, 2018. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2018.09.001. URL `http://www.sciencedirect.com/science/article/pii/S016763931830030X`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

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.

Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION.* Springer, 2016.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. Coarse-to-fine learning for neural machine translation. In Min Zhang, Vincent Ng, Dongyan Zhao, Sujian Li, and Hongying Zan, editors, *Natural Language Processing and Chinese Computing*, pages 316–328, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99495-6.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1163. URL `https://www.aclweb.org/anthology/D16-1163`.

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
AVG	average, mean
BLEU	
BPE	
CS	Czech
E2E	end-to-end
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
K	kilo, thousand
M	mega, million
NMT	
OOV	out-of-vocabulary
P2G	phoneme-to-grapheme
STD	standard deviation
WER	word error rate

A. Attachments

A.1 First Attachment