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

This is not a part of the electronic version of the thesis, do not scan!

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In .... date ....

Author's signature

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 decided for less traditional approach for organization of this work: we work with every chapter as with standalone small paper. Specifically, each chapter contains section about related work dedicated to the chapter's topic. Common related work is summarized in Chapter 1 (such as neuralnetwork 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 5. We also deal with "bringing our work to life" in Chapter 6. Finally, we conclude the work in Chapter 6.

1. Tasks, architectures and data sets

In this chapter we give a brief overview about theoretical foundations needed for this theses. First, we define the main tasks that are part of the solution. Further, we describe used neural networks architectures. Finally, we describe used data sets and introduce evaluations 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 eminent part in this process.

First attempts for ASR systems from 1950s were interested with acoustics-phonetics which describes phonetic elements of speech, including phonemes [Juang and Rabiner, 2005]. Early ASR system worked with syllables, wovels and phonemes. As an example can be taken spoken digit recognizer from Bell laboratories. Their system estimated formant frequencies (as a vowel is spoken, vocal tract sounds with natural modes of resonance called formant) of vowel regions of isolated digit.

Big leap in ASR systems was introduction and popularization of Hidden Markov models (HMMs) in 1980s. This caused architecture shift from pattern recognition to statistical modeling. Their success of HMMs based ASR systems continues even today in form of hybrid models consisting of HMMs and deep neural networks (DNNs).

In 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. Beside their performance, they tend to require less expertise and engineering skills for a particular tasks than other methods. This makes them available for researchers in many areas.

DNNs becomming a standard in ASR currently, but first attempts were already made briefly after introduction of backpropagation algorithm by Rumelhart et al. [1986]. They were used for recognition of phonemes [Waibel et al., 1989] or few words [Lubensky, 1988].

1.1.1 Hybrid models

1.1.2 End-to-End models

1.2 Spoken Language Translation

1.3 Models/architectures

In this section we introduce neural network models/architectures that are employed in our work. First two models, Jasper and QuartzNet, are used as accoustic models. Finally, we present Transformer which serves as 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. One model unites acoustic and pronunciation models. We decided to use this architecture for our final ASR and SLT pipeline.

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 are IPA phonemes.

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

Jasper model consist of B blocks and R sub-blocks. Jasper authors introduce 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.

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

Further, Jasper authors observed that models deeper than Jasper 5x3 require residual connections in order 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 millions parameters.

1.3.2 QuartzNet

QuartzNet [Kriman et al., 2019] is another and-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 millions versus 333 millions) while still achieving near state-of-the-art accuracy.

Same as the Jasper model, network's input is 64 MFCC features computed from windows of length 20 ms and overlap 10 ms. The network outputs probability over 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 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 frame independently. Because of this separation, model has fewer parameters and can even have $3\times$ larger kernel than the bigger Jasper model.

A further reduction of weights can be achieved by using grouped pointwise convolution instead of pointwise convolution layer (see Figure 1.3). When using 4 groups, the number of parameters is halved (for QuartzNet-15x5 from 18.9M to 8.7M) with

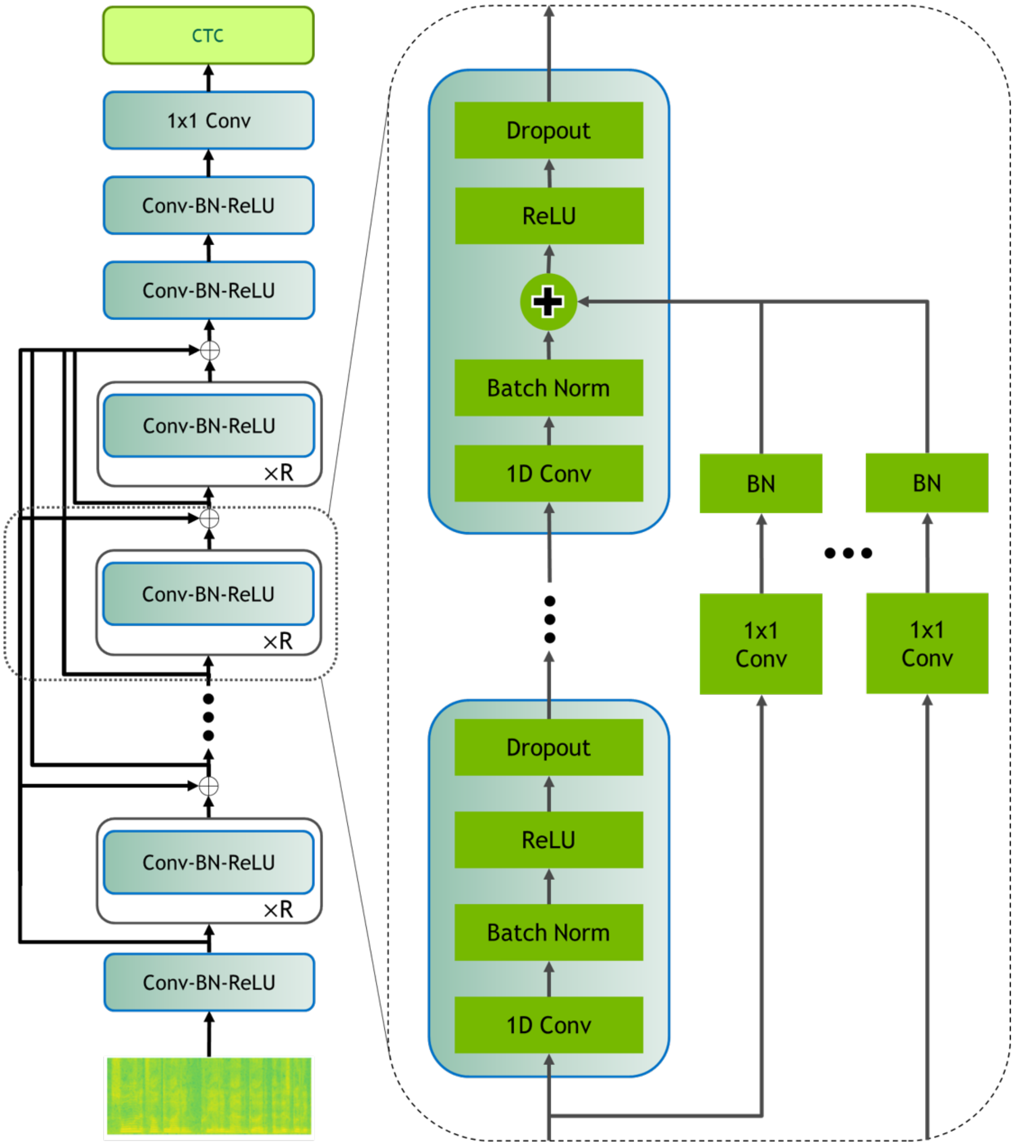

Figure 1.1: Jasper Dense Residual, taken from Li et al. [2019b].

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 grouped poinwise 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 decoder is modified so that the auto-regressive property holds, i.e. the encoder cannot look to right side ("future").

Advantage of self-attention is that it can access arbitrary position in constant number of sequentially executed operations while recurrent networks need $O(n)$ sequentially executed operations. If the sequence length is less than 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 the self-attention leads to better interpretable models. They claim the individual attention heads seems to learn some specific functions that may be related to syntactic and semantic structure of a sentence.

We employ this model in our enhanced ASR pipeline as a correction and language model and as 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 human auditory system perceives voice. Filter in MFCC pipeline is lineary 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 makes the signal less dependent on strong signal from previous time steps.

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

3. **Windowing** In order to avoid potential abrupt changes caused by framing windowing is applied. Windowing is 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 human auditory system. Usually, 40 filters are used. Each filter is of a triangular shape. These are used to compute a weighted sum of filter spectral components approximating Mel scale. Each filter output is then the sum of its spectral filtered components.

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

Example of mel-spectogram is in the Figure 1.5.

1.4.2 Text representation in NMT

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

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

First one is very simple and straightforward method. Character representation enables to encode any word (written in a given characters). Its downside is that it produces longer sequences compared with other methods. Less output classes lead to 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. 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 to handle unknown and rare words. The number of contemporary NMTs that uses 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 idea — they produce more compact text representations. The former is based on “merge” operation that joins the most frequent character sequences together, while the latter is based on unigram language model. 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 enhancement of traditional BPE, which addresses the deterministic nature of the method. BPE-dropout randomly drops some merges from BPE merge table, which results in different segmentation. This 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.

1.5 Data sets

This section is dedicated to data sets used for 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 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 "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, Czech data set was not available.

All utterances are collected and validated by volunteers. Recordings are collected through web form. Speech utterances are stored in MPEG-3 format with 48 KHz sampling rate. After at least two out of tree 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

[1] `https://github.com/VKCOM/YouTokenToMe`
[2] `http://www.voxforge.org/`

of clips in the test set is equal to the number 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 Parliament

1.5.4 CzEng

Filtered CzEng

For our purposes we filtered out all sentence pair that “probably” do not occur in spoken language. We assume following: characters, numbers, apostrophe, punctuation, currency sign, dash and quotation marks on English side mark the sentence pair as a “probable” spoken utterance. Further, we filtered out too long and too short sentences (sentences must have at least 2 character, at most 511 characters).

Our intent is to filter out sentences as for example *“E-006961/11 (PL) Marek Henryk Migalski (ECR) to the Commission (15 July 2011)”*. Such examples could break `phonemizer` and only degrade the transcript/translation quality.

1.6 Error Metrics

In this work 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 evaluation of translation tasks we use BLEU (Bilingual Evaluation Understudy) score introduced by Papineni et al. [2002]. This metric assigns scores in 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 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 brevity penalty.

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

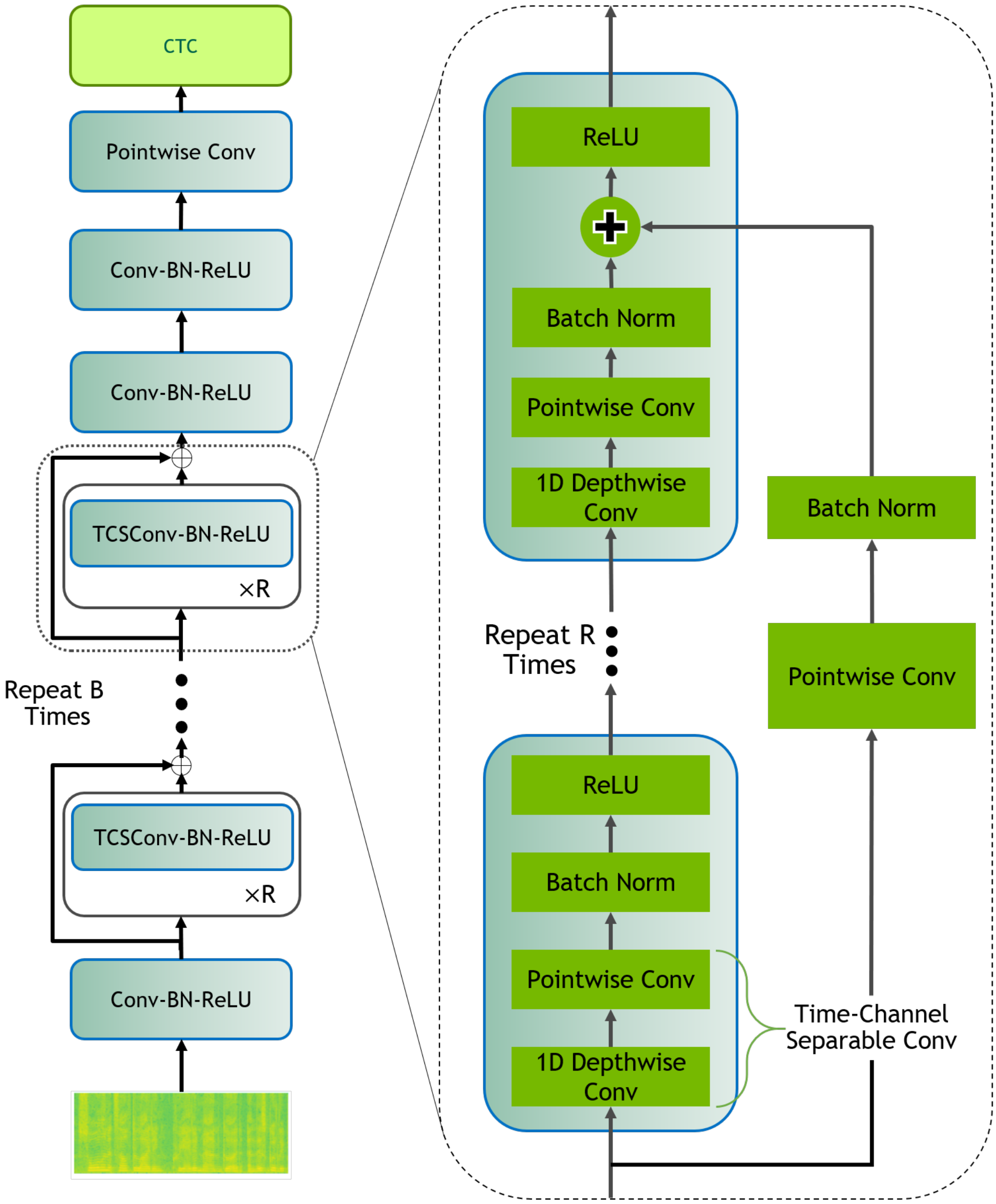

Figure 1.2: QuartzNet BxR architecture. Taken from Kriman et al. [2019].

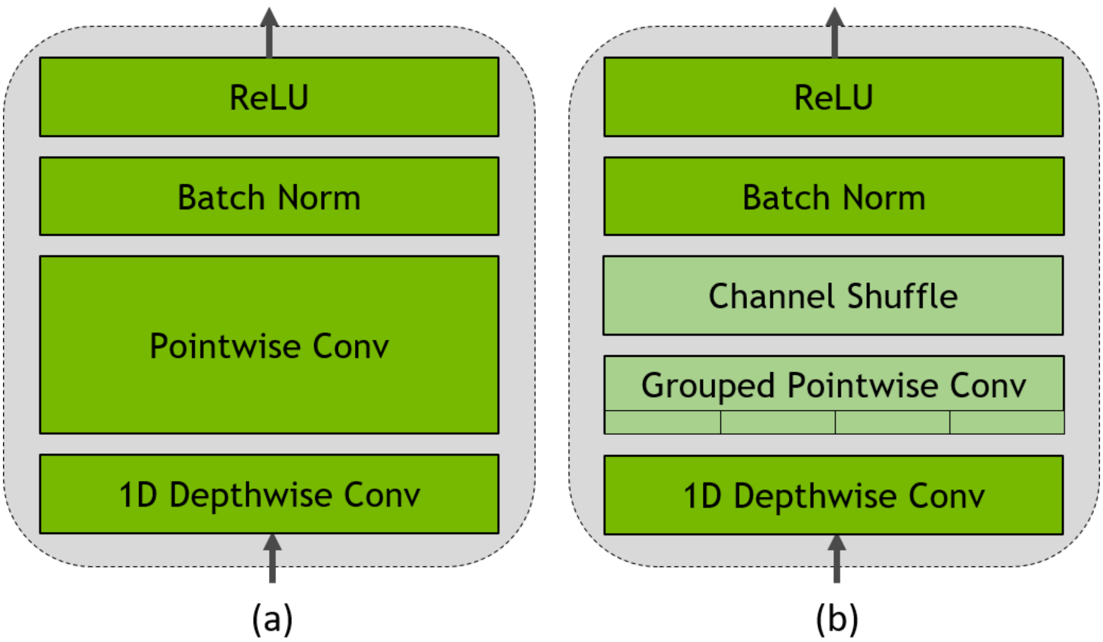

Figure 1.3: (a) Time-channel separable 1D convolutional module (b) Time-channel separable 1D convolutional module with groups and shuffle. Taken from Kriman et al. [2019].

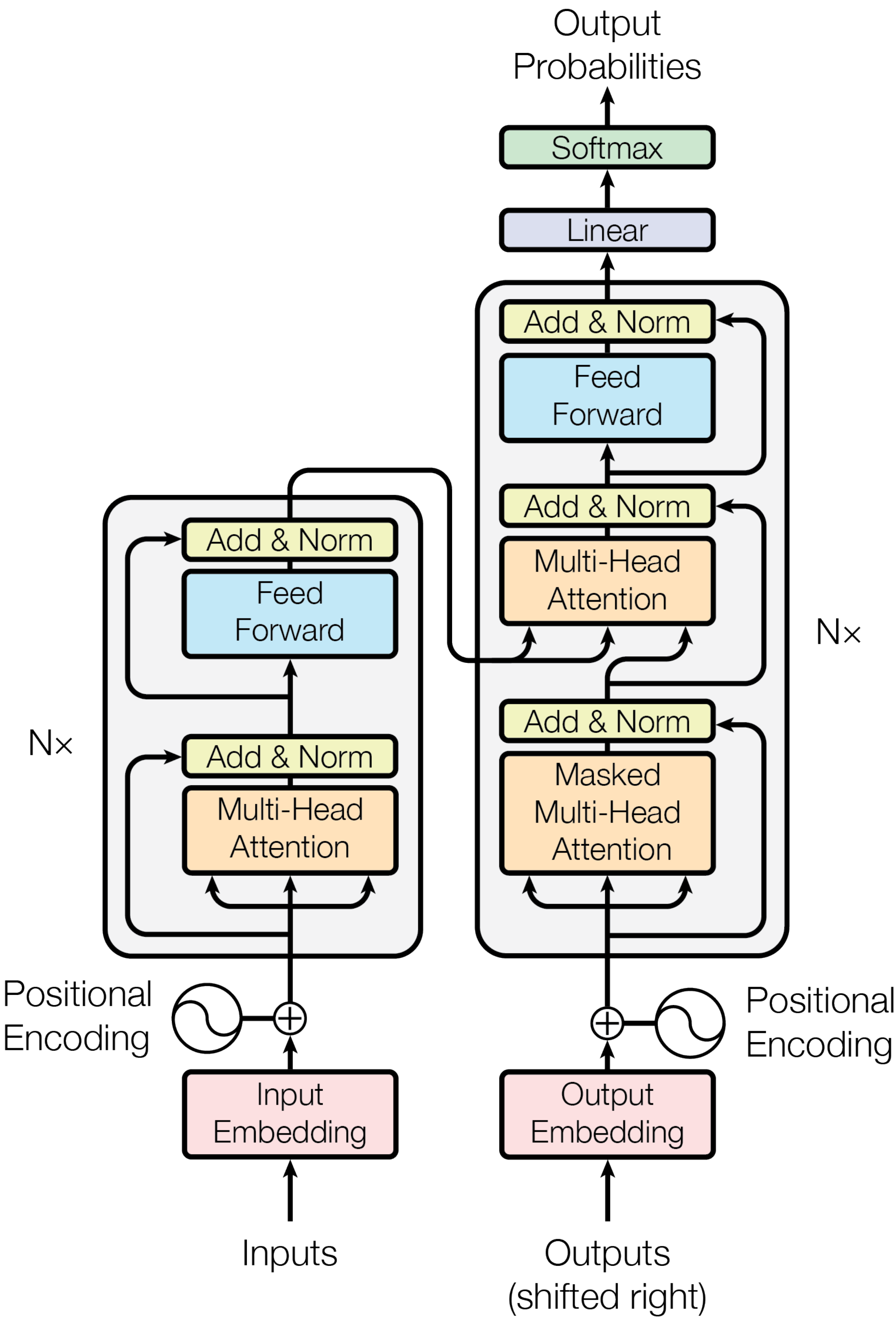

Figure 1.4: Transformer model architecture with detailed encoder (left) and decoder (right). Taken from Vaswani et al. [2017]

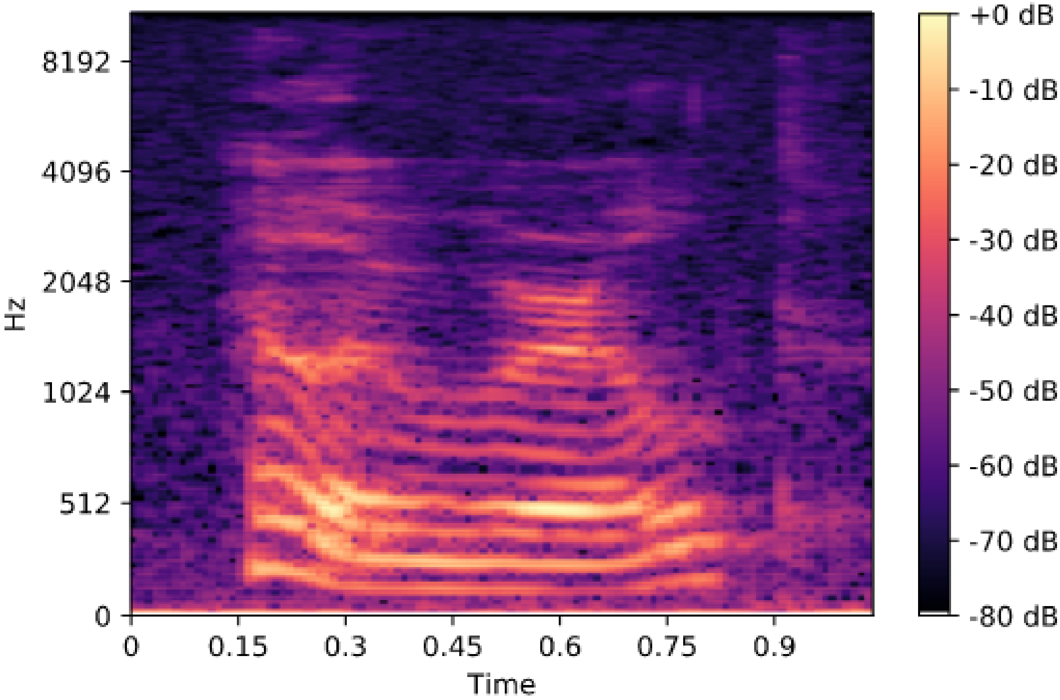

Figure 1.5: Mel spectogram of “Hello World!”.

2. Building ASR

2.1 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].

In order to outperform traditional hybrid models, e.g. Kaldi [Povey et al., 2011], deep-learning ASR systems however must be trained on very large amounts of training data, on the order of a thousand of hours. Currently, there is only a limited number of public datasets that meet this quantity criteria and the variety of covered languages is extremely small. In fact, most of these large datasets contain only English. Although new speech datasets are constantly emerging, producing them is a tedious and expensive task.

Another downside of recent 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 experiments with transfer learning, we reuse the available English ASR checkpoint of QuartzNet [Kriman et al., 2019] and train it to recognize Czech speech instead.

Our paper is organized as follows. In Section 2.2, we give an overview of related work. We follow with a brief description of the neural architecture used in our experiments in **??**. Our proposed method is described in Section 2.3 and the results are presented and discussed in Section 2.4. Finally, in Section 2.5 we summarize the work and we give a brief outlook of our future plans.

2.2 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 a high-resource languages to low-resource ones seems very helpful in machine translation [Zoph et al., 2016, Kocmi and Bojar, 2018].

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

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. They take an English model, freeze weights in the upper part of the network (closer to the input) and 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.

XXX mohli by sme este pridat nakoniec, ze po akceptovani zverejnime checkpointy XXX Other works concerning end-to-end ASR are Tong et al. [2018] and Kim and Seltzer [2018]. The former proposes unificated IPA-based phoneme vocabulary while latter proposes universal character set. The first demonstrates that model with such alphabet is robust to multilingual setup and transfer to other language is possible. The latter proposes language-specific gating enabling languege switching that can increase network's power.

XXX Our work differs from abovementioned twofold: first, we reuse existing models and chechpoints to improve speed and accuracy of unrelated languages. Second, we simplify, rather then unify, Czech character set in order to improve cross-lingual transfer, but also to significantly enhance monolingual training.

Multilingual transfer learning in ASR is studied by Cho et al. [2018]. First, they jointly train one model (encoder and decoder) on 10 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 first train a system on a simpler version of the task 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 the space of possible hyperparameter settings quickly.

2.3 Experiments

2.3.1 Data and Models Used

Pre-Trained English ASR. We use the NeMo 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

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

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

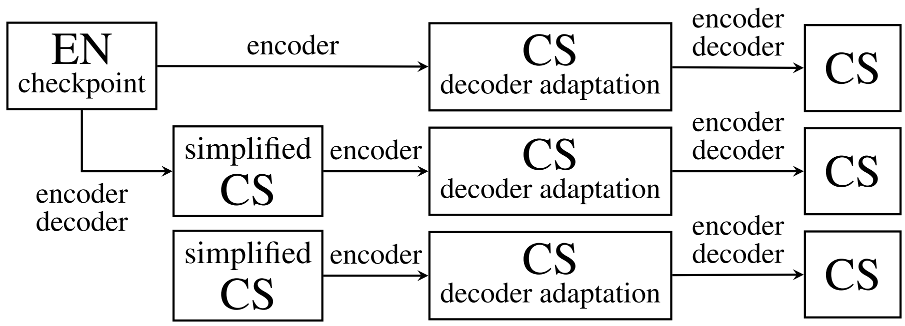

Figure 2.1: Examined setups of transfer learning. The labels on the arrows indicate which model parts are transferred, i.e. used to initialize the subsequent model. No parameter freezing is involved except for the encoder weights in the "CS decoder adaptation" phase.

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 probably the largest available speech corpus for 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 the accuracy of 14.24 % WER on the test set.

2.3.2 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.

2.3.3 Basic Transfer Learning

The first method is very similar 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 very quick 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.

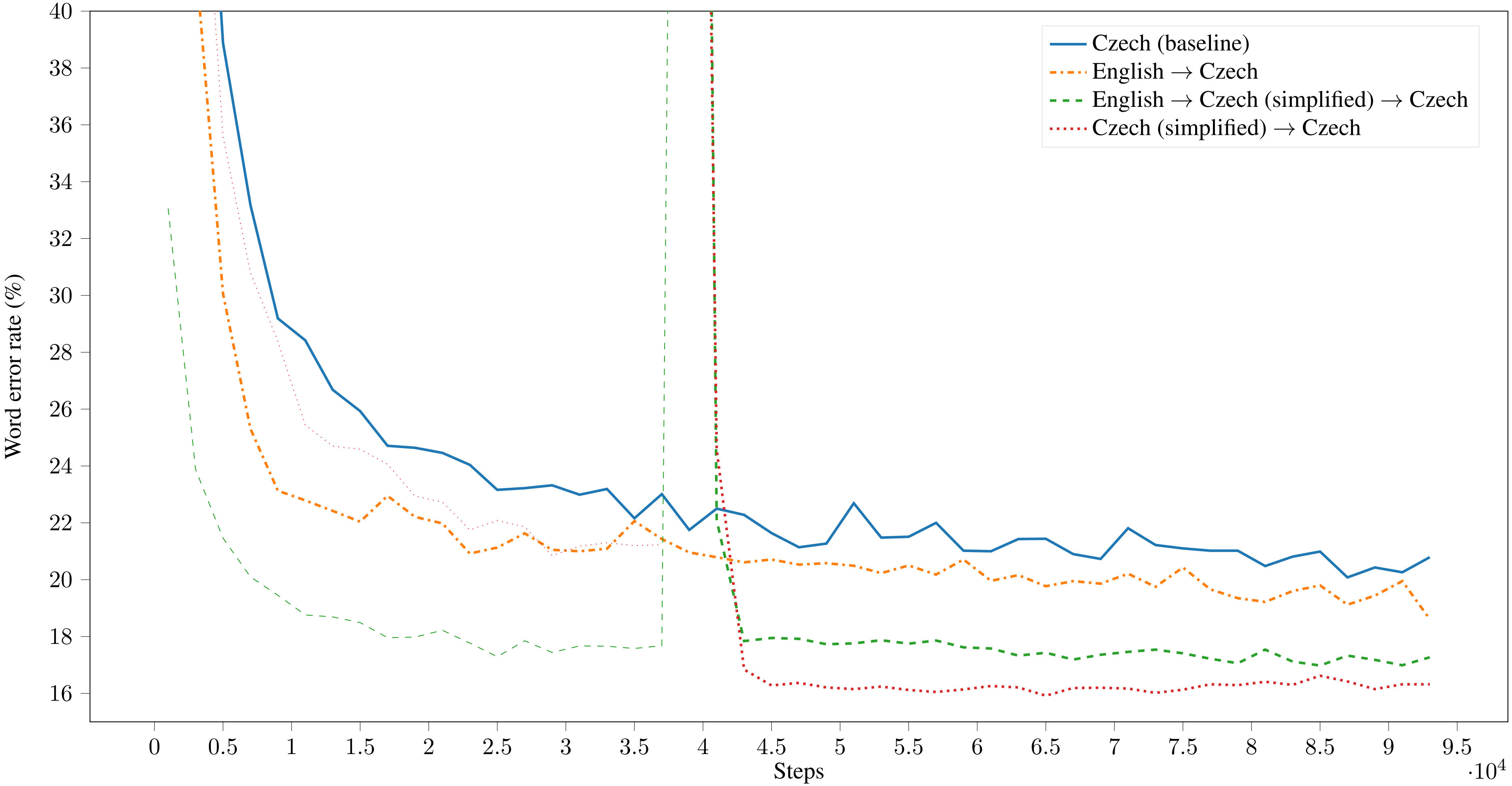

Figure 2.2: Evaluation on test set during training. Note, that WER curves for experiments with simplified vocabulary (thin lines) are not directly comparable with other curves until step 40,000 as the test set is on different (simplified) vocabulary. 10,000 steps takes approximately 5 hours of time.

Experiment	Simpl.	Adapt.	Full
Baseline CS	-	-	20.19
EN → CS	-	97.35	19.06
EN → sim. CS → CS	17.11	22.01	16.88
sim. CS → CS	20.56	24.59	16.57

Table 2.1: Results in % of word error rate on the Czech test set. "Simpl." reflects WER on Czech without accents (both hypothesis and reference stripped). "Adapt." and "Full" already use the original test set.

2.3.4 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 2.3.3. 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.

2.3.5 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 and then switch to full Czech (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.

2.4 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]

2.4.1 Transfer across Unrelated Languages

We observe that initialization with an unrelated language helps to speed-up training. This is best apparent in "English → Czech simplified" where the unchanged vocabulary

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

allows to reuse all the weights. WER drops under 30 % after only 2000 steps (1 hour).[6]

If the target alphabet is altered ("English → 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.

2.4.2 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.[7]

The intermediate simplified vocabulary brings always a considerable improvement: 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; the target alphabet is smaller but more spelling ambiguities may arise) which 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 → Czech" setup could be trapped.

2.5 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. Or 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 or better iteration choice to avoid overfitting.

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

2.6 Cross-Lingual ASR Transfer with Coarse-to-Fine Intermediate Step

XXX Dat odkaz na clanok

[6]This can be particularly useful if the final task does not require the lowest possible WER, such as sound segmentation.

[7]Note that WER curves prior to the spike use the simplified Czech reference, so they are not directly comparable to the rest.

2.7 ASR Transfer from graphemes to phonemes

https://ieeexplore.ieee.org/abstract/document/21701 We describe ASR transfer from graphemes to phonemes. We start with pretrained Jasper ASR model available online[8].

2.7.1 Model and Data preparation

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

3. Enhanced ASR

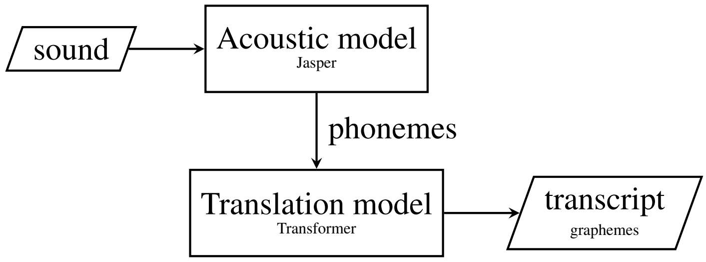

Figure 3.1: Enhanced ASR pipeline.

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 and (2) "translation" model that consumes previously outputted phonemes and translates them into the graphemes. Illustation 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 (see Figure 3.1) 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.

In order to build this part of the pipeline as robust as possible, we train the translation model to be capable of correcting some errors that are created by ASR. Therefore, in Section 3.2 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.

3.1 Related work

3.2 Obtaining "ASR corrupted" training data

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

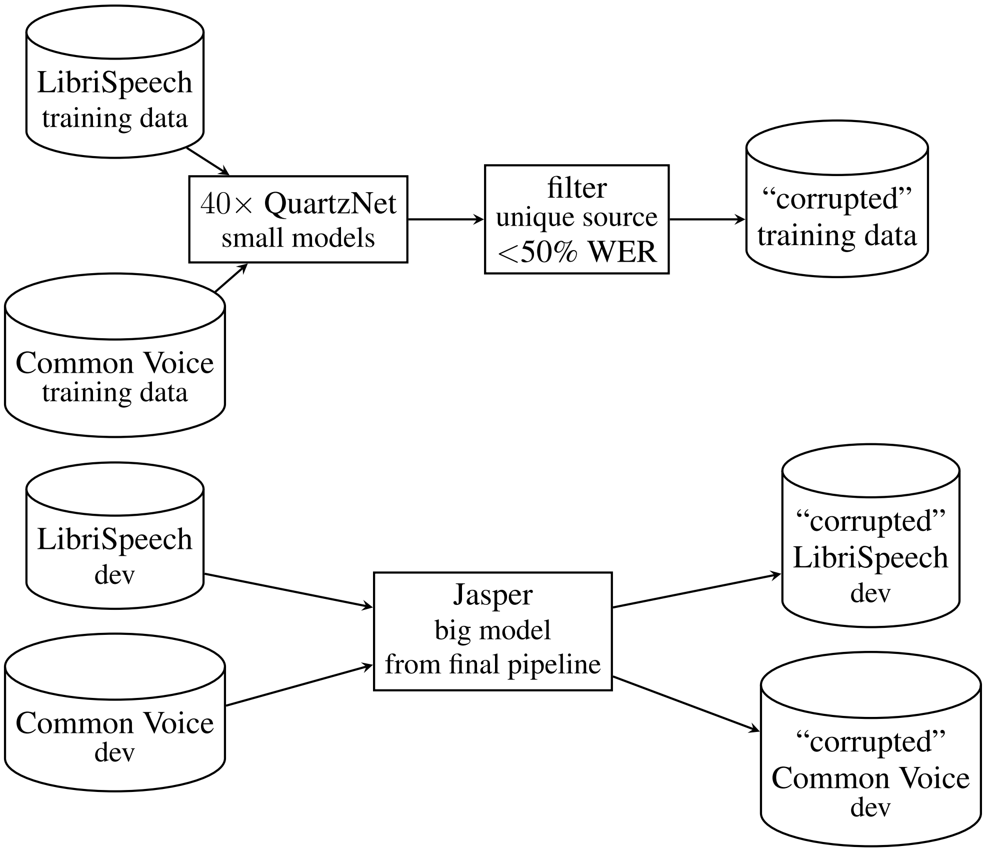

Figure 3.2: After training of 10 "smaller" QuartzNet models with 4 chechpoints made along the way (hence $40\times$), training data are transcribed and filtered. Similarly, Development sets are transcribed using "bigger" Jasper model that will be in the final ASR pipeline (see Figure 3.1).

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 3.2.

3.2.1 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

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

cross-validation manner: i-th model skips i-th fold during training.

3.2.2 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.

3.2.3 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 3.3 and final results are shown in Table 3.1.

3.2.4 "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.

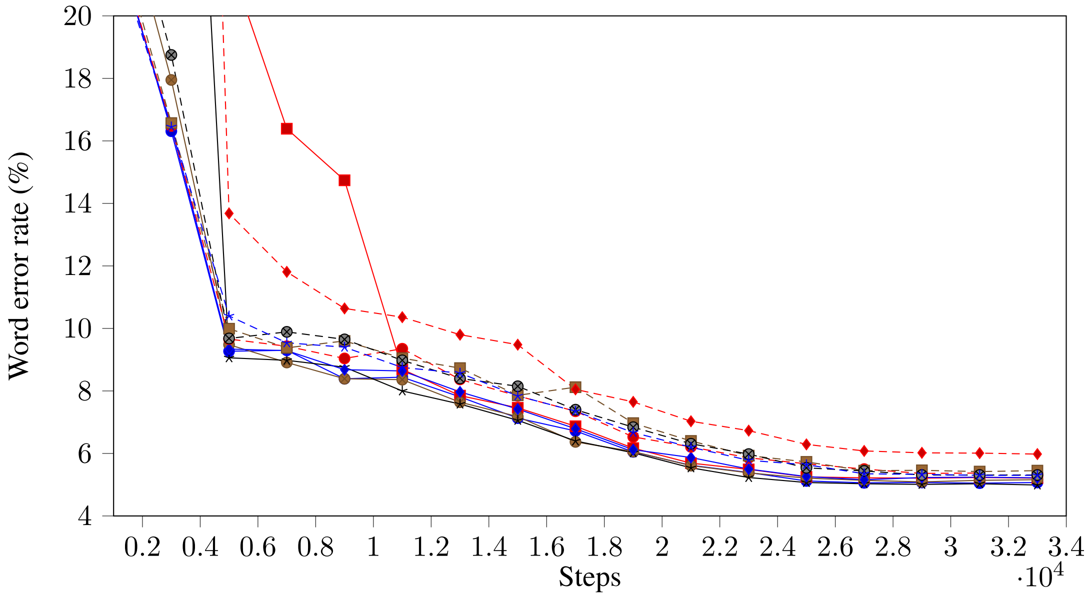

Figure 3.3: Evaluation on dev set during training of 10 models (using greedy decoding). One epoch takes approximately 24400 steps.

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

Distribution of WER in training and dev sets is visible in histogram 3.4. Following Hrinchuk et al. [2019] we filter out all pairs with WER greater than 50 %. As we can see in the histogram 3.4, only 4 % of training data are left out. We can observe that the 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.

3.2.5 Error analysis

XXX description of errors in corrupted data

3.3 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.

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

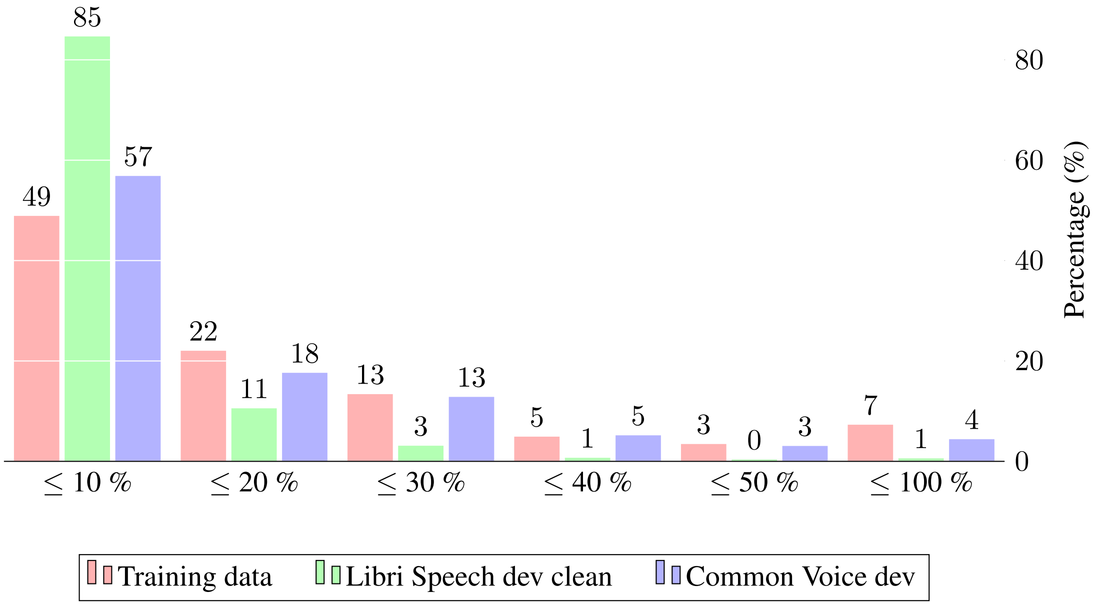

Figure 3.4: Evaluation on test set during training of 10 models.

3.3.1 Task setup

Similarly to the setup described in Section 3.2: 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 %.

3.3.2 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 3.2.

3.3.3 Training results

Table 3.3 and Figure 3.5 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).

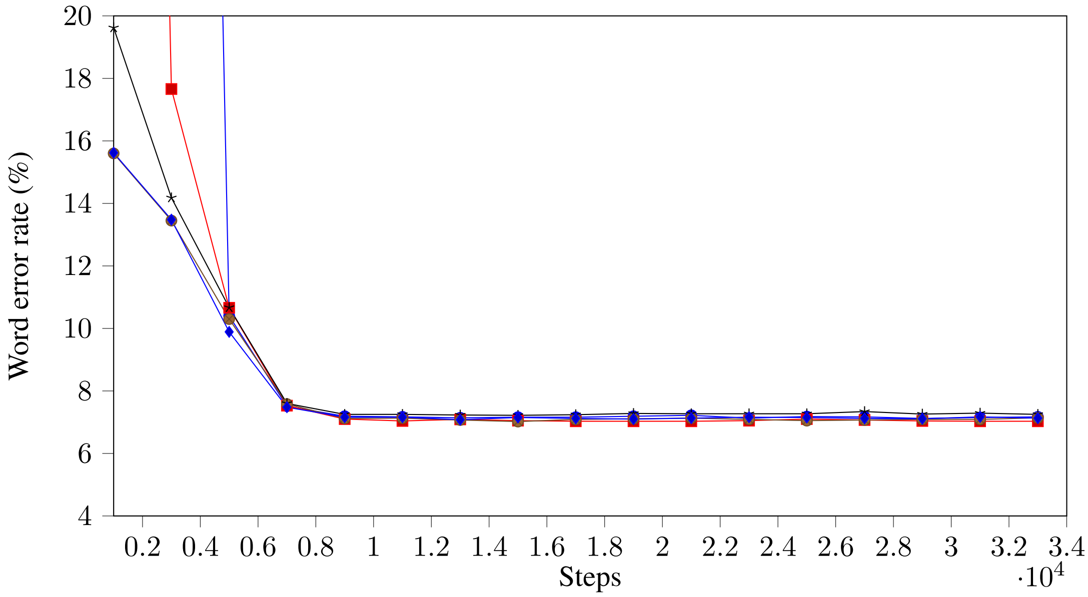

Figure 3.5: Evaluation on dev set during training of 5 models (using greedy decoding). One epoch is approximately 4750 steps.

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

3.3.4 Czech corrupted data collection

3.4 Transformer model

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.

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

As the source and target are in different alphabets — phonemes and graphemes — that share very few characters, BPE vocabularies and embeddings will not be shared. For target tokenization we select vocabulary size of 8K, following NeMo authors' recommendation[2], as this should increase batch size, leading to faster convergence while preserving same level of performance when compared with BPE vocabulary size of 32K. The target should always output valid English, hence we train BPE tokenizer on filtered CzEng (see Section 1.5.4) data set.

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

3.4.1 Source tokenizer

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)?

Related work

In this section we give a brief overview of related work. 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*" in

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. They claim that artificial representations such as BPE are sub-optimal leading to e.g. (linguistically) improbable word fragmentations. Although, they outperform BPE, they recognize 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. 16k-32k is bad for low resource setting - best 0-1K, with large drop after 8K. XXX finish this

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.

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, with each BPE size (except character-level) trained on clean, corrupted and mixture) and same target vocabulary (8k trained on clean graphemes, phonemized filtered CzEng 1.7). Afterwards we evaluate their performance on "corrupted" dev sets that were obtained in Section 3.2.

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

Data preparation

Source BPE vocabularies are trained on: clean filtered Czeng for *clean* setup and corrupted data from ASR ensemble setup (see Section 3.2). For *mixture* setup, we taken subset of 7M from clean filtered Czeng (as it has 57M sentence pairs) of the same size as the ASR data.

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).

Training

As mentioned previously, we use Transformer `base` configuration. We alter maximum sequence length to 1024, as for character-level, 128 and 512 BPE configurations many sentences do not fit into model. We train all models for 70000 steps on 2 GPUs using same batch size for all configurations: 12000 tokens.

Results and analysis

Final results of all experiments are in Tables 3.4 and 3.5. Overview of runs are pictured in Figures 3.6 and 3.7.

We can observe clear dependency between WER and BPE vocabulary size: the bigger size the lower WER. For source of BPE training data, there is not a clear pattern with negligible differences among training source for the same vocabulary size. However, for 128 BPE it seems that the corrupted data are optimal for vocabulary training. Only for small vocabularies, the model is better in translation rather than correcting errors, hence it is better to use data with introduced errors.

Conclusion

Generally, 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. It seems that translation is the most important function of the model, as most of the words in corrupted data are anyway correct (recall, we filtered out sentences with WER over 50 %), We also believe model that uses BPE trained on BPE will make the place around error more fragmented or “odd” to the model and this can be a “clue” for the model to correct the transcript around. This does not hold for small vocabularies. Such model is probably better in translation rather than correcting errors, hence it seems better to use data with introduced errors for this particular setup.

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

XXX add plot of attention with correct and corrupted sentence

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

Size	**Clean**	**Corrupt**	**Mixed**
128	144.56	136.73	103.97
512	98.10	100.9	101.26
2k	64.41	81.20	82.32
8k	56.13	66.68	68.96
32k	50.28	48.34	50.03

Table 3.4: Left: results in % of word error rate on the LibriSpeech dev clean set. Right: average time per sentence in miliseconds.

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

Size	**Clean**	**Corrupt**	**Mixed**
128	23.90	19.11	19.02
512	20.27	17.90	19.78
2k	12.51	15.38	16.70
8k	11.82	13.65	14.13
32k	10.79	12.78	11.07

Table 3.5: Left: results in % of word error rate on the Common Voice dev set. Right: average time per sentence in miliseconds.

3.5 Enhanced ASR

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

3.5.1 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.

3.5.2 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 3.2 while development sets remain same as in the initial training.

3.5.3 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.

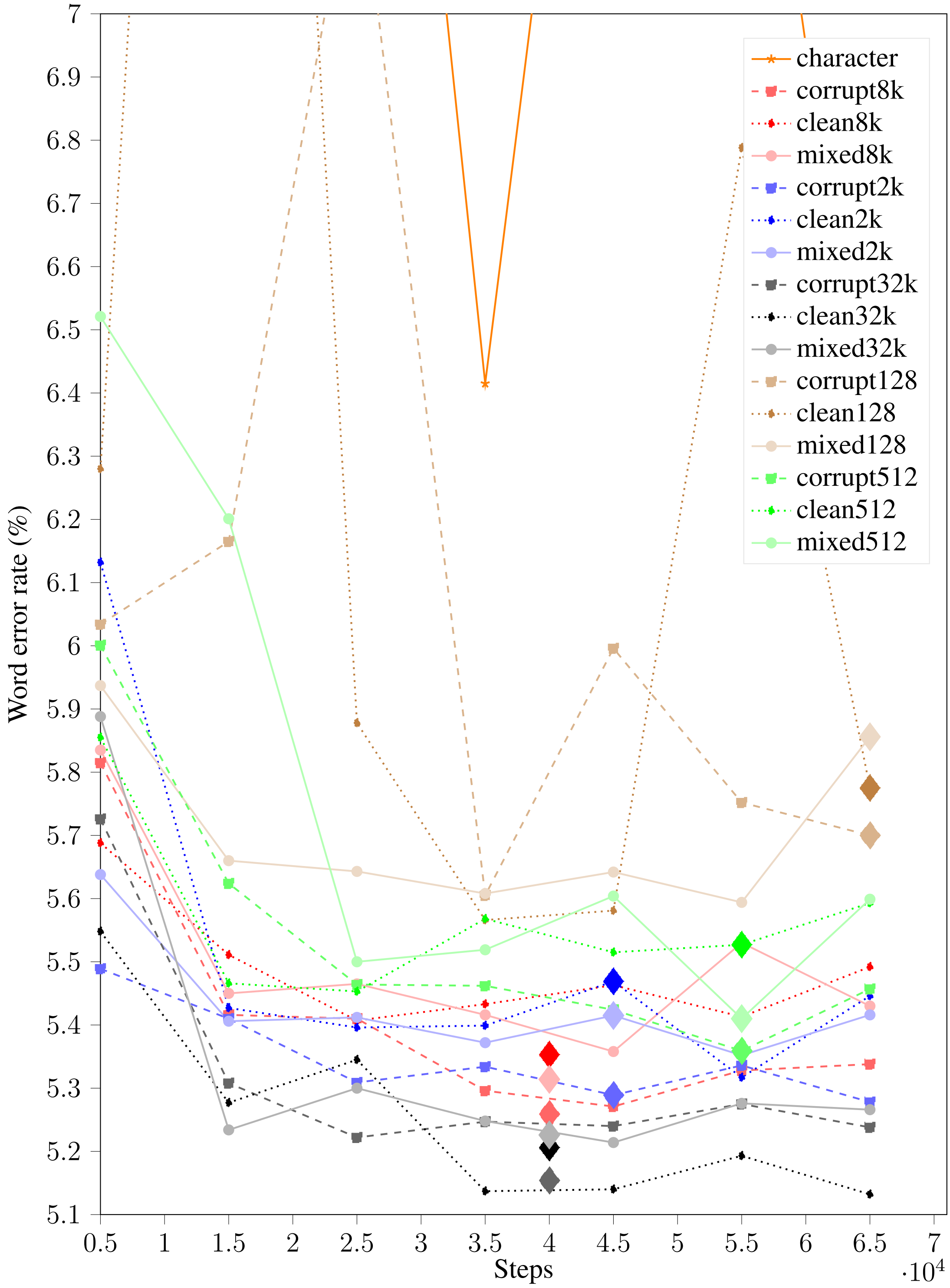

Figure 3.6: Each model is evaluated on LibriSpeech corrupted dev set (see Section 3.2) every 5000 steps. Bigger diamond marks shows where each trained model reached 10th epoch.

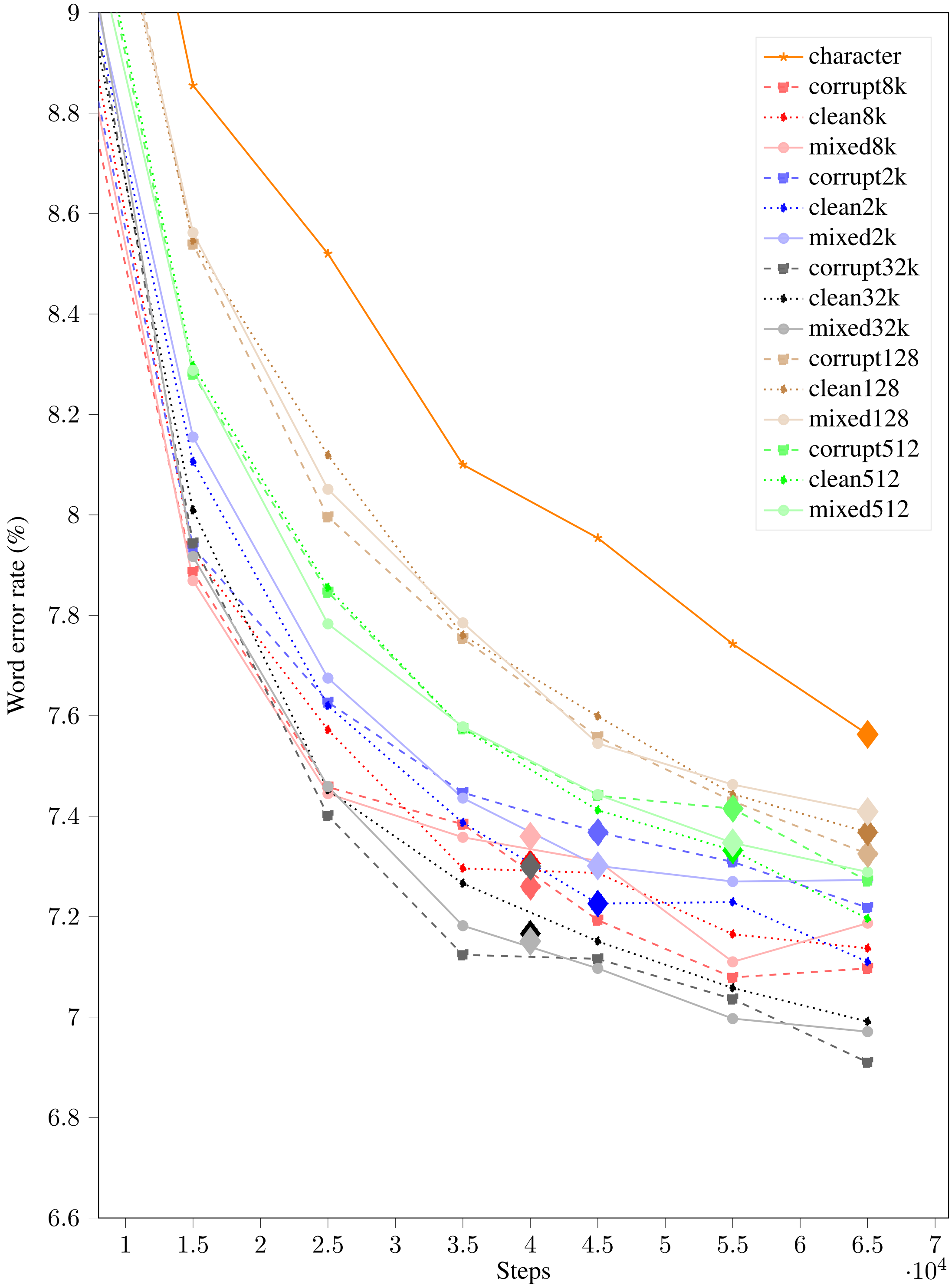

Figure 3.7: Each model is evaluated on Common Voice corrupted dev set (see Section 3.2) every 5000 steps. Bigger diamond marks shows where each trained model reached 10th epoch.

4. Spoken Language Translation

5. Adaptation

https://github.com/clab/fast_align

6. ASR/SLT onlinezation

Conclusion

Bibliography

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.

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`.

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.

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`.

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.

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.

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.

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.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

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.

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

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.

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.

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

ASR	Automatic Speech Recognition
AVG	average, mean
BLEU	
BPE	
K	kilo, thousand
M	mega, million
NMT	
STD	standard deviation
WER	word error rate

A. Attachments

A.1 First Attachment