BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

END-TO-END SPEECH RECOGNITION FOR LOW-RESOURCE LANGUAGES

END-TO-END ROZPOZNÁVÁNÍ ŘEČI PRO JAZYKY S NÍZKÝMI ZDROJI

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **VLADISLAV SOKOLOVSKII**
AUTOR PRÁCE

SUPERVISOR **Ing. MARTIN KARAFIÁT, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2021/2022

Bachelor's Thesis Specification

25174

Student: **Sokolovskii Vladislav**
Programme: Information Technology
Title: **End-to-End Speech Recognition for Low-Resource Languages**
Category: Speech and Natural Language Processing

Assignment:

1. Get acquainted with the fundamentals of classical (hybrid) and end-to-end automatic speech recognition (ASR).
2. Get acquainted with the BUT (Brno) and I2R (Singapore) low-resource setups and reproduce them.
3. Select one or more low-resource languages, prepare data for its training, produce results and compare them with available baselines (BUT and/or I2R).
4. Analyze the results and current trends in low-resource end-to-end ASR (such as new NN architectures, pre-trained Transformer-like models, etc.), implement and test some of them and discuss the results.
5. Prepare a compact package useable by both BUT and I2R labs.
6. Create a short (30s) video presenting your work.

Recommended literature:

- Daniel Povey, Vijayaditya Peddinti ...: "*Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI*". INTERSPEECH2016, pp 2751-2755
- Graves, Alex. "Sequence Transduction with Recurrent Neural Networks." *ArXiv* abs/1211.3711 (2012)
- Dong, Linhao et al. "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition." *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018): 5884-5888.
- Baevski, Alexei et al. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations." *ArXiv* abs/2006.11477 (2020)

Requirements for the first semester:

- Items 1 and 2, significant advance in item 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Karafiát Martin, Ing., Ph.D.**
Consultant: Chen Nancy, A*STAR
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2021
Submission deadline: July 29, 2022
Approval date: November 2, 2021

Abstract

The automatic speech recognition area has started to adopt end-to-end neural network solutions for creating speech recognizers. However, the data hunger nature of these types of systems allows for the creation of recognizers only for high-resource languages, such as English, Chinese or Spanish. In low-resource scenarios, some solutions which alleviate the data scarcity problem have to be developed. One of the most effective techniques for this is fine-tuning a pre-trained model. The problem with the existing approaches of fine-tuning is that the token set of target and source languages does usually differ. That is why previous multi-lingual transfer learning approaches required the output layer to be changed, or mixed tokens from different languages in the output layer, or use universal token sets, or have separate output layers per language. This is undesirable because the sharing across languages in this case latent and not controllable in the output space when the language-specific graphemes are disjoint. Therefore this work proposes to map the tokens to the common set before the beginning of the pre-training. The existing solution was a transliteration of the source language to the target one, the novel approach is romanization where the token set of the target language is romanized to match the English alphabet. Subsequently, the diacritics from the romanized hypotheses can be restored using an additional restoration model. This has the advantage of increasing sharing in the output grapheme space.

Abstrakt

Oblast automatického rozpoznávání řeči začala přijímat end-to-end řešení neuronové sítě pro vytváření rozpoznávačů řeči. Povaha datového hladu těchto typů systémů však umožňuje vytvářet rozpoznávače pouze pro jazyky s velkými zdroji, jako je angličtina, čínština nebo španělština. Ve scénářích s nízkými zdroji je třeba vyvinout některá řešení, která zmírní problém nedostatku dat. Jednou z nejúčinnějších technik je doladění předtrénovaného modelu. Problém se stávajícími přístupy ladění spočívá v tom, že sada tokenů cílového a zdrojového jazyka se obvykle liší. To je důvod, proč předchozí přístupy k učení vícejazyčného přenosu vyžadovaly změnu výstupní vrstvy nebo smíchání tokenů z různých jazyků ve výstupní vrstvě, případně použití univerzální sady tokenů anebo samostatné výstupní vrstvy pro každý jazyk. To je nežádoucí, jelikož sdílení napříč jazyky je v tomto případě latentní a neovladatelné ve výstupním prostoru, když jsou grafémy specifické pro daný jazyk disjunktní. Proto tato práce navrhuje mapování tokenů do společné sady před začátkem předtréninku. Stávající řešení spočívá v transliteraci zdrojového jazyka do cílového, novým přístupem je romanizace, kde je sada tokenů cílového jazyka romanizována tak, aby odpovídala anglické abecedě. Následně lze diakritiku z romanizovaných hypotéz obnovit pomocí dalšího modelu obnovy. To má výhodu ve zvýšení sdílení v prostoru výstupního grafému.

Keywords

ASR, low-resource, transliteration, romanization, model, data, training, transfer learning, speech, end-to-end, augmentation, fine-tuning

Klíčová slova

ASR, low-resource, transliteration, romanization, model, data, trénování, transfer learning, řeč, end-to-end, augmentation, fine-tuning

Reference

SOKOLOVSKII, Vladislav. *End-to-end speech recognition for low-resource languages*. Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Martin Karafiát, Ph.D.

Rozšířený abstrakt

Tato práce zkoumá možnosti využití technik augmentace a přenosového účení pro jazyky s nízkými zdroji, jako je tamilština a vietnamština. Hlavním problémem těchto jazyků je nedostatek dat a cílem zmíněných technik je uměle zvýšit množství trénovacích dat. Novinkou této práce je romanizace zdrojových dat před předtrénováním a obnova diakritiky výstupu doladěného modelu. Dalším používaným přístupem je transliterace. Transliterace převádí anglický referenční text na grafémovou sekvenci cílového jazyka, která může znít podobně jako anglická verze. Model je následně na těchto datech natrénován. Tyto přístupy zavádějí způsob přenosového účení, kde není potřeba měnit vrstvu nebo kombinovat několik výstupních vrstev dohromady, protože cílový i zdrojový jazyk budou mít stejnou sadu tokenů. V případě romanizace se odkaz na cílový jazyk převede na anglickou sadu grafémů. Metodu romanizace popsanou v této práci lze provést pouze na jazyce, který má nastaven grafém podobně jako latina, například vietnamština nebo litevština. Tyto techniky poskytují zvýšení výkonu, ale výsledný výkon end-to-end modelů trénovaných pouze na datech s nízkými zdroji je stále neuspokojivý.

Tato práce také pokrývá základní koncepty a techniky konvenčního rozpoznávání řeči od modelů HMM-GMM až po nejmodernější end-to-end modely založené na attention. V celé práci jsou diskutovány klady a zápory obou přístupů. Proto lze tuto práci využít pro výchovu nové generace studentů, kteří se zajímají o řečové technologie. Také myšlenky sdílené v této práci mohou být výzkumníky důkladněji prozkoumány a implementovány. Sekce “Future work” navrhuje různé možnosti, jak zlepšit výkon end-to-end modelů trénovaných pouze na jazykových datech s nízkými zdroji.

End-to-end speech recognition for low-resource languages

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Martin Karafiat Ph.D.. The supplementary information was provided by Jeremy Wong Ph.D.. The thesis was written within the summer semester internship in A*STAR company. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Vladislav Sokolovskii
July 28, 2022

Acknowledgements

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic and BUT Speech@FIT research group.

I want to thank Jeremy Wong Ph.D. for his outstanding support and patience while working with me and for his desire to help with any issue I had. This internship and your supervision gave me a lot.

I want to sincerely thank Ing. Martin Karafiat Ph.D. for his understanding and endurance throughout the whole last year. Also, I would like to thank both Ing. Martin Karafiat Ph.D. and Ing. Jan Svec for the guidance and help throughout the two last semesters of my study. They provided me with a lot of useful insights and support. Thank you for the regular meetings and productive debates.

I would also like to thank Nancy F. Chen Ph.D., prof. Dr. Ing. Jan Cernocky and A*STAR staff for helping me to organize this internship and for believing in me.

Finally, I want to express my love and gratitude towards my partner Anna for the support and encouragement she gave me throughout this tough and adventurous experience.

Contents

Chapter 1

Introduction

Verbal communication through speech is not only a way of communication between humans but also an interface for communicating between humans and machines. Nowadays, the spoken interfaces may prevail over the conventional physical interfaces in cases when a person is not physically able to interact with a device. This technology can be helpful in education, security, cinematography, and many more fields and industries. Speech is one of the most natural ways of communication for humans, that is why the research in this field is crucial for creating more natural and interactive human-centred interfaces.

There are several ways of how to teach the machine to comprehend speech, one of them is to convert it to text first. Converting speech to text is called Automatic Speech Recognition (ASR). This research area has drastically developed after the rapid improvement of Deep Neural Networks (DNNs). The DNN can be either used as a component in the conventional ASR pipeline or as an independent model capable of direct mapping of extracted audio features to the character sequence. In this work, any speech recognition system based on Hidden Markov Models (HMM) can be referred to as conventional. When DNN is just used for computing some HMM probabilities, this approach is usually called hybrid [7]. The system where a single DNN model directly maps features to characters is called end-to-end (e2e). Both of the mentioned approaches have their advantages and are present in the ASR research and production environments.

1.1 Overview of ASR

The ASR is a supervised machine learning problem, meaning that the model is trained on the beforehand prepared labeled data. During training, the model passes the data several times and updates its parameters with regard to a loss function that computes the "distance" between ground-truth value and model output. The common tendency is that the model will adjust its parameters to produce better and better results on the validation set after each epoch. The evaluation process of the model is usually performed on the previously unseen data. The model is considered well-trained when it is capable of generalizing on the previously unseen test data.

The conventional ASR systems are based on pure statistical techniques such as HMM and Gaussian Mixture Models (GMM). Nowadays, when the ASR community is actively adopting the e2e approach, the main goal of the ASR remains the same, regardless of the chosen approach, and it can be formulated with a single equation. Mathematically speaking, the goal of the ASR systems is to find the most likely sentence $\hat{\mathbf{W}}$ given the input audio features $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_T)$,

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} P\left(\mathbf{W} \mid \mathbf{X}\right), \tag{1.1}$$

where $\mathbf{W}$ is the particular sequence of words, $\mathbf{W} = (w_1, w_2, ...w_N)$.

Using Bayes' theorem, this equation can be rewritten to the form where the single components of conventional model can be explicitly seen:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} \frac{P\left(\mathbf{X} \mid \mathbf{W}\right) * P(\mathbf{W})}{P(\mathbf{X})}. \tag{1.2}$$

Since $P(\mathbf{X})$ is constant with respect to hypothesis $\mathbf{W}$ it can be discarded from the equation. The $P\left(\mathbf{X} \mid \mathbf{W}\right)$ represents the probability of input features giving the word sequence. At first glance, this probability may seem unnatural; however, it can be modeled using the HMM. The HMM is a very natural way to model speech because of its temporal structure and because speech can be encoded as a sequence of spectral vectors. The $P\left(\mathbf{X} \mid \mathbf{W}\right)$ component of the equation is referred to as acoustic model (Sec. 2.3). The $P(\mathbf{W})$ represents the Language Model (LM) (Sec. 2.4). The LM says how likely the given sequence of words is in the particular language and domain. The resulting equation is the underlying equation of statistical speech recognition,

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} P\left(\mathbf{X} \mid \mathbf{W}\right) * P(\mathbf{W}). \tag{1.3}$$

In the conventional model, each part of equation 1.3 is represented by a designated model. In hybrid systems some of these sub-models can be implemented using the DNNs.

In the case of the e2e, the whole model can be represented by several sub-networks which are usually trained together, jointly. The e2e models have a single loss function, whereas in a hybrid model each sub-module has to be optimized separately. The single loss function is a huge advantage of the e2e approach which makes the training easier, however, this property makes e2e models difficult to train in low-resource scenarios. All types of e2e architectures can be roughly described as two neural networks (sometimes more) connected together. Usually, one of them is an encoder. It encodes the audio features to the high-level feature vector representation, and the other is a decoder. It tries to decode the target token sequence from the vector provided by the encoder network. It is a very simplified description of e2e architecture principles, all types of e2e models have their peculiarities, but all of them follow this pattern.

1.2 Thesis goal and structure

The main goal of this thesis is to explore the application of the e2e approach to training the ASR models for languages that lack linguistic resources. These languages are referred to as low-resource. Language extinction, low number of speakers, and lack of linguistic experts who are able to transcribe the speech are some of the traits of low-resource languages. Improving the performance of end-to-end models on low-resource languages may allow ASR to be more accessible to a wider population.

It was decided to glance at the architecture of the conventional speech recognition systems first in chapter 2. Then e2e approach will be examined and compared with the hybrid one. All experiments will be conducted using the Recurrent Neural Network Transducer (RNN-T) e2e architecture in section 3.2.3. The main focus of this work is to alleviate the problem of data scarcity in languages like Vietnamese and Tamil through the adoption of different techniques, it is described in section 4.3.3.

Chapter 2

Conventional speech recognition

This chapter is devoted to introducing what the ASR systems had looked like before the "DNN revolution" happened. What sub-modules does the classical system consist of, and how these sub-modules are interconnected to predict the utterance depending on the input features. Let us explicitly mention that architectures that do not incorporate any DNN solutions will be referred to as classical; otherwise, it will be about hybrid models. However, the underlying architecture and division to sub-modules are the same for classical and hybrid models; all the systems based on the HMM will be referred to as conventional in this work.

It is good to notice that conventional systems are still widely used in production because of their flexibility, which allows adapting only the LM without retraining the whole ASR model from scratch. The transparent division between the language and acoustic models provides this flexibility. This property of conventional solutions saves significant computational resources and time, though training such systems requires linguistic expertise. The researchers and developers have to consult with linguists to understand the specifics of the chosen language and collaborate on modeling the pronunciation lexicon. Also, it is crucial to have a good idea of mathematics and proposed assumptions behind the HMM and GMM. This section is mainly based on the book by Mark Gales, and Steve Young [13]. *Kaldi* [35] toolkit was used to get acquainted with the hybrid approach and train the hybrid (HMM-DNN) model.

2.1 Architecture of classical systems

Generally, the sub-modules of a conventional model are distinguished into the following types:

- **Language model** predicts the probability of sequences of tokens. Tokens can be represented by words, word pieces, or characters. Typically, n-gram approach is used for creating the LM in classical and hybrid systems, (Sec. 2.4). This model is trained on the unlabelled text-only corpus.
- **Pronunciation dictionary (Lexicon)** contains probabilities associated with possible pronunciations of words from the language vocabulary. This model is usually designed in close collaboration with linguists. The main purpose of this sub-module is to map the phonemes to sequence of the words which will be re-scored by the LM. Another option is the graphemic lexicon, it removes the need of linguistic expertise, however, the mapping from sound to graphemes may be ambiguous.

- **Acoustic model** extracts information from the acoustic signal. If you read a line two times it will never be completely the same from the temporal and spectral point of view; this is the main challenge the acoustic model has to cope with. The HMM transitions can resolve the temporal variability problem and the GMM can be used for acoustic modeling. The main goal of the acoustic model is to represent the relationship between phonemes and features extracted from the input audio signal.
- **Decoder** takes outputs from all sub-modules mentioned above and searches through all possible word sequences. An internally created acyclic graph of hypotheses is called a lattice. Of course, different optimization algorithms and dynamic programming approaches are used to make the lattice tractable; otherwise, it would be computationally infeasible. The goal of the decoder is to output the most likely hypothesis in a reasonable time.

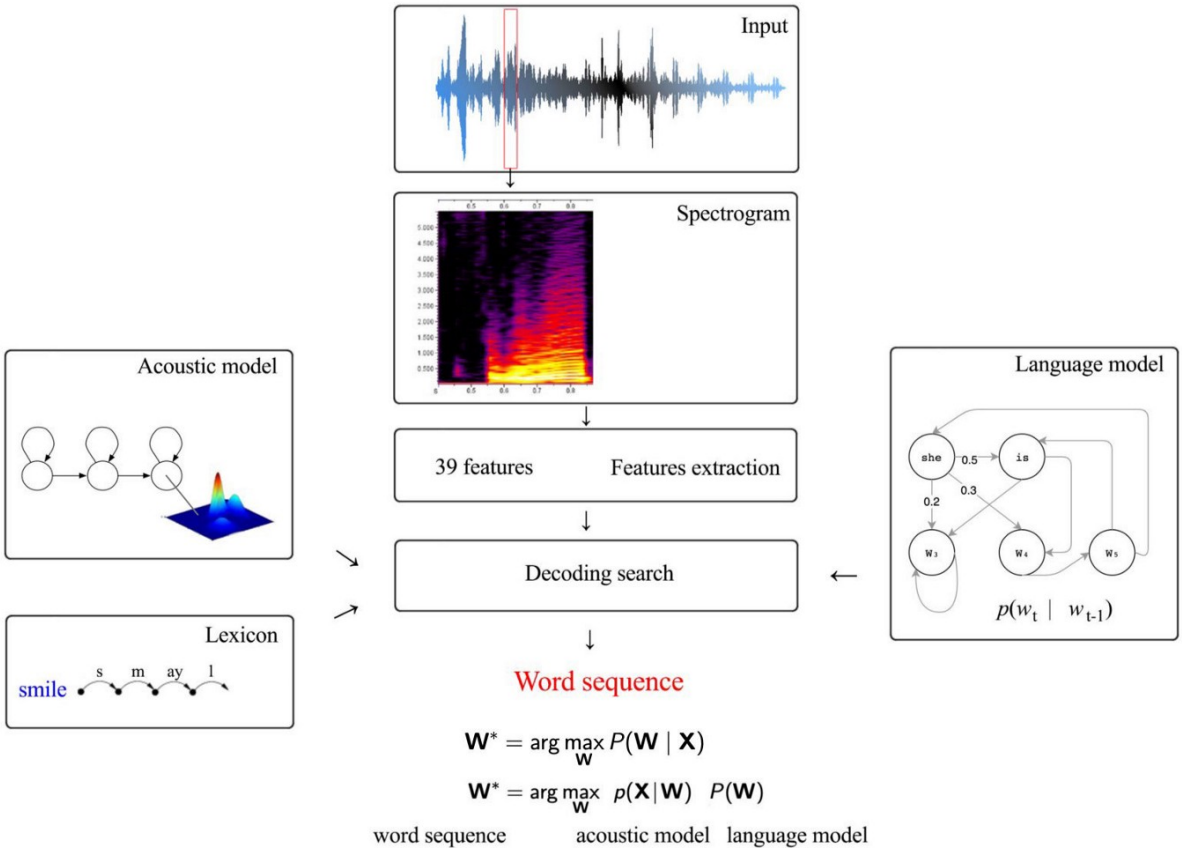


Figure 2.1: Classical speech recognition framework (Jonathan Hui, 2019, Speech Recognition — Acoustic, Lexicon Language Model[1])

The following sections describe mentioned sub-modules and processes in more detail.

[1] https://jonathan-hui.medium.com/speech-recognition-gmm-hmm-8bb5eff8b196

2.2 Feature extraction

For training both conventional and e2e ASR models, speech has to be represented as some feature vectors that can be "fed" directly to the model. Several types of features can be extracted from the audio, but the underlying features for the ASR are Mel-Frequency Cepstral Coefficients (MFCC). These features estimate the shape of the vocal tract filter of the speaker at the concrete moment in time.

MFCC pipeline can be divided to several stages [30]:

1. **Preemphasis** – amplifies energy in the higher frequencies. This improves phone detection accuracy, α is the filter coefficient, usually, it is 0.95 or 0.97,

$$y(n) = x(n) - \alpha * x(n-1). \tag{2.1}$$

2. **Framing and Windowing** – slicing the digital speech signal to frames. It is backed up by the assumption that frequencies in a signal are stationary within one frame. It would not make sense to apply Fourier transform on the whole signal since the frequency contours of the signal will be lost [12]. Frames are usually 25ms long with a 10ms overlap. After the framing process, one of the windowing functions is applied to every frame to prevent aliasing,

$$Y_w(n) = X(n) \cdot W(n), \tag{2.2}$$

   where $Y_w(n)$ is the windowed signal of the frame in the time domain, $X(n)$ is input signal and $W(n)$ is a windowing function. The input signal is multiplied by the windowing function sample by sample.

3. **Fast Fourier Transform** – the transition from time to frequency domain of the signal and decomposes it to frequencies. This operation is applied to every frame of the input signal. Output of this stage is *spectrum* of every frame,

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot exp(-j\frac{2\pi}{N}kn). \tag{2.3}$$

4. **Mel filterbank** – application of this triangle-shaped filter mimics how humans perceive sound. As frequency increases human ear perceives a narrower resolution of frequencies,

$$m = 2595 * log_{10}(1 + \frac{f}{700}). \tag{2.4}$$

5. **Extracting the cepstrum and computing the energy** – by applying Discrete Cosine Transform (DCT) the signal is transformed back to the time domain. For GMM processing, the output of this stage is usually truncated down to 12 coefficients, otherwise, the number of coefficients may be 40 or 80 depending on the setup. The energy of the frame can be easily computed as follows,

$$E = \sum_{t=t_0}^{T} x^2(t). \tag{2.5}$$

In the end, each frame of the signal is represented by 39 MFC coefficients where the first 13 are cepstrum coefficients plus the energy term, the next 13 are the first-order time derivatives with regard to the previous frame features and another 13 parameters are the second-order derivatives, concatenation of the first and second derivatives is an attempt to compensate for statistical independence assumptions made by HMM (Sec. 2.3). After the feature extraction stage, the input utterance is converted to the sequence of acoustic vectors which can be directly passed to the model.

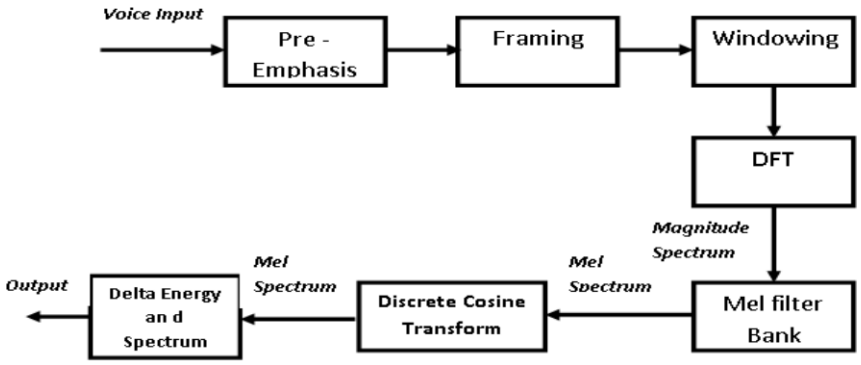


Figure 2.2: Feature extraction pipeline. Reprinted from [30].

2.3 Acoustic model and lexicon

The acoustic model in conventional systems is implemented using the HMM and GMM models. The HMM is a finite state machine with probabilities of transition between its hidden states and probabilities of observing acoustic features given the current hidden state; these are called emission probabilities. The emission probabilities are given by the GMM corresponding to the particular HMM state. GMM is a set of Gaussian models' parameters. Each GMM represents the feature vector probability distribution for single HMM states. A corresponding GMM can model each phonetic unit of the language; however, in practice, one GMM is tied to several similar phonetic units.

The HMM framework does several assumptions which contradict the nature of real speech signals,

- **The Markov assumption:** the next state of the system is dependent only on the current state.
$$P(s_i|s_1, s_2, ...s_{i-1}) = P(s_i|s_{i-1}) \tag{2.6}$$
- **The output independence assumption:** the observable features of the current state are independent of the observables from the previous states. So the observable at time t depends **only** on the current state s_t. The natural language is not that trivial, and the subsequent acoustic features are correlated a lot, so this assumption contradicts the real speech signal properties.

Even though these simplifying assumptions are not valid for the natural speech signal, this system successfully models real speech.

The HMM model considers speech for very short durations (e.g., 25ms) as static. For simplicity, assume that each hidden state represents a single phonetic unit of a language. During the training process, the HMM probabilities and GMM parameters are learned from the paired training data. Each GMM is trained to provide high emission probability to the acoustic vectors, which correspond to the phonetic unit at the current state.

When it comes to decoding, the task is to derive the most likely hidden state sequence from input feature vectors. Since the main focus of this work are e2e systems, going through the concrete algorithms used for decoding and training the HMM-GMM model is beyond this thesis; the book this section is based on offers a detailed explanation [13] of forward-backward algorithms and expectation-maximization training.

Each word can be decomposed into the "bricks" called base phones. With the pronunciation lexicon which models different ways of pronunciation of the vocabulary words the acoustic part of the underlying equation of ASR can be written like this,

$$P\left(\mathbf{X} \mid \mathbf{W}\right) = \sum_{\mathbf{v} \in \mathbf{V}} P(\mathbf{X} \mid \mathbf{v}) P(\mathbf{v} \mid \mathbf{W}), \tag{2.7}$$

where $\mathbf{V}$ is a vector of all possible pronunciations of $\mathbf{W}$ and $\mathbf{v}$ is a particular pronunciation. All speech utterances can be represented by the concatenation of phone models, HMM states. However, as it was already mentioned, the real speech does not meet the output independence assumption. This problem is partially solved by using the particular model for every unique phone plus its 1-neighborhood, triphone. The number of possible states will increase by the power of 3, therefore the number of model parameters will drastically increase. To alleviate this problem, similar triphones are clustered in the groups. Each group is mapped to the particular distribution of observable features. This process is called soft-tying. It will significantly decrease the number of model parameters, thereby reducing the likelihood of overfitting.

One of the main drawbacks of conventional systems is the complex training pipeline. Not only each sub-module has its own objective function, but also the training process of the acoustic model should be done in several stages. It is ineffective to try to represent words with context-independent phones, speech is continuous, and phones depend on their context. That is why the first stage of the training is to train the monophonic system, which will be subsequently used for training the triphone system. The soft-tying can be applied at the end. Take into consideration that after each stage, the re-estimation of the parameters should be done.

The discussed approach where the relationship between the input features and HMM state is modeled using the GMM is usually called classical. The GMM is considered to be a generative model, it learns how the data is distributed in space. However, this generative approach may be mismatched with the discriminative nature of the speech recognition task, and trying to fit such complex data as audio features to GMM may not an ideal solution [22]. To alleviate this problem DNN are trained for the frame classification, the systems with DNNs instead of GMM are called hybrid. In Kaldi to train the hybrid system, the pretrained HMM-GMM model has to be available. The input features will be fed to the network, its prediction will be compared with the HMM-GMM forced alignment and network parameters will be updated according to the loss function [18]. As the result, we have the DNN which is capable of classification of single frames [39].

2.4 Language model

Language modeling is an independent area of research in the natural language processing field. The LM estimates the likelihood of the occurrence of the given tokens sequence $P(\mathbf{W})$ which is the second term of the underlying equation for the ASR (Sec. 1.3). The LMs are trained on the large corpus of text data and it is very important to fine-tune the LM to the domain for which the ASR system is being developed. For instance, the LM trained on the corpus of classical English fairy tales may not be able to correctly predict the likelihood of word sequences consisting of biological terminology and professional slang.

The classical approach to developing the LM is to use an n-gram model. N-gram is a model that returns the probability of the n^{th} word in the sequence depending on the previous context described by $(n-1)$ predecessors. For instance, the bigram model approximates the probability of the word based only on the previous word $P(w_n|w_{n-1})$. The assumption that the current state statistically depends only on the previous state is called the **Markov assumption.** Given the bigram model the probability of a word sequence $w_{1:n}$ can be approximated as follows,

$$P(\mathbf{W}_{1:N}) \approx \prod_{k=1}^{N} P(w_k|w_{k-1}). \tag{2.8}$$

Another approach to implementing the LM is by using a recurrent neural network, LM [28]. A big advantage of RNN approach is that the context of the word is not limited to N words and the network can "remember" longer token sequences thanks to the Long Short-Term Memory (LSTM) architecture of RNN, however, it takes much more computation time to train such model. Another advantage is that the RNN-LM will not assign zero probability to a previously unseen token sequence, whereas the n-gram model will do so. This property of the RNN-LM has a positive impact on the overall generalizability of the model.

Perplexity function is used for assessing how good the LM is,

$$PP = P(w_1, w_2, ..., w_N)^{-\frac{1}{N}}. \tag{2.9}$$

The perplexity takes the test set of tokens and outputs a single number; the lower the number, the better the model is. The lower number means that the model was not "surprised" to see previously unseen data and can predict a token based on its previous context in that particular language. The higher the number, the more confusion the model has; it is not "sure" what token should go next.

2.5 Decoding

Having a trained hybrid system and prepared lexicon, the ultimate goal of the decoding process is to find the most likely sequence of the hidden states given the observable feature vectors – solve the underlying equation introduced in the ASR overview,

$$\hat{W} = \underset{W}{\operatorname{argmax}} \sum_{\mathbf{v} \in \mathbf{V}} P(\mathbf{X} \mid \mathbf{v}) P(\mathbf{v} \mid \mathbf{W}) P(\mathbf{W}). \tag{2.10}$$

The HMM model gives the probability of phone sequences; the lexicon describes pronunciations of the words, and the LM calculates the probability of the word sequences in

the language. Each sub-module provides the decoder with information that is necessary for finding the most likely word sequence.

Solving equation 2.10 by going through all possible state sequences is computationally infeasible for large vocabulary tasks. The beam search algorithm variations are used for the decoding to get the finite lattice of possible sentences and after that, the Vertebi algorithm is applied to obtain the best option. The main idea behind the beam search algorithms is that only a limited number of best hypotheses (beams) is kept in the memory during the decoding.

The acoustic model, lexicon, and language model can be represented as weighted finite state automaton with the probability of transition between the nodes. Some speech processing tool-kits (e.g. kaldi) take advantage of existing toolkits of state automata and compose the sub-modules together. The resulting finite state machine is usually determinized and minimized to make the decoding process more efficient. The resulting decoding graph incorporates phonetic and linguistic information and is capable of mapping the context-dependent phones to sequences of words.

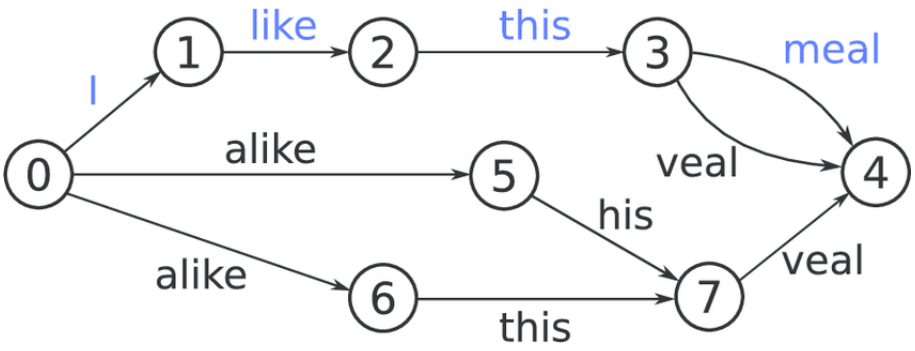


Figure 2.3: Word lattice. Reprinted from [44]

The beam search finds hypotheses, which are subsequently represented as a word lattice (Figure 2.3). The lattice is a compact representation of possible sentences. The final sentence $\hat{W}$ can be chosen by rescoring the lattice with the higher-order LM and choosing the hypothesis with the highest score. The reason why the higher-order LM is not used right during the beam search is computational inefficiency of such approach. Therefore, the balance is drawn by using the beam search for lattice generation and then rescoring the lattice with a higher-order LM. For efficiency purpose, the rescoring can be done with a model of a different type, for instance, RNN-LM. The lattice generation process is one of the reasons why hybrid models cannot be efficiently used on portable user devices, which only have limited computational resources.

The Word Error Rate (WER) metric is used for assessing how good the resulting model is,

$$WER = \frac{I + D + S}{N}, \tag{2.11}$$

where I stands for the number of *Insertions* of words that do not exist in the reference, D is the number of *Deletions* and S is the number of *Substitutions* of the words. N is the total number of words in the reference. Overall WER of the model is computed by averaging the WER for every sample from the test set. The same metric on the character level is called Character Error Rate (CER).

The LibriSpeech recipe (nnet3) was reproduced during the phase of exploring the hybrid ASR. The resulting performance of the hybrid model trained on the full LibriSpeech dataset [31] is in the table 2.1

	validation			test		
	fglarge	tglarge	tgmed	fglarge	tglarge	tgmed
clean	7.00	7.45	9.30	**7.68**	8.06	9.78
other	20.36	21.46	24.55	**21.49**	22.46	25.65

Table 2.1: WER(%) of HMM-DNN ASR model trained on LibriSpeech, fg is 4-gram, tg is 3-gram and large/med indicate the size of the text corpus the LM was trained on.

From table 2.1, it can be seen that the model with the higher order LM performs the best. It is predictable, since 4-gram model has a longer context on which to base its prediction. The difference between performance on clean and other data is quite significant, which may indicate that the augmentation applied during the training was not diverse enough. The performance on validation and test sets differs slightly, sometimes it is not the case. Extrapolating from this trend, it may be expected that in the e2e models trained on even lower-resource data the model may struggle to generalize over previously unseen speakers, environment distortions e.t.c.. The main reason for this may be that the augmentation was not applied during the training of this conventional model. Generally, the model is considered to be generalized when it can perform reasonably on test data with domain shifts from training data, such as by having previously unseen speakers.

Chapter 3

End-to-end speech recognition systems

In comparison to conventional system architectures, e2e architecture is not that complex. The resulting model is a DNN capable of directly mapping audio features to the corresponding text. However, for training an accurate model of this type, it is necessary to have several hundred hours of data. For most spoken languages, collecting and accurately labeling the sufficient amount of resources for building an e2e system is difficult and expensive.

The separation of single components in conventional models may bring better generalizability than e2e models. However, an apparent disadvantage of conventional architectures is using separate objective functions for each sub-module. This property significantly complicates the training process of conventional models. In contrast, e2e models usually use a single objective function. Also, e2e systems for high-resource language datasets have been shown to perform better, they have less WER, they can perform the online decoding without any lattice generation, and less linguistic expertise may be needed to train a recognizer.

For example, the RNN-T model with only 12M parameters, without the external LM, with character-based tokenization and augmentation which was trained on LibriSpeech data performs the same on the clean test set as the HMM-DNN model. Both models were trained on a full LibriSpeech data set. Table 3.1 shows the performance of the e2e model and hybrid models. Most likely the degradation of the HMM-DNN model performance happened due to the fact that no augmentation was applied during the training of the HMM-DNN model. It can be seen what gains the augmentation brings. The e2e model performs comparably well to the hybrid in this case. However, in the low-resource language scenarios, the performance of the e2e models may degrade significantly.

Model type	WER	
	clean	other
HMM-DNN	7.68	21.49
RNN-T	7.68	11.14

Table 3.1: WER and CER(%) of RNN-T and HMM-DNN models trained on LibriSpeech 960, the HMM-DNN was trained without augmentation.

In many enterprise environments, hybrid models still prevail. The reason is that an e2e model is harder to adapt to a particular domain. There is no clear division between acoustic and language models. For example, in RNN-T, the acoustic encoder, joint, and predictor network are inextricably intertwined. It is a monolith DNN with a vague division to acoustic and language models which cannot be easily adapted to the required domain [34], whereas in hybrid systems changing or fine-tuning the LM to the required domain is clear and trivial. In the case of the RNN-T, the adoption requires paired data which is difficult to obtain in large quantities, whereas the hybrid model can be adopted just by using only the text data, which is fairly easy to obtain. However, some of the most recent research try to leverage generalizability and easiness of training of the e2e systems and clear division of the HMM-DNN model to sub-components. They proposed RNN-T architecture with easily adaptable LM [9].

The e2e approach remains a popular topic in ASR today because DNNs are easy to train and they perform better on large datasets. For training the e2e models and conducting experiments *speechbrain* [37] toolkit was used. It is an all-in-one speech toolkit built upon PyTorch.

3.1 Tokenizer

The first decision to make during the development of an e2e model is to choose an appropriate tokenization strategy. The tokenization process is defined as splitting the text data into smaller chunks, such as words, sub-words, or characters, and assigning an identifier to each token. The list of tokens is called vocabulary. Different tokenization methods operate on different levels of granularity and may affect the complexity of the resulting model and its performance. Each approach has its pros and cons. For example, a model using a word-based tokenization strategy will not be able to recognize Out-Of-Vocabulary (OOV) words because it may not have output nodes to represent those words. On the other hand, the spelling of the known words will be very accurate. In contrast, the model with character-based tokenization will be able to recognize the OOV words, however, the spelling accuracy of vocabulary words may be poor.

Choosing an appropriate tokenization strategy may depend upon different factors. For instance, if the target language is fusional (the morphemes in the language usually express more than one meaning and they are hardly separable) and the amount of data is not much, it is better to consider the character-level tokenization rather than byte-pair-encoding (BPE [42]) or unigram. Both BPE and unigram produce a set of sub-word units according to the frequency with which they appear in the data. In the case of a fusional low-resource language, identification of common sub-word units becomes difficult because the morphemes are fused in different inseparable ways. If the BPE or unigram is applied to a low-resource fusional language, it might result in poor vocabulary where there are many tokens that occur in the data quite rarely due to the fusion of morphemes. The level of fusionality varies from language to language. Modern English is not as fusional as Slavic languages; therefore, the BPE tokenization may not produce good results for Russian or Polish, whereas the performance on English data may be significantly better. The languages where the sub-word units are easily separable are called agglutinative languages. These languages work like puzzles, the sub-word units are not modified during the fusion; therefore, it is easier to create a set of common sub-word units. Another essential factor is the character-richness of the language and the fact that the model is trained to be applicable to several languages [43].

It happens quite often that there is no apparent correspondence between spelling and pronunciation. Taking into consideration that an encoder in e2e systems extracts high-level acoustic features and passes them to the decoder, it may be a good idea to incorporate phonetics into the tokenization strategy. Vasileios Papadourakis et al. [32] showed that incorporation of phonetic information to the sub-word tokenization improved the overall performance of the model.

Choosing the correct tokenization method is important because the resulting tokenizer will be subsequently used throughout the whole training process. Furthermore, the same tokenizer must be used for training both acoustic and LM systems. Different tokenizers may result in different orders of tokens in the output layer, and in a multi-lingual transfer learning scenario, it may impact the ability of the target model to adequately learn from the source model of a different language.

3.2 End-to-end architectures

There are several underlying technologies and concepts on which the modern ASR is standing on [27] [36]. The following ideas are currently used in many state-of-the-art ASR models, they may be modified and combined, but the core stays the same. The fundamental ASR architectures are Connectionist Temporal Classification (CTC) [15], Attention-based Encoder-Decoder (AED), and RNN-Transducer models [14]. The CTC was a pioneer model that solved the alignment problem for e2e systems. The AED introduced the principle of attention layer, which will be later incorporated into the Transformer architecture [45]. The RNN-T extended the CTC with auto-repressiveness, which removed the CTC independence assumption.

3.2.1 Connectionist Temporal Classification

Before the CTC was introduced [15] the e2e systems were not considered to be appropriate architecture for labeling unsegmented sequence data because it required pre-segmentation and post-processing to get the predicted labels given the input features. The CTC was the first successful step toward state-of-the-art e2e systems.

Let $\mathbf{l}$ be the target labels sequence of length U, $\mathbf{l} = (l_1, l_2, ..., l_U)$ representing the transcript of the audio and $\mathbf{x}$ is the input features sequence extracted from the same audio, $\mathbf{x} = (x_1, x_2, ..., x_T)$. The CTC makes the assumption that the length of the input vector $\mathbf{x}$ can be at most the same as the target vector, otherwise, it would be impossible to align the targets to the input sequence (Figure 3.1), $T \geq U$. Alignment means that each input feature is mapped to one of the possible output tokens, where the token set consists of all tokens from an alphabet L and the blank token ϵ, which basically means "no label". For example, in hybrid systems, the alignment problem was solved by adding the loop transition to every hidden state which represents the triphone, therefore the system can either transit to the next state or stay where it is without outputting the next token of the sequence.

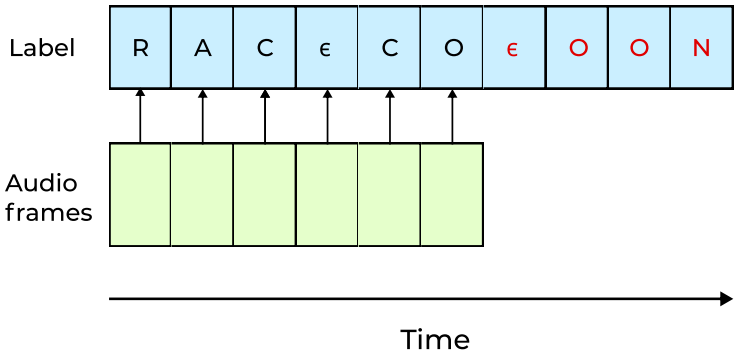


Figure 3.1: Here each audio frame is labeled with a letter from the alphabet L. However, the number of frames is less than the number of labels that is why the correct CTC alignment is impossible in this case.

The CTC introduced the way to interpret the NN output as a probability distribution function over all possible labels given the input sequence. Let $\mathbf{y}$ be the sequence of network outputs, then y_k^t will correspond to the activation value of NN output node k at time t. At each time step, the network makes a prediction if the given input feature represents one of the tokens from the alphabet L or the blank token ϵ.

The CTC output sequence which consists of blanks and tokens needs to be mapped to the grapheme sequence. The function $\mathcal{B}$ is used for this purpose. It does many to one mapping by collapsing all repeating tokens and removing all blank tokens from the CTC output (e.g $\mathcal{B}(c\ c\ a\ \epsilon\ t) = \mathcal{B}(\epsilon\ c\ \epsilon\ a\ t) = cat$). In case when the word has repeating letters, there is a blank token between those two letters to ensure that the resulting spelling will be correct. Therefore, the inverse function $\mathcal{B}^{-1}$ returns all possible alignments of the target sequence $\mathbf{l}$. All alignments have the same length as the input sequence.

To begin the alignment process, the target sequence $\mathbf{l}$ needs to be modified to $\mathbf{l}'$, so that at the beginning, at the end, and between every pair of the target tokens the ϵ token is inserted. It will give the system a choice between transiting to the next state or proceeding to output the blank token. Figure 3.2 contains $\mathbf{l}'$ states for the sequence ab (vertically), another example is for the sequence $HELLO$,

$$\mathbf{l} = (H, E, L, L, O) \quad \rightarrow \quad \mathbf{l}' = (\epsilon, H, \epsilon, E, \epsilon, L, \epsilon, L, \epsilon, O, \epsilon).$$

The probability of the sequence $\mathbf{l}$ given the input equals the sum over probabilities of all possible alignments $\mathbf{q}$ over the lattice (Figure 3.2) with the width equals the number of frames and height is $|\mathbf{l}'|$

$$P(\mathbf{l}|\mathbf{x}) = \sum_{\mathbf{q} \in \mathcal{B}^{-1}(\mathbf{l})} P(\mathbf{q}|\mathbf{x}), \tag{3.1}$$

where the probability of a single alignment can be calculated as the product of activation values at each time step

$$P(\mathbf{q}|\mathbf{x}) = \prod_{t=1}^{T} y_{q_t}^t, \tag{3.2}$$

where the $y^t_{q_t}$ equals the probability of observing label q_t at time t. It is represented by the output from the softmax layer. Dimension of the softmax output layer is $|L| + 1$.

To calculate the probability of observing the sequence $\mathbf{l}$ (Eq. 3.1) the CTC forward-backward algorithm is used. This algorithm computes the probability of occurrence in the given state at a particular time.

During the forward pass, the α is computed. In the case of CTC $\alpha_t(s)$ represents the probability of occurrence in the state s at time t, where each state represents the probability of each character in the dictionary at each time step. $\alpha_t(s)$ is a sum over probabilities of all possible alignments of the target sequence which start at time $t' = 1$ and at the time t go through the state s,

$$\alpha_t(s) = \sum_{\mathbf{q}:\mathcal{B}(q_{1:t})=l_{1:s}} \prod_{t'=1}^{t} y^{t'}_{q_{t'}}. \tag{3.3}$$

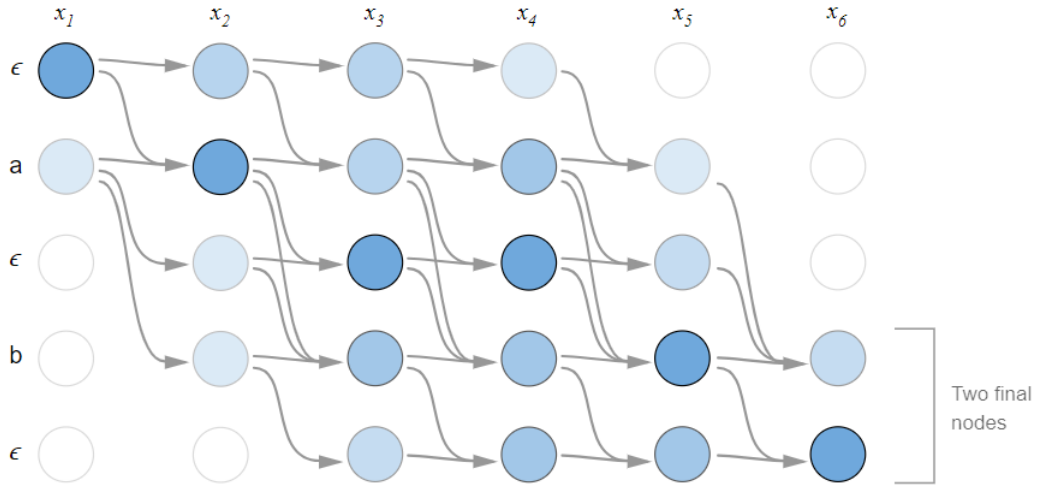


Figure 3.2: The input acoustic vectors are horizontally (t), states from the l' are vertically (s). Node (s, t) in the diagram represents $\alpha_t(s)$ – the CTC score of the sub-sequence $l'_{1:s}$ after t input steps. Reprinted from [17].

Similarly, the backward pass scores β are computed for the remaining sequence:

$$\beta_t(s) = \sum_{\mathbf{q}:\mathcal{B}(q_{t:T})=l_{s:|\mathbf{l}|}} \prod_{t'=t}^{T} y^{t'}_{q_{t'}} \tag{3.4}$$

Since α and β scores are computed as product of the network outputs it tends to underflow, that is why in practice the computations are performed in the logarithmic space. Having α and β scores computed the joint probability of every state in the grid can be calculated. The joint probability is referred to as γ. $\gamma_t(s)$ is the probability of all the paths q which correspond to the target sequence $\mathbf{l}$ and go through the state l'_s at time t,

$$\alpha_t(s)\beta_t(s) = \sum_{q:B(\mathbf{q})=\mathbf{l}, q_t=l'_s} y^t_{l'_s} \prod_{t=1}^{T} y^t_{q_t}. \tag{3.5}$$

The probability of observing the sequence $\mathbf{l}$ given the features $\mathbf{x}$ can be obtained by substitution of equation 3.2 and 3.1 to the 3.5 the probability of the sequence $\mathbf{l}$ given the input $\mathbf{x}$ can be expressed:

$$P(\mathbf{l}|\mathbf{x}) = \sum_{s=1}^{|l'|} \frac{\alpha_t(s)\beta_t(s)}{y^t_{l'_s}}. \tag{3.6}$$

To make the CTC system trainable using the stochastic gradient descent, the system must have a loss function that would be easily differentiable with respect to the network outputs y^t_k. During the differentiation process, the chain rule is applied. It is suggested in [15] to use maximum likelihood training, where the system is trained to output high probabilities to the matching $\mathbf{x}$ and $\mathbf{l}$ vectors from the training dataset. During backpropagation the following function has to be minimized:

$$ML(S, \lambda) = -\sum_{(\mathbf{x},\mathbf{l})\in S} log(P(\mathbf{l}|\mathbf{x})), \tag{3.7}$$

where S is the set of tuples of labeled training data and λ are the network weights.

3.2.2 Attention-based encoder-decoder

The attention encoder-decoder model is another type of e2e architecture. The first paper which applied attention architecture to the ASR problem was [8]. The main idea behind the attention is to teach the model to focus only on relevant information from the input and mask non-relevant. The AED network consists of two sub-networks, an encoder, and an attention-based decoder. The attention layer in the decoder decides what part of the input the decoder should focus on at the current time step.

Usually, the encoder is represented by several Bidirectional LSTM (BLSTM) layers. In the original paper, it was suggested to use the pyramidical structure of the encoder, each stacked layer would decrease the dimensionality of the input sequence by a factor of 2 which reduced the level of computational complexity.

Let $\mathbf{x}$ of the length T be the input sequence that is fed to the input layer of the pyramidical BLSTM encoder. The encoder will extract the high-level features of the input sequence and pass them to the decoder. Let sequence $\mathbf{h}$ of length U be the output from the encoder and $\mathbf{y}$ is the output token sequence. The distribution probability over the characters to be outputted is conditioned by $\mathbf{h}$ and previously outputted tokens:

$$P(y_i|\mathbf{h}, y_0, y_1, ..., y_{i-1}). \tag{3.8}$$

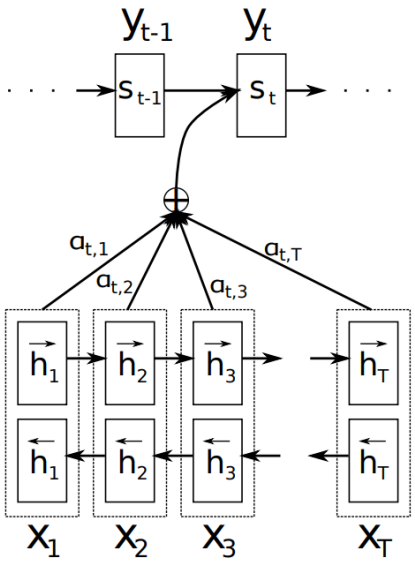


Figure 3.3: Attention example. Reprinted from [3].

Essentially, attention is a weighted sum of vectors from sequence $\mathbf{h}$, it is called context vector c_i where i indicates the output time step. The higher the weight the more attention the model will pay to this particular vector:

$$\mathbf{c_i} = \sum_{u=1}^{U} \alpha_{i,u} h_u, \tag{3.9}$$

where $\alpha_{i,u}$ is the output of softmax over scalar energies which are computed for each time step (Eq. 3.10). The energy e_{iu} depends on the decoder state s_i and input acoustic embedding h_u, $e_{iu} = a(s_{i-1}, h_u)$, where a is alignment model which assesses how well the input and the output match. Ideally, after training α will focus only on several frames of $\mathbf{h}$.

$$a_{i,u} = \frac{exp(e_{iu})}{\sum_{k=1}^{T} exp(e_{ik})} \tag{3.10}$$

In figure 3.3 it can be seen how the attention vector is created and how it is subsequently used during the decoding process for the prediction of the next character in the output.

The decoder is represented by the LSTM with the attention layer [11]. During the training process, the decoder gets $\mathbf{h}$ and $\mathbf{y}$ as the input. The probability distribution of the character to output depends on the current decoder state s_i and the context vector $\mathbf{c_i}$,

$$P(y_i|\mathbf{x}, y_0, y_1, ..., y_{i-1}) = FC(\mathbf{c_i}, s_i), \tag{3.11}$$

where FC is a fully connected layer with the softmax activation function.

The main difference between AED and RNN-T is that AED belongs to the label-synchronous systems which means that the AED model is driven by the text, it processes one label at each step and it stops after the label of the end of the sentence is recognized. On the contrary, frame-synchronous systems stop the decoding after the last input frame was processed. The label-synchronous approach is more effective in the sense that a large number of encoded frames are attended at once and the model can jointly extract relevant

information for each label from different parts and channels of the input [36]. However, the frame-synchronous approaches are the better fit for streaming tasks, because they do not need the whole input to start making predictions, they process the input in the frame-by-frame manner.

3.2.3 RNN-Transducer

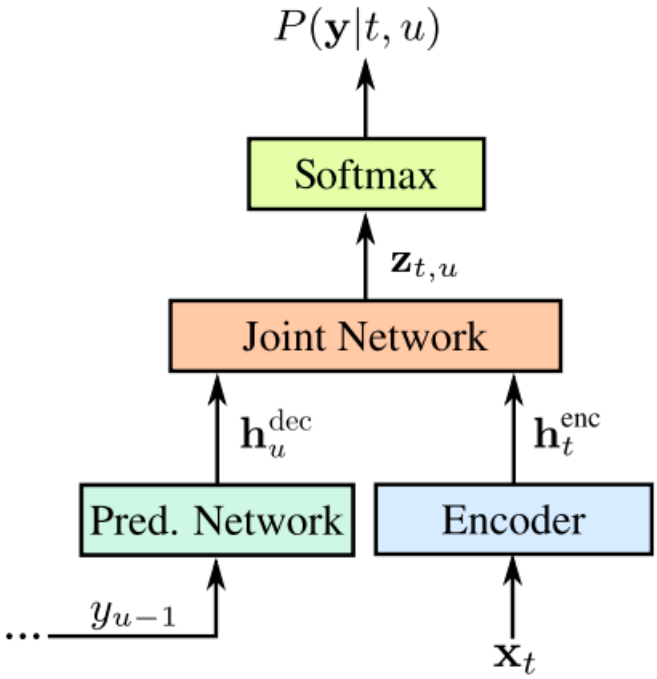


Figure 3.4: RNN-Transducer (Johan Schalkwyk, 2019, An All-Neural On-Device Speech Recognizer[1]).

The next very important e2e architecture, which is used in this work is RNN-Transducer (RNN-T). This architecture was proposed by Alex Graves [14]. The application of this architecture to the ASR problem [16] resolved the main CTC shortcoming and introduced a natural way of streaming ASR. A problem with the conventional CTC architecture is that it cannot produce reasonable results without an external LM because of the conditional independence assumption. The CTC model without an external LM will tend to misspell the words. Essentially, the RNN-T can be seen as an extension of CTC model, the additional component the RNN-T has is the prediction network which acts as an implicit language model. It means that predicting the current token l_u the model considers the past output context $\mathbf{l_{<u}}$ and the whole input sequence $\mathbf{x}$, $P(l_u|\mathbf{x}, \mathbf{l}_{<u})$. In the case of the online decoding the network will consider the frames only up until the current time, $\mathbf{x_{<t}}$.

Up until this architecture was introduced, the usage of RNNs was restricted to problems with beforehand aligned data, in RNN-T due to the incorporation of CTC concepts this restriction could be lifted because the model was able to learn the correct alignments between input and output sequences. In comparison to the CTC, however, the RNN-T does not introduce any requirements regarding the length of the sequences.

The RNN-T network consists of three sub-networks (Figure 3.4): encoder, decoder, and joint network. The encoder extracts embeddings of the input features $\mathbf{x}$, the output of the

[1] `https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html`

encoder is the embedding $\mathbf{h_t^{enc}}$. Predictor acts as an implicit language model, it produces the high-level representation of the utterance $\mathbf{h_u^{pre}}$ based on all previous non-blank outputs of the network. The joint network is a regular feed-forward network that fuses the outputs from both encoder and predictor sub-networks to $\mathbf{z_{t,u}}$, the logits from the joint network are fed to the softmax which subsequently normalizes them and returns probability distribution over output tokens k at each time step t,

$$P(l_u = k|\mathbf{x}, \mathbf{l_{<u}}) = softmax(\mathbf{z_{t,u}^k}). \tag{3.12}$$

Both encoder and prediction network are represented by RNN, to alleviate the problem of vanishing gradient the LSTM or Gated Recurrent Units (GRU) [10] is used. Also, the bidirectionality [4] can be added to the encoder to allow future acoustic context information to be used. However, in this case, it will not be suitable for streaming since it will need to read the whole input sequence before it can start the decoding process. Conventionally, the prediction network is a single RNN layer that finds long contextual information and dependency in the utterance.

The encoder network, also referred to as the transcription network, processes the input features in both directions (in the case of a bidirectional encoder) and computes forward and backward hidden states. The hidden states are then used for computing the output layer activations,

$$o_t = W_{\overrightarrow{h}o}\overrightarrow{h}_t + W_{\overleftarrow{h}o}\overleftarrow{h}_t + bias, \tag{3.13}$$

the output layer sizes of the encoder and the decoder are equal to the number of tokens in the vocabulary plus one additional node for the null output, in the CTC it was called the blank token. The idea behind the fusion of sub-network output vectors is that the model will consider both acoustic information in the current frame and previously emitted tokens. The feed-forward network used after the fusion allows more different combinations of linguistic and acoustic information the network is able to learn.

The RNN-T loss function is very similar to CTC one,

$$L_{RNNT} = -log(P(\mathbf{l}|\mathbf{x})), \tag{3.14}$$

where $P(\mathbf{l}|\mathbf{x})$ can be written as sum of probabilities of all possible alignments $\mathbf{q}$, $\sum_{\mathbf{q}} P(\mathbf{q}|\mathbf{x})$. The optimization of network parameters can be done using a simple chain rule and back-propagation.

During the inference, the beam search algorithm is used. It can return a set of the most possible hypotheses instead of the best one. Also, the length normalization technique is applied to the resulting hypothesis. Due to the fact that in each step the beam search algorithm multiplies probabilities, the overall probability of the sequence will decay. The shorter sequences will prevail on the output, that is why the normalization algorithm will favor the longer sequences with reasonable but not the best probability.

There are several problems with the RNN-T. First, it tends to "forget" the larger context even though LSTM/GRU layers are used. This problem is called "vanishing gradient" [19]. Also, the system built upon recurrent architecture processes the input data sequentially, which is a huge disadvantage from the efficient use of computational resources point of view. The AED architecture solves both problems by simultaneously processing the whole input sequence. The recent research focus [38] [20] is to leverage the pros of both architectures, combine them and make the model stream. This approach is called two-pass. During the second pass, the output of the first pass is re-scored. By performing the second pass, the WER decreases at the expense of an increase in the latency.

Chapter 4

Experiments

This chapter will apply the discussed e2e approach to low-resource language data sets. A problem with the low-resource languages is data scarcity which implies the lack of data diversity. It may be expressed as a small amount of data, a few number of speakers, the same recording environment e.t.c.. One of the previous surveys regarding the low-resource languages suggested that in order to build an accurate ASR model, at least 1000 hours of transcribed speech is required [5], and languages that do not have a sufficient amount of data can be considered low-resource.

Even though e2e architectures tend to outperform the conventional models on relatively clean datasets such as LibriSpeech, e2e solutions for low-resource noisy datasets may perform significantly worse than conventional solutions trained on the same data. For example, RNN-T and a conventional model trained on the same amount of noisy low-resource data perform notably differently [2], Andrei Andrusenko et al. showed that RNN-T can perform 30% worse than conventional HMM-DNN system for some low-resource datasets. This is the main problem the following experiments try to solve. Shrinking the gap between conventional and e2e architectures for languages with limited resources is one of the hot topics in the speech recognition community.

4.1 Data description

For the experiments, Vietnamese and Tamil were chosen as target languages. The Vietnamese dataset consists of the SPEECHOCEAN datasets[1] and the BABEL FLP dataset [2]. The Tamil data is also from the BABEL data collection. The SPEECHOCEAN data has a significant amount of reverberation noise, and BABEL FLP data is telephone conversations with dial-up noise, packet loss, and overlapping; therefore, the training data is considered noisy.

	Train	Dev	Test
Vietnamese	158.8	17.7	18.1
Tamil	62.2	6.9	7.6

Table 4.1: Size of the datasets in hours

[1] https://en.speechocean.com/datacenter/details/1709.html

[2] https://www.iarpa.gov/research-programs/babel

4.2 Baseline systems

The training setups for both Vietnamese and Tamil datasets were predominantly adopted from the *speechbrain* LibriSpeech recipe for RNN-T architecture, where the encoder network was represented by a combination of Convolutional Neural Network (CNN), RNN, and fully-connected DNN layers, and the decoder was just a single-layer GRU network. The goal of this work is not to get state-of-the-art results for the given datasets, therefore there is no need to train the two-pass or transformer model. Consequently, the following experiments were assessed on the conventional RNN-T architecture [14].

4.2.1 Model architecture

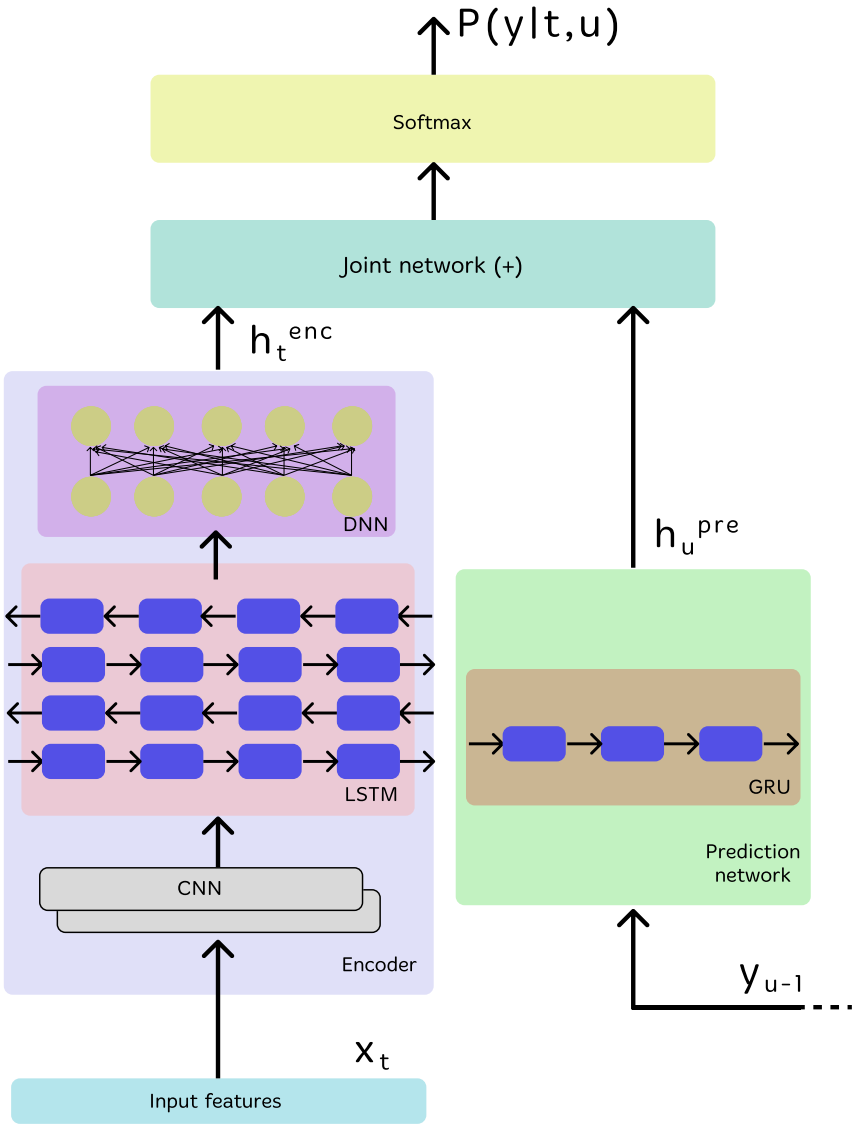


Figure 4.1: Baseline model architecture

First, the time domain augmentation was applied to the raw input audio. Each batch was augmented depending on the given augmentation probability. For example, 0.5 noise environment corruption probability means that the environment corruption augmentation will be applied to 50% of the samples in the current batch. On-the-fly augmentation ensures that the model sees different data in each epoch, it may increase the data diversity compared to offline augmentation where the same data is seen during each epoch. The used augmentation techniques were speed perturbation, reverberation, and noise contamination.

After the augmentation, the audio was pre-processed by the feature extraction block. The feature extraction step extracted the filter banks from the signal and normalized them. The filter bank is the output from the penultimate stage of the MFCC extraction pipeline (Sec. 2.2). All steps for computing the MFCC were motivated by the human physiology of hearing and perception except for the last stage - DCT. The reason is that the previous machine learning algorithms for acoustic modeling, such as GMM, expected non-correlated data; that is why the DCT had to be applied to the filter banks to get less correlated MFC coefficients, though some acoustic information was lost during the dimensionality reduction [12]. Dimensionality reduction is a separate process of omitting some of the output dimensions. Often, both linear transformation and dimensionality reduction are implemented together, by using a non-square transform matrix. In the case of e2e approach the DNN does not make such assumptions, therefore there is no need to apply DCT.

A previous research [41] revealed that vanilla DNN was vulnerable to unseen conditions, therefore feeding the features extracted from the augmented data straight to the feed-forward network would be unwise. Before feeding the features to the RNN they were processed by convolution across both dimensions of the spectrum. In the proposed architecture two-layer CNN was used. They can capture the local dependencies, while BLSTMs are good at modeling longer-span temporal contexts. Therefore, the CNNs are used together with BLSTMs to get the advantages of both temporal modeling approaches. CNN is good at modeling local temporal contexts because the convolution kernel only captures frames that are near the current frame. This method [29] significantly improves the model robustness.

The output from the CNN was passed to the 4-layer bidirectional LSTM with 256 nodes per direction in each layer. The output from the BLSTM was subsequently fed to the 2-layer feed-forward network with rectified linear units [1] activation function. The output from the encoder is then fused with the output from the prediction network.

The prediction network was a 1-layer GRU which may act as an implicit LM predicting the current token based on the previous outputs of non-blank tokens. The output from the encoder and decoder were passed to the joiner. The joiner fused the given vectors to one using the sum operation and passed it to the linear layer with the softmax activation function. The softmax made prediction on the current token.

4.2.2 Training process

After the data description files had been prepared the first component to train was the tokenizer. In this case, character-based tokenization was chosen. During the data preparation stage, all reference text was transformed to uppercase to slightly reduce the level of ambiguity of the resulting model. It resulted to decreasing the vocabulary size. For example, the vocabulary size of English without any special symbols was 26 instead of 52. The output layer scales with the vocabulary size, therefore the model has fewer options to choose from. This technique can be only used in cases when there is no need for differentiation of capital

and lowercase letters, and with languages which have the notion of capital letters, Tamil is not one of these languages.

Attention had to be paid to the processing of meta-pieces which occur in the training dataset, such as `<breath>, <ring>, <lipsmack>` e.t.c.. Incorrect processing of such tokens may result in poor generalizability of the resulting model. For example, the tokenization of `<breath>` as {`<, b, r, e, a, t, h, >`} may not be appropriate, the token `e` from the special meta-piece `<breath>` carries completely different acoustic realization than token `e` extracted from a real word, e.g `endeavor`. Such a problem may cause inconsistency in the training data when to the same grapheme corresponds to completely different acoustic realizations. In the following experiments, all meta-pieces such as `<breath>, <lipsmack>` e.t.c. were mapped to the `<unk>` token. The reason behind it was that different datasets have different meta-pieces and during the transfer learning process we wanted to use a tokenizer with the same token set to simplify the training pipeline. The training of each model was run until its full convergence.

4.2.3 Baselines performance

	Model type	CER(%)		WER(%)	
		clean	mix	clean	mix
Vietnamese	HMM-DNN	-	-	36.26	38.87
	RNN-T	27.90	41.99	42.06	56.79

Table 4.2: CER and WER of Vietnamese baseline model

	Model type	CER(%)	WER(%)
Tamil	HMM-DNN	-	58.80
	RNN-T	38.20	67.48

Table 4.3: CER and WER of Tamil baseline model

Table 4.2 gives the performance of e2e baseline model trained on the available Vietnamese dataset, where the clean test is a subset of the mix dataset. The RNN-T uses character-based tokenization and does not incorporate any external language model. The RNN-T system does not do any data augmentation, whereas the HMM-DNN model does. The following experiments will explore different types of augmentation.

Table 4.3 gives the performance of the models trained on the BABEL FLP Tamil data. The augmentation methods for the hybrid and e2e models are different. In the case of the RNN-T the augmentation is applied randomly on-the-fly, whereas the hybrid model is trained on the data that was augmented offline.

In both cases, the hybrid model[3] performance was shown for reference. Since the HMM-DNN systems do not implement character-based tokenization, the CER metric is not available for this type of system. The performance difference between the hybrid and e2e models is significant. In the following sections, we will explore different techniques which should help to alleviate the data scarcity problem and improve the generalizability of the resulting e2e models for both Vietnamese and Tamil datasets.

[3]The hybrid models were trained by Martin Karafiat

4.3 Alleviating data scarcity problem

The most obvious and useful solution to alleviating the data scarcity problem is to create more data. This data-hunger of e2e approach is one of the concerns of people who are skeptical about the future of pure neural network solutions, though it seems that this data approach still provides performance gains. The following sections explore regularization, data augmentation, and different approaches to model pretraining.

4.3.1 Regularization

The neural networks for the ASR are extremely powerful models with millions of parameters, they can learn very complex patterns and generalize over them. However, sometimes too complex and powerful architecture may lead to overfitting on the training data. Overfitting is dull memorizing of training samples rather than finding more general patterns in them. An overfitted model tends to show very good performance on the train set and significantly degrade on the validation and test sets. Usually, the overfitting indicates that the model architecture is too complex, and the number of trainable parameters is too high for this particular problem. Alternatively, the amount of training data may not be sufficiently diverse to show characteristics that are more general.

On the other hand, the model can be under-fitted. Underfitting is a problem when the model is not powerful enough to learn the pattern in the training dataset, and therefore, also cannot generalize over the previously unseen data. A problem may be with a too simple model architecture or the insufficient number of training epochs. The point when the model is considered to be well-fitted is right at the time when both validation losses reached a local valley of convergence, in the interval after under-fitting but before overfitting.

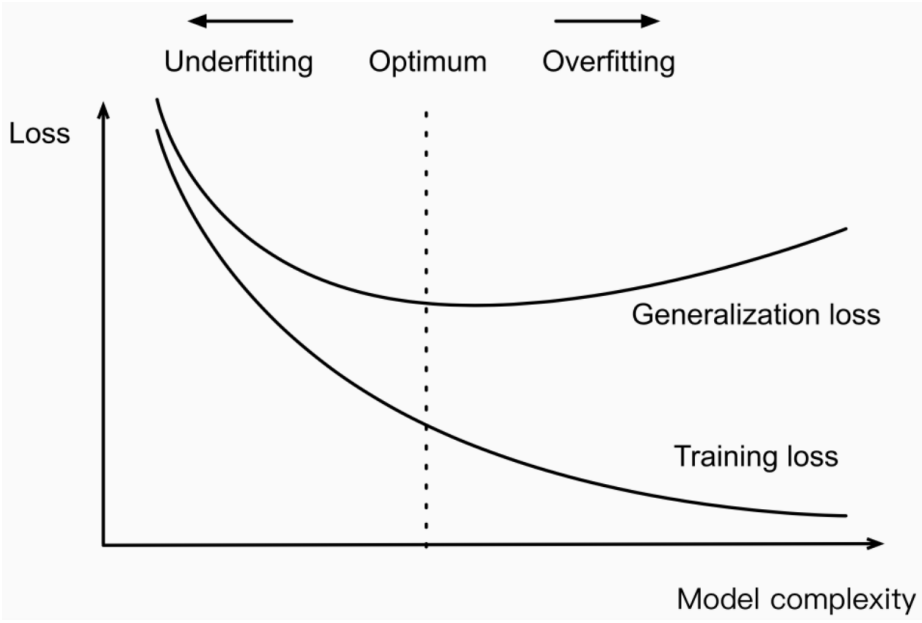


Figure 4.2: Three states of the model (Ajitesh Kumar, 2022, Model Complexity Overfitting in Machine Learning[4])

[4] https://vitalflux.com/model-complexity-overfitting-in-machine-learning/

In figure 4.2 both cases can be seen. It shows how to spot the mentioned model states from the plot of train and validation losses. The first interval shows the under-fitted model which has not reached its peak performance yet. If the model is trained for a sufficient number of epochs, it converges to when the train and validation loss is in the valley. If the model is trained for too many epochs then the parameters may memorize the training data pattern instead of learning more general trends, and the train loss will continue to decrease whereas the validation loss will start growing.

When only several hours of training data are available, which often happens in the case of real low-resource languages, the overfitting can occur much earlier. Some of the methods used in the ASR to prevent the model from overfitting are dropout, data augmentation, and L1/L2 regularization. The goal of these techniques is to discourage the model from learning too complex patterns. Yingbo Zhou et al. [48] showed that combining dropout and augmentation in the ASR tasks leads to significant improvement in the overall test set performance. Therefore, we applied both of these regularization methods within the experiments. L1 and L2 regularizations are components that can be added to the loss function; they will penalize big weights changes in the network,

$$ObjectiveFunciton = Loss + \lambda \sum_{w} w^2, \tag{4.1}$$

where λ is a regularization coefficient and w are the network weights. This is an example of L2 regularization, which is often used in neural networks. In the following experiments, neither L1 nor L2 regularization is used because the amount of training data is considered to be sufficient.

The other mentioned regularization hack is a dropout. The main idea behind the dropout is to let the model train only the random subset of its parameters in each epoch. It will prevent the model from zeroing one subset of model weights and maximizing the other. In the experiments the dropout rate was set to 15%, therefore, for each epoch 15% of randomly chosen network weights will not be updated.

One more way how to alleviate the lack of data problem and improve the generalizability of the resulting model is to artificially increase the amount and diversity of training samples. That is what audio augmentation does. Audio augmentation is the technique of increasing the amount of training data by transforming the original audio [26]. The transformation can be applied both in audio and frequency domains. Some examples of augmentation techniques are perturbation (changing the speed factor), dropping frames and frequencies, and adding noise or reverberation for simulating a noisy environment. The augmentation transformations can be precomputed offline before the actual training, also, they can be computed at the training time, on-the-fly. In *speechbrain* the augmentation module is implemented as a SpecAugment [33] but in the time domain, all transformations are computed on-the-fly. On-the-fly in this context means transforming the audio right before the creation of the batch.

In this report three augmentation techniques were used, speed perturbation, reverberation, and additive noise. Speed perturbation mimics the speaker with a variety of speaking rates present in the training dataset. It encourages the model not to focus too much on the rate of speech and focus on the linguistic content of the utterance. Ko et al. [26] showed that this low implementation cost method provides better results than more sophisticated tempo perturbation and voice tract level perturbation [23]. Reverberation is the convolution of the original signal with Room Impulse Response (RIR). It is used in order to simulate the far-field speech, different positions of the microphone, and environments. The

additive noise augmentation is adding various extraneous noises to the signal. For example, running water noise, white noise, the noise of the city e.t.c.. Even though there are many techniques how to diversify data, it is recommended to use new data and not to rely on the augmentation much. As someone said: "There is no better data than more data". However, large quantities of diverse and accurately transcribed new data is often difficult or expensive to obtain.

Augmentation experiment

	CER		WER	
	clean	mix	clean	mix
no augmentation	27.90	41.99	42.06	56.79
pre-augmented data (offline)	26.96	**36.62**	40.48	**51.48**
speechbrain augmentation (online)	**25.37**	38.94	**38.37**	53.24

Table 4.4: CER and WER (%) of Vietnamese models with different augmentation

Table 4.4 compares RNN-T models trained on Vietnamese language, trained with and without different types of augmentation. The table shows the gains that augmentation brings, the WER and CER drop by several percent. It is expected since the model sees more diverse data during the training process and it contributes to better generalizability. Also, it shows an interesting difference in performance between two types of augmentation, online and offline. The model trained with online augmentation performs better on the clean dataset, whereas the model trained on the pre-augmented data is better at recognizing distorted speech.

Apparently, the attempt to mimic the offline method of augmentation using the online approach was not successful and the difference between the ways how the data is augmented is different. The offline augmentation was performed on the whole dataset, the data set became 4 times larger. 1/4 of all data was reverberated, 1/4 was contaminated with the noise, the perturbation was applied to 1/4 of data and 1/4 of data remained original. In the case of online augmentation, the augmentation rate for all types of augmentation was set to 0.33. The RIRs which were used for the reverberation augmentation were generated by *speechbrain*, whereas the RIRs for the offline augmentation were prepared beforehand. The same with the noise recordings.

4.3.2 Transfer learning

Transfer learning is another technique to artificially increase the amount of training data the model has seen. Transfer learning is using the parameters of a model pretrained on the data from a different domain as a starting point for training a model on the target domain data. It is presumed that during the first round of training the model will learn to extract information which will be useful for the target domain. The process of the second round of training on the target dataset is usually called fine-tuning. Fine-tuning not only shows better results for the low-resource e2e models but also reduces the expenses because for the fine-tuning of the model it is not necessary to have access to a large amount of computational power and the data the model was pretrained on. It has become common

practice to start from large pre-trained foundation models, and apply fine-tuning toward the target task[5].

In one of the first attempts to apply the transfer learning to the low-resource ASR problem [21] connected several softmax output layers with different token sets and trained this model with multilingual data. The resulting model is multi-task model architecture with multiple separate softmax outputs. The proposed shared-hidden-layer architecture allowed transfer learning between languages to yield benefits; however, there are several concerns about this architecture arise. What if several tokens share the same acoustic representation? What impact does the intersection between the token sets have? The opaque nature of DNNs causes this ambiguity.

The other way of leveraging the prior knowledge for training a model for a low-resource language is transferring a pretrained model with output layer change. Usually, there is a disjointment between the high-resource and target language token sets, and to solve this problem the researchers applied different techniques such as training only particular layers of the model and changing the output layer to the target token set. This sharing of hidden states is non-controllable and latent. Also, it requires more engineering efforts.

Recently, a more natural and transparent way of transfer learning was proposed. This approach [25] leveraging transliteration technique does not require any output layer change and relies on high phonetic similarity of the languages.

4.3.3 Romanization and transliteration

For the pretraining without the output layer change, the data used for pretraining can be transliterated to the target language. In this case, transliteration from one language to the target language is transcribing the audio from one dataset using similarly sounding grapheme sequences from the target language. For example, transliteration from English to Czech language is the transcription of English audio using the similarly sounding Czech graphemes, *h e l l o* → *ch e l o u*. Subsequently, the model pretrained on transliterated reference and original audio data of the source language can be fine-tuned only on the target language dataset. In [25] the idea of cross-lingual transliteration, where for a model trained on one language data was fine-tuned on the language with a completely different token set, was improved. The benefit of high-resource language transliteration may depend on the amount of low-resource target language data and the relatedness of the languages. The fewer target language data available the more significant may be the impact of pretraining. Pre-training on the language which is related to the target one may also increase the impact on the overall result.

Speaking of transliteration, it is also worth mentioning the romanization technique, which is denoted as the representation of one language using only Latin graphemes. Basically, romanization is many-to-one transliteration where audio is transcribed using only the Latin graphemes. However, this technique can only be applied to the languages which have the token set which is fairly similar to Latin alphabet, e.g. Czech, German or Lithuanian. In case of this work, romanization means simple removing of all diacritics and tone signs from the Vietnamese reference, có sân → co san.

Fine-tuning experiments

The multi-lingual transfer learning pipeline consists of two steps:

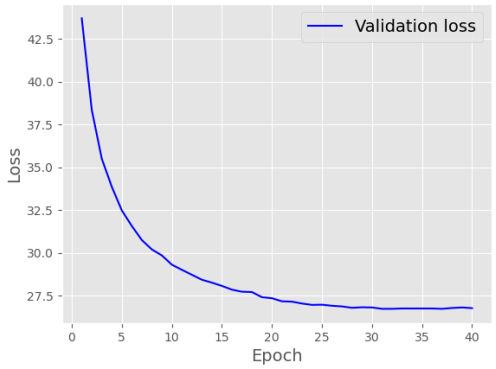
[5] `https://huggingface.co/models`

1. **Pre-training** on the source language is training the model usually on a high-resource language data, the model will learn to extract the relevant features and map them to the tokens. The resulting model parameters will be the starting point for the fine-tuning stage.

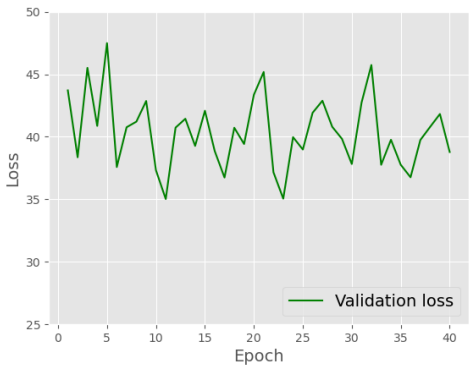
2. **Fine-tuning** on the target language is tweaking the parameters of the pre-trained model to the target space so that the final model performs reasonably well on the target data. This method can speed up the training and help to overcome a small target data set.

During this several-stage training, a problem connected to the learning rate arises. The initial learning rate at the fine-tuning stage has to be adjusted experimentally. Usually, the scheduler module updates the learning rate with respect to the validation loss, in this work NewBob scheduler is used.

There are two systems, the learning rate of the first system 4.4(a) was scheduled with respect to the mix of the target language dataset and transliterated data; the second system 4.4(b) learning rate was scheduled with respect to only transliterated data. Updating the learning rate with respect to the target language validation dataset does not behave well in cases when the model is pretrained only on transliterated data from the source language. In this case, the loss may diverge and the model may not learn any reasonable dependencies. Figure 4.4(b). The validation loss of the first systems was converging gradually 4.4(a). As long as the validation loss continues to decrease it implies that the learning rate is reasonable. Therefore, the model was pretrained on the mix of transliterated LibriSpeech (1000 h) and target language Tamil data (60 h).



(a) The training set is a mix of the target language and transliterated data



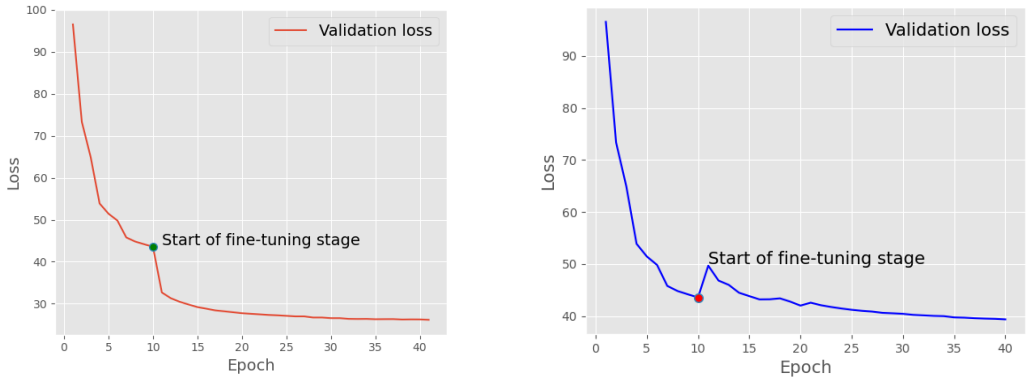
(b) The training set is only the data transliterated to the target language

Figure 4.3: Training the model only on transliterated data from the source language and on the mix of transliterated and target language data.

A model trained only on transliterated data from the source language may not develop accurate language modeling behavior for the target language. Text transliterated to the target language may partially preserve the phonetics but it may not make any sense from the semantic point of view. That is why the learning rate scheduling with respect to the target language may fail. In case the validation during pretraining is done with respect

to the target language, the solution to this would be the pooling or mixing of the target language and high resource training language datasets during the pre-training stage.

The scheduler used in the experiments works in the way that it multiplies the current learning rate by the annealing factor when the validation loss difference between the last and current epochs is less than a pre-set threshold. It may happen that after the beginning of the fine-tuning stage the validation loss will abruptly increase, it may imply that some parts of the model started learning "from scratch", in this case, the main purpose of the pretraining was wasted.



(a) (b)

Figure 4.4: Start of fine-tuning situations

Figure 4.4 shows both of the cases when the learning rate is appropriate (a) and when it is not and the model resets the pretraining effect (b). Of course, the learning rate is not the only hyperparameter which may have an impact on this, the domain mismatch may also cause such behavior during the beginning of the fine-tuning stage.

Romanization

Publicly available resources allowed for easy application of transliteration from English to Tamil. However, an initial search suggested that equivalent resources between English and Vietnamese may not be available. Rather than building a transliteration framework between English and Vietnamese from scratch, this work instead proposes to use romanization as a bridge between the written forms of these two languages. The following section examines the advantages this approach can bring.

Romanization goes in the opposite direction in the sense that the target language is mapped to English (in transliteration English was transliterated to Tamil). During the romanization, all acoustic information from the text had been removed (có sân $\rightarrow$ co san), but it made the fine-tuning of the pre-trained LibriSpeech model easier than changing the output layer. The main motivation for this is to start fine-tuning stage without any layer change, since the token set of both datasets is the same, though they may have different acoustic representations. For this experiment the baseline is taken from the augmentation experiment, the baseline is the Vietnamese model with the online augmentation applied.

	CER(%)		WER(%)	
	clean	mix	clean	mix
Vietnamese Baseline	25.37	38.94	38.37	53.24
Romanized Vietnamese	22.82	36.42	37.29	52.68

Table 4.5: Vietnamese and Romanized Vietnamese models

The models shown in table 4.5 are RNN-T trained on Vietnamese and romanized Vietnamese. The initial proposal was that the discarding of tones and diacritics will result in having fewer possible choices of output tokens. It may reduce the prior probability of choosing the wrong token. Therefore, the error rates can be expected to be lower. Later in the experiment, the attempt to restore the diacritics in hypotheses will be taken.

However, these models cannot be comparable because they are tested on two different datasets (Vietnamese and romanized Vietnamese). Their performance is provided for reference. The romanized model was assessed against the romanized reference. The Vietnamese language becomes ambiguous without the tone marks and diacritics, however, the native speakers should be able to read long context sentences without changing the original meaning.

	CER(%)		WER(%)	
	clean	mix	clean	mix
Romanized Vietnamese	22.82	36.42	37.29	52.68
Fine-tuned Roman. Viet.	**22.18**	**34.71**	**36.24**	**50.95**

Table 4.6: Romanized Vietnamese model performance

The Romanized Vietnamese model in this case is the model from the previous table 4.5 trained only on the romanized Vietnamese data. The fine-tuned model from the table 4.6 is the model pre-trained on the mix of romanized Vietnamese and LibriSpeech (960 h) and fine-tuned only on the romanized Vietnamese data. As can be seen from the table, there is gain in the performance, however, keep in mind that it is hardly readable romanized version of the language and these results are not comparable to the proposed Vietnamese baseline shown in table 4.5.

This work suggests a separate component that takes romanized Vietnamese as input, and hypothesizes Vietnamese with diacritics and tone marks as output. This component was adopted from this[6] public repository. This system is referred to as the restoring model.

To make the Vietnamese baseline and fine-tuned romanized models comparable. I decided to train the system which will restore the diacritics and tone information in the hypotheses of the fine-tuned romanized model right before the WER computation. As such, the final hypothesis can be compared against the original, non-romanized, Vietnamese reference. Since the resulting model will be assessed against normal Vietnamese (non-romanized), the model performance will highly depend on the restoring model accuracy. The model was fine-tuned on the references from the training Vietnamese dataset. The architecture of this restoring model is Transformer which takes a sentence written in romanized Vietnamese and tries to restore the diacritics and tone information of single

[6] `https://github.com/duongntbk/restore_vietnamese_diacritics`

syllables depending on the full context. The accuracy of the resulting model tested on the fraction of the references from the Vietnamese training dataset was 90.82%.

	CER(%)		WER(%)	
	clean	mix	clean	mix
Vietnamese Baseline	**25.37**	**38.94**	**38.37**	**53.24**
Restored diacritics	28.09	40.63	45.85	57.98

Table 4.7: Restored Vietnamese diacritics model performance

It can be seen from table 4.7 that the diacritics restoration method is not a good solution. Presumably, there is a significant text data mismatch caused by the errors from the speech recognition system. The restoration model error is also transferred to the overall error of the model. The lack of information about the audio may be another reason. Another way to fine-tune this model could be training on the pairs of hypotheses outputted by this ASR model and ground truth values, that would decrease the model mismatch and the model would also work as the error correction model, however, this approach seems to be too cumbersome but an idea behind the e2e approach is the simplification of the training process.

Transliteration

The second fine-tuning experiment was conducted on the model pretrained on the transliterated LibriSpeech 1000 hours dataset mixed with target Tamil data. Fine-tuning was done on several intermediate models and on the fully converged model to check whether it is important to pre-train the model till its full convergence. The transliteration from English to Tamil was done using the publically available tool[7].

This work hypothesizes that using a fully converged model for the fine-tuning of the model trained on transliterated data may be less productive than using a model from an intermediate epoch. It is expected that too much training of the model during the pre-training stage may make it difficult to alter the model's behaviour during the fine-tuning stage. This persistence may result from, for example, the model parameters being stuck within a loss function valley, or the model parameters growing too large. If the model gets trapped within a loss function valley from pre-training, then this may manifest as the model utilizing too much of the transliterated language modeling information that it had learned during pre-training, which may not be relevant for the target language. In other words, during the pertaining, the model will likely learn the language-related information which may not be relevant to the target language.

[7] `https://github.com/Ezhil-Language-Foundation/open-tamil`

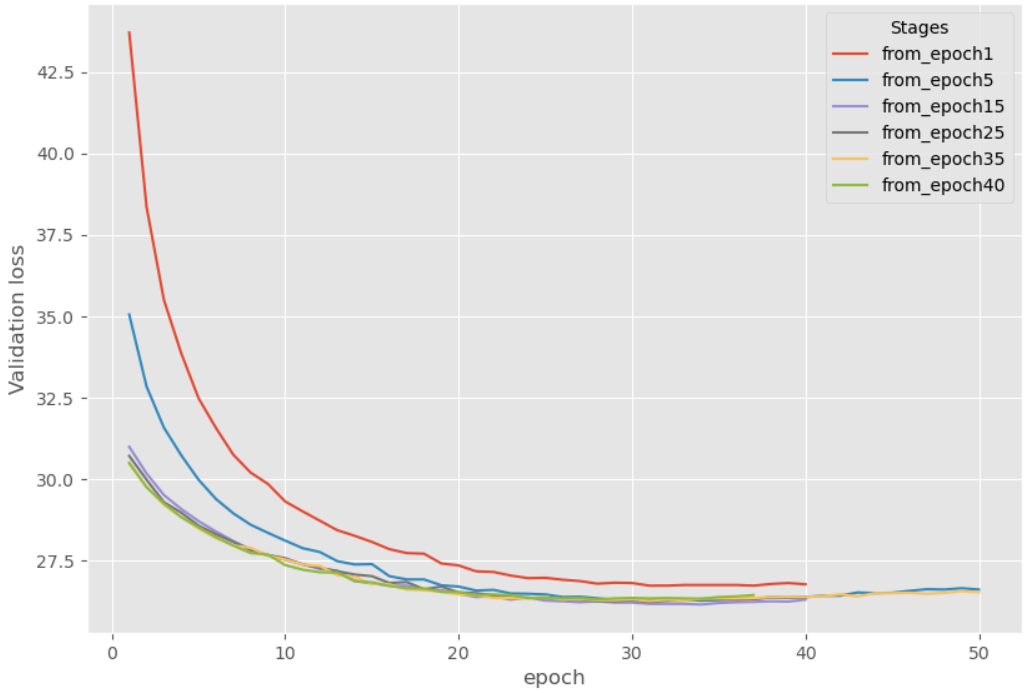


Figure 4.5: Fine-tuning the model from different epochs of the model pre-trained on transliterated data.

In figure 4.5, it can be seen that fine-tuning the pre-trained model from the first epoch resulted in poorer generalization. However, there is no visible difference between fine-tuning of the model from the 25^{th} and 40^{th} epochs of pre-training. Also, no difference in the speed of fine-tuning was observed between usage of the pretrained model from the 20, 25, 30 and 40 epochs. All of these models needed about 30 epochs of fine-tuning to converge.

Model	CER(%)	WER(%)
Tamil baseline	38.20	67.48
Fine-tuned from 1 epoch	37.35	67.28
Fine-tuned from 5 epoch	36.52	66.82
Fine-tuned from 15 epoch	**36.39**	66.05
Fine-tuned from 25 epoch	36.45	**65.96**
Fine-tuned from 35 epoch	36.46	66.17
Fine-tuned from 40 epoch	36.52	66.13

Table 4.8: Fine-tuning of the model pre-trained on mixed transliterated and target data, beginning from different pre-training epochs.

Table 4.8 shows that the most successful model was fine-tuned from the 25^{th} epoch of pre-training. The performance between the model fine-tuned from the 25^{th} and the last epoch seems to be negligible, we should check if this difference is statistically significant. The test used to check the difference between two systems is called Match Pair Sentence-Segment Word Error (MPSSWE) test [15]. This is a two-sided statistical significance test that divides the hypotheses from recognizers and the ground truth to segments, in this case – words. The error in a segment has to be statistically independent of the errors in other segments. The hypothesis that the difference between two models is statistically significant is assessed depending on the average difference in the number of errors between the two systems.

The *SCTK* tool `sclite` and `sc_stats` were used for this testing. Let the level of significance α equal 95%. The output of the models which were fine-tuned from the 25^{th} and 40^{th} epoch and the MPSSWE test was applied to them. The resulting p-value equaled 0.298 which implies that there is no statistical difference between these models on this level of significance. Since the difference in the performance of the resulting fine-tuned models is negligible it is not necessary to waste the resources and pre-train the model till its full convergence.

4.3.4 Components freezing and initialization

In [24] Vikas Joshi et al. applied different methods of transfer learning to RNN-T model. They showed that a randomly initialized model produces significantly worse results on the test data than a model with pretrained parameters. Also, they showed that transferring the whole model gives the highest WER reduction, however, transferring only the encoder part accounts for most of the gains. It is suggested that the encoder may learn language-independent features which can be applicable to any language.

Even though the prediction network is not the pure LM [9] it may still learn some linguistic information during the training. Since the linguistic information carried in transliterated reference may not be relevant from the target language perspective, in this work it is believed that it may be beneficial to *randomly initialize* joint or/and predictor networks before beginning the fine-tuning stage. These sub-networks may reach the local minimum during the pretraining and it may be difficult to correctly fine-tune their parameters to the target language space.

The other technique which might be helpful in this case is *component freezing*. The idea behind it is that the parameters of the frozen part or a layer will not be updated during the backpropagation. I suggest that before the fine-tuning stage it may be helpful to freeze the encoder (and joint) part and focus on updating the rest of the network. After the scheduler will decrease the learning rate below a pre-set threshold all the components will be unfrozen and training will proceed with the updated learning rate. I

The following algorithm will be referred to as two-phase fine-tuning:

1. Freeze part A of the network (after the pre-training stage is finished)
2. Train part B till the learning rate reaches a pre-set threshold
3. Unfreeze part A
4. Train the whole network until full convergence

The following experiments will fine-tune the model from the most successful intermediate epoch 25. Different configurations will be tested, `freeze_enc` – two-phases training with

encoder freezing, `init_pred` – random initializing the prediction network before joint fine-tuning, `freeze_enc_init_pred` – two-phases training with encoder freezing and prediction network initialization, `freeze_enc+joint_init_pred` – two-phases training with encoder and joint freezing and prediction network initialization, `freeze_enc_init_pred+joint` – two-phases training with encoder freezing and prediction and joint network initialization.

Configuration	CER(%)	WER(%)
No freezing/initializing applied	36.45	65.96
freeze_enc	36.59	66.11
init_pred	**36.18**	**65.92**
freeze_enc_init_pred	36.87	66.22
freeze_enc+joint_init_pred	36.89	66.54
freeze_enc_init_pred+joint	36.79	66.52

Table 4.9: Component freezing/initialization

Table 4.9 shows that the model with a randomly initialized prediction network provided a small performance gain. However, the MPSS test showed that there is no statistically significant difference between `init_pred` and baseline models since the p-value is 0.873. It may support the suggestion that in the RNN-T architecture the prediction network does not act as the LM [9] and random initialization of this module may have a negative effect on the correct generation of blank tokens for coordination with the encoder. The freezing also did not show any gains, probably, because the joint training of the RNN-T model is a more natural and efficient way.

Chapter 5

Future work

Some of the techniques used in the experiments helped to get some performance gain, however, there still exists a performance gap between the e2e and conventional model performances on these low-resource tasks. Furthermore, it is reasonable to assume that the same proposed techniques may be applied to conventional models too, bringing further gains to those models, and thereby widening the performance gap even further.

In a more comprehensive and deep experiment, it may be beneficial to compare the discussed approaches with some of the existing multi-lingual transfer learning techniques, such as changing the softmax output layer or multi-language models. However, this thesis did not compare the proposed method against these other methods. This comparison was not done due to the time limits, the time that it would have taken to get each of these alternative approaches to work, and the large amount of information that had to be processed and comprehended within this time span.

The main goal of transliteration and romanization approaches was to make the multi-lingual transfer learning process more smooth and less ambiguous. The fact that the token set during pre-training and fine-tuning is the same ensures that all the network connections which are responsible for mapping between acoustic features and graphemes are optimized smoothly throughout the whole training pipeline. However, in this case, the only tokenization method which makes sense for these approaches is character-wise. It is known that BPE or unigram tokenization techniques usually produce better results, but the constraints of the discussed methods are hardly overcomable.

In the long run, data scarcity is a huge problem of low-resource ASR, small diversity of training data impacts the generalizability of the resulting models. The solutions considered before were trying to artificially increase the amount of paired training data. Many recent works focused on the low-resource ASR try to leverage the real unlabeled data instead which is easier to get. Manual human labor is required to correctly label the data. For some languages, it is difficult to find expert transcribers, due to these problems manual data labeling becomes an expensive thing. In this section, we will briefly go through some of these techniques which try to leverage the unpaired data and define the future work goals and current research problems.

5.1 Semi-supervised training

One of the successful approaches suggested by Jayadev Billa [6] is to try to leverage untranscribed out-of-domain speech for training a low-resource speech recognizer. When a

fraction of the training data is unlabeled, such approach is called semi-supervised training (SST).

To pre-train the model, [6] relies on existing ASR systems, which can be hard to get in real low-resource scenarios. In this case, the unlabeled data is transcribed using an available ASR system and then it is used for the pre-training. However, transcription errors may exist in the ASR output. To alleviate this, the predicted confidence level of the ASR system is used to filter out utterances for which the ASR system is not confident in its transcriptions. Basically, the confidence level will tell you how sure the ASR system is about the returned hypothesis, if the confidence is above a preset threshold then this data is considered good enough for including it to the training set. It showed that the model pretrained on the mix of unlabeled YouTube and paired target data always provided better results after the fine-tuning than the model pretrained only on YouTube data. The effectiveness of pre-training a model on the data pooled from the target and out-of-domain datasets was also shown in the transliteration experiment.

Another recently proposed approach suggested by Rodolfo Zevallos [47] leverages only the target domain text. The introduced approach is suitable primarily for agglutinative languages where single morphemes of words are easily separable because it focuses on a text augmentation technique called token-level delexicalization. In natural language processing delexicalization data augmentation is the substitution of some words which mean places, time, people e.t.c. with tags:

$$Hello,\ Peter,\ have\ you\ been\ to\ London\ yesterday?$$

$$Hello,\ <name>\ ,\ have\ you\ been\ to\ <place><time>?.$$

Afterward, these tags can be substituted with other words from the same semantic category. The delexicalization on the token level does it similarly, but it substitutes single morphemes instead of substituting the words. It would not make sense if it was about the fusion languages where morpheme separation is not that apparent. This technique can be used for the further improvement of the Tamil ASR model because Tamil has agglutinative property.

[47] follows a hybrid approach and uses *kaldi* toolkit for the training. He uses the synthesized text for training an external LM. For training the acoustic model he uses the available Text-To-Speech (TTS) converter. The synthesized audio is then fed to the feature extraction pipeline. Probably, this synthesized audio and text can be used for the e2e training as well. Using this method, the amount of training data can be significantly increased, however, it requires the third model for synthesizing text and speech, which can be a huge challenge in the case of real low-resource languages. This work shows how the data can be created almost from nothing, however, the fact that it relies on the TTS model is a huge disadvantage for real low-resource languages. Also, the resulting model may have difficulties recognizing the real speech, because it may be biased towards the synthesized speech. The TTS model may be limited in its ability to produce diverse speech, from different speakers, emotion states, environments, etc.. This may limit the diversity of audio data that the ASR model can learn from.

5.2 Incorporation of wav2vec

One more technology that is vastly used during the ASR transfer learning is wav2vec [40]. It is a model trained in an unsupervised manner on a large amount of raw audio data. The

main idea behind wav2vec is to extract embeddings from the audio, that contain useful information, by using variations of auto-encoder training. It is then hoped that a wav2vec model that is trained on large quantities of audio data from one domain may also be able to extract useful embedding features for a different low-resource domain. Thus, information from the audio of the source domain may be used to aid the target domain. The wav2vec process uses several convolutional layers which significantly decrease the dimensionality of input data and represent it in a latent space, afterward this data is randomly masked and modified, and the goal of the wav2vec model is to learn the difference between modified and original data. The task gets harder once the model becomes better at predicting further contexts.

Then the pretrained wav2vec model can be placed to any architecture as the encoder and fine-tuned with any criterion just by connecting it via the projection layer. During the fine-tuning on target labeled data wav2vec and the rest of the model are optimized with different learning rates. The paper which applied wav2vec to different low-resource scenarios [46] showed that using the pretrained wav2vec, it is possible to fine-tune the model on 10 hours of data which will be competitive with the model trained on 100 hours of data. In this paper, they applied the wav2vec pretraining on several low-resource datasets and got a significant gain in performance. This method, also, shows that it is resistant to data mismatch. The self-supervised pre-training showed a significant advantage over the training only on the target data; however, the amount of data needed for the self-supervised pretraining is several times more.

Now when the high-resource problems have the existing solutions which perform adequately, the focus of the researchers moves towards the problems with the low-resource data. Hopefully, this impossibility to acquire more and more data will push the research forward and novel solutions and training techniques will appear. The current trends of increasing the amount of training data may face a dead end in foreseeable future and we will need completely new designs and approaches.

Chapter 6

Conclusion

This thesis has explored different methods of overcoming the e2e low-resource problems. The augmentation increased the amount of trained data and improved the overall generalizability of the model. The pretraining on the transliterated data helped the model not to train from scratch but to adjust its weights to cover the target language space. Before the fine-tuning stage, we also tried to freeze and randomly initialize some components of the model which might have learned invalid linguistic information from the transliterated text.

Even though e2e architecture cannot be called a state-of-the-art solution for languages with limited resources yet, it is a very attractive approach because less linguistic expertise may be needed, compared to conventional models, thereby making e2e architectures more accessible. The capability of e2e models to perform online decoding on the device is another reason why big companies invest huge resources in this research field. Incorporating different techniques shown in this work can help to improve the resulting performance of the model, however, it is still not enough for calling the e2e architecture a reliable solution for low-resource languages.

A big problem with examined solutions is that all of them are data-driven. The hunger for data in the deep learning field is an underlying problem of deep learning. That is why it may not be the ideal solution for the problems with the limited amount of data. The current deep learning techniques look more like brute force learning rather than genuine comprehension of the training data. In comparison to a human, DNN will need hundreds or thousands of samples of the same data in order to remember it and generalize beyond it. The ability to generalize outside the training space is crucial in cases when there is a lack of training data but current deep learning approaches may need to be improved to effectively achieve this goal.

Bibliography

[1] Agarap, A. F. *Deep Learning using Rectified Linear Units (ReLU).* arXiv, 2018. DOI: 10.48550/ARXIV.1803.08375. Available at: https://arxiv.org/abs/1803.08375.

[2] Andrusenko, A., Laptev, A. and Medennikov, I. Towards a Competitive End-to-End Speech Recognition for CHiME-6 Dinner Party Transcription. In: *Interspeech 2020*. ISCA, Oct 2020. DOI: 10.21437/interspeech.2020-1074. Available at: https://doi.org/10.21437%2Finterspeech.2020-1074.

[3] Bahdanau, D., Cho, K. and Bengio, Y. *Neural Machine Translation by Jointly Learning to Align and Translate.* arXiv, 2014. DOI: 10.48550/ARXIV.1409.0473. Available at: https://arxiv.org/abs/1409.0473.

[4] Berglund, M. *Bidirectional Recurrent Neural Networks as Generative Models - Reconstructing Gaps in Time Series.* 2015.

[5] Besacier, L., Barnard, E., Karpov, A. and Schultz, T. Automatic speech recognition for under-resourced languages: A survey. *Speech Communication.* january 2014, vol. 56, p. 85–100. DOI: 10.1016/j.specom.2013.07.008.

[6] Billa, J. *Improving low-resource ASR performance with untranscribed out-of-domain data.* arXiv, 2021. DOI: 10.48550/ARXIV.2106.01227. Available at: https://arxiv.org/abs/2106.01227.

[7] Bourlard, H. and Morgan, N. *Connectionist Speech Recognition: A Hybrid Approach.* January 1994. ISBN 978-1-4613-6409-2.

[8] Chan, W., Jaitly, N., Le, Q. V. and Vinyals, O. *Listen, Attend and Spell.* arXiv, 2015. DOI: 10.48550/ARXIV.1508.01211. Available at: https://arxiv.org/abs/1508.01211.

[9] Chen, X., Meng, Z., Parthasarathy, S. and Li, J. Factorized Neural Transducer for Efficient Language Model Adaptation. In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2022, p. 8132–8136. DOI: 10.1109/ICASSP43922.2022.9746908.

[10] Cho, K. *On the Properties of Neural Machine Translation: Encoder–Decoder Approaches.* 2014.

[11] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K. and Bengio, Y. *Attention-Based Models for Speech Recognition.* arXiv, 2015. DOI: 10.48550/ARXIV.1506.07503. Available at: https://arxiv.org/abs/1506.07503.

[12] Fayek, H. M. *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between.* 2016. Available at: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.

[13] Gales, M. and Young, S. The Application of Hidden Markov Models in Speech Recognition. *Foundations and Trends in Signal Processing.* january 2007, vol. 1, p. 195–304. DOI: 10.1561/2000000004.

[14] Graves, A. *Sequence Transduction with Recurrent Neural Networks.* arXiv, 2012. DOI: 10.48550/ARXIV.1211.3711. Available at: https://arxiv.org/abs/1211.3711.

[15] Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. In:. January 2006, vol. 2006, p. 369–376. DOI: 10.1145/1143844.1143891.

[16] Graves, A., Mohamed, A.-r. and Hinton, G. *Speech Recognition with Deep Recurrent Neural Networks.* arXiv, 2013. DOI: 10.48550/ARXIV.1303.5778. Available at: https://arxiv.org/abs/1303.5778.

[17] Hannun, A. Sequence Modeling with CTC. *Distill.* 2017. DOI: 10.23915/distill.00008. https://distill.pub/2017/ctc.

[18] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r. et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine.* 2012, vol. 29, no. 6, p. 82–97. DOI: 10.1109/MSP.2012.2205597.

[19] Hochreiter, S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.* april 1998, vol. 6, p. 107–116. DOI: 10.1142/S0218488598000094.

[20] Hu, K., Sainath, T. N., Pang, R. and Prabhavalkar, R. Deliberation Model Based Two-Pass End-To-End Speech Recognition. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2020, p. 7799–7803. DOI: 10.1109/ICASSP40776.2020.9053606.

[21] Huang, J.-T., Li, J., Yu, D., Deng, L. and Gong, Y. Cross-language Knowledge Transfer using Multilingual Deep Neural Network with Shared Hidden Layers. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).* IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)th ed. May 2013. Available at: https://www.microsoft.com/en-us/research/publication/cross-language-knowledge-transfer-using-multilingual-deep-neural-network-with-shared-hidden-layers/.

[22] Huang, Y., Yu, D., Liu, C. and Gong, Y. A Comparative Analytic Study on the Gaussian Mixture and Context Dependent Deep Neural Network Hidden Markov Models. In: *Interspeech 2014.* Interspeech 2014th ed. September 2014. Available at: https://www.microsoft.com/en-us/research/publication/a-comparative-analytic-study-on-the-gaussian-mixture-and-context-dependent-deep-neural-network-hidden-markov-models/.

[23] Jaitly, N. and Hinton, G. E. *Vocal Tract Length Perturbation (VTLP) improves speech recognition.*

[24] Joshi, V., Zhao, R., Mehta, R. R., Kumar, K. and Li, J. *Transfer Learning Approaches for Streaming End-to-End Speech Recognition System.* arXiv, 2020. DOI: 10.48550/ARXIV.2008.05086. Available at: https://arxiv.org/abs/2008.05086.

[25] Khare, S., Mittal, A., Diwan, A., Sarawagi, S., Jyothi, P. et al. Low Resource ASR: The Surprising Effectiveness of High Resource Transliteration. In: *Proc. Interspeech 2021.* 2021, p. 1529–1533. DOI: 10.21437/Interspeech.2021-2062.

[26] Ko, T., Peddinti, V., Povey, D. and Khudanpur, S. Audio augmentation for speech recognition. In: *Proc. Interspeech 2015.* 2015, p. 3586–3589. DOI: 10.21437/Interspeech.2015-711.

[27] Li, J. Recent Advances in End-to-End Automatic Speech Recognition. Nov 2021. DOI: 10.48550/arXiv.2111.01690. Available at: https://doi.org/10.48550/arXiv.2111.01690.

[28] Mikolov, T., Kombrink, S., Deoras, A., Burget, L. and Cernocky, J. H. RNNLM - Recurrent Neural Network Language Modeling Toolkit. In: *IEEE Automatic Speech Recognition and Understanding Workshop.* IEEE Automatic Speech Recognition and Understanding Workshopth ed. December 2011. Available at: https://www.microsoft.com/en-us/research/publication/rnnlm-recurrent-neural-network-language-modeling-toolkit/.

[29] Mitra, V. and Franco, H. Time-frequency convolutional networks for robust speech recognition. In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU).* 2015, p. 317–323. DOI: 10.1109/ASRU.2015.7404811.

[30] Muda, L., Begam, M. and Elamvazuthi, I. Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. *CoRR.* 2010, abs/1003.4083. Available at: http://arxiv.org/abs/1003.4083.

[31] Panayotov, V., Chen, G., Povey, D. and Khudanpur, S. Librispeech: An ASR corpus based on public domain audio books. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2015, p. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964.

[32] Papadourakis, V., Müller, M., Liu, J., Mouchtaris, A. and Omologo, M. Phonetically Induced Subwords for End-to-End Speech Recognition. In: *Proc. Interspeech 2021.* 2021, p. 1992–1996. DOI: 10.21437/Interspeech.2021-1787.

[33] Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B. et al. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In: *Interspeech 2019.* ISCA, Sep 2019. DOI: 10.21437/interspeech.2019-2680. Available at: https://doi.org/10.21437%2Finterspeech.2019-2680.

[34] Pham, V. T., Xu, H., Khassanov, Y., Zeng, Z., Chng, E. S. et al. Independent Language Modeling Architecture for End-To-End ASR. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2020, p. 7059–7063. DOI: 10.1109/ICASSP40776.2020.9054116.

[35] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O. et al. 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.

[36] Prabhavalkar, R., Rao, K., Sainath, T., Li, B., Johnson, L. et al. A Comparison of Sequence-to-Sequence Models for Speech Recognition. In:. 2017. Available at: http://www.isca-speech.org/archive/Interspeech_2017/pdfs/0233.PDF.

[37] Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S. et al. *SpeechBrain: A General-Purpose Speech Toolkit*. 2021.

[38] Sainath, T. N., Pang, R., Rybach, D., He, Y., Prabhavalkar, R. et al. *Two-Pass End-to-End Speech Recognition*. arXiv, 2019. DOI: 10.48550/ARXIV.1908.10992. Available at: https://arxiv.org/abs/1908.10992.

[39] Sarma, K. and Sarma, M. Acoustic Modeling of Speech Signal using Artificial Neural Network: A Review of Techniques and Current Trends. In:. June 2015, p. 287–303. DOI: 10.4018/978-1-4666-8493-5.ch012. ISBN 9781466684935.

[40] Schneider, S., Baevski, A., Collobert, R. and Auli, M. *Wav2vec: Unsupervised Pre-training for Speech Recognition*. arXiv, 2019. DOI: 10.48550/ARXIV.1904.05862. Available at: https://arxiv.org/abs/1904.05862.

[41] Seltzer, M. L., Yu, D. and Wang, Y. An investigation of deep neural networks for noise robust speech recognition. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, p. 7398–7402. DOI: 10.1109/ICASSP.2013.6639100.

[42] Sennrich, R., Haddow, B. and Birch, A. 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)*. Berlin, Germany: Association for Computational Linguistics, August 2016, p. 1715–1725. DOI: 10.18653/v1/P16-1162. Available at: https://aclanthology.org/P16-1162.

[43] Singh, S., Gupta, A., Maghan, A., Gowda, D., Singh, S. et al. Comparative Study of Different Tokenization Strategies for Streaming End-to-End ASR. In: *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2021, p. 388–394. DOI: 10.1109/ASRU51503.2021.9687921.

[44] Tahir, M. *Discriminative training of linear transformations and mixture density splitting for speech recognition*. Dissertation.

[45] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention Is All You Need. *CoRR*. 2017, abs/1706.03762. Available at: http://arxiv.org/abs/1706.03762.

[46] Yi, C., Wang, J., Cheng, N., Zhou, S. and Xu, B. *Applying Wav2vec2.0 to Speech Recognition in Various Low-resource Languages*. arXiv, 2020. DOI: 10.48550/ARXIV.2012.12121. Available at: https://arxiv.org/abs/2012.12121.

[47] Zevallos, R. *Text-To-Speech Data Augmentation for Low Resource Speech Recognition*. arXiv, 2022. DOI: 10.48550/ARXIV.2204.00291. Available at: https://arxiv.org/abs/2204.00291.

[48] Zhou, Y., Xiong, C. and Socher, R. *Improved Regularization Techniques for End-to-End Speech Recognition*. arXiv, 2017. DOI: 10.48550/ARXIV.1712.07108. Available at: https://arxiv.org/abs/1712.07108.