

BSP S5 - Attention-Based Models for Speech Recognition

Wednesday 6th January, 2021

Le Minh Nguyen

University of Luxembourg

Email: le.nguyen.001@student.uni.lu

Vladimir Despotovic

University of Luxembourg

Email: vladimir.despotovic@uni.lu

Abstract—Deep Neural Networks (DNNs) improved many components of speech recognizers. They are commonly used in the DNN-Hidden Markov model (DNN-HMM) based speech recognition systems for acoustic modelling. Traditionally these components, such as acoustic, pronunciation and language models, have all been trained separately with different objectives. Recent work in this area tries to solve this disjoint training issue by creating models that are trained end-to-end, in other words, converting speech directly to transcripts. Unlike traditional DNN-HMM models, the attention-based model learns all the components of a speech recognizer jointly. This system has two components: a listener and a speller. The listener is a recurrent neural network encoder which processes filter bank spectra. The spelling component is an attention-based recurrent network decoder which outputs characters.

1. Introduction

This paper presents the Bachelor Semester Project (BSP) made by Le Minh Nguyen together with Vladimir Despotovic as his motivated tutor. The main objective of this project is to present the architecture of an attention-based speech recognition model. An additional objective is to reuse an existing implementation of the attention-based model, adapt it to the scope of this project and compare its accuracy to the results of other state-of-the-art speech recognition models. Furthermore, the implemented speech recognition model should be trained on the LibriSpeech dataset which contains a large-scale corpus of read English speech. Finally, its performance will be assessed with different sets of tuned hyperparameters.

2. Project description

2.1. Domains

- Speech Recognition
- Artificial Neural Networks
- Deep Learning
- Feature extraction
- Training, validation and testing set
- Python
- Pytorch
- HPC development environment

2.1.1. Scientific. The scientific aspects covered by this Bachelor Semester Project are the concepts of Speech Recognition and Deep Learning. The Listen, Attend and Spell (LAS) model's architecture will be investigated for the end-to-end Automatic speech recognition task.

Speech Recognition. The objective of speech recognition is to map audio signals which contain a set of spoken natural language expressions to the matching sequence of words produced by the speaker. In the past, Automatic Speech Recognition (ASR) was made up of different modules such as complex feature extraction, acoustic models, language and pronunciation models. [1] Sequential models, such as Hidden Markov Models (HMM), in combination with a pre-trained language model, were used to map sequences of phones to output words. [2] A different approach is to build ASR models end-to-end. With deep learning, it replaces most of the modules with a single module. This alternative method is the main task of this report, focusing on a sequence to sequence model with attention mechanisms to create an end-to-end ASR pipeline.

Artificial Neural Networks (ANN). ANNs are computing systems inspired by the biological brain. These systems are based on a set of connected units called artificial neurons. Each connection can transmit *signals* between units. A unit can process the signal and transmit it to another unit.

Deep Learning. This field deals with learning patterns by decomposing a task's input into smaller and simpler compositions. With Deep Learning, computing systems can build complex concepts from a composition of simpler concepts.

2.1.2. Technical. The technological aspects which are covered in this project are feature extraction, reuse an existing implementation of an attention-based sequence-to-sequence ASR model and benchmark its performance using different configurations and hyperparameters.

Feature extraction. Data preprocessing is an important phase in machine learning. It ensures the quality of the gathered data by eliminating irrelevant and redundant information. Data preprocessing contains tasks such

as cleaning, instance selection, normalization, feature extraction and feature selection. We will focus on feature extraction in this paper with the presentation of Mel filterbanks in Section 4.2.1 which are used as features extracted from a speech signal.

Training, validation and testing set. For a computing system to learn from and make predictions on data, a mathematical model is built from input data. This input data used to create the model consists of three datasets: The training, validation and testing set. The training set contains pairs of an input vector, which represent features and an output vector often called the labels. With the training set, the model learns to map the input vector to the labels. Further, the validation set serves for tuning the hyperparameters of the model. Whereas, the testing set evaluates how well the model generalizes the prediction over the dataset previously not seen by the model.

Python. This is a programming language which is interpreted, high-level and general-purpose. [3]

Pytorch. This library is a Python package which provides high-level features such as GPU accelerated Tensor computation and building Deep neural networks on a tape-based autograd system. It is designed for easy-to-use and efficient experimentation with Deep Learning through a user-friendly front-end. [4]

HPC environment. High-performance computing (HPC) is the capability to process and compute large amounts of data at a fast pace. Implementations of HPC are often referred to as *supercomputers* which consists of many connected compute servers called *nodes*. These nodes work in parallel to compute tasks faster. [5]

2.2. Targeted Deliverables

2.2.1. Scientific deliverables. The scientific aspect covered by this Bachelor Semester Project is the architecture of an attention-based sequence to sequence models. The focus of the scientific presentation will be on the architecture of this model and how it compares to other state-of-the-art models. Further, the model's performance will be assessed when using different hyperparameters. Another important deliverable is to present the extraction of Mel Filterbanks features from the dataset.

2.2.2. Technical deliverables. The technological aspect of this project is the reuse and adaptation of an existing implementation of an attention-based speech recognition model. Additionally, the LibriSpeech dataset will be preprocessed and used for the model's training. The goal is to deploy its training on an HPC environment for efficient computing.

2.3. Constraints

For this BSP, we set different constraints for the speech data set and the LAS model implementation.

Data set. This project does not focus on the collection of a data set. The focus lies on training our model on the LibriSpeech corpus which is an English data set containing approximately 1000 hours of read English speech. This data set is gathered from read audiobooks from the LibriVox project. [6]

LAS implementation. We do not implement the LAS architecture from the ground up since various implementation already exists. Therefore, the focus will lie on choosing a mature existing implementation, explaining its structure and how to use it efficiently for our benchmark.

3. Pre-requisites

To start on the project, certain skills in programming and mathematics are required. In particular, the preliminary requirements of the project is as follows:

- Introduction to deep learning-based speech recognition
- Knowledge of machine learning, data preprocessing and model training
- Understanding of vector and matrix algebra
- Introductory course in Python
- Software development

3.1. Scientific pre-requisites

Speech Recognition. A subfield combining computer science and computational linguistics which develops Automatic Speech Recognition (ASR). ASR is a methodology which enables recognizing and translating spoken language into text by machines. The objective of speech recognition is to map audio signals which contain a set of spoken natural language expressions to the matching sequence of words produced by the speaker. In the past, Automatic Speech Recognition (ASR) was made up of different modules such as complex feature extraction, acoustic models, language and pronunciation models. [1]. A different approach is to build ASR models end-to-end. With deep learning, it replaces most of the modules with a single module to create a single end-to-end ASR pipeline.

Machine learning, data preprocessing and model training. Machine Learning (ML) is the study of algorithms that gives software applications the ability to perform decisions and predictions without explicit programming. ML's basis is to create algorithms that can receive input data and learn by itself. The learning process improves its knowledge or performance through experience. ML is the idea of learning from data examples and deducing patterns. In other words, instead of having an explicit set of instruction in a program, we feed data into an algorithm and let the computer find patterns in the given data set.

Data preprocessing is the process of transforming raw data into more useful representations which make them

more suitable for ML models to better deduce patterns and increase their performance.

Machine learning contains the model training process. This modelling process maximizes the model's performance while mitigating overfitting. During this process, the model will be trained with a set of hyperparameters and these are tuned accordingly over the training phase. Finally, the best performing model will be selected through performance metrics.

Linear Algebra. A sub-field of mathematics, which works with vectors and matrices. Since the input of deep learning is data that are transformed into structures of rows and columns, linear algebra is one of the key foundations of deep learning. It is used to describe the operations of the deep learning algorithms, and implement the algorithms in code. All tasks in deep learning relate to linear algebra, from data preprocessing to the deep learning algorithms. [7]

3.2. Technical pre-requisites

Python is an interpreted, high-level and general-purpose programming language, which is conceived in the late 1980s and released in 1991 by Guido van Rossum. [8] Its design philosophy accentuates readability of the code. As many high-level programming languages, Python is dynamically typed. Further, it supports multiple programming paradigms such as procedural, object-oriented and functional programming.

Software development. Software development is the process of designing and implementation of applications and frameworks. In general, the process of software development includes writing and maintaining the source code which is often a planned and structured process. The software development contains mostly research, prototyping, modification and reuse of existing software. During the technical deliverable section, we use this structured process to analyse existing LAS implementation and adapt it to the scope of this project.

4. Scientific Deliverables

In this section, the defined scientific deliverables will be presented. The architecture of an attention-based sequence to sequence model will be investigated and compared to other state-of-the-art models. Further, the Mel Filterbanks features will be introduced and how to extract them from the data set. Finally, the model's performance trained on the Librispeech dataset will be assessed while using different hyperparameters.

4.1. Requirements

- **FR01** Explain the Mel-Filterbanks features.

- **FR02** Show the extraction of Mel filterbanks features from an audio signal.
- **FR03** Investigate the architectures of an attention-based sequence to sequence model and compare it to other state-of-the-art models.
- **NFR01** Performance evaluation and comparison. During this section, we present and discuss the results obtained from training our LAS model on the LibriSpeech corpus while using different hyperparameters.

4.2. Design

4.2.1. FR01: Feature extraction of Mel-Filterbanks.

Before training an end-to-end ASR model, audio data have to be preprocessed. The first task is to extract features from the data which describes natural language expressions and excludes background noises and emotions. Speech processing is an important process when creating ASR systems. For a long period, Mel-Frequency Cepstral Coefficients (MFCCs) were popular features. However, Mel-Filterbanks are recently getting more prominent. Computing both features contain the same approach where both of them calculate filterbanks and when adding a few more steps MFCCs are retrieved. MFCCs were preferred over filterbanks because some Machine Learning algorithms were more susceptible to highly correlated input found in filterbanks. With the rise of Deep Learning based speech systems, filterbanks are favored since deep Neural Networks are less susceptible to highly correlated input. [9]

To extract the Mel-Filterbanks from the dataset, the following extraction steps have to be executed on a speech signal sampled at $16kHz$:

1. Frame the signal into $25ms$ frames. The frame length of a $16kHz$ signal:

$$0.025 * 16000 = 400samples \quad (1)$$

With a frame step of $160 samples$ or $10ms$, the frames overlap themselves, such that the first frame containing $400 samples$ starts at sample 0 and the second frame starts at sample 160. This pattern repeats until the end of the speech signal. If the audio file is not divisible by an even number, it is padded by zeros.

2. For each frame calculate the periodogram estimate of the power spectrum. To obtain the Discrete Fourier Transform (DFT) from the frame i , we perform:

$$S_i(k) = \sum_{n=1}^{N-1} s_i(n)h(n)e^{-j2\pi kn/N} \quad (2)$$

For each speech frame $s_i(n)$, the periodogram-based power spectral estimate is calculated with:

$$P_i(k) = \frac{1}{N} | S_i(k) |^2 \quad (3)$$

This is called the Periodogram estimate of the power spectrum which identifies for every frame which frequencies are present.

3. Apply the Mel-filterbank to the power spectrum and sum the energy in each filter. The Mel-filterbank is a set of 40 triangular filters. Each filter within the filterbank is triangle shaped with a value of 1 at its frequency center and decreases to 0 reaching the center of the neighboring filter. This Filterbank on a Mel-Scale is shown in Fig. 1.

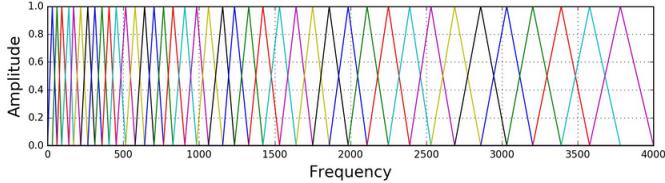


Figure 1: Filterbank on a Mel-Scale

The filterbank energies are calculated by multiplying every filterbank with the power spectrum of the signal. The resulting spectrogram can be seen in Fig. 2.

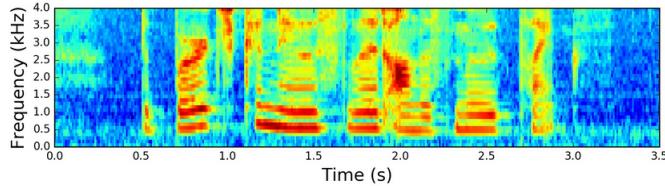


Figure 2: The spectrogram of the signal

To obtain the Mel-filterbank features, the mean of each coefficient is subtracted from all frames. The mean-normalized Mel-Filterbanks is show in Fig. 3.

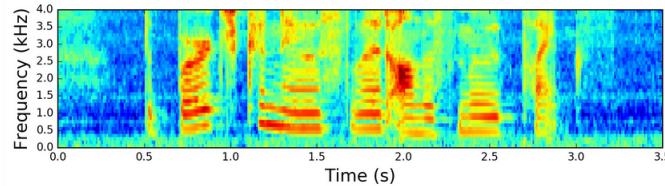


Figure 3: Normalized Mel-Filterbanks

In Section 4.3.1 we will use the *Torchaudio* module from the Pytorch Library to extract the Mel-Filterbank features from the dataset.

4.3. Production

4.3.1. Mel Filterbank feature extraction in Python.

We use the Pytorch module *Torchaudio* [10] to extract the Mel Filterbanks from the dataset. Instead of processing

the whole dataset, the following steps are applied to one single audio file f .

1. Let f be an audio file sampled at 16kHz , *Torchaudio* loads the song as follows:

```
1 import torchaudio
2 import torchaudio.compliance.kaldi.fbank as
   fbank
3
4 waveform, sr = torchaudio.load(filepath)
```

with *waveform* being the audio time series and *sr* the sample rate of *waveform*.

2. The Mel Filterbanks can be extracted with:

```
1 filterbank = fbank(waveform, num_mel_bins=40,
   channel=-1, sample_frequency=sample_rate)
```

The function *fbank()* returns a *torch.Tensor* with the dimension:

$$\text{shape} = (\text{padded_window_size}/2 + 1, m)$$

with *padded_window_size*/2 + 1 being the number of frames and *m* the number of Mel-filters.

3. The last step is to flatten this matrix to a vector which can be stored in a data frame for later training.

```
1 row = filterbank.transpose(0, 1).unsqueeze(0).
   detach()
```

The matrix *filterbank* is transposed and flattened. This concatenates the columns of *filterbank* together forming a vector.

4.4. Assessment

4.4.1. NFR01: Performance evaluation and comparison.

For this study, we use 460h of the roughly 1000h LibriSpeech audio corpus which contains read English speech. It contains 132553 examples of varying lengths, sampled at 16kHz and saved as FLAC formated file. This dataset represents 772 speakers reading the different sentences. The data is gathered from read audiobooks from the LibriVox project. Additionally, these readings were segmented and aligned into example sentences. [6]

5. Technical Deliverables

As technical deliverables, the four ANN models are implemented in Python using the Keras [?] library and a small dataset containing Luxembourgish spoken words is collected.

References

- [1] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. H. Engel, L. Fan, C. Fougner, T. Han, A. Y. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Y. Ng, S. Ozair,

- R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, “Deep speech 2: End-to-end speech recognition in english and mandarin,” *CoRR*, vol. abs/1512.02595, 2015. [Online]. Available: <http://arxiv.org/abs/1512.02595>
- [2] W. S. J. Cai, “End-to-end deep neural network for automatic speech recognition,” 2015. [Online]. Available: <https://cs224d.stanford.edu/reports/SongWilliam.pdf>
- [3] “Python software foundation, python language reference, version 3.7. available at,” <http://www.python.org/>.
- [4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [5] “What is high-performance computing?” <https://www.netapp.com/data-storage/high-performance-computing/what-is-hpc/>, accessed: 12/12/20.
- [6] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5206–5210.
- [7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [8] “General python faq,” <https://docs.python.org/3/faq/general.html#what-is-python>, accessed 19/05/19.
- [9] H. M. Fayek, “Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what’s in-between,” 2016. [Online]. Available: <https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html>
- [10] “torchaudio: an audio library for pytorch,” <https://github.com/pytorch/audio>, accessed: 06/01/21.