BSP S5 - Attention-Based Models for Speech Recognition

Saturday 12th December, 2020

Le Minh Nguyen
University of Luxembourg
Email: le.nguyen.001@student.uni.lu

Vladimir Despotovic
University of Luxembourg
Email: vladimir.despotovic@uni.lu

***Abstract*—Not finale version. 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, its performance will be assessed with different sets of tuned hyperparameters. Finally, the implemented speech recognition model should be trained on the LibriSpeech dataset which contains a large-scale corpus of read English speech.

2. Project description

2.1. Domains

- Speech Recognition
- Artificial Neural Networks
- Deep Learning
- Feature extraction
- Data preprocessing
- Training, validation and testing set
- Python
- Pytorch
- HPC developement 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.

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 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 aspect which is covered in this project is feature extraction and implementation of the end-to-end spoken word recognition model.

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-frequency cepstral coefficients (MFCCs) in Section 4.2.1 which are used as features extracted from the 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 two 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. On the other hand, 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 throught a user-friendly front-end. [4]

HPC environment. Hig-performance computing (HPC) is the capability to process and compute large amounts of data at fast pace. Implementations of HPC are often referred to *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 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. One of the main deliverables is to present how to extract the features from the dataset with Filter Banks.

2.2.2. Technical deliverables. The technological aspect of this project is the reuse and adaptation of an existing implementation of the 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 the different constraints for the speech data set and the LAS model implementation.

Data set. This project does not focus on collection of a data set. we focus on training our model on a dataset containing English spoken numbers. [6]

ANNs implementation. We do not implement the Artificial Neural Networks architecture since existing deep learning libraries such as Keras are already matured. Thereby, we focus on explaining how the APIs work and how to use them efficiently in our model.

3. Pre-requisites

tart 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
- Introductory course in Python
- Understanding of vector and matrix algebra

3.1. Scientific pre-requisites

Speech recognition.

Machine learning, data preprocessing and model training.

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. With a proficient background in programming in Python gained during my high school and university education, I am prepared to work on this project.

4. Scientific Deliverables

The scientific deliverables include; the feature extraction with MFCCs, notions of deep learning and Artificial Neural Networks (ANNs). This section concludes by a performance evaluation of four different ANNs models.

4.1. Requirements

- **FR01** Present the feature extraction of Filter banks.

4.2. Design

4.2.1. FR01: Feature extraction of Filter Banks.

Before training 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.

To extract the Filter banks 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:

$$0.025 * 16000 = 400 samples. \tag{1}$$

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:

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

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

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

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 26 triangular filters. The filterbank energies are calculated by multiplying every filterbank with the power spectrum.

4.3. Production

4.3.1. Feature extraction with MFCC in Python.

For the pilot study, we use an audio dataset of English spoken words [6]. It contains 105829 utterances of 35 words. This dataset contains 2618 persons pronouncing these different words. Every utterance is roughly $1s$ long, sampled at $16kHz$ and saved as a WAVE format file. From this dataset we choose the words $w \in \{'zero', 'one', 'two', \ldots, 'eight', 'nine'\}$.

We use the Python library *librosa* [9] to extract the MFCCs 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 $16kHz$, *Librosa* loads the song as follows:

```
import librosa

y, sr = librosa.load(f, sr=16000, mono=True,
    duration=1)
```

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

2. The next step is to pad the time series y if it is smaller than $1s$:

```
import numpy as np

if y.size < 16000:
  rest = 16000 - y.size
  left = rest // 2
  right = rest - left
  y = np.pad(y, (left, right), 'reflect'))
```

The time series is reflected on both sides to get the correct size.

3. After matching the time series to the correct size, the MFCCs can be extracted with:

```
mfccs = librosa.feature.mfcc(y=y, sr=sr)
```

The function *.mfcc()* returns an *np.ndarray* with the dimension:

$$shape = (n\_mfcc, t)$$

with n_mfcc being the number of MFCCs and t the number of frames.

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

```
row = mfccs.T.flatten()
```

The matrix $mfccs$ is transposed and flattened. This concatenates the columns together forming a vector.

These steps are then applied to the entire dataset. Each computed vector is appended to the data frame which is going to be stored as a $.csv$ file. The entire code snippet is given in the Appendix in Figure **??**.

4.4. Assessment

4.4.1. NFR01: Performance evaluation.

In this section, the performance of the four ANNs investigated in Section **??** are evaluated. We use the English and Luxembourgish dataset for the performance analysis.

The English and Luxembourgish dataset contains spoken words $w \in \{'0', \ldots, '9'\}$. The English dataset consists of 17749 data instances whereas the Luxembourgish consists of 500 data instances.

The first step is to choose an optimal optimizer. The choice is between using the stochastic gradient descent (SGD) or Adam optimizer. The latter is a combination of two extensions of SGD, Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) [10]. The four models will be trained on the English dataset to designate the right optimizer and number of epochs. Their training will be visualized with 60 $epochs$ each. The training visualization is given in the appendix as follows:

1. Feedforward Neural Network:
   - Validation accuracy in Figure **??**.
   - Validation loss in Figure **??**.
2. RNN:
   - Validation accuracy in Figure **??**.
   - Validation loss in Figure **??**.
3. LSTM:
   - Validation accuracy in Figure **??**.
   - Validation loss in Figure **??**.
4. GRU:
   - Validation accuracy in Figure **??**.
   - Validation loss in Figure **??**.

Let $n = 60$ be the number of epochs. The two optimizers return the following testing accuracies (acc) and losses:

Optimizer	FF		RNN		LSTM		GRU	
	acc	loss	acc	loss	acc	loss	acc	loss
SGD	0.79	0.65	0.43	1.51	0.86	0.46	0.85	0.46
Adam	0.87	0.47	0.72	0.97	0.94	0.24	0.94	0.24

TABLE 1: Comparison of the four models' testing results using SGD and Adam. The models were trained on the English dataset.

Table 1 shows that the validation accuracies using the Adam optimizer yield better accuracies compared to SGD. Therefore, for the following investigation, the Adam optimizer is considered.

The next step is to choose the number of epochs which will be used for the performance analysis. The training visualizations show the loss converges around 40 epochs except for the RNN model. Figure **??** shows that the RNN model converges around 220 epochs with an accuracy of 0.88. The training visualization of the RNN model with 300 epochs is given in Figures **??** and **??** in the appendix.

Choosing $epochs = 40$ to evaluate each model's performance, we have:

Optimizer	FF		RNN		LSTM		GRU	
	acc	loss	acc	loss	acc	loss	acc	loss
Adam	0.83	0.54	0.63	1.10	0.937	0.23	0.94	0.23

TABLE 2: The testing results of the four models using Adam. The four models were trained on the English dataset. FF is the abbreviation for feedforward and acc stands for accuracy.

Inspecting the results shows that LSTM and GRU perform better than feedforward and RNN. The overall performance is slightly underneath the performance at 60 epochs. In this case, RNN has the lowest accuracy. This is expected, since standard RNNs can't deal with learning long-term temporal dependencies, because the gradient of the loss function decays exponentially with time (vanishing gradient problem). LSTMs and GRUs use gated units to maintain the information in memory for longer periods of time, and to control which information is kept, and which is forgotten. This explains substantially better performance obtained using LSTM and GRU. Compared to feedforward neural networks, LSTM and GRU have better accuracy since they take advantage of their recurrent units to process series data.

Training on Luxembourgish dataset. The same models used to train on the English dataset were used to train on the Luxembourgish dataset. The following testing accuracies and losses were obtained while training on the Luxembourgish dataset:

Optimizer	FF		RNN		LSTM		GRU	
	acc	loss	acc	loss	acc	loss	acc	loss
Adam	0.84	0.59	0.73	0.83	1.0	0.02	0.99	0.02

TABLE 3: The testing results of the four models using Adam. The four models were trained on the Luxembourgish dataset. The dataset contains 50 instances of each spoken word. FF is the abbreviation for feedforward and acc stands for accuracy.

The results show that LSTM and GRU generalize better than feedforward and RNN. The validation accuracies of LSTM and GRU are both roughly 99%. The high accuracy of LSTM and GRU can be explained by the small dataset which is also speaker-dependent. The results of LSTM and GRU are expected to drop if the dataset grows and contains a variety of speakers.

The training visualization on the Luxembourgish dataset is given in the appendix as follows:

1. Feedforward Neural Network:
   - Validation accuracy in Figure **??**.
   - Validation loss in Figure **??**.
2. RNN:
   - Validation accuracy in Figure **??**.
   - Validation loss in Figure **??**.
3. LSTM:
   - Validation accuracy in Figure **??**.
   - Validation loss in Figure **??**.
4. GRU:
   - Validation accuracy in Figure **??**.
   - Validation loss in Figure **??**.

5. Conclusion

Acknowledgment

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 automaticspeech 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] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *CoRR*, vol. abs/1804.03209, 2018. [Online]. Available: http://arxiv.org/abs/1804.03209

[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] [Online]. Available: http://doi.org/10.5281/zenodo.3478579

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980