Discovering Pseudo Phonemes for Language Modeling from Raw Speech

by

Lingyun Gao

student Number: 4927672

Supervisor: Dr. Odette Scharenborg

Co-supervisor: Dr. Siyuan Feng

DISCOVERING PSEUDO PHONEMES FOR LANGUAGE MODELING FROM RAW SPEECH

By
Lingyun GAO

To be defended publicly on October 25th at 13:30

in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering
Specialization: Signal & Systems
Master of Science in Computer Science
Specialization: Data Science

at Delft University of Technology,
Faculty of Electrical Engineering Mathematics and Computer Science

This Thesis Project is supervised by

Supervisor: Dr. Odette Scharenborg
Co-supervisor: Dr. Siyuan Feng

Thesis Committee:

Dr. Odette Scharenborg (Chair)
Organisation: Delft University of Technology
Faculty: Electrical Engineering, Mathematics and Computer Science
Section: Multimedia Computing

Dr. ir. J. Dauwels
Organisation: Delft University of Technology
Faculty: Electrical Engineering, Mathematics and Computer Science
Section: Circuits and Systems

Dr. Jie Yang
Organisation: Delft University of Technology
Faculty: Electrical Engineering, Mathematics and Computer Science
Section: Web Information Systems

Dr. G. Joseph
Organisation: Delft University of Technology
Faculty: Electrical Engineering, Mathematics and Computer Science
Section: Circuits and Systems

Dr. Siyuan Feng
Organisation: Delft University of Technology
Faculty: Electrical Engineering, Mathematics and Computer Science
Section: Multimedia Computing

CONTENTS

PREFACE

When I started my thesis, I was introduced to the ZeroSpeech Challenge series, which aims at an end-to-end Spoken Dialog system that could be trained in an unknown language with only raw sensory data. The challenge series is held to accumulate open-source techniques progressively to achieve the ultimate goal. The idea of the challenge is very fantastic and I felt it would be interesting to be a possible contributor to this progress. So I joined in and decided my research topic was discovering pseudo-phonemes for language modeling from raw speech. In this work, we built our pipeline method for language modeling from raw speech and have a special focus on developing a novel pseudo-phoneme encoder that helps to discover phoneme-like units from raw speech.

During my thesis year with strict physical distancing, there are hard times but also many good memories to support me. Here in this opportunity, I would like to first express my gratitude to my supervisor Dr. Odette Scharenborg and my daily supervisor Dr. Siyuan Feng for their continual support, feedback, encouragement, and understanding of my thesis work. I would like to also thank their efforts in holding online SALT meetings, where I learned a lot from others and also felt participated in the campus in this difficult year. Secondly, I would like to express my appreciation to my parents and friends who would listen to me and guide me whenever I might need them. Finally, I would like to express my thankfulness to Dr. Richard Hendricks, Dr. Jie Yang, Dr. G. Joseph, and Dr.J. Dauwels for being interested in my thesis and accepting my invitation as my committee member.

Lingyun Gao
Delft, October 2021

THESIS OUTLINE

This thesis report is structured as follows. Chapter 1 provides an introduction to the task of discovering pseudo-phonemes for unsupervised language modeling from raw speech. Research questions are listed in section 1.1. An overview of our novel approach and main contributions are given in the section 1.2.

Chapter 2 introduces the background of ZeroSpeech 2021 Challenge, which proposed the task of unsupervised language modeling for raw speech. Chapter 3 presents supplemental material that provides the deep learning background of our proposed approach.

Our system can be separated into three components, self-supervised speech representation learning, unsupervised segment representation learning, and language modeling. Accordingly, three subsections in chapter 4 give overviews of the related background for these three components and details of the methods we apply in our systems.

Chapter 5 describes the experiment design with regards to our research questions. Then, experimental settings, datasets, evaluation metrics and experimental results follow. Finally, discussion of the limitations of the work, conclusions and potential future works are presented in Chapter 6.

This thesis project is in partial fulfilment of the requirements for the degree of both Master of Science in Electrical Engineering and Master of Science in Computer Science. There are shared 31 credits for both programs and 14 credits independent for each program. In terms of the thesis report, chapters 1,2,3,5,6 belong to the shard part of both programs. Section 4.1 and 4.3 belong to the independent part for Master of Science in Computer Science and section 4.2 belongs to the independent part for Master of Science in Electrical Engineering.

1

INTRODUCTION

Young children can perceive words from raw speech and produce simple and coherent sentences, without being trained on text-annotated speech. Generally, infants first learn to produce single-syllable or repeated single-syllables, such as "no", "mama", and then they develop complex vocabularies and the ability to make sentences [1]. Intuitively, they may achieve this by encoding input speech into their phonetic units (pseudo-phonemes) [2] and use them to form their unique language models (forming vocabulary, syntactic and semantic rules). In the meantime, mimicking such ability on machines and developing a spoken language system without text supervision is of strong research interest, as it could resolve the conflict between the massive text requirement of traditional language models and the shortage of reliable textual resources for most languages in the world [3]. If we could imitate how children build the zero-text-supervision spoken language models on machines, we could provide better language services (translation, etc) for users of languages lacking textual resources (also called low resource languages).

The Zero Resource speech challenge 2021 (ZeroSpeech 2021) is constructed to build such zero-text spoken language models [4]. The challenge offers an open-source pipeline method (baseline) to build a spoken language model without text supervision. Similar to how children build spoken language systems without text, this method tries to encoding raw input speech into discrete units and use them to train language models. The pipeline method consists of three components, a frame-level speech representation learning component, a K-means50 clustering module, and a language model (LM), as shown in figure 1.1. The first two components encode speech into discrete units for training the language models. Specifically, the frame-level speech representation learning component applies the contrastive predictive coding (CPC) [5] model to encode raw audio into frame-level speech representation. The clustering component applies the k-means algorithm to group previous speech representations into fifty phonetic discrete units, while 50 was tested to perform best at the acoustic level. For each audio, a discrete unit sequence is generated by the above two components and is called pseudo-text for this audio piece. The Language model, implemented with LSTM (low GPU budget scenarios) [6] or BERT [7] (both low and high GPU budget scenarios) architecture, is trained

1

Figure 1.1: System Pipeline of ZeroSpeech 2021 Baseline

on these pseudo-text.

The challenge requires evaluating speech understanding of the zero-text language modeling system at four linguistic levels, using four corresponding discrimination tasks. Specifically, they are (1) how well the spoken language modeling system could distinguish different phonemes (acoustic level), (2) how well the system could differentiate existing words from man-made non-words (lexical level), (3) how well the system could distinguish grammatical sentences from non-grammatical ones (syntactic level), and (4) how well the system could recognize the semantic similarity between a pair of words (semantic level).

In this project, our research objective is to obtain a general understanding of advanced research in spoken language modeling from raw speech and apply the learned knowledge and techniques to developing our spoken language models from raw speech. In the next few paragraphs, we will talk about three potential improvements for the ZerosSpeech 2021 baseline model (low-budget scenario) which are encouraging discrete units to be phoneme-like units, reducing speaker-information in speech representations, and replacing CPC with other transformer-based architectures. Our zero-text spoken language modeling system is constructed by modifying the baseline system (low-budget scenario) and these potential improvements. In addition, according to results in [8], the linguistic level performance mainly improves with using the larger or complex LMs. As we do not modify language models but follow almost the same setting in the ZeroSpeech 2021 baseline, we only focus on the performance of acoustic, lexical, and syntactic levels in our project.

The length of a phoneme, defined as the smallest unit of speech distinguishing different words in linguistics, usually varies from 5ms to hundreds of mileseconds [9]. However, in the baseline system, every 10ms speech frame is assigned a discrete unit to form the pseudo-text of speech audios. It is not surprising that we saw many adjacent discrete units are the same. This means words in natural text only differ from each other in linguistic units, but words in pseudo-text differ from each other in both linguistic units and time of pronunciation, and the same words in pseudo-text could have different forms. For example, in the baseline system, the audio of the word "mama" might be converted to pseudo-text "12222133" if the speaker pronounces the vowel longer or converted to "12213" in the shorter scenario. Here we use numerals to represent different discrete units. This suggests that the language model in the baseline has to learn to eliminate the impact of pronunciation time for the same words. Besides, in our experiments, we saw the clustering module in the baseline did not promise to generate dis-

crete units that cover all phonemes. According to [4], the Bert language models trained with forced aligned phonetic transcription to give a nearly full performance on the lexical level, and much better syntactic level performance than the Bert model trained with pseudo-text from the baseline. We then assume that if we could make the discrete units closer to true phonemes, our language modeling system may achieve higher performance on the above three linguistic levels. From this assumption, we tried to build a pipeline method to encode speech into discrete units which mimic phonemes as much as possible. Because we try to force our system to produce phoneme-like units, we use the word 'pseudo-phoneme' to represent the clustered discrete units. The components to generate pseudo-phonemes in our system are together called the pseudo-phoneme encoder.

The frame-level Kmeans clustering module in the baseline may not be optimal in generating pseudo-phonemes. In an early-stage exploration of the baseline method, we found in the pseudo-text, it is common to see several nearby frames were assigned to several different units, while those frames should belong to the same phoneme according to the golden transcriptions. This might be because frame-level clustering in the baseline results in over-fragmented clustering [10, 11], which means the frame-level clustering module tends to give redundant fine-grained clusters. Besides, we saw that those fine-grained clusters could not cover all phonemes especially consonants. An alternative solution is segment-level clustering: first, we obtain phoneme boundaries and cut speech data into variable-length speech segments, then we perform clustering on those segments. Research in [12] showed that, while both frame-level and segment-level clustering use the Kmeans clustering algorithm with the same number of clusters, segment-level clustering can achieve better performance in phone recognition evaluation than frame-level clustering. This suggests that after the segment-level clustering, the discrete units are more likely to represent a true phoneme than that from frame-level clustering. This might be because, after phoneme boundary-guided segment representation learning, a segment representation could contain the more accurate and complete information of a single phoneme than frame representations. This fits the need of us to discover phoneme-like units. In our project, we, therefore, try to improve the baseline model by replacing the frame-level clustering method with a segment clustering method. This method contains three steps: (1) Boundary Learning, which is used to identify phoneme boundaries in speech frames (2) Segment Representation Learning for combining speech frame representations between two boundaries into segment representations (3) Kmeans Clustering to cluster those segments into several discrete units, following[12].

Our pseudo-phoneme encoder starts with a frame-level representation learning module applying the CPC technique. If we could find better representation learning modules than CPC, with regards to the spoken language modeling performance, we might be able to construct a better pseudo-phoneme encoder. We thus proposed two potential replacements for CPC in the next two paragraphs.

Considering our goal is spoken language modeling, we expect that the speech representations would encode speech content and eliminate other information as much as possible. However, experiment results in [13] show that CPC representations not only encode linguistic information but also preserve most of the speaker information. On other

hand, speaker information remained in speech representations has been seen to cause performance deterioration on downstream acoustic and lexical discrimination tasks [14, 15]. Thus another improvement we explore for the baseline is to reduce speaker information in frame-level speech representations. Besides, according to the submission results from Niekerk [8], applying the speaker normalization method after the speech representation learning method in the ZeroSpeech 2021 baseline system could improve performance on all four evaluation scores. The results suggest applying a post step speaker-invariant technique after the frame-level speech representation learning is possible to give better language modeling performance. We prefer an independent post-speaker-invariant technique rather than modify the representation learning method because training a post method could save much more time. Prevalent traditional speaker invariant techniques include feature-space maximum likelihood linear regression (fMLLR), disentangled speech representation learning, and speaker adversarial training. The fMLLR estimation based methods relies on the out-of-domain acoustic speech recognition (ASR) model, which means text transcription is used during training. Thus fMLLR does not fully meet the requirement of zero-text in the challenge. Besides, [16] shows that speaker-invariant techniques implemented in the front-end (fMLLR and disentangled approach) might perform better than techniques implemented in the back-end (speaker adversarial training) for speech representations. In addition, among the three methods, the disentangled approach is the easiest to implement. In our project, based on the above observations, we apply a disentangled technique (FHVAE) [17] in our model, trying to reduce speaker information in the frame-level representations.

The second potential replacement for CPC is using self-supervised transformer-based representation learning methods. Before ZeroSpeech 2021, this challenge series have concentrated mainly on discovering discrete linguistic units (subword and word units) from raw speech data [2, 18–20]. Meanwhile, recent research applying self-supervised representation learning methods, which generate supervisory resources from unlabelled data, shows great improvements in downstream speech recognition performance (speech to text recognition tasks evaluated by word error rate) [21] under the low-resource scenario. Here low-resource scenario means only a few textual resources are used during fine-tuning the self-supervised method. This suggests that those self-supervised methods may learn their own language models without text resources. This indicates it is possible to construct language models without text supervision. ZeroSpeech Challenge 2021 then takes one step forward from previous iterations and aims at learning a language model directly from audio. ZeroSpeech 2021 baseline system chose CPC for self-supervised representation learning, which uses LSTM to obtain contextual information in the representations. However, state of art research of self-supervised speech recognition models in low resource scenarios mainly uses transformer architecture to generate contextualized speech representations [22–26]. Considering the assumption that we expect self-supervised methods to learn their own language models, transformer-based architectures are potentially superior over LSTM based architectures, as most state-of-art supervised language models mainly apply transformer architectures. Therefore, we think it is worth exploring transformer-based self-supervised representation learning methods to replace CPC in the baseline system. As in the low resource scenario, wav2vec 2.0 shows the best speech recognition performance [21] which suggests a strong ability

to learn its own language models, in our project, we try to apply wav2vec 2.0 to replace CPC in the baseline.

Finally, in our zero-text spoken language system, the pseudo-phoneme encoder is trained on a large raw English speech dataset. However, practically, English is actually a high-resource language with a large amount of speech data and textual resources. It would be interesting to test our system on a true low resource language such as Mboshi [27], which do not have their own writing system and there are only a small number of speech data available, to investigate its transferability on other languages. In addition, we do not perform language modeling on Mboshi, as above mentioned evaluation on language modeling needs dataset and professional annotations for a certain language. So we did not consider this in our project.

1.1. RESEARCH QUESTIONS

In our project, we try to build a language modeling system from a raw speech by replacing the discrete encoder in the ZeroSpeech 2021 baseline with a designed pseudo-phoneme encoder. Out system will be evaluated on phonetic, lexical, and syntactic levels using metrics from ZeroSpeech 2021 [4]. Besides, we would like to investigate two potential replacements (wav2vec2 and a speaker-invariant CPC) and see if these two could bring better language modeling performance compared to CPC. Moreover, we would like to test if our pseudo-phoneme encoder is transferable across different languages and we plan to use Mboshi for testing.

This leads to the following research questions:

- Could we use a pseudo-phoneme encoder to improve the language modeling performance?
- Will applying a transformer-based self-supervised speech representation method (wav2vec 2.0) in the ZeroSpeech 2021 baseline system brings better language modeling performance?
- Will applying a post speaker invariant method (FHVAE) to the CPC representation-learning method improve the performance at the acoustic level?
- Transferability: Is our pseudo-phoneme encoding approach transferable to other true zero-resource languages (tested by Mboshi)?

1.2. OVERVIEW OF THE PROPOSED PIPELINE APPROACH

In this project, we developed a novel pipeline method to build a language modeling system from raw speech. As shown in figure 1.2, it consists of a pseudo-phoneme encoder for discovering pseudo-phonemes and a language model. Specifically, in the pseudo-phoneme encoder, the frame-level speech representation module encodes the raw audio into frame-level representations. The boundary learning module produces the boundary labels from raw audio waves. The segment learning module utilizes the frame representations and the boundary labels to produce the segment representations and the clustering module groups those segment representations into pseudo-phonemes and

generates pseudo-phoneme sequences. Finally, the pseudo-phoneme sequences are used to train the language model.

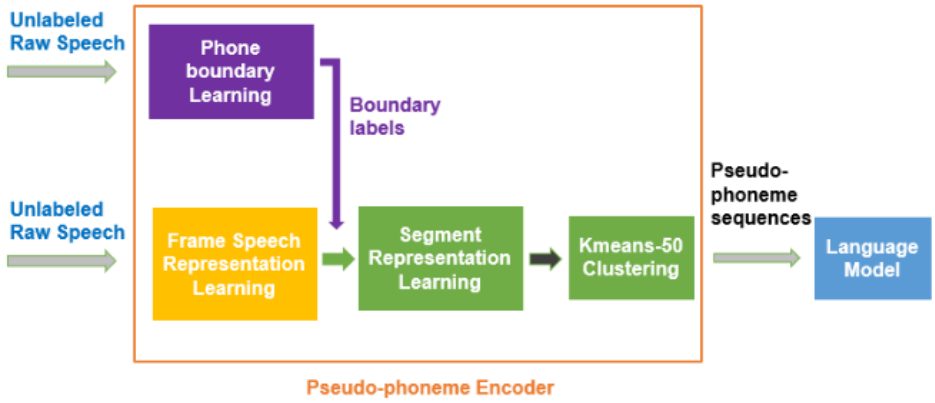

Figure 1.2: System Pipeline of Our Pipeline Approach

Figure 1.3: Transformation process from audio to pseudo-phoneme in the pseudo-phoneme encoder

An example of the Transformation process from audio to pseudo-phoneme in our pseudo-phoneme encoder is shown in figure 1.3. A signal was first cut into numbers of 10ms speech frames and encoded to be frame representations (yellow vectors). Then boundary labels (purple bracket) indicate the start and end of a segment. Those frames that belong to the same segment were encoded into segment representations (green vectors). Then in the clustering module, segments are assigned to different pseudo-phonemes.

REFERENCES

[1] K. R. Fahey, L. M. Hulit, and M. R. Howard, “Born to talk,” 2019.

[2] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic,

C. Dugrain, L. Ondel, A. W. Black *et al.*, "The zero resource speech challenge 2019: Tts without t," *arXiv preprint arXiv:1904.11469*, 2019.

[3] M. A. Hedderich, L. Lange, H. Adel, J. Strötgen, and D. Klakow, "A survey on recent approaches for natural language processing in low-resource scenarios," *arXiv preprint arXiv:2010.12309*, 2020.

[4] T. A. Nguyen, M. de Seyssel, P. Rozé, M. Rivière, E. Kharitonov, A. Baevski, E. Dunbar, and E. Dupoux, "The zero resource speech benchmark 2021: Metrics and baselines for unsupervised spoken language modeling," *arXiv preprint arXiv:2011.11588*, 2020.

[5] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] "Result page for ZeroSpeech 2021," https://zerospeech.com/2021/results.html, 2021, [Online; accessed September-2021].

[9] H. Kuwabara, "Acoustic properties of phonemes in continuous speech for different speaking rate," in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, vol. 4. IEEE, 1996, pp. 2435–2438.

[10] B. Wu, S. Sakti, J. Zhang, and S. Nakamura, "Optimizing dpgmm clustering in zero resource setting based on functional load." in *SLTU*, 2018, pp. 1–5.

[11] S. Feng and T. Lee, "Exploiting cross-lingual speaker and phonetic diversity for unsupervised subword modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2000–2011, 2019.

[12] S. Feng, P. Żelasko, L. Moro-Velázquez, and O. Scharenborg, "Unsupervised acoustic unit discovery by leveraging a language-independent subword discriminative feature representation," *arXiv preprint arXiv:2104.00994*, 2021.

[13] L. van Staden and H. Kamper, "A comparison of self-supervised speech representations as input features for unsupervised acoustic word embeddings," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 927–934.

[14] ——, "Improving unsupervised acoustic word embeddings using speaker and gender information," in *2020 International SAUPEC/RobMech/PRASA Conference*. IEEE, 2020, pp. 1–6.

[15] Y. Miao, H. Zhang, and F. Metze, "Speaker adaptive training of deep neural network acoustic models using i-vectors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 11, pp. 1938–1949, 2015.

[16] S. Feng, "Exploiting cross-lingual knowledge in unsupervised acoustic modeling for low-resource languages," *arXiv preprint arXiv:2007.15074*, 2020.

[17] W.-N. Hsu and J. Glass, "Scalable factorized hierarchical variational autoencoder training," *arXiv preprint arXiv:1804.03201*, 2018.

[18] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, "The zero resource speech challenge 2015: Proposed approaches and results," *Procedia Computer Science*, vol. 81, pp. 67–72, 2016.

[19] E. Dunbar, X. N. Cao, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera, and E. Dupoux, "The zero resource speech challenge 2017," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 323–330.

[20] E. Dunbar, J. Karadayi, M. Bernard, X.-N. Cao, R. Algayres, L. Ondel, L. Besacier, S. Sakti, and E. Dupoux, "The zero resource speech challenge 2020: Discovering discrete subword and word units," *arXiv preprint arXiv:2010.05967*, 2020.

[21] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[22] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[23] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," *arXiv preprint arXiv:1911.03912*, 2019.

[24] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.

[25] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. v. d. Oord, "Learning robust and multilingual speech representations," *arXiv preprint arXiv:2001.11128*, 2020.

[26] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3497–3501.

[27] A. Rialland, M. Adda-Decker, G.-N. Kouarata, G. Adda, L. Besacier, L. Lamel, E. Gauthier, P. Godard, and J. Cooper-Leavitt, "Parallel corpora in mboshi (bantu c25, congo-brazzaville)," in *11th edition of the Language Resources and Evaluation Conference (LREC 2018)*, 2018.

2

BACKGROUND

The task of language modeling from the raw speech is proposed in the Zerospeech Challenge 2021, which belongs to a series of Zero Resource Speech Challenges [1–4]. The eventual target of this series of challenges is building an end-to-end Spoken Dialog (SD) system with only sensory data of unknown languages. This ultimate SD system could help information retrieval, automatic annotation for low-resource languages, and speech recognition in abnormal situations. Towards this goal, this series of challenges propose achievable but progressively harder tasks and make all essential techniques of the target SD system open to public use.

In the ZeroSpeech challenge 2015 and 2017, tasks focused on developing distinguishable speech representations to encode phonetic units/word units, which are robust across different languages and speakers [2, 3]. The ZeroSpeech challenge 2019-2020 aims to improve the quality of encoding phonetic information of phonetic units and requires that those phonetic units should be able to build a speech synthesizer (text-to-speech without text)[1, 4].

In the ZeroSpeech challenge 2021 [5], the organizers wish to push one step forward from previous iterations by aiming at building a language model from raw speech audio without any text supervision. While training the spoken language modeling system from raw speech, the challenge requires that spoken dataset, LibriSpeech[6] and (optional) Libri-Light [7]) are the only sources that can be used. The organizers have provided a pipeline solution [5], consisting of a representation learning method, a clustering module, and a language model. This pipeline is the foundation on which we build our system. The challenge gives baselines of two budget scenarios. The difference between the two scenarios is the size of the language model applied in the pipeline, while the low resource scenario uses a language model of a 60-hour GPU budget and the high resource scenario uses a 1536-hour LM.

To evaluate language models, perplexity is the most popular evaluation metric. However, this metric varies with the granularity of the training data for language models. In our context, this metric varies with the granularity of the discrete units sequences (intervals of each discrete unit) fed to the language models. In the challenge, the interval

is allowed to be self-defined by participants of the Challenge. Thus it cannot be used in this challenge for model comparison. To resolve this problem, the challenge uses a zero-shot strategy: to evaluate the speech understanding of the system at a certain linguistic level, the language modeling system is given a simple discrimination task on a man-made speech dataset. This enables direct human-machine comparison at a certain linguistic level. To give more details, the challenge requires that the language models should be able to understand the speech from a natural language at four different linguistic levels: acoustic, lexical, syntactic, and semantic. The following four discrimination tasks and corresponding datasets in the ZeroSpeech Challenge 2021 to evaluate speech understanding of the language models at the four linguistic levels :

- Acoustic Level(Libri-Light dataset [7]), where the system has to judge whether two input audios, while each contain a triphone such as 'apa' each and these two only differ in the middle phoneme), have the same phoneme in the middle or not.
- Lexical Level(sWUGGY dataset) - where the system needs to judge if the word in the input audio is an existed word or a man-made non-word.
- Syntactic Level(sBLIMP dataset) - where the system needs to judge whether a sentence in a speech utterance aligns with the grammatical rules of the training language or not
- Semantic Level(sSIMI dataset) - the system has to assess the semantic similarity between a pair of words, and judge if the similarity score is proportional to scores made by experts

REFERENCES

[1] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black *et al.*, "The zero resource speech challenge 2019: Tts without t," *arXiv preprint arXiv:1904.11469*, 2019.

[2] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, "The zero resource speech challenge 2015: Proposed approaches and results," *Procedia Computer Science*, vol. 81, pp. 67–72, 2016.

[3] E. Dunbar, X. N. Cao, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera, and E. Dupoux, "The zero resource speech challenge 2017," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 323–330.

[4] E. Dunbar, J. Karadayi, M. Bernard, X.-N. Cao, R. Algayres, L. Ondel, L. Besacier, S. Sakti, and E. Dupoux, "The zero resource speech challenge 2020: Discovering discrete subword and word units," *arXiv preprint arXiv:2010.05967*, 2020.

[5] T. A. Nguyen, M. de Seyssel, P. Rozé, M. Rivière, E. Kharitonov, A. Baevski, E. Dunbar, and E. Dupoux, "The zero resource speech benchmark 2021: Metrics and baselines for unsupervised spoken language modeling," *arXiv preprint arXiv:2011.11588*, 2020.

[6] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[7] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.-E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen *et al.*, "Libri-light: A benchmark for asr with limited or no supervision," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7669–7673.

3

DEEP LEARNING BASICS

In our project, we investigate one segment-level and two frame-level deep representation learning methods. There is no doubt that we have a special focus on speech representation learning in a deep learning scenario. These three methods are usually applied in acoustic speech recognition (ASR) tasks. Thus in this chapter, we will first review the deep learning background in speech representation learning of acoustic speech recognition (ASR) tasks.

The performance of machine learning (ML) tasks depends on how to represent speech data [1]. Before the deep learning era, Mel-frequency cepstral coefficients (MFCCs), perceptual linear predictive coefficients (PLPs) [2], and hand-crafted versions of the previous two, obtained by data pre-processing pipelines, are mainstream speech representation techniques for speech technologies [3]. Gaussian Mixture Model and Hidden-Markov-Model models (GMM-HMM) with those features are popular models for analyzing speech. In 1990 [4], researchers found that in phoneme recognition, applying neural nets (NN) can get better recognition performance than traditional GMM-HMM models. This suggests NN is the potential to construct better ASR models than GMM. Besides, humans have few degrees of freedom to produce speech. Thus only a limited range of acoustic signals can be produced and speech data is reckoned as lying on a low dimensional nonlinear manifold [5], embedded in a high dimensional data space. Yet, HMM is not efficient to model data distributed in a nonlinear manifold as a lot of parameters and expensive computation are usually needed. Feature learning models like neural nets, which are the potential to model speech in a less costly way[6, 7], are thus explored as alternatives to HMM. In 2012 [7], DNNs achieves better performance (lower word error rate) on ASR tasks than traditional GMM-HMM models. The better performance is attributed to the ability of DNNs to learn better representations from input speech data. Since then, varied deep learning architectures, including convolutional neural networks (CNN), recurrent neural networks (RNN) especially long-short term memory (LSTM) and gated recurrent units (GRUs), transformers, and a combination of the previous architectures are explored in developing speech representation learning methods.

The speech representation learning models can be trained in different ways, such

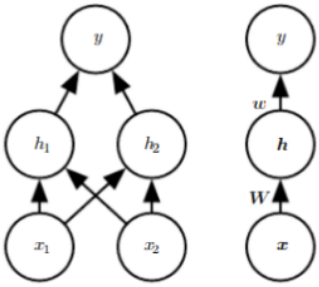

Figure 3.1: Feed Forward NN, from [1]

as supervised, unsupervised, and self-supervised. In supervised learning, DNNs are trained by datasets with annotations. This manner usually leads to representations with more accurate performance in downstream tasks. While in unsupervised learning, annotations are not required and this manner is popular for low-resource languages in which text annotations are lacking. Self-supervised learning [8] in a way converts the unsupervised scenario into supervised ones by using part of the training data as labels for other training data. As self-supervised learning does not need annotations, it is usually recognized as a special type of unsupervised learning. Recent state-of-art ASR models as well as the models we investigated are constructed by applying self-supervised representation learning techniques [9, 10].

In the next several sections, we will introduce the basic deep learning concepts and basic architectures used in constructing the three speech representation methods we investigated in our system. Those sections are mostly referring to [1] and [11].

3.1. FEEDFORWARD NEURAL NETWORKS

Feedforward neural networks or multilayer perceptrons (FNN or MLP) are basic deep learning models and form the basis of other complex architectures. A feedforward neural network defines a mapping or approximates a function from the output to the input vector. The "feedforward" indicates that the data flow from the input goes straight to the output and there is no backward data flow. Feedforward neural networks usually contain multiple layers and can be represented by a compound function.

As an example shown in figure 3.1, these feedforward neural networks have an input layer (first layer) with input vector x, a hidden layer (second layer) with weight parameter vector W and intermediate output h, and an output layer (Third layer) with weight parameter vector w and output y. The left subimage shows the node view of the networks and the right shows the vector view. The mapping $f(x;W,w)$ of the feedforward neural networks can be represented as

$$
\begin{aligned}
f(\boldsymbol{x};\boldsymbol{W},\boldsymbol{w}) &= f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right) \\
h &= f^{(1)}(\boldsymbol{x}) = \boldsymbol{W}^{\top}\boldsymbol{x} \\
f^{(2)}(\boldsymbol{h}) &= \boldsymbol{h}^{\top}\boldsymbol{w} \\
y &= f(\boldsymbol{x}) = \boldsymbol{x}^{\top}\boldsymbol{W}\boldsymbol{w}
\end{aligned}
\tag{3.1}
$$

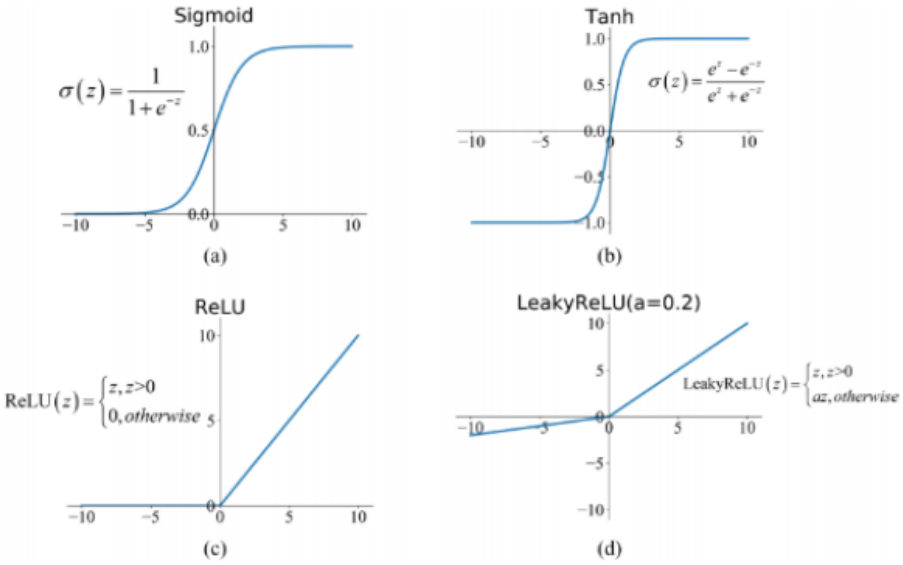

Figure 3.2: Commonly used activation functions, from[39]

3.2. NONLINEARITY

Speech data is proposed to be lying on nonlinear manifolds. Yet as stated in the last section, naturally, the feedforward neural networks can only approximate linear functions for data modeling. Differentiable activation functions are then used to introduce nonlinearity to feedforward neural networks. The rectified linear activation function (ReLU function), the Leaky-ReLU function, the sigmoid function, and the tanh function are commonly used activation functions.

As indicated by its name, in practice, those functions determine if a node in this layer could be activated or not. Coming back to the example in figure 3.2, if we introduce a ReLU function in the hidden layer, the mapping of the feedforward neural networks will be represented as

$$f(\boldsymbol{x};\boldsymbol{W},\boldsymbol{w}) = \boldsymbol{w}^{\top}\max\left\{0,\boldsymbol{W}^{\top}\boldsymbol{x}\right\}$$

in which if a hidden node is smaller than zero, it will not be calculated/activated by the networks.

3.3. OPTIMIZATION

To train a neural network, a loss/objective function and an optimization algorithm (optimizer) to minimize the loss function are needed. During training, the loss function measures how close the practical output of the network is to the desired output under current weights. Usually, loss functions are designed for certain tasks. Common loss functions include Mean Squared Error (MSE) for regression problems and Cross-Entropy for classification problems. In general, Cross-Entropy and modified versions are commonly used in ASR tasks. Because of the nonlinearity caused by activation functions, many loss functions applied in neural networks become non-convex. Thus, iterative and gradient-based optimizers such as stochastic gradient descent are used in training deep neural networks.

3.3.1. STOCHASTIC GRADIENT DESCENT

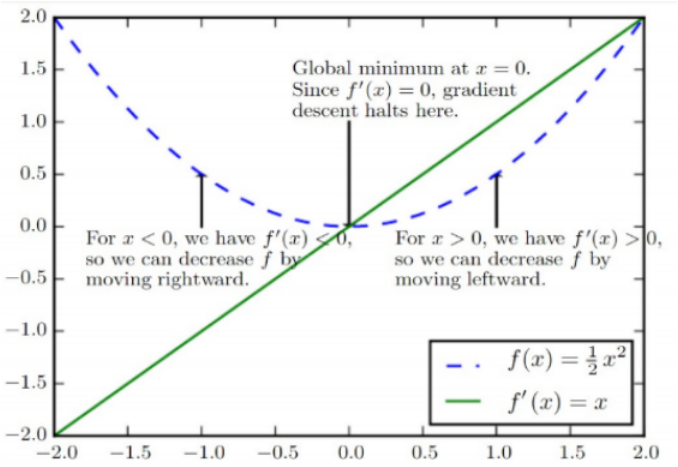

Figure 3.3: Example of Gradient Descent from [1]

Assume we have the loss function $f(x)$, the goal is to find the $x*$ that we could obtain the minima of $f(x)$. If we have a small movement according to Taylor expansion we have

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + \mathcal{O}\left(\epsilon^2\right) \tag{3.2}$$

For a convex problem, such as the example shown in figure 3.3, moving along the direction of the negative gradient could decrease the value $f(x)$. We could also prove it mathematically. If we choose a small fix step size, also called learning rate $\eta > 0$, and $\epsilon = -\eta f'(x)$ we have

$$f\left(x - \eta f'(x)\right) = f(x) - \eta f'^2(x) + \mathcal{O}\left(\eta^2 f'^2(x)\right). \tag{3.3}$$

if $f'(x)$ is not zero and η is small enough so that high order terms are approximately zero then $f(x - \eta f'(x))$ will be smaller than $f'(x)$ or we could say $f'(x)$ will decrease.

In practice, we have loss functions with the form of $f(x;\theta)$, where x is the input vector and θ is the parameter vector. Our goal is to find the optimal $\theta*$ that gives the minima of the loss function. The optimization procedure is

- Calculate the gradient $\nabla_\theta f(x,\theta)$
- Updating the parameters $\theta' = \theta - \eta\nabla_\theta f(x,\theta)$
- Repeat untill conditions of θ acomplished

When the amount of data is too large, this calculation is not realistic, an alternative is to randomly select a batch of m' data samples and use them to update the weights. The updating step is then adapted to $\theta' = \theta - \eta\frac{1}{m'}\sum_{i=1}^{m'}\nabla_\theta f\left(x^{(i)},\theta\right)$

3.3.2. ADAM

Because of the existence of large amounts of local minima in realistic non-convex problems, researchers developed many additional algorithms [12]. Mini-batch stochastic gradient descent can be more efficient by collecting larger sets of observations in one mini-batch [13]. The momentum method can accelerate convergence by aggregating history gradients [14]. RMSprop and Adagrad [15, 16] proposed per-coordinate scaling-related optimization algorithms. Adam [17] applies all the above advanced modules and is proved to be a robust and effective optimization method. Now Adam is a default optimization method for training neural networks.

3.4. CONVOLUTIONAL NEURAL NETWORK

Convolutional neural networks (CNN) [18] are originally proposed for image recognition problems with their unique advantage of capturing spatial structures of images, assuming nearby pixels are correlated with each other. CNN is also superior in enormous reduction of the number of parameters compared to fully connected layers. This architecture and assumption can also be applied to speech. In [18], CNN-based systems were presented theoretically to obtain speech features that are robust during short temporal shifting, then the HMM with CNN hybrid models were successfully applied and showed better performance in speech recognition tasks than HMM-DNN [19]. Pure CNN models with pre-training such as convolutional RBM are also proved to achieve the comparative results with hybrid models [20]. Now many state-of-art systems in speech recognition will also use CNN structure as part of the model [9, 21]. As an example shown in figure 3.4, CNN typically contains convolutional layers with several feature maps, pooling layers, and other fully connected layers. Activation functions such as ReLU are used if there are multiple convolution layers. In this example [20], the input of CNN is three-channel visualized audio filter banks, while each feature map in the convolution layer learns from a limited frequency range of the input filter banks. The pooling layer subsamples the feature maps. Other fully connected layers flatten the representation and are used for classification.

3.4.1. CONVOLUTION LAYER

The convolution layer is the core of CNN. In this layer, as a 2-D example shown in figure 3.5, the input feature vector I and the kernel K also called a filter or feature detector

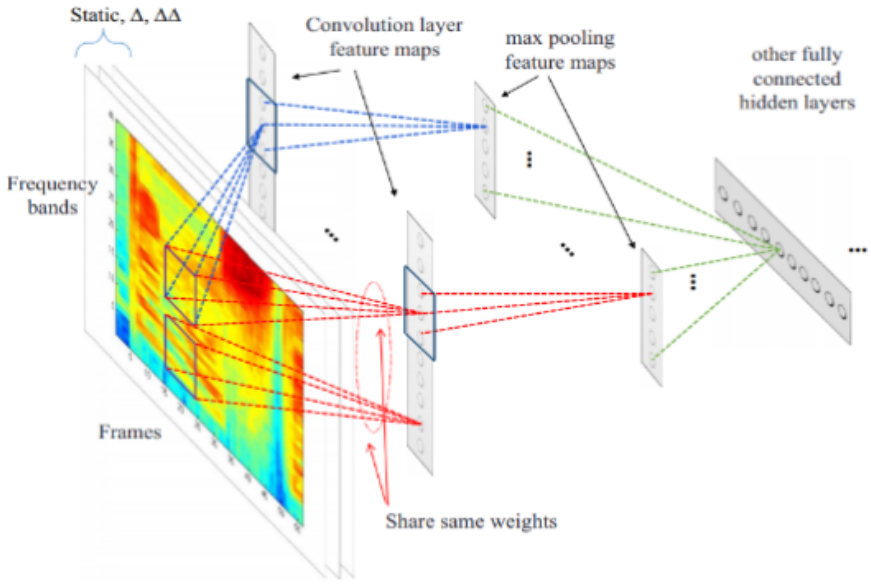

Figure 3.4: Convolutional neural networks in speech recognition, from [20]

are processed with a cross-correlation operator to generate the output S. The mark * represents the cross-correlation operator. The computing between these three feature vectors can be shown as

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{3.4}$$

Convolution Layers have some conventional terminologies. For a certain kernel, the output of a convolutional layer is also called a feature map. Each node/neuron stores a feature map. The receptive field for an element x in the convolutional layer means the number of all previously used input space elements that join in the calculation of x. Stride for the convolutional layer means how many elements in the input we move when the kernel starts the next calculation. In practice, there might be multiple input feature vectors or the input is a tensor In this case, the number of input feature vectors is usually called channels.

3.4.2. POOLING LAYER

The pooling layer, or usually known as the subsampling layer in speech processing, reduces the size of the feature maps over a region. Pooling can be realized by maximum or average functions. The pooling layer could help to control the usage of memory.

3.5. RECURRENT NEURAL NETWORKS

While training DNNs and CNNs, we assume that data samples are independent and identically distributed (i.i.d.) [11]. In speech recognition tasks, however, this assump-

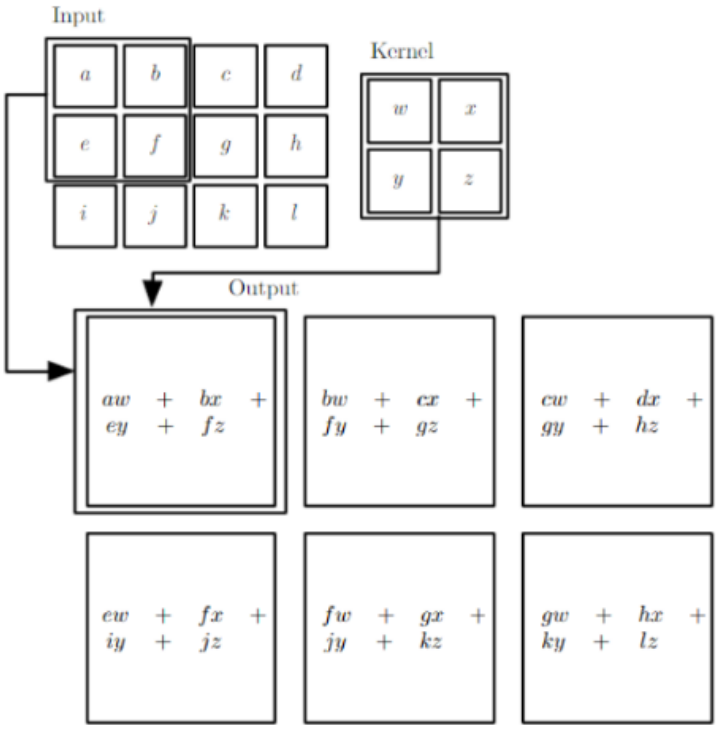

Figure 3.5: Two-dimensional cross-correlation for the input and the kernel to generate the output. The first windowed output value calculated by the windowed input and the kernel, from [1]

tion is usually not true, since speech data is in ordered sequences. Random permutation of such data will cause information change. For example, if we randomly permute the audio signal in a conversation, then some natural language information in such data might be lost. Besides, in speech-to-text tasks, the input size might be variable-length and we expect a corresponding variable-length sequenced output.

To deal with such sequence modeling tasks, recurrent neural networks (RNN) were proposed [22]. The idea behind RNN is to share parameters in different parts of the model which guaranteed to extend or apply the model to data of different lengths or more general, different forms. This idea could be applied in convolution layers across 1-D temporal sequences, but compared to RNN it is rather short in temporal modeling, which means the output is usually related to several neighbor inputs, while RNN theoretically allows its output to be related with a long-range of previous input sequences.

To explain the RNN, here in this section we used computational graphs. We first introduce a classical recurrent process as shown in figure 3.6. The process can be represented as the following formula. It is recurrent because the definition of state s at time t is the same as the definition of the previous state s−1 at time t1. They share the same parameters θ

Figure 3.6: A computational graph for a classical recurrent process, from [1]

$$s^{(t)} = f\left(s^{(t-1)};\theta\right) \tag{3.5}$$

To adapt to the realistic problem, where we expect the current state to be influenced by both the previous states and the current input, we introduce the following model shown in figure 3.7.

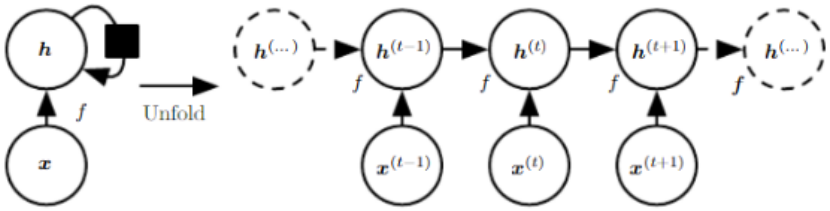

Figure 3.7: The left is a simplified/circuit diagram for a recurrent network without outputs. The right is the unfolded version of the left, from [1]

In this case the input x at time step t is given to the state h at time step t. In RNN, those states are called hidden states because they are not observed. The formula becomes

$$\begin{aligned} h^{(t)} &= g^{(t)}\left(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \ldots, x^{(2)}, x^{(1)}\right) \\ &= f\left(h^{(t-1)}, x^{(t)};\theta\right) \end{aligned} \tag{3.6}$$

It can be seen that the current hidden states h at time t is related to all previous input. In practice, RNN will add a layer of output o, utilizing hidden states for prediction as an example shown in figure 3.7.

According to the computational graph of the RNN in figure 3.8, we could see some insights into the difficulties of updating parameters U, W, V. On one hand, the parameters can only be updated when a total sequence is completely processed. Besides, for long-hidden state chains, the derivative can easily become zero or infinite, which is often known as the gradient vanishing or explosion problem [11].

The standard RNN introduced in this section has limitations in long-range relationship learning. Specifically, for long-range relationship learning, the gradient explosion or vanishing problem might be caused in the following three situations.

- A. Some early observations in the sequences are very important which makes their gradient too large and then impacts the latter gradient.

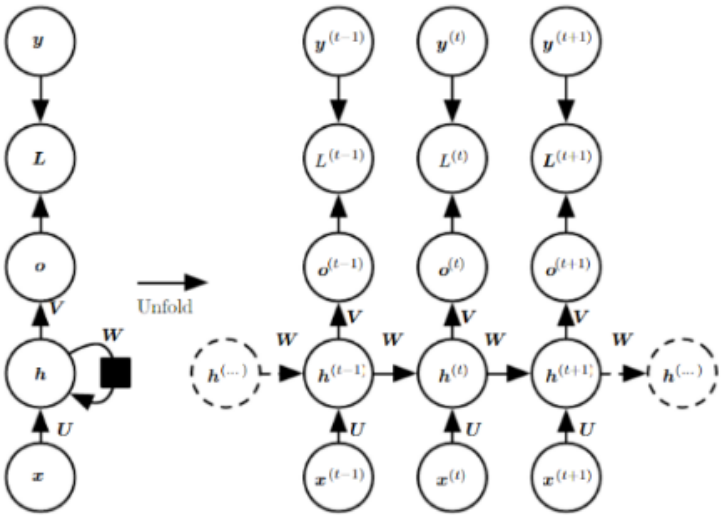

Figure 3.8: An example of RNN model, from [1]

- B. There will be some uninformative input in the middle of sequences.
- C The sequences might consist of several different logical chunks.

Correspondingly, to resolve the gradient explosion or vanishing problem, we would like to add some techniques that could store some early important information, skip uninformative input and reset the internal state while it is necessary.

3.5.1. GATED RECURRENT UNITS

To improve the long-range relationship learning of RNN models, Gated Recurrent Units were proposed [23, 24].

GRU achieved the above functions by adding gating mechanisms. As shown in figure 3.9, given a previous hidden state and current input, the reset gate determines how much we preserve the previous hidden state Ht-1 in the candidate hidden state Hat Ht. The update gate determines in the final calculation of the current hidden state Ht, what the proportion is the previous state and the current candidate state. Specifically, this process can be represented as

$$\begin{aligned}
\mathbf{R}_t &= \sigma\left(\mathbf{X}_t\mathbf{W}_{xr} + \mathbf{H}_{t-1}\mathbf{W}_{hr} + \mathbf{b}_r\right) \\
\mathbf{Z}_t &= \sigma\left(\mathbf{X}_t\mathbf{W}_{xz} + \mathbf{H}_{t-1}\mathbf{W}_{hz} + \mathbf{b}_z\right) \\
\tilde{\mathbf{H}}_t &= \tanh\left(\mathbf{X}_t\mathbf{W}_{xh} + \left(\mathbf{R}_t \odot \mathbf{H}_{t-1}\right)\mathbf{W}_{hh} + \mathbf{b}_h\right) \\
\mathbf{H}_{t-1} &= \mathbf{H}_{t-1} \odot \mathbf{Z}_t + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t
\end{aligned} \tag{3.7}$$

Where W_{xr}, W_{hr}, b_r are weights and bias for reset gate, a W_{xz}, W_{hz}, b_z are weights and bias for update gate, W_{xh}, W_{hh}, b_h are for hidden forward process. The activation func-

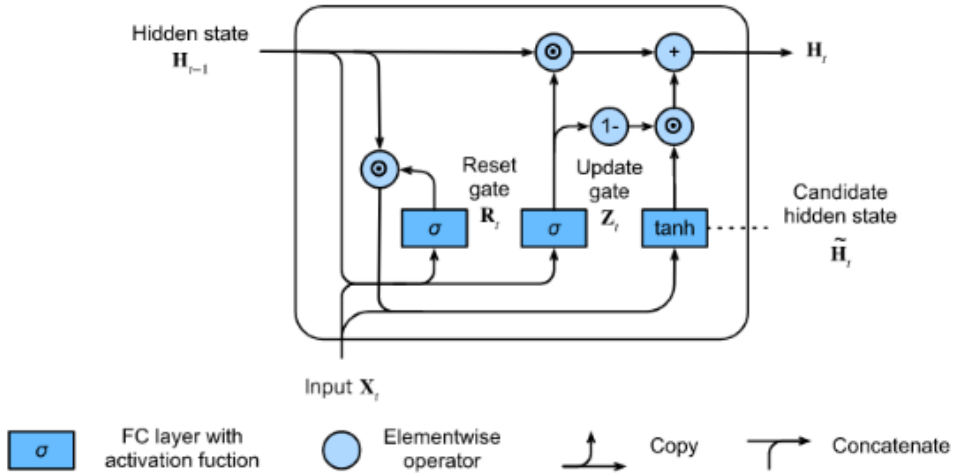

Figure 3.9: An illustration for Gated Recurrent Units, from [11]

tion makes the values of reset gate vector falls in interval [0,1], the $tanh$ function promises the candidate vector falls in interval [-1,1].

When there are some important early states, the reset gate can promise in candidate hidden states, the new input will occupy a smaller proportion. If uninformative input occurs, the update gate could just discord the new input information. When different logical chunks occur, reset gate could help to increase the input proportion and the update gate could help reduce the proportion of the old state.

3.5.2. LONG SHORT-TERM MEMORY

Similar to GRU, as shown in figure 3.10, Long Short-Term Memory(LSTM) [24] is also proposed to resolve gradient vanishing and explosion in learning of long-range temporal relationships but with a more complex design. Comparable with functions of reset and update gates in GRU, three gates, including input gate It, output gate Ot and forget gate Ft, are implemented in LSTM. Their calculations are shown as following

$$\begin{aligned}\mathbf{I}_t &= \sigma\left(\mathbf{X}_t\mathbf{W}_{xi} + \mathbf{H}_{t-1}\mathbf{W}_{hi} + \mathbf{b}_i\right)\\ \mathbf{F}_t &= \sigma\left(\mathbf{X}_t\mathbf{W}_{xf} + \mathbf{H}_{t-1}\mathbf{W}_{hf} + \mathbf{b}_f\right),\\ \mathbf{O}_t &= \sigma\left(\mathbf{X}_t\mathbf{W}_{xo} + \mathbf{H}_{t-1}\mathbf{W}_{ho} + \mathbf{b}_o\right)\end{aligned} \tag{3.8}$$

where $\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{xo} \in \mathbb{R}^{d\times h}$ and $\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{ho} \in \mathbb{R}^{h\times h}$ are weight parameters for input, forget, output gate respectively and $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o \in \mathbb{R}^{1\times h}$ are bias parameters.

A gated memory cell C_t is introduced into LSTM to determine how much to add new information. This cell is in shape the same as hidden states, thus it is sometimes regarded as a special form of hidden states. Similar to candidate hidden state in GRU, In LSTM, a candidate memory cell $\hat{C}_t$ is computed by

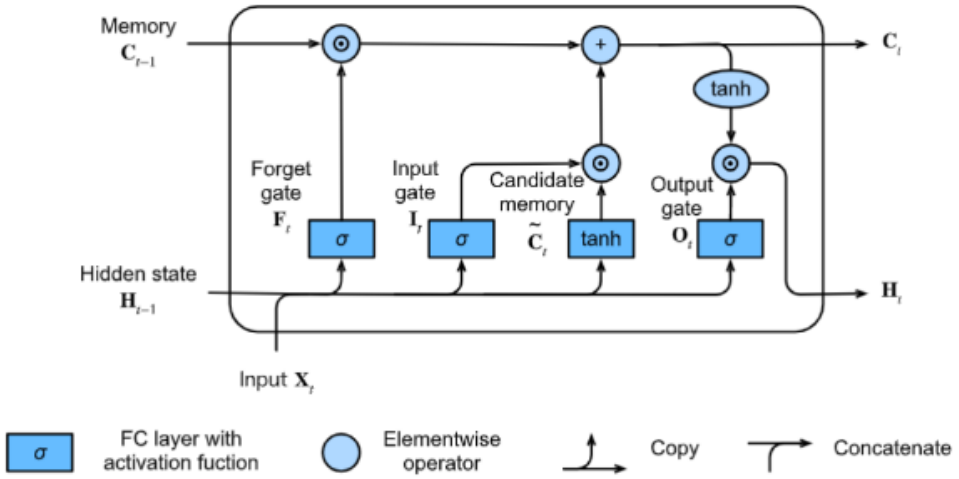

Figure 3.10: An illustration for Long Short-Term Memory, from [11]

The information in the current state of the memory cell C_t is determined by the input gate, which controls the amount of additional new information stored in the candidate hidden states, and the forget gate, which decides the remaining proportion of the old memory cells. Then the final calculation of the current hidden state Ht is defined by the output gate dot product with the current memory cell activated by tanh function. To be specific, this can be shown as

$$\begin{aligned} \mathbf{C}_t &= \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t \\ \mathbf{H}_t &= \mathbf{O}_t \odot \tanh(\mathbf{C}_t) \end{aligned} \tag{3.9}$$

3.6. TRANSFORMER

In this section, we would like to introduce the transformer, a replacement for sequential RNN models that struggle from costly training due to long-time dependency. The transformer is a sequence-to-sequence (S2S) architecture that only employs self-attention blocks [25] for sequential learning. It was originally proposed for natural language tasks but is now pervasively applied in all kinds of deep learning tasks. The next few sections will explain the basic operation of transformers: self-attention operation and multi-variable version: multi-head self-attention. Then we introduce how to leverage position information through positional coding techniques. Finally, we introduce a common encoder-decoder structure to build a transformer.

3.6.1. SELF-ATTENTION

The core technique behind transformers is the self-attention mechanism and is inspired by the attention mechanism of humans. Biologically, attention is a limited resource. For example, at each second, humans receive massive sensory input from eyes, ears, and other sensory organs, but humans usually focus on a very small piece of sensory data

such as an image. This important technique is learned in evolution because not all input data are equally important for surviving [11].

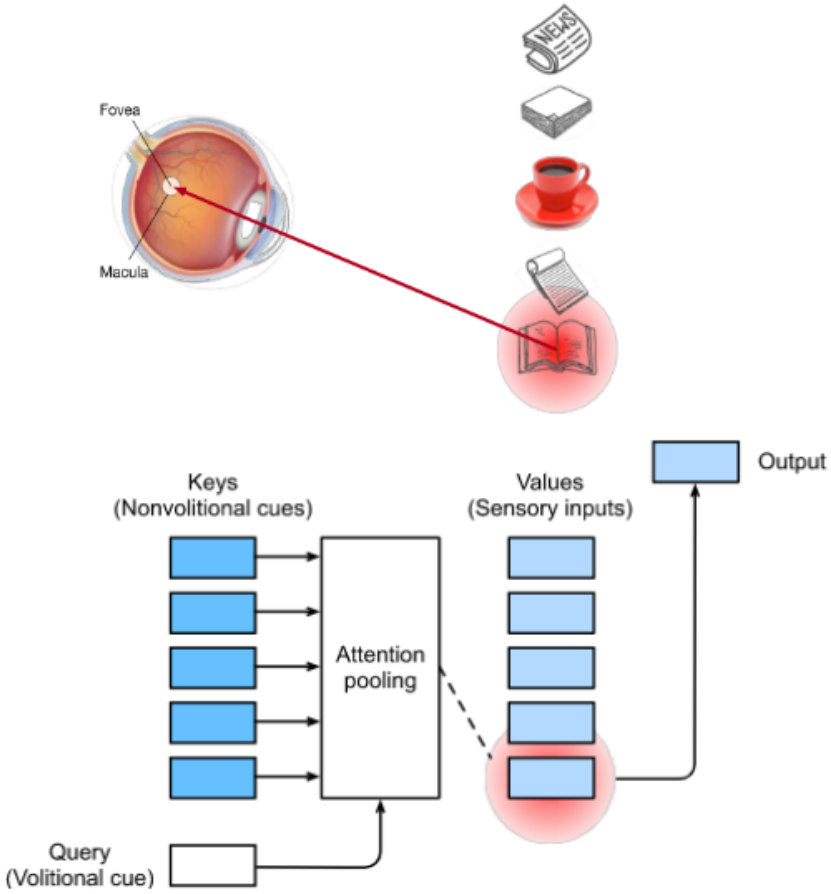

Figure 3.11: Inspiration of Attention Mechanism from human to machine, from [11]

In [26], attentional cues that direct how humans arrange attention are divided into two different types, nonvolitional cue, and volitional cue. Nonvolitional cue is based on how salient and conspicuous an object is, while the volitional cue is task-dependent. For example, as shown in the upper subimage of figure 3.11, there are five objects in the right column, and in the middle, the cup of coffee is conspicuously red as other objects are not colored. This feature (color) which shows saliency and conspicuity of the cup coffee is a nonvolitional cue. However, because our task is to read a book, our attention is directed by this idea to find a book for this task and this task-dependent idea is a volitional cue. Translating this idea in the context of the machine attention mechanism, queries are volitional cues and on a certain query, we use keys (nonvolitional cue) to find the suitable values (objects like a book) to generalize the output (accomplish the task: reading a

book), as shown in the bottom sub-image of figure 3.11.

Self-attention is generally a weighted average sequence-to-sequence operation. The calculation can be shown as follows

$$\begin{aligned} \mathbf{q}_i = \mathbf{W}_{\mathrm{q}} \boldsymbol{x}_{\mathrm{i}} \quad \mathbf{k}_{\mathrm{i}} = \boldsymbol{W}_{\mathrm{k}} \boldsymbol{x}_{\mathrm{i}} \quad \boldsymbol{v}_{\mathrm{i}} = \boldsymbol{W}_{v} \mathbf{x}_i \\ w'_{\mathrm{ii}} = \mathbf{q}_i^{\top} \mathbf{k}_{\mathrm{j}} \\ w_{ij} = \mathrm{softmax}\left(w'_{ij}\right) \\ \mathbf{y}_i = \sum_j w_{ij} \boldsymbol{v}_j \end{aligned} \tag{3.10}$$

Where q_i, k_i, v_i, x_i, y_i means the $i-th$ query vector, key vector, value vector input and output vector, weights for self-attention or weighted average operation are generated by the dot product of query vector and key vector. As shown in figure 3.12, a random $i-th$ input vector will be used in three ways, form the query vector for $i-th$ output, form the key vector, and value vector for every output.

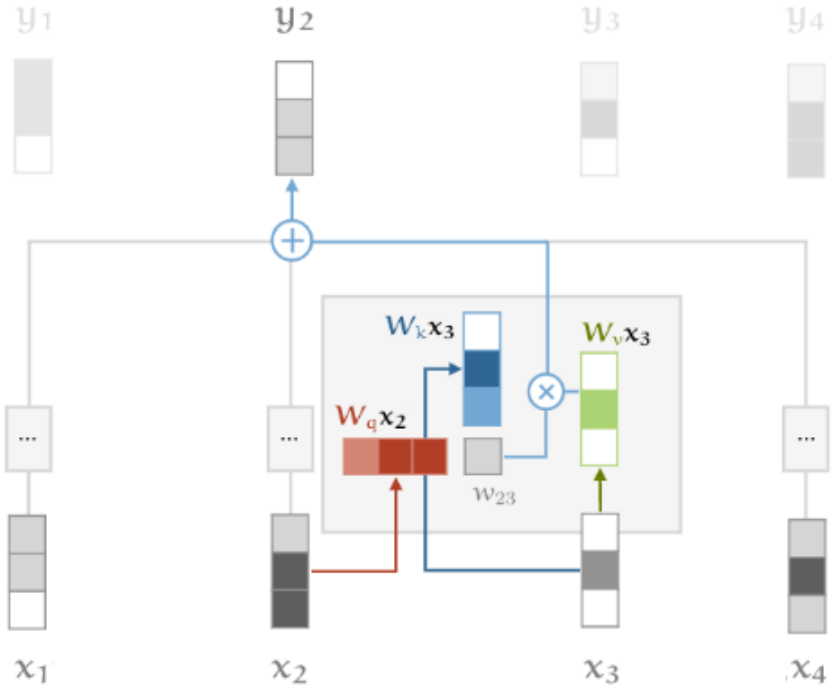

Figure 3.12: Illustration of the self-attention calculation, query vector colored red, key vector colored blu and value vector colored green, from [27]

3.6.2. MULTI-HEAD SELF-ATTENTION

Sometimes researchers would like to train more than one group of key, query, value parameters to learn different properties. In this case, multi-head attention (MHA) is useful.

We could train two groups in parallel and concatenate them after the attention operation and reduce dimension to suitable numbers by final fully connected (FC) layer as shown in figure 3.13. It is also possible to cut input into several chunks and assign corresponding groups of k,q, and v parameters for them.

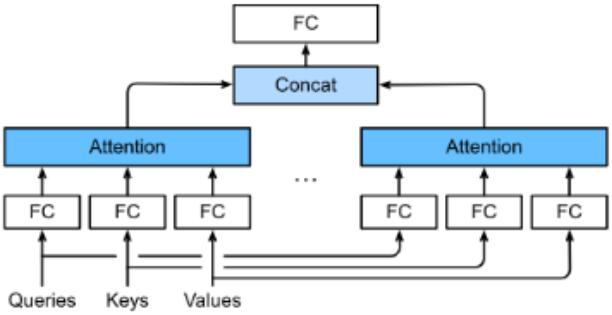

Figure 3.13: Multi-head illustration of self-attention,from [11]

3.6.3. POSITIONAL CODING

In many sequential tasks, we would like to use self-attention in an autoregressive way, which means we expect output at time step i is only related to previous input (time $< i$). To achieve this, we could multiply the weights of self-attention by a masking vector as shown in figure3.14, weights that are used to future input compared to output are arranged to zero. As the self-attention operation itself does not consider position information/sequence order, we usually add positional coding to mark the positions for each input vector [].

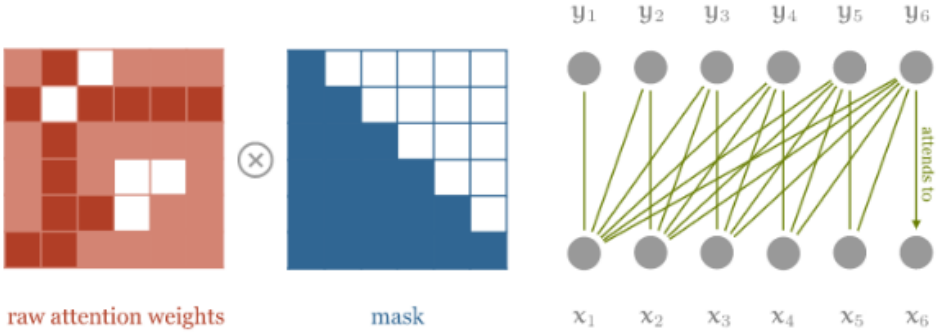

Figure 3.14: Masking operation to use self-attention in an autoregressive way,from [27]

3.6.4. ENCODER-DECODER STRUCTURE OF TRANSFORMERS

The transformer encoder-decoder structure proposed in [25] is shown in figure 3.15. The Encoder consists of n repeated blocks in the left dotted frames. Each encoder block con-

tains two sublayers. The first sublayer consists of a multi-head attention layer and an add & norm layer. The second is composed of an FFN layer and an add & norm layer. The add & norm layer is used to add a residual connection followed by a layer normalization, which helps to train deep neural networks faster and more accurately. The input is first converted to embeddings by convolutional layers and the positional encoding is added to the input embedding before being fed to the encoder.

The design of the decoder is very similar to the encoder. The decoder also consists of n repeated blocks in the right dotted frames. Each block consists of three sublayers, two sublayers are the same with encoder and one sublayer consisting of masked multi-head attention and an add & norm layer. The middle sublayer is also called encoder-decoder attention whose queries are from the previous decoder sublayer but keys and values are from the encoder. The masked multi-head self-attention is allowed to reach future positions and the masking ensures that the later encoder-decoder attention would not reach those future positions.

During the training of this transformer structure, the encoder attention receives a preprocessed input embedding sequence and maps it to a latent embedding sequence. This latent embedding sequence is passed to the encoder-decoder attention as key and values. Meanwhile, the masked decoder attention received preprocessed embeddings and map it to another latent embedding and passed to encoder-decoder attention as queries. The decoder then uses them to generate desired sequences. The softmax algorithm in the FC layer is applied and a loss function of cross-entropy is adopted to compare final decoder sequences and labels to update transformer parameters.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[3] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Qadir, and B. W. Schuller, "Deep representation learning in speech processing: Challenges, recent advances, and future trends," *arXiv preprint arXiv:2001.00378*, 2020.

[4] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[5] R. Togneri, M. Alder, and Y. Attikiouzel, "Dimension and structure of the speech space," *IEE Proceedings I (Communications, Speech and Vision)*, vol. 139, no. 2, pp. 123–127, 1992.

[6] S. S. A. Zaidi, M. M. Ali, F. Aftab, Y. Shahid, and M. Khurram, "Name spotting over low signal-to-noise ratio (snr) using blind source separation and connectionist temporal classification," in *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*. IEEE, 2017, pp. 320–325.

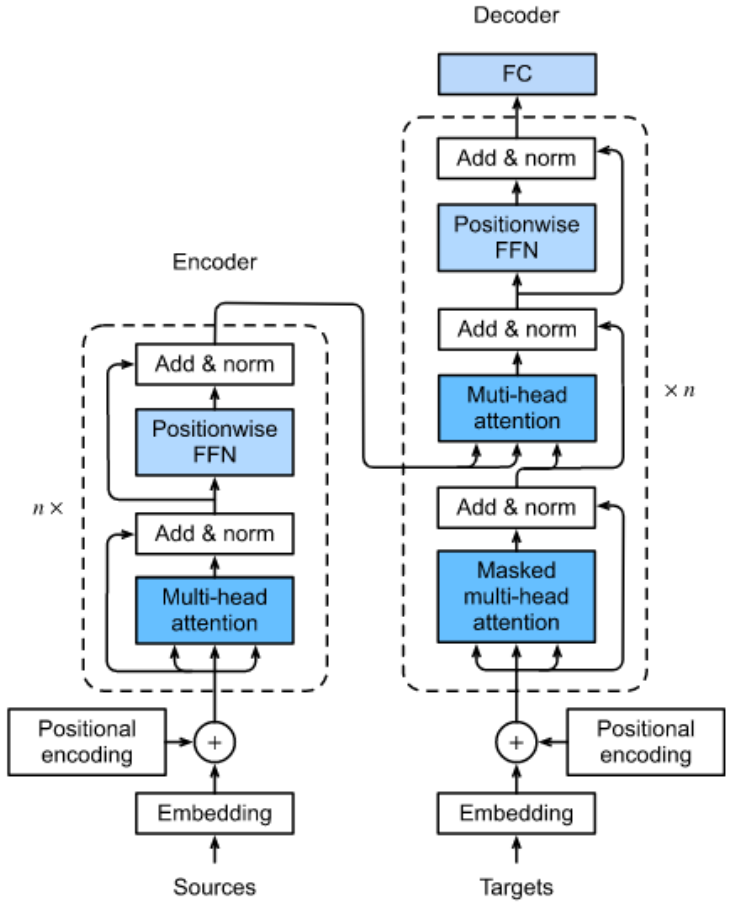

Figure 3.15: Architecture of Transformers, from [11, 25]

[7] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[8] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1476–1485.

[9] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[10] W. Wang, Q. Tang, and K. Livescu, "Unsupervised pre-training of bidirectional

speech encoders via masked reconstruction," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6889–6893.

[11] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[12] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[13] J. Konečnỳ, J. Liu, P. Richtárik, and M. Takáč, "Mini-batch semi-stochastic gradient descent in the proximal setting," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 242–255, 2015.

[14] L. K. Chan, N. Jegadeesh, and J. Lakonishok, "Momentum strategies," *The Journal of Finance*, vol. 51, no. 5, pp. 1681–1713, 1996.

[15] T. Tieleman, G. Hinton *et al.*, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[16] A. Lydia and S. Francis, "Adagrad—an optimizer for stochastic gradient descent," *Int. J. Inf. Comput. Sci*, vol. 6, no. 5, 2019.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[19] H. Lee, P. Pham, Y. Largman, and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," *Advances in neural information processing systems*, vol. 22, pp. 1096–1104, 2009.

[20] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. Li, "Reconstruction of porous media from extremely limited information using conditional generative adversarial networks," *Physical Review E*, vol. 100, no. 3, p. 033308, 2019.

[21] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[22] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[26] W. James, *The principles of psychology*. Cosimo, Inc., 2007, vol. 1.

[27] P. Bloem, "Transformers from scratch," http://www.peterbloem.nl/blog/transformers, 2021, [Online; accessed September-2021].

4

PIPELINE OF CONSTRUCTING LANGUAGE MODELLING SYSTEM FROM RAW SPEECH

In this project, we develop a pipeline method to construct a language modeling system from raw speech. The pipeline consists of three components, the self-supervised frame-level speech representation module which encodes the raw audio into frame-level representations. The unsupervised segment learning module learns the segment representations and clusters those segment representations into pseudo-phonemes. The first two components are together called pseudo-phoneme encoders. Finally, the pseudo-phoneme sequences are used to train the language model module. In the next few sections, we will introduce those components in more detail.

4.1. PSEUDO PHONEME ENCODER

In phonetic linguistics, a word is often represented as a string of phonemes, as an example of the Transcription column shown in table 4.1. A phoneme (examples shown in first row in 4.1) is a speech sound and is defined as the smallest unit to change a word to another [1]. Generally, a phoneme in speech signals lasts from 5ms to hundreds of ms. The International Phonetic Alphabet (IPA) is a standard phonetic alphabet developed for most languages in the world, and there are many other alphabets, such as ARPAbet [2] for American English, developed for a more compact phonetic alphabet for certain languages.

In our problem, a low-resourced context is given, while we assume the speech data is from a language without any text resources. The goal of the pseudo-phoneme encoder is to construct a pseudo-phoneme alphabet with discovered phoneme-like units that are able to used by language models distinguishing different words and learn high-level linguistic knowledge for an unknown language. Our idea of discovering such phoneme-

Symbol	Word	Transcription
[zh]	ambrosia	[ae m b r ow zh ax]
[l]	licorice	[l ih k axr ix sh]
[w]	kiwi	[k iy w iy]
[r]	rice	[r ay s]

Table 4.1: Examples of ARPAbet phone symbol transcription to represent words,from [1]

like units is first to divide a speech audio into segments with boundaries close to true phoneme boundaries of the language, while neighborhood segments should be discriminative to each other. Then, we cluster those segments into pseudo-phoneme units.

To achieve this idea, we develop an phoneme encoder consists of a frame-level representation learning method, which is responsible for generate frame-level representations which are discriminative to each other when they belong to different phonemes, and a segment-level clustering module to learn segments, constrained by reliable boundary lables, from previous frames and cluster the segments into units.

The next two sections will introduce the frame-level representation learning method and segment-level clustering method we investigate in our project.

4.2. SELF-SUPERVISED SPEECH FRAME REPRESENTATION LEARNING

In our spoken language modeling system, the first step is to transform the raw audio input into frame-level speech representations, which preserves the content of the speech utterance and eliminates other information as much as possible. This step is done by a self-supervised speech frame representation learning module.

In the next few subsections, we will briefly review the background of speech representation learning. Then an introduction for self-supervised speech representation learning and the two self-supervised speech representation learning methods we investigated in our pipeline method will be given.

4.2.1. SPEECH REPRESENTATIONS

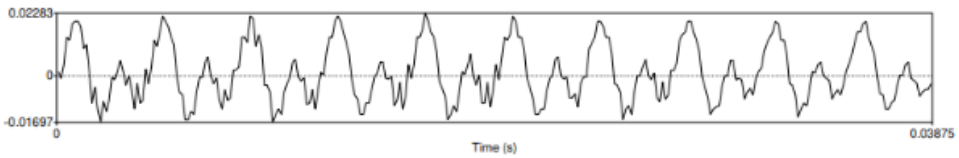

Figure 4.1: An illustration of the sound waveform of phoneme in the air, from [1]

The raw speech audio is a complex series of continuous changes in air pressure generated by human vocal organs [1], as shown in figure 4.1. To record human speech in machines for analyzing and processing, speech receivers such as microphones convert the raw audio into analog electric signals and digitize those signals by sampling and

quantization. As frequencies of general human speech information are below 10k Hz, according to Nyquist rules, the sampling rate of 20k Hz is enough and the 16k Hz sampling rate is widely applied in practical applications. After digitizing, 1-second speech audio is represented by 16k sampled values.

The performance of machine learning tasks highly depends on input features. In machine learning tasks for speech processing and analyzing, more informative and compact speech features, also called representations, compared to sampled audio values are expected. Traditionally, characteristics of speech in the frequency domain and perceptual domain (pitch, loudness) are utilized for building more informative and compact representations. A general procedure of constructing such representations [3] is

- A. slice speech signals into short-term frames, for example, 10ms frame with 50% overlapping and a 30ms window.
- B. Perform some calculations on each speech frame such as energy computing.
- C. (optional) Perform some calculations on a batch of short frames to obtain long-term features. For example, we would like a feature containing the content of a word. In this case, we may need to combine hundreds of short-term frames to generate one feature.

Simple speech representations such as log-energy, zero-crossing rate, and complex ones such as Mel-frequency cepstral coefficients (MFCC) are popular traditional features for speech-related tasks. For decades, feature engineering where experts manually design features like the above ones by leveraging characteristics of speech, and designing classification or prediction models considered separate problems. However, feature engineering usually requires a lot of labor. Researcher expect a less hand-crafted way to obtain features for downstream tasks. Then, in 2006, a work of training deep neural networks for representation learning automatically appeared[4] and was soon followed by large amount of research in representation learning with deep models. In speech processing area, the success of representation learning methods with DNNs first appeared in decreasing word error rate (WER) on ASR task [5]. Since then varied deep architectures were applied in speech representation learning.

4.2.2. SELF-SUPERVISED SPEECH REPRESENTATION LEARNING

Deep models for speech representation learning can be trained in different ways such as supervised, unsupervised, and self-supervised. Supervised deep learning can learn well-fitted representations for certain tasks. However, manuel labels are required and for speech data, annotations are usually expensive and scarce, especially for low-resource languages.

Self-supervised learning is a particular type of learning method, which do not need labels for training a model. Figure 4.2 shows an illustration of how self-learning tasks are constructed. In general, the strategy of self-supervised learning is to used one part of input data as labels (pale-blue data) and train the network predict this part from other input data (light-purple data). The loss functions are calculated between predicted output and labels made by part of data. This strategy has been widely used in time-series data representations (predict the data of a certain moment from other moment, future from

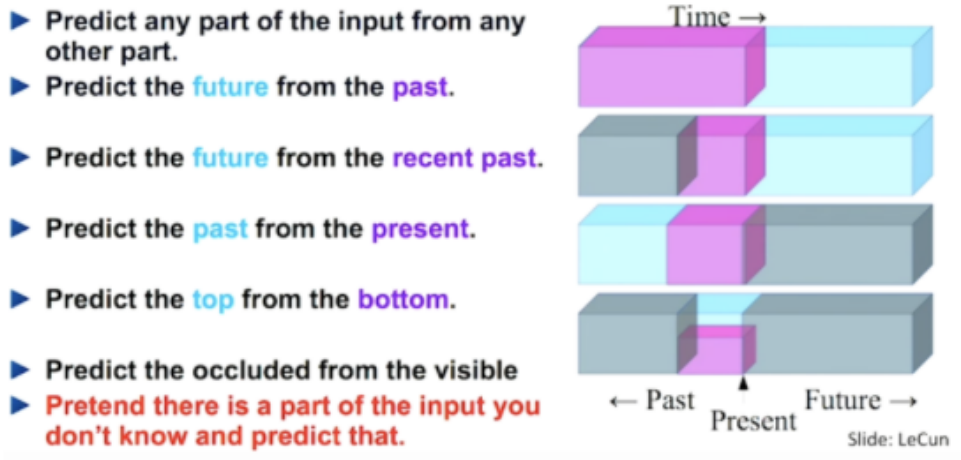

Figure 4.2: A summary of self-supervised learning constructing manners, from [6]

past or past from future). In self-supervised speech representation learning, common tasks are usually constructed in an autoregressive manner, by using the past to predict the future [7, 8].

In the context of self-supervised speech representation learning, the prediction performance of self-supervised tasks/pretext tasks is often not important. What we care about is the learned representation. Specifically, we concern if the learned intermediate representation can capture expected linguistic knowledge, structural or contextual information and if those representations can perform well in downstream ASR or other speech understanding tasks. In the next two subsections, we will introduce two self-supervised structures, wav2vec 2.0 and contrastive predictive coding, with which we experimented within our system.

CONTRASTIVE PREDICTIVE CODING

The Contrastive Predictive Coding (CPC) model overview is shown in figure 4.3 [9]. The input raw audio data are transformed into a compact latent space by encoder g_{enc}. Then the encoded representation z is fed into the autoregressive model g_{ar} to generate contextualized representation c. The autoregressive models use context representation at the present time step to predict the encoded representation in several future steps (positive example).

In practice, each future step prediction corresponds with one predictor and a loss function. Each future prediction is compared to several irrelevant predictions, in which the autoregressive models use context representation at present time step to predict the encoded representation in random selected time steps (negative examples). The goal is to generate representations that are highly related with neighboring future representations and highly unrelated with distant random representations. To train this end to end model, the Noise-Contrastive Estimation is used for the InfoNCE loss function, which

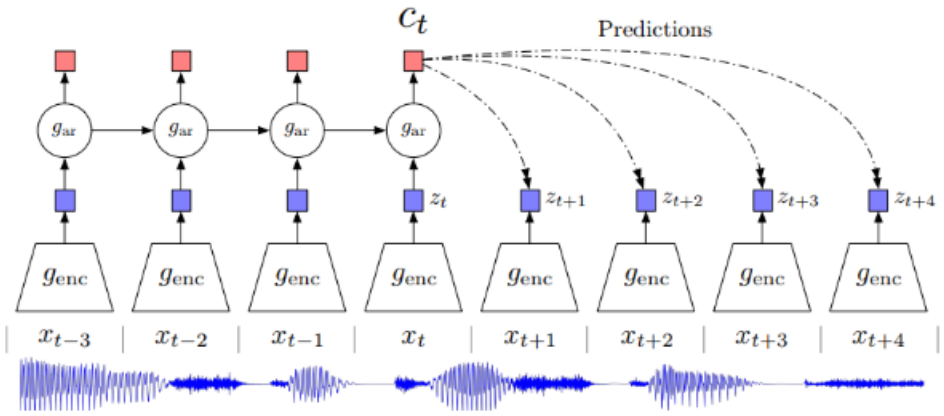

Figure 4.3: Illustration of Contrastive Predictive Coding framework which learns contextualized speech representations, from [9]

can be shown as

$$\mathscr{L}_{\mathrm{N}} = -\underset{X}{\mathbb{E}}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)}\right] \tag{4.1}$$

wherein the numerator is the $t+k$ step prediction result with the kth prediction function. The denominator is the sum of prediction results of c_t with other random time step representations. The intermediate representation z and c can be used in different downstream tasks and the contextualized ones are recommended for speech recognition representation.

WAV2VEC 2.0

The diagram of Wav2vec 2.0 [7] is shown in figure 4.4. The convolutional neural networks referred to as CNN, encode the raw audio signals into latent speech representations. These representations are then passed to a masked transformer network to embed context information into the speech representations. In the meantime, the model learns to quantize these latent representations to form two quantizer codebooks. Each codebook contains hundreds of entries which are called discrete units. Two codebooks are used to enable product quantization, which is used in the prior work of wav2vec2 and shown to result in better ASR performance. The self-supervised task or, in this case, the contrast task is to distinguish the true discrete unit (positive) from the distracted ones (positive).

The loss function consists of two parts, contrastive loss and diversity loss with a tuned hyperparameter α

$$\mathscr{L} = \mathscr{L}_m + \alpha \mathscr{L}_d \tag{4.2}$$

The goal of optimizing contrastive loss is to distinguish true quantized latent representation q at time t (in the numerator) with other K uniformly random-selected distracted latent representation (in the denominator). The calculation of the contrastive

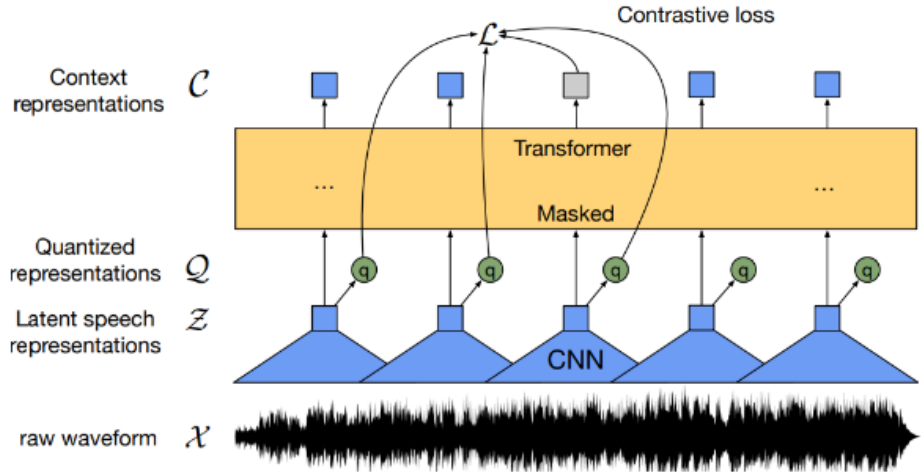

Figure 4.4: Illustration of wav2vec 2.0 which learns contextualized speech representations using quantized speech units, from [7]

loss is following

$$\mathscr{L}_m = -\log \frac{\exp\left(\text{sim}\left(\mathbf{c}_t, \mathbf{q}_t\right)/\kappa\right)}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp\left(\text{sim}\left(\mathbf{c}_t, \tilde{\mathbf{q}}\right)/\kappa\right)} \tag{4.3}$$

where sim represents the similarity calculation.

$$\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\| \tag{4.4}$$

There are two codebooks with hundreds of entries each in the wav2vec 2.0 model. To encourage two codebooks to be learned differently and entries to be used in an equal way, the diversity loss is calculated by maximizing the entropy of the averaged softmax distribution of codebook G over the codebook G and entry V.

$$\mathscr{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H\left(\bar{p}_g\right) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v} \tag{4.5}$$

4.2.3. SPEAKER-INVARIANT SPEECH REPRESENTATION LEARNING

The above two self-supervised structures do not specifically aleviate speaker information, but speaker information is shown as a harmful noise in speech recognition tasks [10]. In this section, we will introduce the model that we used to learn speaker-invariant speech representations.

4.2.4. FACTORIZED HIERARCHICAL VARIATIONAL AUTOENCODER

Speech is usually considered stationary in short frames (10-30ms), but many noise might be stationary in a longer segment. This means the speech information are encoded in frame levels but noisy information are encoded in segment or utterance levels. Thus if

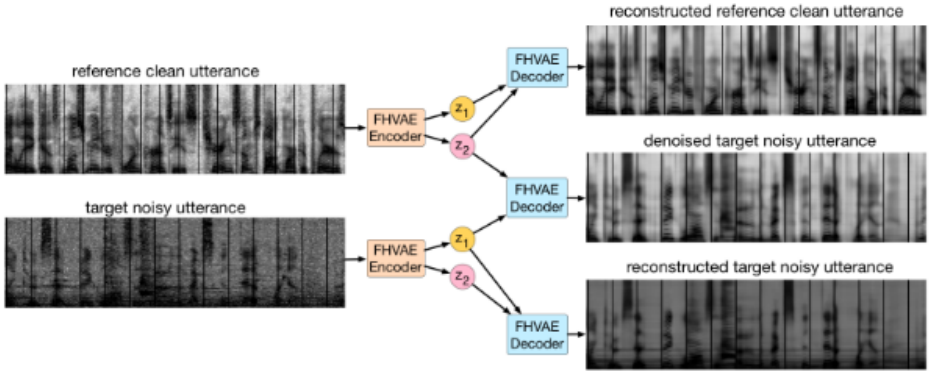

Figure 4.5: Illustration of the factorized hierarchical variational autoencoder, from[11]

we could factorize the speech into different levels in a suitable way, it is possible to extract or disentangle solelyclean speech from noise information signals. The illustration of how factorized hierarchical variational autoencoder (FHVAE) [11] works is shown in figure 4.5. The encoder in the yellow box learns to generate two types of latent representations z_1, z_2 from sequential data with a factorized hierarchical graphical model. As an example in figure 4.5, a clean utterance (reference speaker) and a noisy one (noisy speaker) are fed to the encoder and the encoder learns to encode linguistic content into z_1 and speaker-noise content into z_1. The decoder model is designed to replace the noisy z_1 with clean z_1. Noisy speech representation z_1 and clean z_2 are combined to form a new denoised clean output speech utterance containing the speech content of original noise utterance.

4.3. UNSUPERVISED SPEECH SEGMENT UNIT LEARNING

As mentioned in previous sections, our pseudo-phoneme encoder consists of a frame-level representation learning method and a segment-level clustering method. After frame representation learning, we use the segment-level clustering method to discover pseudo-phonemes from frame-level speech representations, preparing for later language modeling. In this chapter, we will introduce our three-step segment-level clustering method to learn unsupervised speech segment units from speech frame representations.

As shown in figure 4.6, we first construct a self-supervised boundary detector, consisting a phone boundary learning with two additional voice activity detection modules (VAD), to detect boundary labels. Then we use the averaging to obtain segment-level representations and finally we use Kmeans-50 Clustering to generate pseudo-phonemes.

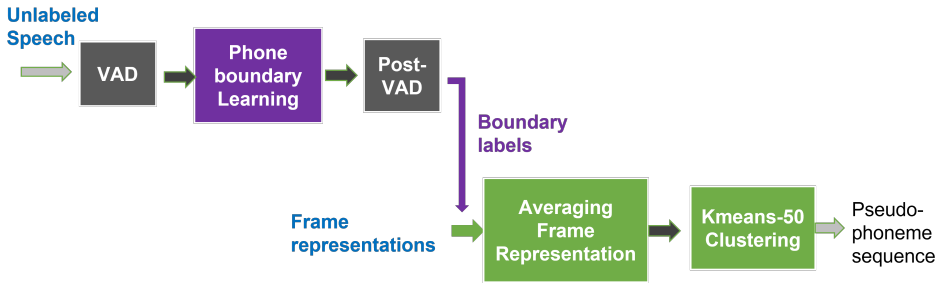

Figure 4.6: An overview of segment level clustering in our pipeline method

4.3.1. BOUNDARY DETECTOR

In our system, we employ self-supervised boundary detection [12] as our framework of boundary detector. This architecture was the state-of-art boundary detection algorithm in TIMIT and Buckeye dataset by the beginning of our project, but has now been slightly surpassed currently by [13].

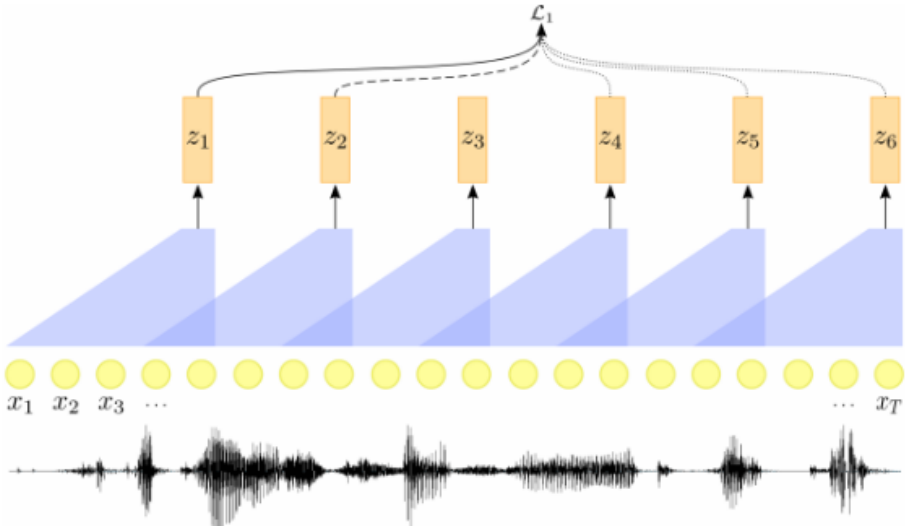

Figure 4.7: overview of the self-supervised boundary detection method, from [12]

As shown in figure 4.7, the system contains an encoder which encodes speech $x_1, x_2, ...$ into frame-level representations $z_1, z_2, ...$ This system was optimized on a contrast loss, in which lower loss is obtained when the current frame z_1 is close to the one neighbor frame z_2 and far from some random selected frames. This model is very similar to the contrast predictive coding architecture but it do not apply autoregressive models(AR). This is because researchers found applying AR will decrease the performance of boundary detection tasks. By using the contrast loss, the system encourages speech representation z_1 to be learned similar to the next step speech representation z_2 and unsimilar to

a randomly selected set of speech representations. The loss function can be shown as

$$
\begin{aligned}
D\left(\mathbf{z}_i\right) &= \left\{\mathbf{z}_j : |i-j| > 1, \mathbf{z}_j \in \mathbf{z}\right\} \\
\hat{\mathscr{L}}\left(\mathbf{z}_i, D_K\left(\mathbf{z}_i\right)\right) &= -\log \frac{e^{\operatorname{sim}\left(\mathbf{z}_i, \mathbf{z}_{i+1}\right)}}{\sum_{\mathbf{z}_j \in \left\{\mathbf{z}_{i+1}\right\} \cup D_K\left(\mathbf{z}_i\right)} e^{\operatorname{sim}\left(\mathbf{z}_i, \mathbf{z}_j\right)}} \\
\mathscr{L} &= \sum_{\mathbf{x} \in S} \sum_{\mathbf{z}_i \in f(\mathbf{x})} \hat{\mathscr{L}}\left(\mathbf{z}_i, D_K\left(\mathbf{z}_i\right)\right)
\end{aligned}
\tag{4.6}
$$

where $D(zi)$ represents the random negative example set and the similarity is calculated by

$$
\operatorname{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^{\top} \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\| \tag{4.7}
$$

The boundary is determined by a peak detection algorithm [14] among the disimilarity score. The hyperparameter prominence, which measures how far a peak stands out from the lowest contour line of a signal, is automatically optimized by compare network prediction and validation label using a chosen metric such as precision. It could also be hand-crafted after the training.

$$
\operatorname{score}\left(\mathbf{z}_i\right) = -\operatorname{sim}\left(\mathbf{z}_i, \mathbf{z}_{i+1}\right) \tag{4.8}
$$

4.3.2. REMOVAL OF SILENCE SEGMENTS

In terms of the VAD, in practice we found too many silence frames in the inpur will lead to performance decrease in phone boundary learning. Thus we need a preprocessing step to remove silent segments in the input. In our project, a voice activity detector (VAD) is used to detect the silent frames and generate speech-present/unpresented labels. We use those labels to cut out the silent audio in the input. The VAD algorithm applied in our system is an unsupervised segment-based method for robust voice activity detection (rVAD) [15] because it is current state-of-art unsupervised VAD method in Aurora, one of the commonly used dataset for silence recognition [16].

As an overview shown in figure 4.8, rVAD consists of two denoising steps/passes and a voice activity detection step. In step 1, high-energy speech segments are detected by a SNR estimation method, posteriori signal-to-noise ratio weighted energy difference (pSNRD). Then, detected noise is set to zero. In step 2, these segmentations are enhanced to remove the remaining noise by minimum mean-square error (MMSE). In step 3, the enhanced segments were extended to include voiced sounds, unvoiced sounds, and the pSNRD is used again to detect voice activity. In this process, the enhanced segments are used as anchors for detecting speech in other unprocessed sounds, which is more efficient in processing large amount of data. Labels of 1 (speech present) and 0 (not presented) indicates the preserving or cutting out of speech data in our system.

In our system, we use voice activity detectors (VAD) in advance to remove large silence segments (>80ms). This is to avoid removing low-energy speech sounds. Instead, the post-VAD is used to check if there are still boundary labels inside shorter silence segments (<80ms). If yes, remove it.

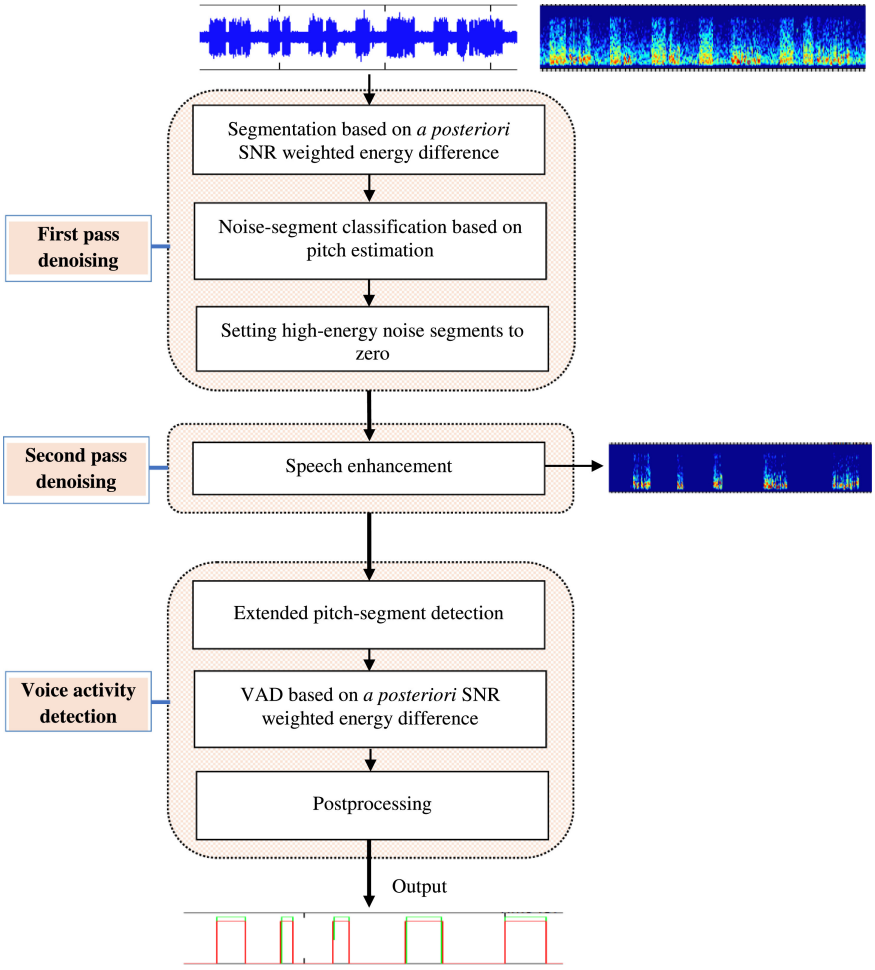

Figure 4.8: Overview of the voice activity detection of rVAD, from [15]

4.3.3. AVERAGING

Downsampling and averaging are two common segment learning methods. However, as downsampling usually leads to increasing of dimensions of the representation. Considering that our representations are already 512 dimension, we prefer averaging method in case increasing dimensions cause higher difficulty for latter clustering. Thus we chose the averaging as our segment learning method.

As shown in figure 4.9, the averaging process in our system is that given the frame representation $z1, ..., zn$ and boundary labels which indicate the boundary frame with label 1 and others with label 0, the frame representations between two boundary labels are averaged equally to form segment representations $s1, ..., sq$.

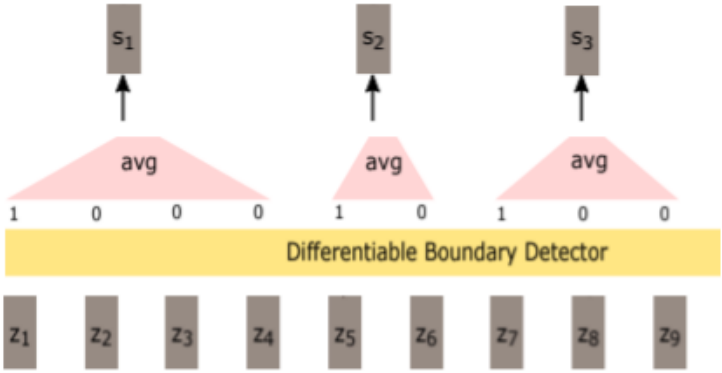

Figure 4.9: illustration of averaging process in our method

4.3.4. KMEANS CLUSTERING

In the third step of segment-level clustering, we need the Kmeans algorithm to cluster segment representations into pseudo-phonemes. As an example shown in figure 4.9, given a certain distant function such as euclidean distance, the K-means clustering procedure is calculated as follows:

- Initialize cluster centroids randomly.
- Repeat the following two steps until conditions are satisfied or until convergence:
  A. For every point i, set its cluster label c to be the closest cluster centroid

$$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2 \tag{4.9}$$

  B. For each cluster j, set the new centroid in this cluster to be the center of all new points in this cluster

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\}\, x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}} \tag{4.10}$$

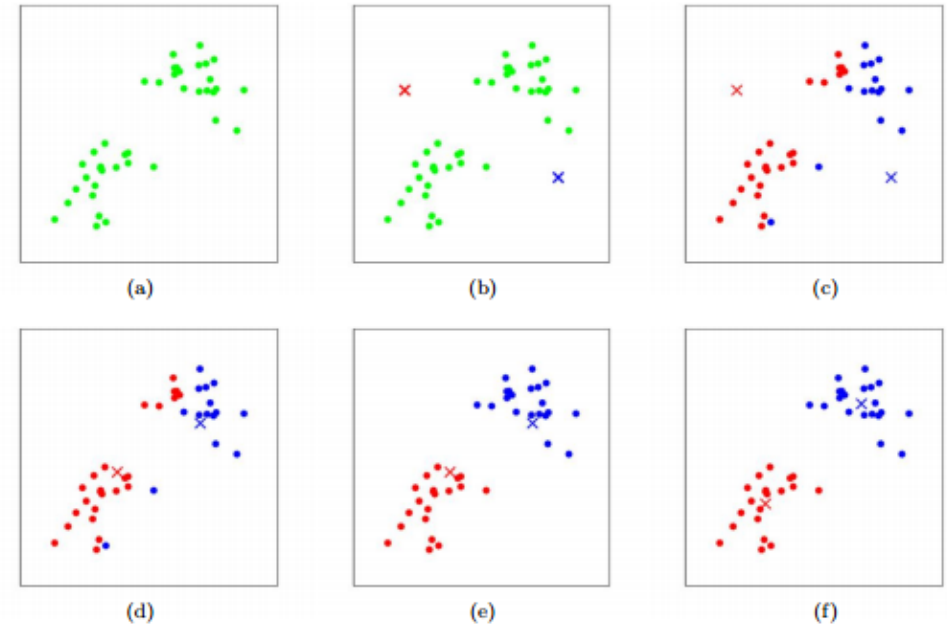

Figure 4.10: Illustration of how a Kmeans-2 clustering is constructed in steps from (a) to (f), from [1]

4.4. LANGUAGE MODELS

Language models are also called probabilistic language models because they assign probabilities to judge how probable can the sequences of linguistic units (such as words, characters) be seen in the training dataset. Given some previous context information of a sequence, language models can predict the next linguistic unit. The N-gram model is the most popular traditional language model. This model estimates the probability model of the current linguistic unit over up to n of previous units. Recently, in language modeling tasks, state of art results are achieved by RNN-based or Transformer-based language models, which have the advantage of the unfixed size of the previous context.

There are two types of evaluation of language models, extrinsic and intrinsic evaluation [1]. Extrinsic evaluation means evaluating different language models in a certain context of application and see how much performance improvement can be brought by new language models. This is also the most popular evaluation. For example, in wav2vec 2.0 [7], LSTM language models and 4-gram models are compared by comparing the accuracy of speech transcriptions. This is usually cost-inefficient as it requires running large end-to-end models several times. Another option is intrinsic evaluation, which measures intrinsic characteristics could be obtained directly from LMs and independent of any other application such as perplexity, which calculated by normalized inverse probability of test dataset.

In our spoken language modeling system from raw speech, we use pseudo-phonemes, a self-defined language annotation to train language models. Following the setting of the

ZeroSpeech Challenge 2021, where LSTM and BERT are applied in the baselines of different GPU budgets, we also explored these two architectures in our system for scenarios of different GPU budgets. We will give an introduction to these two language models in the next two subsections. Perplexity metric varies with the granularity of the training data fed into language models. In our context, this metric varies with the time intervals between each discrete unit. This time interval can be of any time length, determined by the discrete unit encoder or pseudo-phoneme encoder in our case. Thus different spoken language systems cannot be compared using the perplexity scores. To resolve this difficulty, Zerospeech 2021 proposed their language modeling evaluation metrics in which four discrimination tasks with corresponding datasets are used to evaluate the language models at four linguistic levels. Those evaluation metrics are explained in more detail in chapter 5.

4.4.1. DNN BASED LANGUAGE MODELS

RNN-based language models especially LSTM-based and transformer-based language models are current state-of-art models for natural language processing (NLP) [17, 18]. However, in what situation should we choose one of the mentioned two models is still an open question, depending on datasets and tasks. A general assumption is that transformer-based architectures have potential to be trained with much larger amount of input data [18] and parameters to obtain better performance than other language models. While under the context of a small dataset (thousands of utterances) for text classification problems[19], some results shows that LSTM performs better than transformers. In our case, we mainly consider a low-budget scenarios, with fewer than 60 hours on an V100 GPU required for language training, the dataset for training language model contains 960 hours speech data. A pair of low-budget LSTM and comparable-scale BERT are provided forthis scenario and we also explore both structures in our projects. In this section, we will introduce the basic concepts of the two language models used in our experiments.

LSTM

As shown in figure 4.11 [20], given a LSTM unit on the left, a general LSTM-based language model contains an input layer, output layer, hidden layer, and projection layer (optional). In the input layer, the linguistic units are generally encoded by 1-of-K coding where K is the number of types of a certain linguistic unit (if K is a phoneme, K is the number of the type of phonemes). In the right of the figure 4.11, the input layer encode an input as "1" Softmax activation function with cross-entropy loss is often used in the output layer to produce probabilities in the output layer. Hidden layers consist of several LSTM units to learn linguistic knowledge, and the projection layer is usually used to normalize/standardize the input.

LSTM based models are relatively harder to use hardware acceleration compared to transformer based models. Thus time cost of training LSTM with the same parameters are usually higher. Transformer-based models, in the opposite, can be easily implemented in a parallel way. Int the Next section, we will introduce the most popular transformer based models, BERT [21] in natural language modelling tasks.

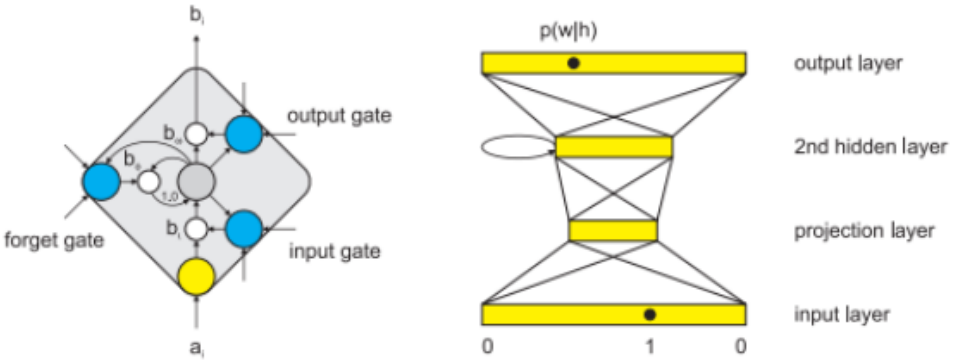

Figure 4.11: lstm-based natural language model architecture. The left represents an LSTM unit and the right is the LM overview, from [20]

BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS

BERT stands for Bidirectional Encoder Representations from Transformers [21], which is proven to achieve great performance on many language-based tasks and is now a default model for NLP research. The core of BERT is a stack of transformer blocks.

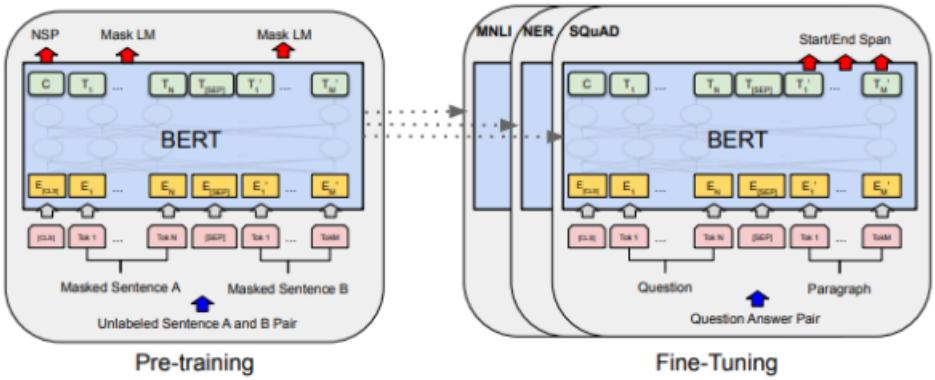

Figure 4.12: General working procedure of BERT. First, pre-training of masked LM is done by feeding unlabeled text data, then a downstream task training is used to fine-tune the BERT, from [21]

A general workflow of BERT is shown in figure 4.12. Pretraining BERT in a certain unsupervised way and then fine-tune BERT with labels for downstream tasks such as Question- Answer System. There are two different tasks can be used in pretraining. Next Sentence Prediction (NSP), focusing on understanding semantic relationships between two sentences, and Masked LM, for language modelling. In our project, we only use the BERT for training language models with Masked LM task, so fine-tune step is not performed. Specifically in our task, we mask part of the input randomly, and then predict those masked part.

REFERENCES

[1] J. Daniel and H. M. James, "Speech and language processing," 2000.

[2] J. E. Shoup, "Phonological aspects of speech recognition," *Trends in speech recognition*, pp. 125–138, 1980.

[3] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Qadir, and B. W. Schuller, "Deep representation learning in speech processing: Challenges, recent advances, and future trends," *arXiv preprint arXiv:2001.00378*, 2020.

[4] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[5] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[6] Y. LeCun, "Self-supervised learning: could machines learn like humans?" https://www.youtube.com/watch?v=7I0Qt7GALVk, 2021, [Online; accessed September-2021].

[7] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[8] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. v. d. Oord, "Learning robust and multilingual speech representations," *arXiv preprint arXiv:2001.11128*, 2020.

[9] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[10] L. van Staden and H. Kamper, "Improving unsupervised acoustic word embeddings using speaker and gender information," in *2020 International SAUPEC/RobMech/PRASA Conference*. IEEE, 2020, pp. 1–6.

[11] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," *arXiv preprint arXiv:1709.07902*, 2017.

[12] F. Kreuk, J. Keshet, and Y. Adi, "Self-supervised contrastive learning for unsupervised phoneme segmentation," *arXiv preprint arXiv:2007.13465*, 2020.

[13] S. Bhati, J. Villalba, P. Żelasko, L. Moro-Velazquez, and N. Dehak, "Segmental contrastive predictive coding for unsupervised word segmentation," *arXiv preprint arXiv:2106.02170*, 2021.

[14] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey,

İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[15] Z.-H. Tan, N. Dehak *et al.*, "rvad: An unsupervised segment-based robust voice activity detection method," *Computer speech & language*, vol. 59, pp. 1–21, 2020.

[16] H. Hirsch, "The aurora experimental framework for the performance evaluations of speech recognition systems under noisy conditions, isca itrw asr2000," *Automatic Speech Recognition: Challenges for the Next Millennium*, 2000.

[17] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," *arXiv preprint arXiv:1611.06639*, 2016.

[18] I. Tenney, D. Das, and E. Pavlick, "Bert rediscovers the classical nlp pipeline," *arXiv preprint arXiv:1905.05950*, 2019.

[19] A. Ezen-Can, "A comparison of lstm and bert for small corpus," *arXiv preprint arXiv:2009.05451*, 2020.

[20] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

5

EXPERIMENTS AND RESULTS

In our project, our goal is to learn and leverage advanced speech techniques to build a spoken language modeling system from raw speech. We plan to construct our system by modifying the ZeroSpeech 2021 baseline. Specifically, we replace the discrete unit encoder in the baseline with a designed pseudo-phoneme encoder, which is the potential to generate phoneme-like units. We would like to investigate if applying this pseudo-phoneme encoder in our spoken language modeling system can improve the baseline system.

Besides, our pseudo-phoneme encoder relies on the frame-level representation learning module applying the CPC technique. If we could find representation learning modules that can achieve better language modeling performance, we might be able to construct a better pseudo-phoneme encoder. We thus proposed two potential replacements (wav2vec2 and a speaker-invariant CPC) for CPC and we plan to explore if they could bring better language modeling performance compared to CPC in the experiments.

Moreover, we would like to test if our pseudo-phoneme encoder is transferable across different languages and we plan to use Mboshi for testing the transferability.

Based on the above ideas, we set up four research questions and four corresponding experiment series. In this chapter, we will first present datasets and evaluation metrics involved in the four experiment series. Then in the next four sections, we will explain the experiment content, settings, and results for each experiment series.

5.1. DATASET

To construct the language modeling system from raw speech, we use LibriSpeech 960h dataset (460h clean and 500 noisy English speech) [1] and clean-6k version of the Libri-light [2] dataset (6kh hours English speech). These two datasets only contain raw speech and thousands of speaker labels. Besides, a 10 minutes Libri-light labeled dataset is used in the experiments of the pseudo-phoneme encoder, to evaluate the boundary detection performance of the boundary detector. Moreover, Zero speech challenge 2021 provides access to its Benchmark evaluation dataset [3]. In our project, the Libri-light ABX dataset,

sWUGGY dataset, and sBLIMP dataset are used for the language modeling evaluation at acoustic, lexical, and syntactic levels.

In terms of transferability experiments, we use the Mboshi (Bantu language spoken in Congo-Brazzaville) corpus [4] consists of 5k speech utterances (approximately 4 hours of speech) with text translations in French and phonetic speech translations. The Mboshi dataset provides a test set of 0.5k utterances with high quality of phoneme transcription (golden standard) and a training set of 4.5k utterances with relatively lower quality of phoneme transcription (silver standard).

5.2. EVALUATION METRICS

5.2.1. EVALUATION METRICS FOR LANGUAGE MODELING AT THREE LINGUISTIC LEVELS

In our project, we will use the following three discrimination tasks to evaluate the speech understanding of the language modeling system at three linguistic levels.

In acoustic Level, the system has to judge whether two input audios, while each contain a triphone such as 'apa' each and this two only differ in the middle phoneme), have the same phoneme in the middle or not. The ABX metric is to access acoustic level. Given a and x, which are tokens belonging to the same triphone group A (of cardinality n_A) and b belonging to a different triphone group B (n_B), we compute the ABX score as following

$$\hat{e}(A,B) := \frac{1}{n_A\,(n_A-1)\,n_B} \sum_{\substack{a,x\in A \\ x\neq a}} \sum_{b\in B} \left[\mathbb{1}_{d(b,x)<d(a,x)} + \frac{1}{2}\mathbb{1}_{d(b,x)=d(a,x)} \right] \tag{5.1}$$

This score is computed and normalized across all possible pairs from different triphone groups. If the wrong case happens, when the distance between b and x is not larger than the distance between a and x, the score will increase. Thus, lower ABX scores mean fewer wrong discrimination cases and better acoustic level performance and the ABX score is between [0,1].

At the lexical level, the system needs to judge if the word in the input audio is an existed word or a man-made non-word. Similarly, at the syntactic level, the system needs to judge whether a sentence in a speech utterance aligns with the grammatical rules of the training language or not. To obtain the interpretable scores of the two levels, we compute and compare the pseudo-probability of an input pair. At the lexical level, for example, a pair of an existed word and a non-word are given to the Language model to compute the pseudo-probability. In the correct case, the pseudo-probability of an existed word should be higher than a non-word. The lexical score is calculated by aggregating all correct cases divided by the number of all cases. A similar calculation is done for the syntactic score, which is calculated by aggregating the number of correctly discriminated grammar/non-grammar pairs divided by the number of all input pairs. Thus, higher scores mean fewer wrong discrimination cases are preferred for both lexical and syntactic levels. Both scores are within the range of $[0,1]$.

5.2.2. EVALUATION METRICS IN BUILDING THE PSEUDO-PHONEME ENCODER

The phoneme segmentation/boundary detection of the pseudo-phoneme encoder is evaluated by following four commonly used boundary detecting metrics, Precision, Recall, F1 score, R-value, and a self-defined metric, limited precision. Precision, Recall, F1-score are used to measure the performance of the phone segmentation. R-value is to evaluate the degree of over-segmentation when input audio is segmented into too many subcomponents (too many boundary labels where wrong ones are much more than correct ones). A higher R-value means lower over-segmentation. This limited precision is a special version of precision eliminating multiple counting. Multiple counting means when the differences between a true boundary time label and multiple predicted labels all fit the tolerance, all of the multiple predicted labels are regarded as correctly predicted. Over-segmentation happens, but the precision and R-value score cannot detect it. The limited precision will only count one predicted label as true and others are not counted.

Those metrics are calculated in the following formulas, where tp represents true positive, FP represents false positive, fn represents false negative, tp(limited) represents true positives which have a one-to-one corresponding true label in the ground truth.

$$
\begin{aligned}
\text{P} = \text{Precision} &= \frac{tp}{tp+fp} \\
\text{LimitedPrecision} &= \frac{tp(limited)}{tp+fp} \\
\text{R} = \text{Recall} &= \frac{tp}{tp+fn} \\
\text{F1} &= 2 * \frac{P*R}{P+R} \\
\text{OS} &= R/P - 1 \\
\text{R-value} &= 1 - \frac{|r_1|+|r_2|}{2} \\
r_1 = \sqrt{(1-R)^2+(OS)^2}, \quad r_2 &= \frac{-OS+R-1}{\sqrt{2}}
\end{aligned}
\tag{5.2}
$$

Higher scores of the five metrics are preferred, within the range of [0,1].

5.2.3. EVALUATION METRICS IN BUILDING SPEAKER-INVARIANT CPC

In experiments of speaker-invariant CPC. We need to evaluate if speaker information is reduced in the speaker-invariant CPC representations compared to the original CPC representation. We train an extrinsic speaker classification task for this evaluation. Specifically, two uniform speaker classification models are trained by the two different speech representations, respectively. Lower classification performance means, using the certain representation learning method, the speaker classification model is harder to recognize speakers. This indicates there is less speaker information remained in the representations.

Simple classification accuracy, calculated by the number of correct predictions di-

vided by the total number of examples, is used for evaluating the speaker classification performance. In our case, lower classification accuracy indicates less speaker information embedded in the representations.

5.2.4. THE BASELINE AND SHARED SETTINGS IN FOUR EXPERIMENTS

In this section, we give an introduction to the shared settings in four experiments. Especially, as the first three experiments all compared to the baseline performance and our spoken language modeling system also utilized parts of the baseline, it would be helpful to first explain the baseline in detail in this subsection.

The low budget baseline [3] in the ZeroSpeech 2021 is set as the baseline of the first three experiments. The low-budget means 60 training hours on one V100 GPU for training language models (LSTM or BERT-small). The baseline is consisted of

- CPC-big+Kmeans50+LSTM

while the language models are with the following parameters:

Language Models	L	HD	ED	FFD	H	Number of Parameters
BERT-small	8	512	512	2046	8	28M
LSTM	3	200	1024	200		22M

Table 5.1: Paramters of LSTM and BERT-small

L is the number of hidden layers; Row in ED, HD, and FFD shows the dimension of the embedding layer, hidden layer, and feed-forward layer respectively; H represents the number of attention heads, in BERT-small architecture.

Training language models with LSTM and BERT-small all used the same hyperparameters follow [3]. In all experiments, the language models are trained with Librispeech 960h dataset. The language models are implemented by fairseq [5]. All experiments are conducted with PyTorch [6] on TU Delft clusters with V100 GPUs. If without extra explanation, the LSTM model is trained with a total batch size of 163k tokens, while The BERT models were trained with a total batch size of 524k tokens. The learning rate for both models was prepared to a peak of $1 * 10^5$.

5.3. EXPERIMENTS FOR APPLYING PSEUDO-PHONEME ENCODER

In this set of experiments, we want to answer the first research question: Could we use a pseudo-phoneme encoder to improve the language modeling performance compared to the baseline?

To answer this question, we first need to construct a pseudo-phoneme encoder consisting of a frame-level representation learning method and a segment-level clustering method. Then we train the language models with pseudo-phoneme sequences generated by our encoder.

To produce phone-like units, the pseudo-phoneme encoder should fulfill two requirements: (1) Reliable boundary labels are obtained to divide speech into segments (2) Segments are discriminative to each other when they belong to different phonemes/phonemes sequences. As the CPC representations are shown to be discriminative to each other

when belonging to different phonemes and close to each other when belonging to the same ones (proved by very low ABX scores), the second requirement is satisfied while the first is satisfied. Besides, In our project, we define reliable boundary labels as at least three of the five phone segmentation scores calculated from the labels, are higher than scores calculated by unit label boundaries in the baseline system. The pseudo-phoneme encoder with reliable boundary labels is regarded as qualified and can be considered for language modeling tasks.

The goal of the phoneme encoder is to generate phoneme sequences with which language models can differentiate words and learn high-level linguistic knowledge. Thus, to find the best encoder, we set up the BERT-small language model, to investigate different settings of an encoder and choose one of the highest language modeling performances as our final pseudo-phoneme encoder. Then, as with different settings of the encoder, the generated pseudo-phoneme sequences can differ in size. This means the parameters of the language models might need to be modified to get the best performance. Thus we also adapted the language models to fit the low-budget criteria, to make our model compared to the baseline.

In this experiment series, we compare boundary detection and language modeling performance and for different settings of constructing the pseudo-phoneme encoder, to select the best one. In addition, as acoustic level, ABX evaluation metric requires equal intervals between each unit, yet our pseudo-phoneme sequences are of a variable interval, so we only evaluate our system in the lexical and syntactic levels. Besides, during the selection of the best pseudo-phoneme encoder, we only use lexical level evaluation as a reference while selecting the best phoneme encoder, because the evaluation of syntactic is time-consumption (at least 24 hours for each model).

5.3.1. SETTINGS

The pseudo-phoneme encoder investigated in the next section consists of the pretrained CPC-big followed by a segment-level clustering method. The segment-level clustering method includes a boundary detection CPC, trained by 100 hours of clean librispeech data, with the same hyperparameters in [7], rVAD method (The hyperparameter of rVAD is chosen to be 0.4), and a Kmeans clustering.

A series of different settings of the pseudo-phoneme encoder is explored to find the best phoneme encoder. Specifically, (1) if a lower handcrafted prominence rather than the default one is used (2) if a post-vad method is used are explored. The best encoder is chosen when (1) the phone segmentation scores of the encoder are more reliable than scores calculated by the unit boundary of the baseline. (2) the lexical performance is better. To keep all models comparable to the baseline, kmeans-50 is chosen for all models. Tolerance of 25ms is used to obtain the phone segmentation score.'

After we selected the best encoder, we adapt the language models for the best encoder to obtain the best language modeling performance. We finally chose the BERT-middle architecture with a smaller batch of masked tokens at 160k, more training epochs to form the final language modeling system compare to the baseline. The detailed parameters of the BERT-middle are in the following table compared to the standard low-budget language model BERT-small. The only difference is we add up one more layer for training.

Language Models	L	HD	ED	FFD	Number of Parameters
BERT-small	8	512	512	2048	28
BERT-middel	9	512	512	2048	31.5

Table 5.2: Parameters comparison of the language models used in the baseline and our pseudo-phoneme encoder based system

5.3.2. RESULTS

In table 5.3, to select the best encoder as our pseudo-phoneme encoder, and our encoder performs better on precision, R-value and limited-precision. The F1 scores of the two models are very close.

Method	Precision	Recall	F1	R-value	LPrecision	Lexical score
Baseline	0.60	**0.96**	**0.74**	0.40	0.30	0.61
Boundary CPC+vad	**0.71**	0.74	0.73	**0.76**	**0.58**	0.52
Boudary CPC+vad (High recall, Low Prominence)	0.61	0.82	0.70	0.64	0.49	0.560
Boudary CPC+vad+postvad (High recall,Low Prominence)	0.63	0.83	0.71	0.64	0.49	0.579

Table 5.3: Boundary detection and acoustic level performance with Tolerance 25ms

In table 5.3, we can see that compared to baseline, all three encoders perform better on precision, limited precision, and R-value, and the encoder with default prominence 0.09 and vad method performs the best among three encoders. The best encoder, however, for language modeling is chosen to be BoundaryCPC+vad+postvad, with a prominence at 0.02 who gives the best lexical scores among all other encoders. Compared to all other three encoders, the best encoder performs better on recall scores.

Method	Lexical	Syntactic	GPU Budget	Practical GPU cost
Baseline	0.61	0.52	60	/
pseudo-phoneme encoder+BERT-small	0.579	0.51	60	fewer than 20
pseudo-phoneme encoder+BERT-middle	**0.620**	**0.54**	>60	47

Table 5.4: A comparison of language models with different clustered unit encoders and different LM scale on 960 hours training data

In ZeroSpeech 2021 Challenge, the GPU budget of Language models, which is the required highest training hours for language models, is used to determine if two models can be compared to each other. When we train the language model with BERT-small, which has a theoretical GPU budget of 60 in ZeroSpeech 2021, we actually use fewer than 20 hours to get our model converged. This model gives a lower lexical and syntactic performance than the baseline. Thus we investigated other size and parameters of the BERT model as language models and finally chose BERT-middle and trained it for 60 hours (converged at around 47 hours) for comparison. In this model, the lexical and syntactical levels are shown to be better than the ZeroSpeech baseline, with 1.6%

Methods	dimension	Traning data	frame-interval	Training cost
CPC-small	256	LibriSpeech clean-100h	10ms	60h 1GPU
CPC-big	512	Libri-Light clean-6kh	10ms	3d 8GPU
wav2vec2 base-960	512	LibriSpeech 960h	20ms	1.6d 64GPU

Table 5.5: The difference in parameters of the three representation learning method

System	Pipeline for Language Modelling from Raw Speech
Chance	random guess results for each evaluation
CPC-small	CPC-small+Kmeans50+LSTM
Baseline	CPC-big+Kmeans50+LSTM
wav2vec2+k50	wav2vec2+Kmeans50+LSTM

Table 5.6: Pipeline composition for three language modeling system in Experiments for RQ1

improvement in the lexical level and 3.8% improvement in the syntactic level.

5.4. EXPERIMENTS FOR REPLACING CPC WITH WAV2VEC2

In this set of experiments, we want to answer the second research question: Will applying a transformer-based different speech representation method (wav2vec 2.0) in the ZeroSpeech 2021 baseline system bring better language modeling performance in acoustic, lexical, and syntactic levels?

We train the language models by replacing the CPC with the pretrained wav2vec base-960 model and compare it to the baseline with the language modeling performance at the three linguistic levels.

5.4.1. SETTINGS

In this experiment, we use the pretrained wav2vec2 base-960 model as our representation learning. As shown in table 5.5, training a wav2vec2 is very time-consuming, while the wav2vec2 base-960 model was trained for 1.6 days on 64 V100 GPU. Thus in our system, we only use the pretrained model rather than training from scratch. However, this wav2vec2 model is trained with 960 hours of speech, while the baseline CPC is trained with 6k hours of speech. Considering the amount of dataset that may impact the performance, we also compare the language modeling system trained with CPC-small (trained with 100 hours speech) [3] to the wav2vec2 model. The default frame-interval is 20ms for the wav2vec2 model and we preserve the default settings.

In total, we compare the language modeling performance among the following four systems in 5.6. The chance model is to indicate if the trained language model performs better than a random guess for each linguistic level. The last three models differ only in the representation learning method.

5.4.2. RESULTS

As shown in table 5.7, in the acoustic level (tested in clean/other dataset) and the lexical level, wav2vec2 model-based LM performs worse than the baseline and the CPC-small.

System	Acoustic Scores	Lexical Scores	Syntactic Scores
Chance	0.49/0/50	0.5	0.5
CPC-small	0.10/0.14	0.55	0.51
Baseline	**0.06/0.09**	**0.61**	0.52
Wav2vec2	0.31/0.32	0.52	0.494

Table 5.7: Language modeling performance with our wav2vec2 model and other models)

Besides, according to [3], only when the performance of LMs is better than the chance model can we describe an LM as available in modeling at a certain linguistic level. For the wav2vec2 model, however, in this set of experiments, only acoustic and lexical performance are better than chance models.

5.5. EXPERIMENTS FOR CPC WITH SPEAKER-INVARIANT TECHNIQUE

In this set of experiments, we want to answer the question: Will applying a post speaker invariant method (FHVAE) to the CPC representation learning method bring better acoustic level performance?

In this experiment, we want to use FHVAE to remove the speaker information in the CPC representations and we are interested in how this could improve the CPC representations for acoustic performance. There are two sub-questions for this question. (1) Can FHVAE be applied after CPC eliminates the speaker information in CPC representations? (2) Can FHVAE applied after CPC achieve better acoustic language modeling performance?

We first trained the FHVAE with 512-dimension CPC representations as input to generate 280-d speaker-invariant speech representations as output, following the settings and parameters in the original paper of FHVAE [8]. Then we train two speaker-classification models with speaker-invariant speech representations and CPC representations respectively. As speaker-invariant speech representations is targeted as an improved version of CPC representations to remove speaker information. We mainly compare the acoustic level to see if we could improve the CPC representations by this speaker-invariant technique.

5.5.1. SETTINGS

In the experiment of developing speaker invariant CPC representations, we have two sub-questions. For the first sub-question, we train two softmax-regression classifiers for speaker classification with ZeroSpeech baseline CPC and speaker invariant CPC representations, respectively, to see if the accuracy of the speaker could decrease in the latter case. For the second sub-questions, we use the same baseline model in the wav2vec2 experiment and see how LSTM with speaker invariant CPC representations and kmeans-50 perform on language modeling evaluations. The speaker invariant CPC representation is obtained by training an FHVAE, with hyperparameters same in . with ZeroSpeech baseline CPC representations of 100 hours Librispeech clean data.

The training data consists of 100 hours of randomly selected data from Librispeech-

Representation Learning Method	Speaker Classification Accuracy	Acoustic Scores
CPC-big	81%	**0.06/0.09**
CPC-big + FHVAE	**26%**	0.19/0.21

Table 5.8: Speaker classification accuracy and acoustic level performance for speaker invariant CPC representation methods, compared to CPC

960 with 297 speakers. they are first processed to CPC representations and then fed to FHVAE along with its speaker labels.

5.5.2. RESULTS

In this experiment set, we try to obtain speaker-invariant representations by CPC + FHVAE based model to see if it could improve the acoustic performance of the CPC model.

In table 5.8, for CPC + FHVAE representations, the speaker recognition accuracy is 26% compared to 81% in ZeroSpeechCPC. It means applying FHVAE successfully reduces the speaker information. However, by comparing the acoustic scores, we can see that this speaker-invariant representation performs much worse at the acoustic level compared to the CPC models.

5.6. EXPERIMENTS FOR TRANSFERABILITY OF THE PSEUDO-PHONEME ENCODER

In this set of experiments, we want to answer the fourth research question: Is our pseudo-phoneme encoding approach transferable to other true zero-resource languages (tested by Mboshi)?

In our project, we expect the pseudo-phoneme encoder can produce units that (1) the boundary indicates by pseudo-phoneme units is closer to true boundaries (2) pseudo-phoneme units can be used for language models learning lexical and other high-level linguistic knowledge.

As we do not have language modeling evaluation datasets for Mboshi, we only focus on the transferability of the first characteristics.

In this experiment, we adapted the pseudo-phoneme encoder to Mboshi and compared our boundary detection performance to other research working on the same task and the same dataset.

5.6.1. SETTINGS

To adapt the pseudo-phoneme encoder, we follow the following procedure:

- Retrain the ZeroSpeech baseline CPC-big with Mboshi for 15 epochs
- Train boundaryCPC+vadp encoder with Mboshi from scratch to get boundary labels
- Use boundary labels for segmentation of retrained CPC representations
- Kmeans clustering and get boundary labels indicated by clustered unit sequences

- Calculate the boundary detection performance metrics

We use 4k utterances raw Mboshi audio for training data and 0.5k utterances for testing (golden standard). We trained our boundary detection CPC and kmeans-50 clustering with Mboshi training data with 4k training utterances and retrained the CPC representation learning module with Mboshi training data. We did not train ZeroSpeech baseline CPC from scratch with Mboshi because in practice we found this model requires a large amount of data to achieve fairly good modeling performance while the boundary detection CPC does not. The boundary detection performance was evaluated by the 0.5k utterances dataset.

We used an F1 score with a tolerance of 20ms, according to the research work we compared with. Besides, two settings of the pseudo-phoneme encoder are trained and tested, one with the VAD method and another without the VAD method. This is because VAD is mainly to improve the Librispeech dataset, so we tested two settings to see if VAD is still required for the Mboshi dataset.

5.6.2. RESULTS

Method	F1 score
S Feng,et al.[9]	**62.90 ± 0.15**
Yusuf et al. [10]	59.15 ± 1.51
Ondel et al. [11]	59.50 ± 0.78
Our pseudo-phoneme encoder	58.91
Our pseudo-phoneme encoder (remove post-VAD)	59.04

Table 5.9: Boundary Detection Performance of the Two Pseudo-Phoneme Encoders

In table 5.9, the first three rows show boundary detection performance with an F1 score in other research. The last two rows are the performance of our pseudo-phoneme encoders. The best performance is achieved by Feng [9] in the first row but we can see that our results are comparable to other research. Besides, removing post-VAD methods in our pseudo-phoneme encoder leads to a slight performance improvement of 0.1%.

REFERENCES

[1] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[2] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.-E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen *et al.*, "Libri-light: A benchmark for asr with limited or no supervision," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7669–7673.

[3] T. A. Nguyen, M. de Seyssel, P. Rozé, M. Rivière, E. Kharitonov, A. Baevski, E. Dunbar, and E. Dupoux, "The zero resource speech benchmark 2021: Metrics and baselines for unsupervised spoken language modeling," *arXiv preprint arXiv:2011.11588*, 2020.

[4] A. Rialland, M. Adda-Decker, G.-N. Kouarata, G. Adda, L. Besacier, L. Lamel, E. Gauthier, P. Godard, and J. Cooper-Leavitt, "Parallel corpora in mboshi (bantu c25, congo-brazzaville)," in *11th edition of the Language Resources and Evaluation Conference (LREC 2018)*, 2018.

[5] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," *arXiv preprint arXiv:1904.01038*, 2019.

[6] 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

[7] F. Kreuk, J. Keshet, and Y. Adi, "Self-supervised contrastive learning for unsupervised phoneme segmentation," *arXiv preprint arXiv:2007.13465*, 2020.

[8] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," *arXiv preprint arXiv:1709.07902*, 2017.

[9] S. Feng, P. Żelasko, L. Moro-Velázquez, and O. Scharenborg, "Unsupervised acoustic unit discovery by leveraging a language-independent subword discriminative feature representation," *arXiv preprint arXiv:2104.00994*, 2021.

[10] B. Yusuf, L. Ondel, L. Burget, J. Černockỳ, and M. Saraçlar, "A hierarchical subspace model for language-attuned acoustic unit discovery," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3710–3714.

[11] L. Ondel, H. K. Vydana, L. Burget, and J. Černockỳ, "Bayesian subspace hidden markov model for acoustic unit discovery," *arXiv preprint arXiv:1904.03876*, 2019.

6

DISCUSSION AND CONCLUSION

In our project, we built a language modeling system from raw speech with a pseudo-phoneme encoder, investigated two potential replacements we proposed for the CPC representation learning method in the low-budget ZeroSpeech 2021 baseline system, and explored transferability on Mboshi for our pseudo-phoneme encoder to answer the following research questions (RQ):

1. Could we use the pseudo-phoneme encoder to improve the language modeling performance?
2. Will applying a transformer-based self-supervised speech representation method (wav2vec 2.0) in the ZeroSpeech 2021 baseline system brings better language modeling performance?
3. Will applying a post speaker invariant method (FHVAE) to the CPC representation learning method improve the performance at the acoustic level?
4. Transferability: Is our pseudo-phoneme encoding approach transferable to other true zero-resource languages (tested by Mboshi)?

In the last chapter, we see that for RQ1 we achieved a better performance than the low-budget ZeroSpeech 2021 baseline at the lexical and syntactic levels by using the selected best pseudo-phoneme encoder. For RQ2 and RQ3, the result shows that both wav2vec2 and speaker-invariant CPC did not perform better than CPC for language modeling. Moreover, for RQ4 the results show that pseudo-phonemes of Mboshi give comparable boundary recognition scores with other research thus we can conclude the pseudo-phoneme encoder is transferable to Mboshi. An discussion for the results will be shown in the next section.

6.1. DISCUSSION

1. **Could we use the pseudo-phoneme encoder to improve the language modeling performance?**

In experiments we applied a pseudo-phoneme encoder, considering the practical training time fit the low-budget requirements, our pseudo-phoneme encoder-based language modeling system achieved better performance than the ZeroSpeech baseline at the lexical and syntactic level. Our assumption is that words represented by pseudo-text from the baseline differ from each other in both linguistic units and time of pronunciation, and the same words in pseudo-text could have different forms. This increased difficulties of language modeling. The segmentation step in our system may help to resolve this problem by the group of the frames that belong to the possibly same phoneme. However, though our goal is to discovering phoneme-like units, we do not directly evaluate how our pseudo-phoneme sequences are similar to the true phonetic transcription but only rely on the language modeling performance to indirectly judge the performance of the pseudo-phoneme encoder.
Future work could be exploring better metrics and loss functions to train an end-to-end pseudo-phoneme encoder and evaluate the similarity between pseudo-phoneme sequences and phonetic transcription.

2. **Will applying a transformer-based self-supervised speech representation method (wav2vec 2.0) in the ZeroSpeech 2021 baseline system brings better language modeling performance?**

In experiments to apply wav2vec2 in spoken language modeling, the acoustic-level evaluation is much worse than ZeroSpeech CPC. However, wav2vec2 is shown in [1] to perform well in predicting word transcription in low resource scenarios (fine-tuned with 10 minutes labeled data). There might be several reasons for the bad performance.

   (a) The quantized wav2vec2 representation is not suitable for distance-based clustering. The loss function of wav2vec2 encourages current quantized frame representation to be distinguished enough from other distant time steps, but it does not constrain nearby quantized frame representation to be close to the current frames. If the quantized representations are lying in the data space sparsely, distance-based Kmeans clustering may find it difficult to cluster those representations.

   (b) The quantized representation is generated by the encoder of wav2vec rather than a contextual representation from transformer layers. We chose the quantized representation for clustering because authors of wav2vec2 use it for downstream ASR tasks. However, considering that CPC in the baseline used contextual representations for clustering, it is possible that contextual representations are more suitable to be fed into unsupervised clustering.

   (c) The wav2vec2 pre-trained model we used is not the best model for ASR performance in a low-resource scenario.

Future work involved with wav2vec2 architecture could consider training it from scratch with the same loss function with ZeroSpeech CPC or exploring representations from a series of transformer layers in wav2vec2. Also, representations of the best wav2vec model (wav2vec2-XL) could be explored.

3. **Will applying a post speaker invariant method (FHVAE) to the CPC representation learning method improve the performance at the acoustic level?**

   In experiments to apply FHVAE to construct speaker-invariant representations, the speaker information is proved to be reduced than ZeroSpeech CPC representations so FHVAE is successful to remove part of speaker information. However, the speaker-invariant representations do not achieve better acoustic level performance. The decreased performance of the speaker-invariant representation at acoustic level might be due to the the dimension reduction process of FHVAE removing too much phonetic information in the representations. According to the original paper of FHVAE [2], whose input is 200 dimensional speech features and the output is 32-d features. From this paper, we set a similar proportion of input/output size which prune the CPC representations from 768-d to 280-d representations. However, the ABX score is very high for this 280-d representation which indicates, a lot of wrong discrimination happens. This might because in this pruning process, some information important for spoken language modeling have been lost. **Future researchers** who plans to modify CPC representations might need be cautious in reduce representation dimensions.

4. **Transferability: Is our pseudo-phoneme encoding approach transferable to other true zero-resource languages (tested by Mboshi)?**

   In transferability experiments, the results show that the pseudo-phonemes encoder of Mboshi give comparable boundary recognition scores with other research which indicates our encoder approach could be transferable to other languages. We also see that additional processing steps such as removing post-VAD could further improve the performance, which suggests rVAD method is less efficient in Mboshi that in English Librispeech dataset.

6.2. CONCLUSIONS

In this thesis project, we proposed a new pipeline method for language modeling from raw speech. Specifically, we developed a pseudo-phoneme encoder and applied it to modify the low-budget ZeroSpeech 2021 baseline. The system was evaluated in acoustic, lexical, and syntactic levels and achieves better performance than the low-budget ZeroSpeech 2021 baseline at lexical and syntactic levels. Besides, we investigated two speech representation learning methods, wav2vec2 and CPC with a speaker-invariant approach (FHVAE), and the result shows that both did not perform better than CPC for language modeling at the three linguistic levels. Moreover, we investigated the transferability of our pseudo-phoneme encoder in other languages (tested by Mboshi). The results show that the pseudo-phonemes of Mboshi give a comparable boundary recognition score with other research which indicates our encoder approach can be transferable to other languages.

REFERENCES

[1] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*,

2020.

[2] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," *arXiv preprint arXiv:1709.07902*, 2017.

7

APPENDIX

REFERENCES

[1] K. R. Fahey, L. M. Hulit, and M. R. Howard, "Born to talk," 2019.

[2] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black *et al.*, "The zero resource speech challenge 2019: Tts without t," *arXiv preprint arXiv:1904.11469*, 2019.

[3] M. A. Hedderich, L. Lange, H. Adel, J. Strötgen, and D. Klakow, "A survey on recent approaches for natural language processing in low-resource scenarios," *arXiv preprint arXiv:2010.12309*, 2020.

[4] T. A. Nguyen, M. de Seyssel, P. Rozé, M. Rivière, E. Kharitonov, A. Baevski, E. Dunbar, and E. Dupoux, "The zero resource speech benchmark 2021: Metrics and baselines for unsupervised spoken language modeling," *arXiv preprint arXiv:2011.11588*, 2020.

[5] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] "Result page for ZeroSpeech 2021," https://zerospeech.com/2021/results.html, 2021, [Online; accessed September-2021].

[9] H. Kuwabara, "Acoustic properties of phonemes in continuous speech for different speaking rate," in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, vol. 4. IEEE, 1996, pp. 2435–2438.

[10] B. Wu, S. Sakti, J. Zhang, and S. Nakamura, "Optimizing dpgmm clustering in zero resource setting based on functional load." in *SLTU*, 2018, pp. 1–5.

[11] S. Feng and T. Lee, "Exploiting cross-lingual speaker and phonetic diversity for unsupervised subword modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2000–2011, 2019.

[12] S. Feng, P. Żelasko, L. Moro-Velázquez, and O. Scharenborg, "Unsupervised acoustic unit discovery by leveraging a language-independent subword discriminative feature representation," *arXiv preprint arXiv:2104.00994*, 2021.

[13] L. van Staden and H. Kamper, "A comparison of self-supervised speech representations as input features for unsupervised acoustic word embeddings," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 927–934.

[14] ——, "Improving unsupervised acoustic word embeddings using speaker and gender information," in *2020 International SAUPEC/RobMech/PRASA Conference*. IEEE, 2020, pp. 1–6.

[15] Y. Miao, H. Zhang, and F. Metze, "Speaker adaptive training of deep neural network acoustic models using i-vectors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 11, pp. 1938–1949, 2015.

[16] S. Feng, "Exploiting cross-lingual knowledge in unsupervised acoustic modeling for low-resource languages," *arXiv preprint arXiv:2007.15074*, 2020.

[17] W.-N. Hsu and J. Glass, "Scalable factorized hierarchical variational autoencoder training," *arXiv preprint arXiv:1804.03201*, 2018.

[18] M. Versteegh, X. Anguera, A. Jansen, and E. Dupoux, "The zero resource speech challenge 2015: Proposed approaches and results," *Procedia Computer Science*, vol. 81, pp. 67–72, 2016.

[19] E. Dunbar, X. N. Cao, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera, and E. Dupoux, "The zero resource speech challenge 2017," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 323–330.

[20] E. Dunbar, J. Karadayi, M. Bernard, X.-N. Cao, R. Algayres, L. Ondel, L. Besacier, S. Sakti, and E. Dupoux, "The zero resource speech challenge 2020: Discovering discrete subword and word units," *arXiv preprint arXiv:2010.05967*, 2020.

[21] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[22] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[23] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," *arXiv preprint arXiv:1911.03912*, 2019.

[24] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.

[25] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. v. d. Oord, "Learning robust and multilingual speech representations," *arXiv preprint arXiv:2001.11128*, 2020.

[26] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3497–3501.

[27] A. Rialland, M. Adda-Decker, G.-N. Kouarata, G. Adda, L. Besacier, L. Lamel, E. Gauthier, P. Godard, and J. Cooper-Leavitt, "Parallel corpora in mboshi (bantu c25, congo-brazzaville)," in *11th edition of the Language Resources and Evaluation Conference (LREC 2018)*, 2018.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[29] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.-E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen *et al.*, "Libri-light: A benchmark for asr with limited or no supervision," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7669–7673.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[31] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[32] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Qadir, and B. W. Schuller, "Deep representation learning in speech processing: Challenges, recent advances, and future trends," *arXiv preprint arXiv:2001.00378*, 2020.

[33] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[34] R. Togneri, M. Alder, and Y. Attikiouzel, "Dimension and structure of the speech space," *IEE Proceedings I (Communications, Speech and Vision)*, vol. 139, no. 2, pp. 123–127, 1992.

[35] S. S. A. Zaidi, M. M. Ali, F. Aftab, Y. Shahid, and M. Khurram, "Name spotting over low signal-to-noise ratio (snr) using blind source separation and connectionist temporal classification," in *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*. IEEE, 2017, pp. 320–325.

[36] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[37] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1476–1485.

[38] W. Wang, Q. Tang, and K. Livescu, "Unsupervised pre-training of bidirectional speech encoders via masked reconstruction," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6889–6893.

[39] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[40] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[41] J. Konečnỳ, J. Liu, P. Richtárik, and M. Takáč, "Mini-batch semi-stochastic gradient descent in the proximal setting," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 242–255, 2015.

[42] L. K. Chan, N. Jegadeesh, and J. Lakonishok, "Momentum strategies," *The Journal of Finance*, vol. 51, no. 5, pp. 1681–1713, 1996.

[43] T. Tieleman, G. Hinton *et al.*, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[44] A. Lydia and S. Francis, "Adagrad—an optimizer for stochastic gradient descent," *Int. J. Inf. Comput. Sci*, vol. 6, no. 5, 2019.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[46] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[47] H. Lee, P. Pham, Y. Largman, and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," *Advances in neural information processing systems*, vol. 22, pp. 1096–1104, 2009.

[48] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. Li, "Reconstruction of porous media from extremely limited information using conditional generative adversarial networks," *Physical Review E*, vol. 100, no. 3, p. 033308, 2019.

[49] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[50] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[52] W. James, *The principles of psychology*. Cosimo, Inc., 2007, vol. 1.

[53] P. Bloem, "Transformers from scratch," http://www.peterbloem.nl/blog/transformers, 2021, [Online; accessed September-2021].

[54] J. Daniel and H. M. James, "Speech and language processing," 2000.

[55] J. E. Shoup, "Phonological aspects of speech recognition," *Trends in speech recognition*, pp. 125–138, 1980.

[56] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[57] Y. LeCun, "Self-supervised learning: could machines learn like humans?" https://www.youtube.com/watch?v=7I0Qt7GALVk, 2021, [Online; accessed September-2021].

[58] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," *arXiv preprint arXiv:1709.07902*, 2017.

[59] F. Kreuk, J. Keshet, and Y. Adi, "Self-supervised contrastive learning for unsupervised phoneme segmentation," *arXiv preprint arXiv:2007.13465*, 2020.

[60] S. Bhati, J. Villalba, P. Żelasko, L. Moro-Velazquez, and N. Dehak, "Segmental contrastive predictive coding for unsupervised word segmentation," *arXiv preprint arXiv:2106.02170*, 2021.

[61] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[62] Z.-H. Tan, N. Dehak *et al.*, "rvad: An unsupervised segment-based robust voice activity detection method," *Computer speech & language*, vol. 59, pp. 1–21, 2020.

[63] H. Hirsch, "The aurora experimental framework for the performance evaluations of speech recognition systems under noisy conditions, isca itrw asr2000," *Automatic Speech Recognition: Challenges for the Next Millennium*, 2000.

[64] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," *arXiv preprint arXiv:1611.06639*, 2016.

[65] I. Tenney, D. Das, and E. Pavlick, "Bert rediscovers the classical nlp pipeline," *arXiv preprint arXiv:1905.05950*, 2019.

[66] A. Ezen-Can, "A comparison of lstm and bert for small corpus," *arXiv preprint arXiv:2009.05451*, 2020.

[67] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[68] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," *arXiv preprint arXiv:1904.01038*, 2019.

[69] 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

[70] B. Yusuf, L. Ondel, L. Burget, J. Černockỳ, and M. Saraçlar, "A hierarchical subspace model for language-attuned acoustic unit discovery," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3710–3714.

[71] L. Ondel, H. K. Vydana, L. Burget, and J. Černockỳ, "Bayesian subspace hidden markov model for acoustic unit discovery," *arXiv preprint arXiv:1904.03876*, 2019.

[72] C. Chakraborty and P. Talukdar, "Issues and limitations of hmm in speech processing: a survey," *International Journal of Computer Applications*, vol. 141, no. 7, pp. 13–17, 2016.

[73] M. Löffler, A. Y. Zhang, and H. H. Zhou, "Optimality of spectral clustering in the gaussian mixture model," *arXiv preprint arXiv:1911.00538*, 2019.

[74] P. Godard, G. Adda, M. Adda-Decker, J. Benjumea, L. Besacier, J. Cooper-Leavitt, G.-N. Kouarata, L. Lamel, H. Maynard, M. Müller *et al.*, "A very low resource language speech corpus for computational language documentation experiments," *arXiv preprint arXiv:1710.03501*, 2017.

[75] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8. Citeseer, 2015, pp. 18–25.

[76] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[77] G. Le Godais, T. Linzen, and E. Dupoux, "Comparing character-level neural language models using a lexical decision task," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, pp. 125–130.

[78] A. Warstadt, A. Parrish, H. Liu, A. Mohananey, W. Peng, S.-F. Wang, and S. R. Bowman, "Blimp: The benchmark of linguistic minimal pairs for english," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 377–392, 2020.

[79] Y.-A. Chung and J. Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," *arXiv preprint arXiv:1803.08976*, 2018.

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[81] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition." in *Interspeech*, vol. 11. Citeseer, 2013, pp. 73–5.

[82] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[83] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A.-r. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 2011, pp. 30–35.

[84] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[85] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962*, 2019.