Self-Supervised Learning for Speech Processing

by

Yu-An Chung

B.S., National Taiwan University (2016)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....
Department of Electrical Engineering and Computer Science
May 13, 2022

Certified by .....
James R. Glass
Senior Research Scientist, CSAIL
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Self-Supervised Learning for Speech Processing

by

Yu-An Chung

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Deep neural networks trained with supervised learning algorithms on large amounts of labeled speech data have achieved remarkable performance on various spoken language processing applications, often being the state of the arts on the corresponding leaderboards. However, the fact that training these systems relies on large amounts of annotated speech poses a scalability bottleneck for the continued advancement of state-of-the-art performance, and an even more fundamental barrier for deployment of deep neural networks in speech domains where labeled data are intrinsically rare, costly, or time-consuming to collect.

In contrast to annotated speech, untranscribed audio is often much cheaper to accumulate. In this thesis, we explore the use of self-supervised learning—a learning paradigm where the learning target is generated from the input itself—for leveraging such easily scalable resources to improve the performance of spoken language technology. Specifically, we propose two self-supervised algorithms, one based on the idea of "future prediction" and the other based on the idea of "predicting the masked from the unmasked," for learning contextualized speech representations from unlabeled speech data. We show that our self-supervised algorithms are capable of learning representations that transform high-level properties of speech signals such as their phonetic contents and speaker characteristics into a more accessible form than traditional acoustic features, and demonstrate their effectiveness in improving the performance of deep neural networks on a wide range of speech processing tasks. In addition to presenting new learning algorithms, we also provide extensive analysis aiming to understand the properties of the learned self-supervised representations, as well as disclosing the design factors that make one self-supervised model different from the other.

Thesis Supervisor: James R. Glass
Title: Senior Research Scientist, CSAIL

Acknowledgments

Ten years ago when I was just about to start as a freshman at National Taiwan University, I knew nothing about Computer Science and thought the subject was about teaching me how to assemble computers. At that time I would say you must be out of your mind if you told me that I would actually receive a PhD degree in Computer Science ten years later. Now it is hard to believe that not only the statement has come true, I am even receiving this degree from MIT, one of the best institutes for Computer Science in the world.

The past five years at MIT are undoubtedly amazing and unforgettable. Throughout my PhD journey, I have received helps from many people. First and foremost, I would like to express my enormous appreciation to my doctoral advisor, Jim Glass. I have always considered Jim as the type of advisors that I really could not ask more from. As an advisor, Jim allows immense research freedom, letting me make my own plans and explore whatever topics that interest me. He also has great knowledge and vision in speech and language, which often guides me to discover new research directions. As a leader, Jim maintains good atmosphere within the group where members are not just collaborators but also friends. He also spends lots of his time improving and maintaining our computational resources and systems, doing his best to support his students and post-docs with the best research environment. As a person, Jim is considerate and always takes good care of everyone in the group. Among all the people I knew, Jim has the best communication skills. He has the ability to speak with people softly while still making his points accurately and standing on his opinions strongly, and that is one of the many things I am still trying to learn from him. All in all, I feel extremely fortunate to have Jim as my advisor. Because of him, I had a very pleasant and smooth PhD journey.

I am also indebted to my thesis committee members, Jacob Andreas and Phillip Isola. Their insights and careful examination on my research outcome have helped me improve this thesis. I would like to express a special thanks to Jacob, who also served on my RQE committee and let me TA his Natural Language Processing class

in 2020 Spring with Jim.

I am extremely thankful to my undergrad research advisors, Hsuan-Tien Lin, Lin-Shan Lee, and Hung-Yi Lee. Hsuan-Tien led me into the field of machine learning and sparked my interests in doing research, which obviously made a huge impact on my career and life. I still clearly remember the excitement when he told me that I was ready to write my very first paper in 2015, which was later accepted by IJCAI. Lin-Shan introduced me to the field of speech processing, and encouraged me to pursue a PhD degree after I finished my undergrad study. His strong connections with MIT along with his great endorsement in my letter were some of the key factors that helped me get accepted by MIT to fulfill my dream. Hung-Yi is an expert in machine learning's applications to speech and language processing. During our discussions I always learned a lot about the state-of-the-art methods as well as hands-on experience on implementing them. Joining their respective groups was one of the best decisions I ever made in my life.

I was very fortunate to have the opportunities to work with many great researchers throughout my PhD journey, from both inside and outside of MIT. Inside MIT, I want to thank Hao, Wei-Ning, and Yonatan for collaborations that are directly related to this thesis. I admire Hao's knowledge in speech, as well as his rigorousness and dedication in research; I covet Wei-Ning's genius in coming up with mathematical solutions and novel algorithms for machine learning problems; I yearn for Yonatan's expertise and experience in natural language processing. They are all my role models and I hope one day I can be as good as a researcher as them. I also want to thank Alex, Yuan, and Jeff for letting me participate in their interesting projects and publish papers together. Outside of MIT, I want to thank all my past internship hosts: Yuxuan Wang (now at ByteDance), Yu Zhang, Peter J. Liu, and Bo Li from Google, and Chenguang Zhu from Microsoft. I truly enjoyed every moment during all my internships, from discussing the research projects to sharing the things that happened in our daily lives. I want to express my special gratitude to Yuxuan and Yu, who hosted my first internship. It was because of my interactions with them during that internship that made me decide to pursue a career as a research scientist in the

industry after receiving my PhD. I still remember during the first week of my first internship, I had a hard time understanding the huge and complicated codebase of Google, feeling frustrated and exhausted. It was Yuxuan who sat down and patiently walked me through the codebase, taught me new tools, and even led me to read through some files almost line-by-line. Because of his kindness and patience, I was able to start making contributions to the team on the second week, which greatly boosted my confidence. My mindset has grown since then, and I am no longer afraid of new environments and unseen technologies, which is a very important ability for my career.

I am also very grateful to many friends that I made during the journey. Thank you to my labmates Hao, Wei-Ning, Di-Chia, Mandy, François, Suwon, Tainxing, Yonatan, Dave, Sameer, Wei, Hongyin, and all the other SLSers who I had overlaps with. The countless times when we had lunch together sharing our lives were invaluable memories to me, and I hope we can have a reunion in the future (maybe in one of the future conferences?) to resonate the good old times. Also, a huge thank you to Marcia for making the administrative aspects of my career as a graduate student as painless as possible. Thank you to Wei-Hung, Schrasing, as well as many members from the Taiwanese Students Association at MIT, for making my PhD life more colorful and letting me feel at home.

Last but not least, I would like to thank my girlfriend Ashley and my family for their endless and unconditional love and support. Thank you to my parents, Chun-Wei and Yu-Hsin, and my older brother, Yu-Hao. It has been extremely difficult for me to live so far away from them, and all I can do is to take good care of myself to not let them worry about me. I cherished every video call we had and enjoyed every photo/video of my cute nephews, Ning-Fan and Mu-Chen, my parents and brother sent me. I feel lucky to be born in this family. Thank you to my girlfriend, Ashley, for being with me during the ups and downs in my life for the past seven years. I am not good at expressing myself, but all I can say to her is that, I am sure there will be more unknown challenges in the future, but I am not afraid of any of them because I know we will face them together.

This work was sponsored in part by the Singapore Defense Science and Technology Agency. I would like to thank them for their generous support.

Bibliographic Note

A large portion of this thesis is based on peer-reviewed publications. The work presented in Chapter 3 was largely published in Chung et al. (2019a) and Chung and Glass (2020a). The content of Chapter 4 first appeared in Chung et al. (2020). Much of the content in Chapter 5 was published in Chung and Glass (2020b). The work presented in Chapter 6 was published in Chung et al. (2021b). The content of Chapter 7 was published in Chung et al. (2021a).

Part of the code in this thesis is available at `https://github.com/iamyuanchung`.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, deep neural networks, or deep learning techniques, empower the state-of-the-art artificial intelligence systems for a wide range of applications across diverse data types—image classification (He et al., 2016; Liu et al., 2022), machine translation (Vaswani et al., 2017), and speech recognition (Gulati et al., 2020) to name a few. However, the conventional paradigm for training these systems has been supervised learning, where performance of the systems has been growing roughly logarithmically with the size of labeled data used for training them (Sun et al., 2017). The cost of acquiring such annotated data has proven to be a scalability bottleneck for the continued development of state-of-the-art systems, and an even more fundamental barrier for deployment of deep neural networks in application areas where data and annotations are intrinsically rare, costly, or time-consuming to collect.

The aforementioned situation has motivated a wave of research in self-supervised representation learning, where freely available labels generated from carefully designed *pretext tasks* are used as the supervision signals to *pre-train* deep neural networks. The parameters from the pre-trained deep neural networks are then entirely or partially used to initialize the parameters of task-specific deep neural networks to solve downstream tasks of interest using comparatively little annotated data compared to conventional supervised learning.

Self-supervision refers to learning tasks that ask deep neural networks to predict one part of the input data (or a label programmatically derivable thereof) given

another part of the input. This is in contrast to supervised learning, which asks the deep neural networks to predict a manually provided target output; and generative modeling, which asks the deep neural networks to estimate the density of the input data or learn a generator for input data. Self-supervised learning algorithms differ primarily in their strategies for defining the derived labels to predict. This choice of pretext task, determines the (in)variances of the resulting learned representations, and thus how effective they are for different downstream tasks.

Self-supervised learning techniques have been successfully leveraged to improve sample efficiency of learning across a variety of modalities, ranging from image (Chen et al., 2020; Grill et al., 2020; Caron et al., 2020), video (Xu et al., 2019; Alwassel et al., 2020), speech and audio (Baevski et al., 2020b; Gong et al., 2022), text (Mikolov et al., 2013; Peters et al., 2018b; Devlin et al., 2019; Liu et al., 2019), to graphs (Velickovic et al., 2019), to name a few. Some results suggest that the quality of self-supervised representations is also a logrithmic function of the amount of unlabeled pre-training data (Goyal et al., 2019). If this trend holds, then achievable performance may improve for "free" over time since improvements in data collection and computational power allow increasingly large pre-training sets to be used without the need for manually annotating new data.

In this thesis, we focus on applying self-supervised learning strategies to the domain of speech, with the goal of pushing the state-of-the-art performance of spoken language technology and improving the data efficiency for training them. We present our efforts in developing new self-supervised speech representation learning methods, as well as analyzing the properties of their learned representations.

1.1 Thesis Contributions

The primary contributions made by this thesis are as follows:

1. **Introduction of one of the earliest successful self-supervised speech representation learning frameworks.** We exploit the idea of "future prediction" and propose a simple yet effective self-supervised objective called Au-

toregressive Predictive Coding (APC) for training deep neural networks. The designed future frame prediction task is able to leverage unlabeled speech data to learn representations that make high-level properties of speech utterances such as their phonetic contents and speaker characteristics more accessible (defined as linear separability) to downstream tasks. APC is one of the earliest works that showed the superiority of self-supervised representations over traditional hand-crafted acoustic features such as Mel-frequency cepstral coefficients (MFCCs) and log Mel spectrograms, indicating the potential of using self-supervised learning for boosting spoken language technology performance.

2. **Introduction of one of the current state-of-the-art self-supervised speech representation learning frameworks.** We exploit the idea of "predicting the masked from the unmasked" and propose w2v-BERT, which is one of the current state-of-the-art frameworks for pre-training very deep neural networks for speech applications. We train a speech discretizer (through optimizing a contrastive loss) for representing continuous speech signals as discriminative tokens, and use them to train a BERT-like model. In contrast to existing frameworks such as vq-wav2vec and HuBERT that also make use of the "predicting the masked from the unmasked" methodology, in w2v-BERT the discretizer and the context network can be optimized in an end-to-end fashion, avoiding the need of coordination between multiple training stages that could often involve brittle modeling choices. We demonstrate the effectiveness of w2v-BERT by showing its superiority over the state of the arts, including HuBERT and wav2vec 2.0, on both a well-benchmarked speech recognition dataset and a Google-collected voice search dataset.

3. **Introduction of an analysis method capable of bridging connections between self-supervised objectives and properties of the representations they learn.** We explore the use of vector quantization for controlling the amount of information flow inside deep neural networks to obtain a spectrum of models trained with the same self-supervised objective but with decreasing

model capacity. We apply this analysis method to study APC, and diagnose the preferences of APC in preserving information while its model capacity becomes constrained. Our analysis results provide an explanation to why APC can learn representations that capture high-level phonetic and speaker information. The analysis method is general and can be applied to analyzing other self-supervised objectives as well.

4. **Demonstration of several shared natures of different self-supervised models.** When analyzing our own and other existing self-supervised models, we find that there exist several properties that most of those models share in common regardless of their differences in training objectives and neural network architectures. One of such properties is the ability of implicit discovery of an inventory of meaningful acoustic units. We find that there usually exist some layers in the self-supervised models where representations have considerably high mutual information with English phones (when the models are trained on an English corpus), even though the models are not explicitly trained towards discovering them. Another properties shared by most self-supervised models is that different levels of speech information are captured in different layers, although the information distribution could vary model to model. For instance, in APC, the lower layers tend to be more discriminative for speakers, while the upper layers provide more phonetic content. Being aware of this insight is useful for selecting proper layers to extract representations from for the best performance on the tasks of interest.

5. **Identification of the order of importance of modeling factors for training self-supervised models that impact their representational similarity.** We compare a collection of self-supervised models with diverse modeling choices during their training, and use measures such as canonical correlation analysis (CCA) to quantify their pairwise similarities. We consider three modeling factors: training objectives, model directionality (i.e., whether the model is unidirectional or bidirectional), and neural network building blocks

(CNN/RNN/Transformer), and show that the three factors have different weights in making one self-supervised representation different from another. Specifically, we find that training objective has the highest impact on representational similarity among all the factors; under the same training objective, a model's directionality affects representational similarity more than its neural network building blocks.

1.2 Chapter Guide

The remaining chapters of this thesis are organized as follows:

- Chapter 2 reviews background materials for topics related to this thesis. The goal of this chapter is to provide readers with sufficient background to understand this thesis.

- Chapter 3 presents a self-supervised objective for speech representation learning based on the idea of "future prediction", and shows its preliminary positive results on phone and speaker classification tasks.

- Chapter 4 analyzes the objective proposed in the previous chapter to draw a connection between the objective and the properties of the representations it learns. The goal of this chapter is to provide an explanation to why the proposed objective is capable of learning good speech representations.

- Chapter 5 proposes an auxiliary objective that, when optimized together with the objective proposed in Chapter 3, improves the generalization of the main objective, which leads to better speech representations. The results of the proposed objective on standard speech applications, including speech recognition and speech-to-text translation, are also presented in this chapter.

- Chapter 6 presents another self-supervised pre-training framework that makes use of the idea of "predicting the masked from the unmasked", and demonstrates

its superiority over existing pre-training frameworks on two speech recognition datasets.

- Chapter 7 compares a collection of self-supervised models with diverse modeling choices during their training, aiming to identify the key factors that impact the representational similarities between self-supervised models.
- Chapter 8 summarizes this thesis and discusses possible future directions.

Chapter 2

Background and Related Work

In this chapter, we provide background materials for several topics that are relevant to this thesis. We start by giving an overview on automatic speech recognition in Section 2.1, which is one of the core applications of speech processing as well as the main task for evaluation the systems in this thesis. Next, in Section 2.2 we provide a review on three types of commonly used neural network architectures for speech processing: LSTM, Transformer, and Conformer. In Section 2.3 we first briefly review the history of neural network pre-training, then discuss its recent breakthroughs in visual, textual, and speech representation learning brought about by self-supervised techniques. Finally, in Section 2.4 we review related work in neural representation analysis, as some methods from the literature will be adopted in this thesis for analyzing self-supervised speech representations.

Note that the review here is by no means a comprehensive one that includes all the existing studies (in fact, not even all the state-of-the-art papers will be mentioned) or mathematical details on the discussed topics, but only aims to provide background knowledge on them to the extent that is sufficient for the readers to understand this thesis.

2.1 Automatic Speech Recognition

The goal of automatic speech recognition (ASR) is to enable machines to automatically transcribe human speech into text. ASR is undoubtedly one of the core applications of speech processing, and often serves as the first step of several other speech processing applications such as speech translation and spoken language understanding. The following sections provide an overview on ASR, starting with its problem formulation.

2.1.1 Problem Formulation

Modern ASR systems are built based on the noisy channel model (Jelinek, 1976). Under such a model, the recognizer perceives a speech utterance X, and its job is to recover the underlying text sequence $\hat{Y}$. By applying Bayes' rule, the inference of the text sequence Y given the speech utterance X can be written as:

$$Y^* = \underset{Y}{\operatorname{argmax}}\, P(Y|X) = \underset{Y}{\operatorname{argmax}}\, \frac{P(X|Y)P(Y)}{P(X)}, \tag{2.1}$$

where Y^* stands for the best guess of $\hat{Y}$ based on the three statistical models $P(X|Y)$, $P(Y)$, and $P(X)$. Since the denominator term $P(X)$ is actually independent from Y and hence does not affect the search of $\hat{Y}$, Equation 2.1 can be simplified as:

$$Y^* = \underset{Y}{\operatorname{argmax}}\, \frac{P(X|Y)P(Y)}{P(X)} = \underset{Y}{\operatorname{argmax}}\, P(X|Y)P(Y). \tag{2.2}$$

In speech literature, Equation 2.2 is often referred to as the "fundamental equation of speech recognition," and the statistical models that are used for estimating $P(X|Y)$ and $P(Y)$ are typically called the *acoustic model* and *language model*, respectively. The ASR problem has thus become finding the optimal parameterizations for the acoustic model and language model.

Acoustic Model For some text sequence Y, an acoustic model estimates $P(X|Y)$ that tells us the likelihood of a speech utterance X given Y. To give a concrete

example, let us consider a speech recording of someone speaking the phrase "how to recognize speech." An ideal acoustic model would then assign a higher likelihood score to $P(X|\text{"how to recognize speech"})$ than to another phrase that sounds completely different from "how to recognize speech" such as the phrase "give me a drink."

Obtaining such a statistical model typically requires a collection of audio recordings along with their parallel text transcripts $\{(X^{(i)}, Y^{(i)})\}_{i=1}^{N}$, where N is the number of audio-transcript pairs in the collection. An audio recording (for simplicity, here we ignore the superscript i that indicates the i-th pair in the collection) is usually represented as a sequence of spectral feature vectors $X = (x_1, x_2, \ldots, x_T)$ such as the log Mel-filterbank features and Mel-frequency cepstral coefficients (MFCCs) (Mermelstein, 1976), where T is the sequence length that varies recording to recording.[1] The text transcript Y is a sequence of textual tokens that—depending on the ASR system's final usage—can be at different levels when training the acoustic models. One of the most common ways to represent Y is to use a handcrafted pronunciation model to map each word in Y to its pronunciation along with its associated probability, where the pronunciation is a sequence of phones. The use of such a pronunciation model allows a more efficient modeling of $P(X|Y)$, since instead of having to estimate a different density for every unique word in the vocabulary set, the acoustic model now only needs to cover a set of elementary acoustic units such as phones, which are merely dozens and shared across words in a language. Let U denote the phone sequence converted from Y using the pronunciation model, the problem of modeling the conditional distribution of an audio sequence X given a text sequence Y, $P(X|Y)$, becomes the problem of modeling the conditional distribution of X given the phone sequence U, $P(X|U)$.

Traditionally, a Hidden Markov Model (HMM) is used to model $P(X|U)$ (Baker, 1975). Each phone owns a dedicated HMM for modeling that usually consists of three hidden states that are meant to capture the transient acoustic dynamics within a phone (one can think of it as attempting to model the beginning, middle, and end

[1] The complete derivation of these spectral features is out of scope of this thesis. For details on the frond-end acoustic feature extraction scheme, please refer to Davis and Mermelstein (1980).

of the phone). These HMMs are then concatenated to represent the entire phone sequence U. The acoustic feature vectors in X are treated as the state emissions, and their densities are usually modeled by either a set of Gaussian Mixture Models (GMM) (Bilmes, 1998) or a deep neural network (DNN) (Mohamed et al., 2011), where the latter usually outperforms the former. For GMM-HMM acoustic models, their parameters can be estimated by maximum likelihood estimation using the forward-backward algorithm (more details can be found in Rabiner and Juang (1993)). To estimate the parameters of a DNN-based acoustic model, it usually still starts with a trained baseline GMM-HMM speech recognizer, which is used to compute the target state label for each frame in the audio sequence (the obtained target state sequence is typically referred to as the forced-alignment). Once the target state sequence is obtained, the DNN-based acoustic model can then be trained using backpropagation with common gradient descent techniques. The usage of different types of neural architectures, such as feed-forward neural networks (Dahl et al., 2011; Seide et al., 2011), recurrent neural networks (Graves et al., 2013; Sak et al., 2014), and convolutional neural networks (Abdel-Hamid et al., 2012), have been explored. During inference time, the maximization (Equation 2.2) is typically solved using a decoding algorithm such as Viterbi search (Viterbi, 1967).

Language Model The statistical model that is used to estimate $P(Y)$ in Equation 2.2 is typically called the language model in the literature, as it provides an *a priori* probability of how likely a text sequence Y is to appear in human language (i.e., to be spoken or written by a person) in the first place. To give a concrete example on how a language model plays its role in speech recognition, let us reuse the example recording of someone speaking the phrase "how to recognize speech," and assume there is another recording of someone speaking the phrase "how to wreck a nice beach." Now, even a strong acoustic model would have trouble assigning the two recordings different likelihood scores $P(X|Y)$ due to the fact that the two phrases have very similar pronunciations and hence their U—the phone sequence converted from their original text sequence Y using the pronunciation model—would

also be very similar that give rise to very similar X as well. This is where a language model jumps in and breaks the tie: it would simply assign a higher likelihood score to P("how to recognize speech") than to P("how to wreck a nice beach") because the former is just more likely to be spoken or written by a person.

Traditionally, language models are parameterized using the count-based n-gram models (Manning and Schutze, 1999). Given a text sequence $Y = (y_1, y_2, \ldots, y_M)$, where M denotes the number of tokens (e.g., words) in the sequence, an n-gram model factorizes the probability of generating Y into the product of the probabilities of each individual token in the sequence, each conditioned on all the tokens that appeared before it:

$$\begin{aligned} P(Y) &= P(y_1, y_2, \ldots, y_M) \\ &= P(y_1)P(y_2|y_1)P(y_3|y_1, y_2)\ldots P(y_M|y_1, y_2, \ldots, y_{M-1}) \\ &= \prod_{m=1}^{M} P(y_m|y_1, y_2, \ldots, y_{m-1}). \end{aligned} \tag{2.3}$$

As can be seen in the factorization shown in Equation 2.3, there can be a combinatorially large number of possible token histories, which would often cause the language models to generalize poorly on unseen combinations of tokens. To alleviate such problem, in practice, we often construct the n-gram models with a small n that is usually between two to five such that the models would only consider the previous $n-1$ tokens when predicting the current token. For instance, a bi-gram model (i.e., $n = 2$) would approximate Equation 2.3 as:

$$\begin{aligned} P(Y) &= P(y_1, y_2, \ldots, y_M) \\ &= P(y_1)P(y_2|y_1)P(y_3|y_1, y_2)\ldots P(y_M|y_1, y_2, \ldots, y_{M-1}) \\ &\approx P(y_1)P(y_2|y_1)P(y_3|y_2)\ldots P(y_M|y_{M-1}), \end{aligned} \tag{2.4}$$

and estimating the bi-gram probabilities of one token y_i coming before another token y_j is accomplished by simply counting the number of times the sub-sequence

(y_i, y_j) appears in the text corpus:

$$P(y_j|y_i) = \frac{\text{Count}(y_i, y_j)}{\text{Count}(y_i)}. \tag{2.5}$$

Smoothing techniques (Kneser and Ney, 1995; Katz, 1987) are commonly applied to handle the case of zero probabilities assigned to unseen n-grams.

Recently, recurrent neural networks (RNN) have been used to replace n-gram models for parameterizing language models (Mikolov et al., 2010). Due to the mechanism of an RNN, when processing each token y_m in a text sequence $Y = (y_1, y_2, \ldots, y_M)$, the RNN maintains a hidden state h_m at each time step that is naturally encoded with the information of all the previous tokens y_1 to y_{m-1}:

$$\begin{aligned} P(y_1, y_2, \ldots, y_M) &= \prod_{m=1}^{M} P(y_m|y_1, y_2, \ldots, y_{m-1}) \\ &= \prod_{m=1}^{M} P(y_m|h_m), \\ h_m &= \text{RNN}(y_m, h_{m-1}). \end{aligned} \tag{2.6}$$

Such property allows RNN-based language models to model the long-term context-dependencies in text sequences better than n-gram models such that, when deployed, they often help the ASR systems produce better results.

Unlike training acoustic models, training a language model requires only text data $\{Y^{(i)}\}_{i=1}^{N}$. Maximum likelihood estimation is commonly used, which trains the language model to maximize the probabilities of generating the text sequences in the text corpus:

$$\theta^* = \underset{\theta}{\text{argmax}} \prod_{i=1}^{N} P(Y^{(i)}; \theta) = \underset{\theta}{\text{argmax}} \sum_{i=1}^{N} \log P(Y^{(i)}; \theta), \tag{2.7}$$

where θ represents the parameters of the language model.

Evaluation A widely used evaluation metric for evaluating ASR systems is word error rate (WER). Let Y_{ref} denote the reference transcript and Y_{pred} be the transcript

predicted by an ASR system, the WER is computed as:

$$\text{WER}(Y_{\text{ref}}, Y_{\text{pred}}) = \frac{\text{Levenshtein-distance}(Y_{\text{ref}}, Y_{\text{pred}})}{\text{Length}(Y_{\text{ref}})}, \tag{2.8}$$

where the Levenshtein distance (Levenshtein, 1966) is a type of edit distance that computes the minimum number of single-token edits required to change Y_{pred} to Y_{ref}.

2.1.2 End-to-End ASR

Based on Equation 2.2 (the "fundamental equation of speech recognition"), for decades ASR systems have been built in a "hybrid" fashion, where their components such as the acoustic models, pronunciation models, and language models, are developed separately and combined later to form the complete ASR systems. Recently, however, there has been a trend of transiting from hybrid modeling to end-to-end modeling, which attempts to directly transcribe speech utterances into textual tokens using a single model.

In end-to-end ASR, the conditional distribution $P(Y|X)$ is modeled by a single model, usually a neural network. End-to-end systems have several advantages over traditional hybrid systems:

- End-to-end models use a single objective function that is consistent with the ASR objective for optimizing the entire network. On the other hand, hybrid models develop their individual components separately, where each component's errors can compound.

- End-to-end models directly learn to output textual tokens from speech utterances, hence the pipeline for speech recognition is greatly simplified when compared to traditional hybrid systems, whose design is complicated and often require lots of expert knowledge with years of ASR experience.

- Since only a single neural network is used for modeling, end-to-end systems have the potential to be much more compact than traditional hybrid systems,

which makes end-to-end systems more suitable for being deployed to devices with high accuracy and low latency.

Given these advantages, end-to-end ASR has attracted great attention recently (Graves and Jaitly, 2014; Hannun et al., 2014; Chorowski et al., 2014; Miao et al., 2015; Bahdanau et al., 2016; Chan et al., 2016; Collobert et al., 2016; Tang et al., 2017; Sak et al., 2017), with some end-to-end systems already outperforming hybrid systems that have been optimized at production level for decades (Watanabe et al., 2017; Sainath et al., 2020; Li et al., 2020c). Below we will give a brief introduction to three of the most popular techniques for modeling end-to-end ASR: Connectionist Temporal Classification, Attention-Based Encoder-Decoder, and Recurrent Neural Network Transducer.

Connectionist Temporal Classification One of the earliest work on end-to-end ASR is connectionist temporal classification (CTC) (Graves et al., 2006). Recall that the goal of end-to-end modeling is to directly map an input speech utterance $X = (x_1, x_2, \ldots, x_T)$ to its textual transcript $Y = (y_1, y_2, \ldots, y_M)$ with a single neural network. The key problem to optimizing such a network is that we do not know the alignment between X and Y (in other words, we do not know which x_t corresponds to which y_m) and T is usually much longer than M in speech.

The idea of CTC is to ask the model—usually a recurrent neural network—to make a token prediction $\tilde{y}_t$ for each speech frame x_t. In addition to predicting tokens from the original token set, the model can also predict a blank token ϵ for $\tilde{y}_t$. The predicted sequence $\tilde{Y}$ has the same length as X with allowable repetition of tokens that construct a valid "path" for Y. To give the final prediction of the ASR system, in $\tilde{Y}$, all the blank tokens ϵ will be removed, and any repeated tokens will be collapsed into just one unless an ϵ is inserted between them. There can be multiple valid paths for a given Y. For instance, for a speech utterance X with a $T = 8$, and the corresponding transcript Y is the word "team" ($M = 4$), then (ϵ, ϵ, t, ϵ, e, a, m, m), (t, ϵ, e, ϵ, ϵ, a, ϵ, m), and (ϵ, t, e, a, a, ϵ, m, ϵ) are three valid paths based on CTC's rules. Let $\mathcal{B}^{-1}(Y)$ denote the set of all valid paths that can be reduced to Y, the

CTC loss is defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{CTC}} &= -\log \sum_{\tilde{Y} \in \mathcal{B}^{-1}(Y)} P(\tilde{Y}|X) \\ &= -\log \sum_{\tilde{Y} \in \mathcal{B}^{-1}(Y)} \prod_{t=1}^{T} P(\tilde{y}_t|X),\end{aligned} \tag{2.9}$$

where $\tilde{Y} = (\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_T)$ denotes a valid CTC path.

Attention-Based Encoder-Decoder The attention-based encoder-decoder (AED) model is another popular technique for modeling end-to-end ASR. Largely inspired by those used for end-to-end neural machine translation (Cho et al., 2014b; Bahdanau et al., 2015), an AED model for end-to-end ASR is composed of an encoder network, an attention module, and a decoder network, and models the conditional distribution of Y given X as:

$$P(Y|X) = \prod_{m=1}^{M} P(y_m|X, Y_{1:m-1}), \tag{2.10}$$

where m indexes the sequence Y. An AED model is also trained to minimize $-\log P(Y|X)$. The function of the encoder network is to convert the input speech sequence X into a high-level representation H. For each decoding time step, the attention module will first compute attention weights between the previous decoder output and each feature vector in H using attention functions such as additive attention (Bahdanau et al., 2016) or dot-product attention (Chan et al., 2016), and will then use those attention weights to compute the weighted sum of the feature vectors in H, resulting in a context vector. Finally, the decoder network will take both its previous output and the context vector to generate its output that will be used for computing $P(y_m|X, Y_{1:m-1})$.

One assumption CTC makes that is problematic for tasks like ASR is the conditional independence assumption, which assumes that every output is conditionally independent of the other outputs. AED models' attention mechanism, on the other hand, naturally allows them to take into account the other outputs when generating the current output. This advantage makes AED models generally more powerful than

CTC models. In practice, however, it is common to optimize an AED model along with a CTC model by sharing their encoder (Watanabe et al., 2017; Kim et al., 2017; Ueno et al., 2018; Kim et al., 2019). This is because such a multi-task training strategy is found to assist the AED models with learning a better alignment that greatly improves their convergence.

Recurrent Neural Network Transducer Another popular technique for end-to-end ASR modeling is recurrent neural network transducer (RNN-T) (Graves, 2012), which consists of an encoder network, a prediction network, and a joint network. As in the encoder network for CTC and AED, the encoder in an RNN-T extracts a high-level feature representation h_t^{enc} for each x_t in X. In parallel, the prediction network extracts a high-level feature representation h_u^{pre} for RNN-T's previous output label y_{u-1}, where u is the label index. The joint network is a feed-forward network (FFN) that combines h_t^{enc} and h_u^{pre} as:

$$z_{t,u} = \text{FFN}(h_t^{enc}, h_u^{pre}) = \psi(\mathbf{Q}h_t^{enc} + \mathbf{V}h_u^{pre} + b_z), \tag{2.11}$$

where $\mathbf{Q}$ and $\mathbf{V}$ are weight matrices, b_z is a bias vector, and ψ is a non-linear activation function such as ReLU or tanh. Then, a linear transformation is applied to $z_{t,u}$:

$$h_{t,u} = \mathbf{W}_y z_{t,u} + b_y, \tag{2.12}$$

where $\mathbf{W}_y$ and b_y denote a weight matrix and a bias vector, respectively. Finally, the probability for each output token k from the token set is calculated as:

$$P(y_u = k|X_{1:t}, Y_{1:u-1}) = \text{softmax}(h_{t,u}^k). \tag{2.13}$$

To train an RNN-T model, we minimize $-\log P(Y|X)$, where

$$P(Y|X) = \sum_{\tilde{A} \in A^{-1}(Y)} P(\tilde{A}|X) \tag{2.14}$$

is the sum of the probabilities of all valid paths, which are denoted as $A^{-1}(Y)$, that can be mapped to the label sequence Y. Note that the name "RNN-T" itself could be confusing, as Transformers (Vaswani et al., 2017) or Conformers (Gulati et al., 2020) can also be used as the encoder networks, which will be discussed in Section 2.2. The term RNN-T is just commonly used for historical reason.

For all ASR experiments in this thesis, we do not use hybrid models but use end-to-end models. Specifically, we will be using an AED in Chapter 5 and an RNN-T in Chapter 6. In both cases, only the encoder networks are pre-trained with the self-supervised objectives, while the rest of the networks are trained from scratch during the fine-tuning stage.

2.1.3 Semi-Supervised ASR

So far, we have posed ASR as a purely supervised learning problem: regardless of modeling ASR with hybrid or end-to-end approaches, it always requires audio-transcript pairs for training the systems to make them learn good speech representations. On the other hand, how to leverage large-scale and easily collectable unlabeled speech data to improve supervised ASR performance has been a longstanding research problem. To date, there have been two major streams for utilizing unlabeled speech data for tackling such a semi-supervised ASR task.

The first line of work is self-training (Riloff and Wiebe, 2003; Yarowsky, 1995; Scudder, 1965), also known as pseudo-labeling, where the system starts with training a teacher model using initially available labeled data. Next, the teacher model is used to annotate the unlabeled data. The combined labeled and pseudo-labeled data are then used to train a student model. The pseudo-labeling process can be repeated multiple times to improve the quality of the teacher model. Self-training has been a practically useful and extensively studied technique in ASR (Kahn et al., 2020a; Synnaeve et al., 2020; Li et al., 2019; Parthasarathi and Strom, 2019; Novotney and Schwartz, 2009; Zavaliagkos and Colthurst, 1998).

The second direction of taking advantage of unlabeled speech data is unsupervised pre-training, or self-supervised pre-training. In unsupervised pre-training, a model

is first trained to complete a proxy task that is designed to consume only unlabeled data (hence unsupervised). Such a proxy task has been empirically verified and hence is commonly believed to be capable of initializing the parameters of the model at a good starting point before it is being trained on the supervised data. Significant recent research effort has been made to develop proxy tasks that allow models to perform well when the models are fine-tuned on ASR tasks (Oord et al., 2018; Schneider et al., 2019; Ling et al., 2020; Liu et al., 2020b, 2021; Wang et al., 2020; Ling and Liu, 2020; Bai et al., 2021). Finally, there have also been studies that show that the gains brought by self-training and unsupervised pre-training are additive in downstream ASR (Zhang et al., 2020b; Xu et al., 2021).

In this thesis, we focus on improving the unsupervised pre-training aspect of semi-supervised ASR by proposing and studying two novel pre-training frameworks.

2.2 Neural Networks for Speech Processing

In this section we review two of the most widely used neural network architectures for speech processing: recurrent neural networks and self-attention networks. Here we assume the readers are already familiar with their basic mechanisms and variants (e.g., long short-term memory), and focus on reviewing their usage in speech processing, especially in modeling end-to-end ASR.

2.2.1 Recurrent Neural Networks

In end-to-end models such as CTC, AED, and RNN-T, the most important component is their encoder network, whose goal is to transform the input audio sequence $X = (x_1, x_2, \ldots, x_T)$ into a high-level feature representation $H = (h_1, h_2, \ldots, h_T)$.

When end-to-end models first came out (Graves et al., 2006; Graves, 2012; Chan et al., 2016), long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997)—a variant of recurrent neural network (RNN)—had been largely used to construct the

encoder networks. The encoder can be either a multi-layer unidirectional LSTM:

$$h_t^\ell = \text{LSTM}(x_t^\ell, h_{t-1}^\ell), \tag{2.15}$$

or a multi-layer bidirectional LSTM:

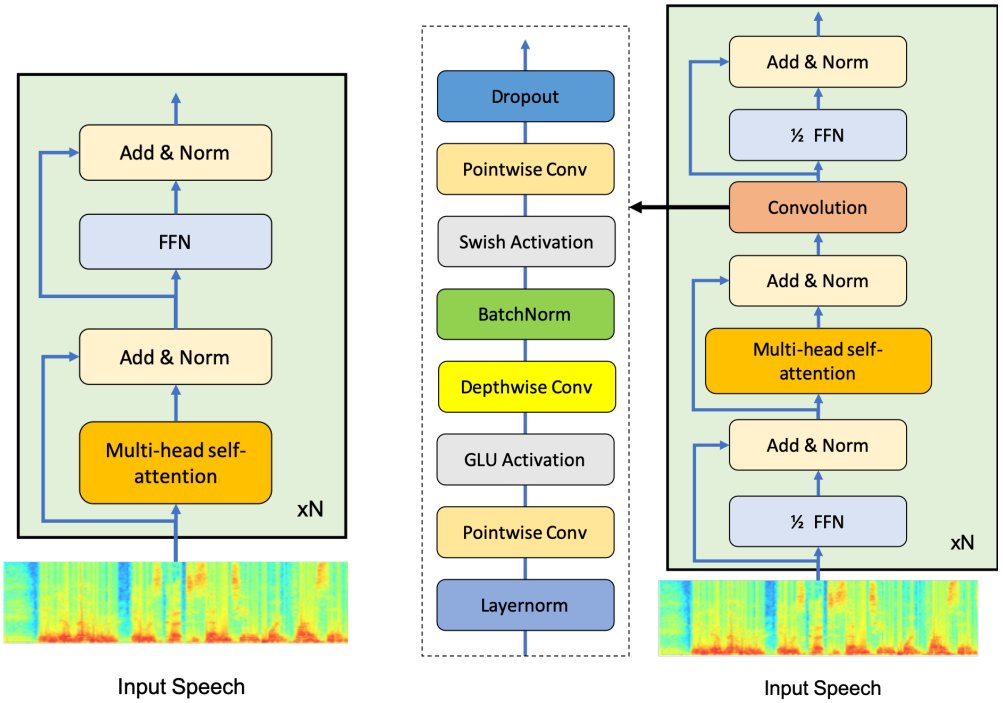
$$h_t^\ell = [\text{LSTM}(x_t^\ell, h_{t-1}^\ell), \text{LSTM}(x_t^\ell, h_{t+1}^\ell)], \tag{2.16}$$

where $\text{LSTM}(\cdot)$ denotes the standard LSTM unit, h_t^ℓ denotes the hidden output of the ℓ-th layer at time t, and x_t^ℓ is the input vector for the ℓ-th layer defined as $x_t^\ell = x_t$ if $\ell = 1$ (i.e., the first layer input) else $h_t^{\ell-1}$. The last layer output of the LSTM network is taken as the encoder output. Whether to use a unidirectional and bidirectional LSTM as encoder network depends on the streaming request for the ASR system: the former allows streaming but sacrifices performance, while the latter performs better but does not allow streaming.

2.2.2 Self-Attention Networks

Although an LSTM can capture short-term dependencies in sequential data, self-attention networks (Vaswani et al., 2017), or *Transformers* in most literature, have been shown to be better at capturing long-term dependencies in sequences since their self-attention mechanism allows Transformers to access the entire sequence when extracting h_t for every x_t in X. Due to the modeling power of Transformers, currently there is a trend of replacing the LSTM encoder network in end-to-end ASR models with Transformers (Dong et al., 2018; Zeyer et al., 2019; Karita et al., 2019; Li et al., 2020b; Zhang et al., 2020a).

As illustrated in Figure 2-1a, the encoder network of a Transformer-based end-to-end model is composed of a stack of Transformer blocks, where each block consists of a multi-head self-attention layer and a feed-forward network (FFN). Residual connections (He et al., 2016) and layer normalization (Ba et al., 2016) are used to connect different layers and blocks. In each Transformer block, every input vector x_t (here x_t



(a) Transformer encoder. (b) Conformer encoder.

Figure 2-1: The structures of Transformer and Conformer as encoder networks in end-to-end ASR models. The green area depicts a Transformer/Conformer block, which is stacked N times to form the encoder network.

represents not only the acoustic feature vector in the original speech sequence X at time step t but also the corresponding representation vector output by the previous Transformer block) is linearly transformed into a query vector q, a key vector k, and a value vector v with matrices $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$, respectively. A self-attention mechanism is used to compute the attention distribution over the input sequence with the dot-product similarity function as:

$$\alpha_{t,\tau} = \frac{\exp(\beta(\mathbf{W}_q x_t)^T(\mathbf{W}_k x_\tau))}{\sum_{\tau'} \exp(\beta(\mathbf{W}_q x_t)^T(\mathbf{W}_k x_{\tau'}))}, \tag{2.17}$$

where $\beta = \frac{1}{\sqrt{d}}$ is a scaling factor, τ indexes the input sequence, and $\alpha_{t,\tau}$ represents the attention weight for x_τ to put on x_t. All the attention weights are then used to

combine the value vectors to generate the layer output at the current time step as:

$$z_t = \sum_{\tau} \alpha_{t,\tau} \mathbf{W}_v x_\tau = \sum_{\tau} \alpha_{t,\tau} v_\tau. \tag{2.18}$$

z_t is then used as the input vector at time step t to the next Transformer block. Multi-head self-attention is used to further improve the model capacity by applying multiple parallel self-attention modules on the input sequence, and the outputs of each attention module are concatenated.

2.2.3 Convolution-Augmented Self-Attention Networks

While the Transformer model is good at capturing global context, it is less effective in extracting local patterns. To further improve modeling capability, a convolutional neural network (Waibel et al., 1989), which works well at capturing local information, is combined with Transformer as a *Conformer* (Gulati et al., 2020). As illustrated in Figure 2-1b, each Conformer block contains two half-step FFNs sandwiching the multi-head self-attention module and the convolution module. Such convolution-augmented Transformer architecture enjoys the best of both worlds, being able to model both local and global dependencies in audio sequences, and are gaining more and more popularity in speech processing (Guo et al., 2021).

2.3 Neural Network Pre-Training and Self-Supervised Learning

As introduced in Section 2.1.3, self-supervised pre-training is one of the two main streams of semi-supervised techniques (the other one is self-training) that leverage large quantities of unlabeled speech data for improving ASR performance. Self-supervised pre-training is also the aspect this thesis focuses on for tackling the semi-supervised ASR problem. In this section we first provide a brief overview on neural network pre-training, then review some of the most representative self-supervised

pre-training frameworks for speech processing.

2.3.1 Background

The wide adoption of the "pre-training followed by fine-tuning" paradigm for semi-supervised learning stems from computer vision. An instance of the pre-training paradigm is supervised pre-training—the approach of learning representations with a supervised task. The representations learned from image classification have been shown to be useful for object detection and semantic segmentation (Girshick et al., 2014). This approach has been extended to other supervised tasks without manual annotations, and is termed *self-supervised learning* by Doersch et al. (2015).[2] Various tasks, such as colorization (Larsson et al., 2016), solving jigsaw puzzles (Noroozi and Favaro, 2016), and inpainting (Pathak et al., 2016) in the vision domain have been proposed for self-supervised learning of visual representations. Subsequent developed pre-training techniques in the text domain based on language modeling such as ELMo (Peters et al., 2018b), BERT (Devlin et al., 2019), and XLNet (Yang et al., 2019), also belong to self-supervised learning.

The key difference between supervised pre-training and self-supervised pre-training is the type and the amount of information about the input retained in the learned representation. The representations learned through a supervised task tend to only include information useful to perform the task, while information irrelevant to the task tends to be discarded (Belinkov and Glass, 2017; Chowdhury et al., 2021). As a result, representations learned by supervised pre-training are only suitable for tasks similar to the supervised tasks. This explains why, for instance, representations trained on image classification can be useful for object detection and semantic segmentation, given the similarity among these tasks. The representations learned from self-supervised tasks, however, tend to include various aspects of the input, not for a specific task, in particular when the self-supervised task includes reconstructing parts of the in-

[2]From this point of the thesis, we will not differentiate the two terms "unsupervised" and "self-supervised". Both terms are now used to indicate that manually labeled data are used during training.

put. This explains the wide applicability of representations learned by self-supervised learning.

2.3.2 Self-Supervised Tasks for Speech Representation Learning

Self-supervised methods for learning speech representations can be roughly categorized into two groups: those that are based on contrastive learning and those based on generative (or sometimes called reconstructive) learning.

Contrastive Learning The key idea behind contrastive learning is "learning by comparison." Contrastive Predictive Coding (CPC) (Oord et al., 2018), which we will introduce in more detail and experiment with extensively in Chapter 3, is arguably the most representative method that falls into this category. CPC defines a task where a unidirectional, autoregressive model is asked to predict the frames in the near future. The model learns the representations by distinguishing the target future frames from frames from other audio sequences, or frames from a more distant time in the same sequence. After CPC was proposed, several improvements, such as changing unidirectional to bidirectional Kawakami et al. (2020a) and making architectural modifications (Riviere et al., 2020), have also been explored.

The wav2vec series also belongs to the family of contrastive learning. Schneider et al. (2019) proposed the first wav2vec model, which is actually very similar to CPC. The notable difference is that wav2vec uses a fully-convolutional architecture while CPC uses an RNN. In vq-wav2vec proposed by Baevski et al. (2020a), the self-supervised training procedure is divided into two steps: the authors first augmented the original wav2vec model with a vector-quantization layer (Oord et al., 2017) to learn a codebook of discretized speech units, then trained a BERT model on top of the discretized speech units. Although vq-wav2vec is able to represent continuous speech as distinctive speech tokens and can hence make use of already well-developed NLP pre-training frameworks such as BERT, the two-stage training scheme still has some problems such as unrecoverable incorrect speech ID assignments. Observing

such problem, Baevski et al. (2020b) further proposed wav2vec 2.0, which improves vq-wav2vec into a single-stage training scheme and achieves remarkable results on ASR.

Reconstructive Learning In contrast to contrastive learning approaches that often define objectives that operate in the space of hidden vectors projected from the input, reconstructive learning approaches define tasks that ask the models to reconstruct parts of the input explicitly in the output space that is often the same as the input space. This line of research is primarily inspired by the remarkable progress made in the text domain such as ELMo, BERT, and XLNet.

One of the earliest approaches that falls into the category of reconstructive learning is Autoregressive Predictive Coding (APC) (Chung et al., 2019a; Chung and Glass, 2020a), which is a direct adaptation from language modeling pre-training. APC employs a unidirectional, autoregressive model for summarizing the past acoustic frames to predict/reconstruct a near future frame, where the difference between the predicted and target future frames is measured by the $\ell 1$ loss. APC, including its variant (Chung et al., 2020) and improved version (Chung and Glass, 2020b), is one of the two pre-training frameworks proposed in this thesis, and is presented from Chapter 3 to Chapter 5. Inspired by ELMo, Ling and Liu (2020) have also explored bidirectional APC.

Another line of research in reconstructive learning is largely inspired by the masked language modeling task from BERT (Liu et al., 2020b; Wang et al., 2020; Chi et al., 2021; Jiang et al., 2021; Liu et al., 2021) and the permutation language modeling task from XLNet (Song et al., 2020), where these NLP pre-training techniques are adapted to operate on continuous speech.

Finally, there are some recently proposed methods that are difficult to classify as either contrastive or reconstructive learning, such as HuBERT (Hsu et al., 2021) and w2v-BERT (Chung et al., 2021b). w2v-BERT is the second pre-training framework proposed in this thesis to be presented in Chapter 6, where HuBERT will also be discussed and compared with.

Although self-supervised learning approaches have been mainly developed for improving different ASR benchmarks, there have been studies that show self-supervised pre-trained models are also useful for speech tasks other than ASR. In particular, WavLM (Chen et al., 2021) was developed and has achieved remarkable performance on full-stack speech processing tasks that require the models to capture a variety of speech characteristics such as the phonetic, speaker, emotion, and semantic information of the spoken utterances.

2.4 Neural Representation Analysis

In this section, we review two lines of techniques for analyzing the representations learned by deep neural networks: the probing task approach and the similarity measure approach, which will be used in Chapter 7 for analyzing a variety of self-supervised speech representations.

2.4.1 Background

Deep neural networks have long been treated as a black-box due to the fact that their high architectural complexity makes it hard for humans to understand the networks' inner workings. Recent new developments in deep neural networks, including new architectures, pre-training objectives, and parameter optimization algorithms, have led to significant improvements over previous state-of-the-arts on various benchmarks across different domains such as text, speech, and vision. Such progress has attracted researchers' interest to study the models' internal representations to assess what linguistic/acoustic/visual properties they capture that make them so powerful. Below we review two lines of techniques for analyzing the representations learned by deep neural networks: the probing task approach and the similarity measure approach.

2.4.2 The Probing Task Approach

The idea of the probing task approach is to use so-called *probing classifiers* to predict certain properties from representations extracted from the deep neural models under investigation. This approach contains three steps. A model is first trained on some tasks (the task can be either a supervised one like neural machine translation or a self-supervised one such as masked language modeling). Next, the pre-trained model is used for generating feature representations for another task by running it on a corpus with linguistic annotations (or acoustic annotations, depending on the properties of interest). Finally, a probing classifier, which is usually a linear classifier or shallow multi-layer perceptron, is used for predicting the properties of interest using the extracted feature representations as input. The performance of the probing classifier evaluates the quality of the feature representations extracted from the pre-trained model for capturing the properties of interest.

The probing task approach has been used in a numerous of studies for associating linguistic properties, such as part-of-speech tags and named entities, with supervised (e.g., neural machine translation) and self-supervised (e.g., BERT) textual representations (Ettinger et al., 2016; Belinkov et al., 2017; Adi et al., 2017; Conneau et al., 2018; Hupkes et al., 2018; Belinkov and Glass, 2019). For speech, the probing task approach has also been applied to investigate the acoustic properties captured by pre-trained ASR (Belinkov and Glass, 2017; Li et al., 2020a) and speech synthesis models (Zhu, 2020).

Although the probing task approach can sometimes yield compelling insights about the models that are helpful for improving them, the approach's applicability is constrained by the availability of annotations for the properties of interest. Additionally, comparing different model representations is indirect by observing their probing performance, which makes it hard to draw conclusions on the similarities and dissimilarities between different model representations.

2.4.3 The Similarity Measure Approach

In contrast to the probing task approach, the similarity measure approach does not rely on annotated corpora for analysis, and is able to compare different model representations directly without the need of proxy probing classifiers. The idea is to take advantage of existing similarity measures to output a score that quantifies the similarity between two model representations, where the two representations can be from the same (e.g., representations from different layers of the same model) or different pre-trained models, and then use such information to comment on the inter- and/or intra-similarity of the models.

In the text domain, Bau et al. (2019) used this approach to analyze the role of individual neurons in neural machine translation models. Studies such as Morcos et al. (2018); Saphra and Lopez (2019) used the similarity measure approach to investigate the learning dynamics in language models by comparing a series of checkpoints of models. In Wu et al. (2020), a variety of similarity measures—each capturing a different similarity notion—were explored to study the effect of diverse aspects of modeling choices (such as building blocks, training objectives, directionality, and model sizes) for building self-supervised models. For speech, the similarity measure approach was used by Pasad et al. (2021) as one of the tools for analyzing a particular self-supervised speech model (wav2vec 2.0).

Similarity measures are the key to this analysis method. Given two model representations $\mathbf{h}^{(a)} = (h_1^{(a)}, h_2^{(a)}, \ldots, h_L^{(a)})$ and $\mathbf{h}^{(b)} = (h_1^{(b)}, h_2^{(b)}, \ldots, h_L^{(b)})$, where a and b represent two layers that can be either from the same or different models, and L denotes the number of words in the text corpus (or number of frames in the speech corpus) used for analysis, a similarity measure outputs a score $\texttt{sim}(\mathbf{h}^{(a)}, \mathbf{h}^{(b)}) \in \mathbb{R}$ that quantifies the similarity of the two representations. Let d_a denote the dimensionality of layer a (i.e., $h_i^{(a)} \in \mathbb{R}^{d_a}$), $\mathbf{h}^{(a)}$ can be viewed as a $L \times d_a$ matrix, in which the i-th row vector is essentially $h_i^{(a)}$. The k-th neuron is denoted as $\mathbf{h}^{(a)}[:, k]$, which can be viewed as a $L \times 1$ column vector. The same notations apply to $\mathbf{h}^{(b)}$. Below we introduce three groups of similarity measures that are designed to capture different

levels of localization/distributivity of information.

Neuron-level similarity Similarity measures that belong to this group aim to capture localization of information, and their approach is to estimate the similarity of behaviors between pairs of individual neurons in layers a and b. For example, Bau et al. (2019) proposed `neuronsim`, where for every neuron k in layer a, `neuronsim` finds the maximum correlation between it and another neuron in layer b. `neuronsim` then outputs the average of maximum correlations of all neurons in layer a:

$$\begin{aligned} \texttt{neuronsim}(\mathbf{h}^{(a)}, \mathbf{h}^{(b)}) &= \frac{1}{d_a} \sum_{k=1}^{d_a} \texttt{neuronsim}(\mathbf{h}^{(a)}[:, k], \mathbf{h}^{(b)}) \\ &= \frac{1}{d_a} \sum_{k=1}^{d_a} \max_{k'} |\rho(\mathbf{h}^{(a)}[:, k], \mathbf{h}^{(b)}[:, k'])|, \end{aligned} \tag{2.19}$$

where ρ is the Pearson correlation.

Mixed neuron-representation similarity Unlike neuron-level similarity measures, mixed neuron-representation measures aim to capture the similarity between a neuron in layer a with the entire layer b, since it is possible that some information is localized in one layer but distributed in another. A mixed neuron-representation similarity captures such a phenomenon, while a neuron-level similarity measure fails to. An example measure is `mixedsim` (Wu et al., 2020), where for every neuron k in layer a, a regressor is trained to regress to it from all neurons in layer b. `mixedsim` then outputs the average of the goodness of fit of all k regressors:

$$\begin{aligned} \texttt{mixedsim}(\mathbf{h}^{(a)}, \mathbf{h}^{(b)}) &= \frac{1}{d_a} \sum_{k=1}^{d_a} \texttt{mixedsim}(\mathbf{h}^{(a)}[:, k], \mathbf{h}^{(b)}) \\ &= \texttt{lstsq}(\mathbf{h}^{(a)}[:, k], \mathbf{h}^{(b)}).\texttt{r}, \end{aligned} \tag{2.20}$$

where `lstsq` stands for least square regression and `.r` denotes the R^2, the goodness of fit, associated with each regressor.

Representation-level similarity Rather than trying to capture the behaviors of individual neurons to learn the localization of information, representation-level similarity measures emphasize the distributivity of information. Measures that fall into this category are more capable of identifying similar behaviors of two layers overall, even if no individual neuron has a a similar matching pair (lack of neuron-level similarity) or is represented well by neurons in the other layer (lack of mixed neuron-representation similarity).

Lots of studies have proposed measures and for evaluating such representation-level similarity (Kriegeskorte et al., 2008; Morcos et al., 2018; Bouchacourt and Baroni, 2018), from which two of them will be used in this thesis (in Chapter 7): singular vector canonical correlation analysis (SVCCA) (Raghu et al., 2017) and linear centered kernel alignment (CKA) (Kornblith et al., 2019). Let $\mathbf{Z}$ be a column centering transformation such that given a matrix $\mathbf{V}$, the sum of each column in $\mathbf{ZV}$ is zero. SVCCA first transforms $\mathbf{h}^{(a)}$ and $\mathbf{h}^{(b)}$ into $\mathbf{A}$ and $\mathbf{B}$, where

$$\mathbf{A}, \mathbf{B} = \mathbf{Zh^{(a)}}, \mathbf{Zh^{(b)}}. \tag{2.21}$$

Let $\mathbf{U}_a$ and $\mathbf{U}_b$ be the left singular vectors of $\mathbf{A}$ and $\mathbf{B}$, and l_a and l_b be the index required to account for 99% of the variance, respectively. SVCCA then computes the truncated principle components $\mathbf{A'}$ and $\mathbf{B'}$, where

$$\mathbf{A'}, \mathbf{B'} = \mathbf{U}_a[:, : l_a], \mathbf{U}_b[:, : l_b]. \tag{2.22}$$

The SVCCA correlations ρ_{svcca} are defined as:

$$\mathbf{u}, \rho_{\text{svcca}}, \mathbf{v} = \text{SVD}(\mathbf{A'}^T\mathbf{B'}), \tag{2.23}$$

and the SVCCA similarity $\texttt{svsim}(\mathbf{h}^{(a)}, \mathbf{h}^{(b)})$ is the mean of ρ_{svcca}. The CKA similarity is defined as:

$$\texttt{ckasim}(\mathbf{h}^{(a)}, \mathbf{h}^{(b)}) = \frac{||A^T B||^2}{||A^T A||||B^T B||}, \tag{2.24}$$

where $||\cdot||$ is the Euclidean norm.

2.5 Chapter Summary

We have provided an overview on several topics that are relevant to this thesis, including automatic speech recognition (ASR), neural networks for speech processing, self-supervised representation learning, and neural representation analysis.

In Section 2.1, we first formulated the ASR problem, then discussed both traditional hybrid and modern end-to-end techniques for modeling ASR. The introduced end-to-end models AED and RNN-T will be used in Chapter 5 and Chapter 6 for ASR experiments, respectively. We also introduced two lines of research directions for semi-supervised ASR that aims to leverage large quantities of unannotated speech to improve ASR performance: self-training and self-supervised pre-training, and the latter direction is the focus of this thesis.

In Section 2.2, we reviewed three neural network architectures widely used for speech processing: LSTM, Transformer, and Conformer, all of which will be used extensively in this thesis for building different speech processing systems.

In Section 2.3, we first provided some background of self-supervised pre-training in different domains including text, speech, and vision, and compared it with supervised pre-training. Then, we moved our focus to discussing existing works for self-supervised speech representation learning, positioning the two pre-training frameworks we are going to propose and study in this thesis: Autoregressive Predictive Coding (Chapter 3 to Chapter 5) and w2v-BERT (Chapter 6).

In Section 2.4, we introduced two lines of techniques for analyzing representations learned by deep neural networks: the probing task approach and the similarity measure approach. In particular, for the probing task approach, we will use phone and speaker classifications for understanding the constituents of a self-supervised speech representation; for the similarity measure approach, we will use singular vector canonical correlation analysis (SVCCA) and centered kernel alignment (CKA) to quantify the similarities between pairs of self-supervised speech representations.

In the next chapter, we will present our first efforts in self-supervised speech representation learning, proposing the Autoregressive Predictive Coding pre-training framework.

Chapter 3

Self-Supervised Objective: Predicting the Future from the Past

This chapter proposes a novel self-supervised autoregressive neural model, which we call Autoregressive Predictive Coding (APC), for learning generic speech representations. In contrast to most existing speech representation learning methods that aim to remove noise or speaker variabilities, APC is designed to preserve information for a wide range of downstream tasks. In addition, APC does not require any phonetic or word boundary labels, allowing the model to benefit from large quantities of unlabeled data. Speech representations learned by our model significantly improve performance on both phone classification and speaker verification over the surface features and other supervised and self-supervised approaches. Further analysis shows that different levels of speech information are captured by APC at different layers. In particular, the lower layers tend to be more discriminative for speakers, while the upper layers provide more phonetic content.

The content of this chapter was first published in Chung et al. (2019a).

3.1 Introduction

Speech signals encompass a rich set of acoustic and linguistic properties, ranging from the individual lexical units, such as phonemes and words, to the characteristics of the

speakers, their intent, or even their mental status. However, these high-level properties of speech are poorly captured by the surface features, such as the amplitudes of a wave signal, log Mel spectrograms, or Mel frequency cepstral coefficients. The goal of speech representation learning is to find a transformation from the surface features that makes high-level properties of speech more accessible to downstream tasks.

In this chapter, we propose an autoregressive model for learning speech representations that can be transferred to different tasks across different datasets. Our model is able to retain much information from the surface features, allowing a wide range of tasks across different datasets to benefit from the learned representations, while also being unsupervised and able to leverage large quantities of unlabeled data. As a first step, we focus on learning general speech representations from log Mel spectrograms, but it is straightforward to extend our approach to the amplitudes of the wave signals.

We use linear separability (or separability with a shallow network) to define the accessibility of information for the downstream tasks. Others such as Schatz et al. (2013) have argued that there are many nuisance factors that might affect the performance of linear classifiers, and have proposed to use a contrastive loss for evaluation. However, there has been evidence (Settle and Livescu, 2016) and theories (Arora et al., 2019) supporting the idea that low contrastive loss implies the existence of a linear classifier with low error. In other words, we aim to learn speech representations that allow linear classifiers to perform well on many downstream tasks.

When the downstream tasks are known, supervised learning, specifically multitask learning (Caruana, 1997), is the most successful approach for learning specialized representations of those particular tasks. In general, however, when a transformation is trained against a certain set of tasks, information independent of the tasks (such as noise or speaker variability, depending on the tasks) tends to be discarded after training (Tishby et al., 1999). We risk discarding useful information for other unseen tasks when learning representations in a supervised fashion. Instead of having targeted tasks in advance, we focus on learning representations for a wide range of, potentially unknown, tasks. Due to the required generality, it is necessary to retain in the representations as much information about the original signal as possible. Two

commonly used loss functions, i.e., the autoencoding and autoregressive loss functions, satisfy this criterion. However, when no additional constraints are imposed, there is a trivial solution, the identity mapping, for the autoencoding loss function. This makes the autoregressive loss more appealing, because no additional techniques, such as denoising (Vincent et al., 2010), are required to avoid the trivial solution as for the autoencoding loss. The autoregressive approach also does not require other types of linguistic constraints, such as phonetic or word boundaries (Kamper et al., 2016).

The autoregressive loss belongs to a large family of self-supervised loss functions (Wang and Gupta, 2015; Doersch et al., 2015; Larsson et al., 2017). There also exists some work on unsupervised speech representation learning (Chorowski et al., 2019; Chung and Glass, 2018; Milde and Biemann, 2018; Hsu et al., 2017b,a; Chung et al., 2016). However, none of the studies are able to show the transferability of the learned representations across different datasets. Our work is largely motivated by the recent success in transfer learning from large-scale pre-trained language models (Peters et al., 2018b; Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019), and we aim to learn general speech representations that can be transferred to different tasks across different datasets.

3.2 Models

We propose a novel autoregressive architecture, which we call Autoregressive Predictive Coding (APC), for self-supervised speech representation learning. Predictive coding on wave samples (Schroeder and Atal, 1985) has a long and influential history in speech processing, and its recent neural version (Oord et al., 2016) and variants, such as Contrastive Predictive Coding (CPC) (Oord et al., 2018), have also been used to learn speech representation (Chorowski et al., 2019). In contrast to these studies, our work mainly focuses on predicting the spectrum of a future frame rather than a wave sample. We will briefly review CPC here and compare extensively with it in Section 3.3.

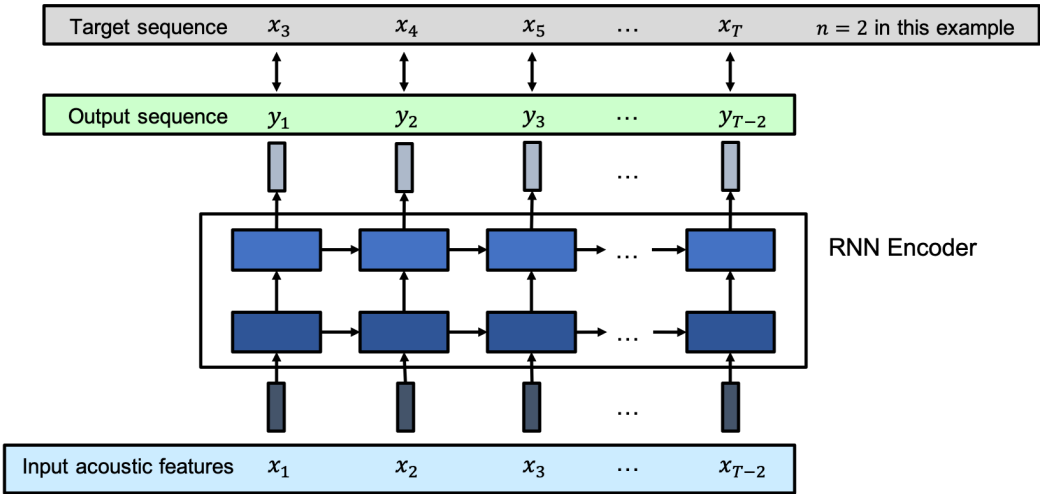


Figure 3-1: Illustration of the Autoregressive Predictive Coding model. The input to the model is a sequence of acoustic feature vectors $(x_1, x_2, \ldots, x_T)$, and the goal of the model is to predict a future frame that is n steps ahead (in this example, $n = 2$) from the current time step. The RNN Encoder can be replaced with any other autoregressive models such as a Transformer with proper masking.

3.2.1 Autoregressive Predictive Coding

The methodology of the proposed APC model is largely inspired by language models (LMs) for text, which are typically a probability distribution over sequences of N tokens $(t_1, t_2, ..., t_N)$. Given such a sequence, an LM assigns a probability $P(t_1, t_2, ..., t_N)$ to the whole sequence by modeling the probability of token t_k given the history $(t_1, t_2, ..., t_{k-1})$:

$$P(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} P(t_k \mid t_1, t_2, ..., t_{k-1}). \tag{3.1}$$

It is trained by minimizing the negative log-likelihood:

$$\sum_{k=1}^{N} -\log P(t_1, t_2, ..., t_{k-1}; \theta_t, \theta_{\text{rnn}}, \theta_s), \tag{3.2}$$

where the parameters to be optimized are θ_t, θ_{rnn}, and θ_s. θ_t is a look-up table that maps each token into a vector of fixed dimensionality. θ_{rnn} is a Recurrent Neural Network (RNN) used to summarize the sequence history up to the current time step.

θ_s is a Softmax layer appended at the output of each RNN time step for estimating probability distribution over the tokens. Language modeling is a general task that requires the understanding of many aspects in language in order to perform well.

Following most of the neural LMs in the literature, we use an RNN (Mikolov et al., 2010) for modeling the temporal information within an acoustic sequence. For speech data, each token t_k corresponds to a frame rather than a word or character token, hence we do not need the look-up table θ_t as we do in LMs and directly feed each frame into the RNN θ_{rnn}. Since there does not exist a finite set of target tokens (such as the vocabulary set as in text), we choose to replace the Softmax layer with a regression layer θ_r. In other words, the RNN output at each time step attempts to directly fit the target frame with a linear mapping. The learnable parameters in APC are θ_{rnn} and θ_r.

Given the history $(t_1, t_2, ..., t_{k-1})$, an LM aims to maximize the probability of the next token to be the t_k in the data. However, for APC, exploiting the local smoothness of the speech signal might be sufficient to predict the next frame. To encourage APC to infer more global structures rather than the local information in the signals, we ask the model to predict a frame n steps ahead of the current one. In other words, given an utterance represented as a sequence of acoustic feature vectors $(x_1, x_2, ..., x_T)$, the RNN processes each sequence element x_t one at a time and outputs a prediction y_t, where x_t and y_t have the same dimensionality. The model, which consists of a RNN and a linear regression layer, is optimized by minimizing the $\ell 1$ loss (as is done when predicting spectral frames in some speech synthesis models such as Wang et al. (2017); Chung et al. (2019b)) between the input sequence $(x_1, x_2, ..., x_T)$ and the predicted sequence $(y_1, y_2, ..., y_T)$:

$$\sum_{i=1}^{T-n} |x_{i+n} - y_i|. \tag{3.3}$$

Figure 3-1 illustrates the APC model. The target sequence can be easily generated by right-shifting the input sequence by n time steps.

3.2.2 Contrastive Predictive Coding

Instead of learning to predict future frames like APC, Contrastive Predictive Coding (CPC) (Oord et al., 2018) aims to learn representations that separates the target future frame x_{i+n} and randomly sampled negative frames $\{\tilde{x}\}$, given a context $h_i = (x_1, x_2, ..., x_i)$.

Specifically, CPC consists of three modules: a frame encoder E_{frm}, a uni-directional RNN E_{ctx}, and a scoring function f. A sequence of frames is first encoded to a sequence of frame representations $z_i = E_{frm}(x_i)$ using the frame encoder. The encoded sequence is then passed to the recurrent context encoder to obtain a sequence of context representations $(c_1, c_2, ..., c_T)$, where c_i is a fixed-dimensional representation computed from $E_{ctx}(z_1, z_2, ..., z_i)$. The scoring function assigns a positive scalar to a pair of frame and context, formulated as $f(x, h) = \exp\left(z^T W c\right)$, where z is the frame representation of x, and c is the context representation of h.

Suppose the target frame is n steps away. Given a context h_i, the target future frame x_{i+n}, and a collection of negative frames $\tilde{X}$, CPC jointly optimizes the three modules by minimizing a contrastive loss:

$$\mathcal{L}_n(h_i, x_{i+n}, \tilde{X}) = \log \frac{f(x_{i+n}, h_i)}{\sum_{x \in \tilde{X} \cup \{x_{i+n}\}} f(x, h_i)}. \tag{3.4}$$

As shown in Oord et al. (2018), minimizing this loss will result in $f(x, h)$ estimating the density ratio $p_n(x \mid h)/q(x)$, where p_n denotes the conditional distribution of x at n steps ahead of the given context h, and q is the proposal distribution where negative samples are drawn from. In other words, the choice of the number of steps ahead and the proposal distribution would both affect the estimated target density ratio, and therefore would change what is learned in the representations z and c. For example, using a proposal distribution that draws samples from the same sequence as the target frame would encourage the model to learn the phonetic content but not the speaker information, because the latter do not help distinguishing a target frame from negative ones. We will study such differences in our experiments.

Both CPC and the proposed APC consider the sequential structures of speech,

and predict information about future frames. However, the two models differ significantly in the type of information the corresponding loss function enforces them to capture. While CPC representations are encouraged to focus on information that is most discriminative between the target and negative frames, APC has to encode information sufficient for predicting the target frame, and are allowed to only discard information that is common across the train dataset.

3.3 Experiments

In this section, we empirically demonstrate the effectiveness of the learned representations from the proposed APC model. Since phone and speaker information are two of the most important characteristics that differentiate one speech utterance from another, we choose to use phone classification and speaker verification to examine how much phone and speaker information are captured by the representations.

3.3.1 Experimental Data and Setup

We use the LibriSpeech corpus (Panayotov et al., 2015) for training the feature extractors (all APC and CPC models). Specifically, the 360-hour subset, which contains 921 speakers in total, is used. We use 80-dimensional log Mel spectrograms (normalized to zero mean and unit variance per speaker) as input features.

An ideal feature extractor should extract representations that generalize to datasets of different domains. To examine the robustness to shift in domains, rather than on the LibriSpeech test set, we conduct phone classification and speaker verification on the Wall Street Journal (WSJ) (Paul and Baker, 1992) and TIMIT corpora. For phone classification, we follow the standard split of WSJ, use 90% of `si284` for training, use the rest of the 10% for development, and report numbers on `dev93`. The phone alignments are generated with a speaker adapted GMM-HMM model. For speaker verification, we follow the standard split of TIMIT, use the training set for training the universal background model, the i-vector extractor (Dehak et al., 2011), a linear discriminant analysis (LDA) model. We follow the standard practice of speaker

verification and only consider female-female and male-male pairs in the 50-speaker development set. We note that speaker verification on TIMIT is not common, and we mainly use it to check if the representations contain speaker information.

We model our APC with a multi-layer unidirectional LSTM (Hochreiter and Schmidhuber, 1997) network with residual connections (He et al., 2016) between two consecutive layers as is done in Wu et al. (2016), and the dimensionality of each layer is 512. For CPC, we follow the implementation for the context encoder and the scoring function in Oord et al. (2018), but change the acoustic feature x from a window of 400 samples (25ms) to a 80-dimensional vector of Mel spectra computed from that segment, and replace the 5-layer strided Convolutional Neural Network with a 3-layer, 512-dim fully-connected neural network with ReLU activations for the frame encoder. Such modification aims for a fairer comparison between APC and CPC models in terms of their training objectives, while eliminating the source of variation due to the choice of acoustic features. All APC and CPC models (except cpc-ctx-exhaust, which we will describe more below) are trained for 100 epochs using the Adam optimizer (Kingma and Ba, 2015) with a batch size of 32 and an initial learning rate of 10^{-3}.

Note that the proposed approach is unsupervised, and we do not and should not tune hyperparameters according to the downstream tasks. The goal of hyperparameter tuning is to show how the hyperparameters affect what is learned in the speech representations. Recall that we define the accessibility of categorical information as the linear separability among classes. For phone classification, we simply use a linear classifier to predict the phoneme classes for each frame. The frame error rates indicate how much phonetic content is contained in the speech representations. Similarly, for speaker verification, we train an LDA model on top of the speech representations.

3.3.2 Phone Classification

Table 3.1 compares APC with a series of CPC models that use different training variants. Phone error rates (PER) are reported, and each of the first four rows corresponds to a CPC variant. We use cpc-n9all to denote a CPC model that draws 9

Table 3.1: Comparing APC with a series of CPC models on phone classification. PERs are reported.

Method	#(step)			
	2	5	10	20
cpc-n9all	51.3	48.8	50.8	54.6
cpc-n9same	47.5	48.2	50.0	53.0
cpc-ctx-n9same	42.1	46.1	48.8	53.8
cpc-ctx-exhaust	42.9	43.1	45.6	49.1
apc (proposed)	36.5	35.6	35.4	37.7

negative samples from utterances within the same minibatch, and cpc-n9same to denote a CPC model that draws 9 negative samples from the same utterance. For both cpc-n9all and cpc-n9same, we take the outputs of the frame encoder (i.e., the outputs of the 3-layer fully-connected neural networks) as the extracted features and feed them to the linear classifier. The training approach of cpc-ctx-n9same is the same as cpc-n9same, except that the RNN outputs are taken as the extracted features instead of the frame encoder outputs. We use ctx, short for context, to indicate such difference. The final CPC variant we try is cpc-ctx-exhaust, which follows the exact same training procedure in Oord et al. (2018) that combines contrastive losses for all steps $k \leq n$ with equal weights for training (i.e., $\sum_{k=1}^{n} \mathcal{L}_n$), uses all non-target samples in a minibatch as negative samples, and are trained with mini-batches of 8 utterances that are randomly chuncked to 128 frames each. For APC, the outputs of the last RNN layer are taken as the extracted features. All models in Table 3.1 consist of one RNN layer, and the effect of predicting different time steps ahead is also investigated.

A comparison of models from the CPC-family. From Table 3.1, we observe that cpc-n9same outperforms cpc-n9all across all time steps we try. This is an expected outcome, since for cpc-n9all, the negative samples are drawn from different utterances within a minibatch that could possibly be uttered by different speakers, and thus cpc-n9all is not required to really capture phonetic content to differentiate the positive and negative samples. In contrast, cpc-n9same draws negative samples from the same

Table 3.2: PERs on phone classification. All features are fed to a linear classifier unless otherwise stated. The number of steps to the target #(steps) is not relevant in the first four rows.

Method	#(step)					
	1	2	3	5	10	20
Mel	50.0					
Mel + MLP-1	43.4					
Mel + MLP-3	41.3					
cpc best	42.1					
apc 1-layer	39.4	36.5	35.4	35.6	35.4	37.7
apc 2-layer	38.5	34.6	35.9	35.7	34.6	38.8
apc 3-layer	37.2	36.7	33.5	36.1	37.1	38.8
apc 4-layer	36.2	34.4	34.5	35.3	36.9	39.6

utterance, and in such case, speaker information is identical for each sample and cpc-n9same is forced to learn other non-trivial features such as phone information so as to differentiate positive and negative samples. In addition, we find that representations extracted from RNN contain more phonetic content than those extracted from the frame encoder, as cpc-ctx-n9same often outperforms cpc-n9same especially when the number of steps to the target is small. By using all non-target samples as negative samples from the minibatch, cpc-ctx-exhaust further lowers the PER, suggesting that richer phonetic content is learned in the representations.

Comparing CPC with APC. Our APC, as shown in the last row in Table 3.1, significantly outperform all CPC models in spite of its much simpler architecture and training approach. These results demonstrate that more phonetic content is immediately accessible from a linear classifier in the representations extracted by APC compared to CPC models.

There are other aspects of APC worth investigating. In Table 3.2, we present the phone classification results of using deeper RNNs for APC and with more target time steps. For all APC models, we take the outputs of the last RNN layer as the extracted features. Three supervised baselines, a linear classifier, a 1-layer multi-layer perceptron (MLP), and a 3-layer MLP, are implemented, taking the *surface features*, i.e., spectrograms, as input features. For MLPs, each layer consist of 512

units with ReLU activations. These three baselines are meant to help us understand how accessible the phonetic content is from the surface features, even under some amount of nonlinear transformations. We also include the best number of CPC models from Table 3.1 to bridge the two tables.

Surface features with non-linear phone classifier. From Table 3.2, we observe that incorporating non-linearity in the phone classifier does improve PER[1]. When using a 3-layer MLP as the classifier, the surface features are transformed into higher-level representations that are more linearly separable than the best CPC features. However, we can see there is still a significant gap between the transformed spectrogram representations with features extracted by APC models.

A comparison of APC models. Overall, we find that deeper APC models produce better representations especially for small #(steps). There also exists a sweet spot when we vary the amount of time steps to the target for APC models to predict—the PER continues to drop as we increase #(steps) until a certain point, which is usually when #(steps) equals 3; after that the PER begins to increase as #(steps) increases.

3.3.3 Speaker Verification

For speaker verification, we compare APC with the i-vector representation. We train a GMM with 256 components as the universal background model on the TIMIT training set. We then extract 100-dimensional i-vectors and project them down to 24 dimensions with LDA trained on the training set. The cosine similarity is used for evaluation. We also include the best results from all CPC models. The equal error rates (EER) on speaker verification are presented in Table 3.3. Same as what we do in the phone classification experiments, the outputs of the last RNN layer are taken as the extracted representations. The representation of the entire utterance is a simple average of the frame representations. For the last two rows , i.e., apc 3-layer-1 and apc 3-layer-2, it means that we take the outputs of the first and the second RNN layer as the extracted representations. We explain our motivation of doing so below.

[1]The best performing supervised 3-layer LSTM with minimal lookahead on this particular task can achieve 16.3 (Tang and Glass, 2018).

Table 3.3: EER on speaker verification. The number of steps to the target #(steps) is not relevant for the first two rows.

Method	#(step)					
	1	2	3	5	10	20
i-vector	6.64					
cpc best	5.00					
apc 1-layer	4.71	4.07	4.14	4.14	5.14	5.29
apc 2-layer	4.71	4.64	5.71	4.86	5.57	6.07
apc 3-layer	5.21	4.93	4.43	4.57	5.79	6.21
apc 3-layer-1	3.43	3.86	3.79	3.86	4.07	4.86
apc 3-layer-2	3.79	4.64	4.14	4.29	5.14	5.00

Comparing APC with i-vector and CPC. From Table 3.3, we can see that the best CPC model outperforms the i-vector baseline, and APC further outperform CPC when #(steps) is smaller than 10. This demonstrates that representations learned by APC contain not only phonetic information but also speaker information.

Speaker information across different APC layers. Unlike phone classification, where we find increasing the depth of APC improve PER, deeper APC somehow performs worse in speaker verification. Studies have shown that in a deep LM, lower layers tend to focus more on local syntax, while the upper layers usually induce more semantic content (Peters et al., 2018a). Motivated by the fact that LMs for text could exhibit different kinds of information across different layers, we are interested in investigating whether other layers besides the last one contain more information of our interest, that is, the speaker information. Specifically, instead of taking the outputs of the last RNN layer of apc 3-layer, we try using the outputs of the first and second RNN layers of it to perform speaker verification, denoted by apc 3-layer-1 and apc 3-layer-2 in Table 3.3, respectively. Surprisingly, for all #(steps), we see that apc 3-layer-1 consistently outperforms apc 3-layer-2, which further outperforms apc 3-layer. This indicates that lower layers indeed contain more speaker information than higher layers, or at least the speaker information is represented in a more accessible form in lower layers. Additionally, we observe that apc 3-layer-1 outperforms apc 1-layer and apc 3-layer-2 outperforms apc 2-layer although the representations are extracted

from the same RNN depth. Combining all of our observations from both tasks, we conclude that a deep APC is a very powerful speech feature extractor, whose higher layers capture phonetic information while more speaker information resides in its lower layers.

3.4 Chapter Summary

In this chapter, we have proposed Autoregressive Predictive Coding (APC) for self-supervised speech representation learning. The backbone of APC is a deep LSTM network, and the model is trained in an autoregressive fashion. We introduce a time shifting factor that asks the model to predict further steps ahead of the current frame during training in order to encourage it to discover more general structures rather than the local ones within the speech signal. Our experimental results show that the number of steps to the target frame controls what is learned in the representation. How this hyperparameter is set depends on how the representation is going to be used and can be thought of as a prior. Despite its simplicity, APC has demonstrated a strong capability of extracting useful phone and speaker information through our experiments, outperforming the surface features and other supervised and self-supervised approaches such as CPC on phone classification and speaker verification. Transfer learning from large-scale pre-trained language models has shown great success recently, and we believe it is promising and useful to develop similar transfer learning techniques for the domain of speech and audio. APC proposed in this chapter sets our initial step towards this goal.

Although we have shown the effectiveness of APC in learning representations that are useful in phone and speaker classification tasks through empirical results, the reason why this seemingly unrelated self-supervised task can learn such representation remains unclear. In the next Chapter, we will investigate this question, with the goal of bridging the connection between the self-supervised task and the properties of representations it learns.

Chapter 4

Understanding Future Prediction: A Study on Why It Is Useful for Speech Representation Learning

In Chapter 3, we proposed Autoregressive Predictive Coding (APC) and showed that APC, as a self-supervised objective, has enjoyed success in learning representations from large amounts of unlabeled data, and the learned representations are rich for many downstream tasks. However, the connection between low self-supervised loss and strong performance in downstream tasks remains unclear. In this chapter, we propose Vector-Quantized Autoregressive Predictive Coding (VQ-APC), a novel model that produces quantized representations, allowing us to explicitly control the amount of information encoded in the representations. By studying a sequence of increasingly limited models, we reveal the constituents of the learned representations. In particular, we confirm the presence of information with probing tasks, while showing the *absence* of information with mutual information, uncovering the model's preference in preserving speech information as its capacity becomes constrained. We find that there exists a point where phonetic and speaker information are amplified to maximize a self-supervised objective. As a byproduct, the learned codes for a particular model capacity correspond well to English phones.

The content of this chapter was first published in Chung et al. (2020).

4.1 Introduction

Recall that Autoregressive Predictive Coding (APC) defines a prediction task that trains an autoregressive neural model (e.g., a unidirectional RNN or a Transformer decoder) to predict a future frame considering the past context. Although the learned representations contain highly accessible phonetic and speaker information, the reason why this seemingly unrelated self-supervised objective produces such a representation remains unclear. In this chapter, we aim to provide an explanation, investigating the constituents that lead to low objective values, and connect them with the properties of the learned representations.

The proposed approach is to study the properties of the learned representation as we limit the model capacity. The models with limited capacity are forced to retain information to achieve maximal prediction, thereby allowing us to study the constituents of the task and the learned representations. Several options are available to obtain a spectrum of models with different capacity, including reducing the number of layers, reducing the hidden layer size, or enforcing a bottleneck along the feed-forward process. The impact of different numbers of hidden layers has been studied in the previous chapter. Regardless, it is difficult to quantify the amount of information by changing the number of layers, changing the hidden layer size, or using low-rank matrices to enforce bottlenecks. In this chapter, we study the use of vector quantization (VQ), where the amount of information (i.e., bits required to transmit the codebook and the sequence of codes) can be exactly quantified, to control the capacity of the models.

Recent studies on VQ for representation learning, mostly motivated by the discrete nature of phonetic units, attempt to show that enforcing the quantization leads to a better representation for acoustic unit discovery (Oord et al., 2017; Harwath et al., 2020) and automatic speech recognition (Baevski et al., 2020a; Liu et al., 2020a). In contrast, our goal is not to discover the discrete units in speech. We treat VQ as a general approach to limit the model capacity, and study its impact on the information encoded in the learned representations.

4.2 Vector-Quantized Autoregressive Predictive Coding

Given a sequence of acoustic feature vectors $(x_1, x_2, ..., x_t)$ as context, APC incorporates an autoregressive neural model g_{AR}, e.g., a unidirectional RNN or a Transformer decoder (Liu et al., 2018; Radford et al., 2018), to summarize the sequence for predicting a future frame x_{t+n} that is n steps ahead of x_t. Let y_t denote the prediction of g_{AR} at time t. In practice, for a speech utterance $\mathbf{x} = (x_1, x_2, ..., x_T)$, where T is the sequence length, g_{AR} is trained by minimizing the $\ell 1$ frame-wise loss between the predicted sequence $(y_1, y_2, ..., y_{T-n})$ and the target sequence $(x_{1+n}, x_{2+n}, ..., x_T)$:

$$\sum_{t=1}^{T-n} |x_{t+n} - y_t|. \tag{4.1}$$

Once g_{AR} is trained, one can extract features by taking its hidden representations, e.g., the last layer output, to replace the surface features as the new input to downstream models.

Figure 4-1 illustrates the VQ-APC architecture, which is based upon APC with additional quantization layer(s). Assume g_{AR} has L layers. Let $g_{AR}^{(\ell)}$ denote the ℓ-th layer of g_{AR}. After feeding $\mathbf{x} = (x_1, x_2, ..., x_T)$ to g_{AR}, each $g_{AR}^{(\ell)}$ will produce a sequence of hidden vectors $\mathbf{h}^{(\ell)} = (h_1^{(\ell)}, h_2^{(\ell)}, ..., h_T^{(\ell)})$. Next, we add a vector quantization (VQ) layer (Oord et al., 2017) that replaces $h_t^{(\ell)}$ by $z_t^{(\ell)}$, where $z_t^{(\ell)}$ is one of the V elements in a codebook $\{c_1, \ldots, c_V\}$. We then pass the resulting hidden vectors $\mathbf{z}^{(\ell)} = (z_1^{(\ell)}, z_2^{(\ell)}, \ldots, z_T^{(\ell)})$ to the next layer $g_{AR}^{(\ell+1)}$ and continue the feed-forward process.

We use the Gumbel-Softmax with the straight-through estimator (Jang et al., 2017) for selecting discrete codebook variables in a fully differentiable way. Specifically, we apply a linear layer to map $h_t^{(\ell)}$ to a vector $r \in \mathbb{R}^V$. At test time, we simply choose the largest index in r. At training time, the probability p_i of selecting the i-th

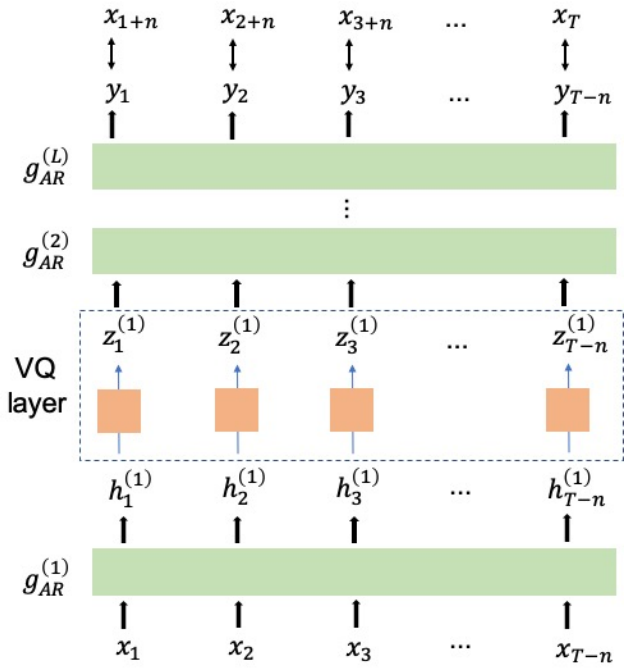


Figure 4-1: A diagram of VQ-APC. g_{AR} is an autoregressive model with L layers with $g_{AR}^{(\ell)}$ denoting the ℓ-th layer and $h_t^{(\ell)}$ denoting the output vector of $g_{AR}^{(\ell)}$ at time t. The figure is an example of inserting a VQ layer (area inside the dashed box) between the first and second layers $g_{AR}^{(1)}$ and $g_{AR}^{(2)}$. The orange block represents the code variable lookup process that replaces $h_t^{(\ell)}$ by $z_t^{(\ell)}$, where $z_t^{(\ell)}$ is one of the elements in a codebook. The quantized hidden vectors are fed to the next layer and the feed-forward process continues. The training objective is the same as APC: to minimize the $\ell 1$ loss between the predicted frame y_t and the target future frame x_{t+n}.

code variable c_i is computed as follows:

$$p_i = \frac{e^{(r_i+v_i)/\tau}}{\sum_{j=1}^{V} e^{(r_j+v_j)/\tau}}, \tag{4.2}$$

where $v = -\ln(-\ln(u))) \in \mathbb{R}^V$ and u is uniformly sampled from $\mu(0,1)$. τ controls how close the approximation is to argmax. During the forward pass, the argmax code c_k where $k = \text{argmax}_i p_i$ is chosen; during the backward pass, the true gradients of the Gumbel-Softmax outputs are used. The training objective is the same as APC (Equation 4.1).

The codebook size and the code dimension of a VQ layer control the amount of information from the previous layer flowing to the next, and changing either of these

two factors allows us to explicitly control the capacity of the models. As we limit the model capacity, the model is forced to retain information to achieve maximal prediction. By studying a sequence of increasingly limited models, we are able to reveal the constituents of the prediction task and the learned representations.

4.3 Experiments

We conduct experiments to study the properties of the learned representations and their connection to the self-supervised objective. Since VQ layers are known to significantly disrupt model training, we first examine where VQ layer(s) should be inserted. Next, by using phonetic and speaker classification as probing tasks, we show the model's preference in preserving speech information as its capacity becomes constrained. We then visualize the learned VQ codes to show the presence and absence of phonetic information and the correspondence between codes and phones. Finally, we compare VQ-APC with other self-supervised speech representation models.

4.3.1 Experimental Data and Setup

Training of self-supervised models. All self-supervised models, including the VQ-APC variants and other models to be compared, are trained on the clean 360-hour subset of LibriSpeech (Panayotov et al., 2015). We use 80-dimensional log Mel spectrograms (normalized to zero mean and unit variance per speaker) as input features, that is, $x_t \in \mathbb{R}^{80}$, and train all models for 100 epochs using Adam (Kingma and Ba, 2015) with a batch size of 32 and an initial learning rate of 10^{-3}.

Probing tasks. We consider phonetic and speaker classification for measuring the accessibility of the phonetic and speaker information contained in the representations, respectively. Both experiments are carried out on the Wall Street Journal corpus (WSJ) (Paul and Baker, 1992). For phonetic classification, the goal is to correctly classify each frame in an utterance into one of the 42 phones. The phone alignments are generated with a speaker adapted GMM-HMM model. We follow the

Table 4.1: Phonetic classification results of VQ-APC with different VQ configurations. The layers where VQ is inserted are denoted as a set, and $\emptyset$ is equivalent to the regular APC. We compare both the hidden vectors $\mathbf{h}^{(\ell)}$ and their corresponding VQ codes $\mathbf{z}^{(\ell)}$ (when applicable) for $\ell = 1, 2, 3$ as extracted features. Training loss on LibriSpeech is also reported. The lowest phone error rate is highlighted in bold.

VQ config.	Train loss	Phone error rate					
		$\mathbf{h}^{(1)}$	$\mathbf{h}^{(2}$	$\mathbf{h}^{(3)}$	$\mathbf{z}^{(1)}$	$\mathbf{z}^{(2)}$	$\mathbf{z}^{(3)}$
$\emptyset$	0.68	46.5	38.6	33.3	–	–	–
$\{1\}$	0.70	43.0	37.5	32.3	43.4	–	–
$\{2\}$	0.72	45.8	35.4	31.5	–	36.0	–
$\{3\}$	0.73	46.4	35.7	**30.5**	–	–	30.8
$\{1, 2\}$	1.22	75.0	72.3	70.7	74.8	72.6	–
$\{1, 3\}$	0.83	59.9	54.1	51.0	61.2	–	51.4
$\{2, 3\}$	0.87	63.1	58.7	54.7	–	59.9	55.2
$\{1, 2, 3\}$	1.21	75.3	74.1	68.5	75.4	75.2	67.8

standard split of WSJ, using 90% of `si284` for training, the rest 10% for validation, and reporting phone error rate on `dev93`. For speaker classification, the goal is to correctly predict the speaker identity of an utterance. We consider a 259-class classification task where each class corresponds to an unique speaker, using 80% of `si284` for training, the other 10% for validation, and reporting classification error rate on the rest 10%. We note that speaker classification is not a typical task for WSJ, and only serves as a sanity check for the presence of speaker information (and its potentially correlated channel information) (Oord et al., 2018; Liu et al., 2020b). The classifier for both tasks is a linear logistic regression that takes the features extracted from the self-supervised models as input. For speaker classification, the features from the same utterance are averaged before being fed to the classifier. All self-supervised models are kept frozen when training the linear classifier. All reported numbers are an average of 5 runs, of which variances are negligibly small and not included.

4.3.2 Preliminary Experiments with Vector Quantization

We first explore several potential places to insert VQ layers. For all VQ-APC variants in our experiments, the autoregressive model g_{AR} is a 3-layer unidirectional GRU

with 512 hidden units, and the target frame in the future, n, is set to 5 when training (Equation 4.1) on LibriSpeech. Whenever a VQ layer is added, the embedding dimension of each code is 512, and τ for the Gumbel-Softmax straight-through estimator (Equation 4.2) is a fixed value of 0.1 throughout training.

Table 4.1 presents the phonetic classification results of adding VQ layers to different layers in g_{AR}. In the "VQ config." column, the numbers inside the parenthesis denote the layers we insert a VQ layer. For example, {1} means that we only add VQ layer after $g_{AR}^{(1)}$. $\emptyset$ denotes the case where no VQ layer is applied, equivalent to the regular APC. The codebook size here is 128. We try using both the hidden vectors $\mathbf{h}^{(\ell)}$ and their quantized codes $\mathbf{z}^{(\ell)}$ (when applicable) for $\ell = 1, 2, 3$ as extracted features. We also include the final VQ-APC training loss on the LibriSpeech 360-hour subset after 100 epochs (not the downstream linear classifier's training loss on WSJ).

Quantizing one layer. As indicated by the training loss, we see that the bottleneck imposed by the VQ layer indeed handicaps the models' ability to predict the future, as {1}, {2}, and {3} all have higher training loss than $\emptyset$. In terms of phone error rate, regardless of where VQ is inserted, we see improvement over the APC representations. Inserting VQ at the third layer leads to the most improvement, from 33.3 to 30.5. The quantized codes $\mathbf{z}^{(\ell)}$, when applicable, could also be used as extracted features, which perform similarly to their corresponding pre-quantized representations. For example, in {3}, $\mathbf{z}^{(3)}$ (30.8) is close to $\mathbf{h}^{(3)}$ (30.5).

Quantizing multiple layers. We find that our VQ-APC models with multiple VQ layers have trouble fitting the training set. Their representations are also much worse than the regular APC on phonetic classification. One potential remedy is to enable VQ with a schedule (Harwath et al., 2020), but is beyond the scope of the thesis.

4.3.3 Analysis of the Constituents of Representations

Experiments so far suggest that the phonetic information is still present (if not better) after using VQ. For the rest of the chapter, we will focus on the case where VQ is inserted at the third layer, i.e., the case of {3}. To study the constituents of the

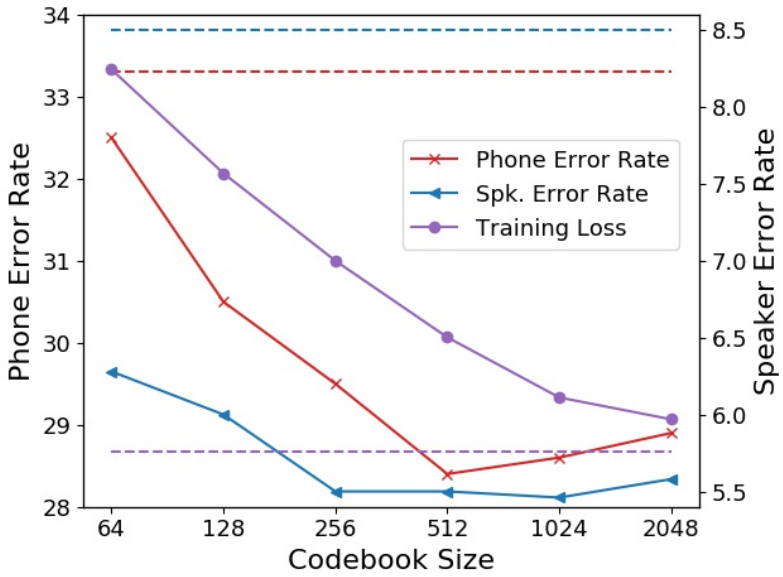


Figure 4-2: Training loss (purple), phonetic classification (red), and speaker classification (blue) results of VQ-APC with VQ configuration {3} using varying codebook sizes. The x-axis is the codebook size, decreasing from 2048 to 64, and the y-axis on the left is the corresponding phone error rate and on the right the speaker (spk.) error rate. For clarity, the vertical axis for training loss is not displayed. The three dash horizontal lines show the corresponding results of a regular APC, i.e., $\emptyset$.

learned representations, we train a series of increasingly limited VQ-APC models by decreasing the codebook size from 2048 to 64 while fixing the code dimension to 512. As the codebook size becomes smaller, the model is forced to choose what information to encode and what to discard, thus revealing the constituents of the learned representations. We show the training losses of these models at convergence and the respective phone and speaker error rates in Figure 4-2. The dashed lines are the training loss, phone error rate, and speaker error rate of a regular APC model.

First, the training loss (purple curve) increases as expected, showing worse fit on the training set as we limit the codebook size. Note that in theory, when the codebook size goes to infinity, we recover the regular APC. The phone error rate (red curve) obtains a minimum at codebook size 512, and starts to worsen with smaller codebook size. The sharp degradation in phone error rate suggests that the model discards certain phonetic information to achieve maximal self-supervised objective.

The speaker error rate (blue curve), on the contrary, does not change by much as

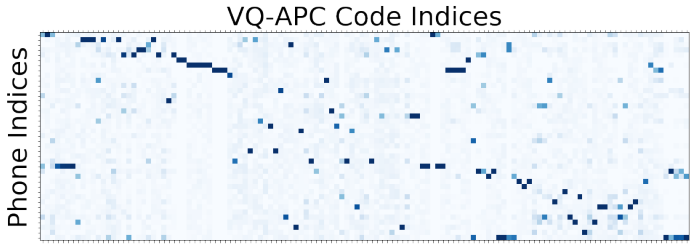
we limit the codebook size. This shows that the speaker information (and its potentially correlated channel information) is mostly retained. Given the sharp degradation in phone error rate, we can conclude that the model prefers to retain speaker information over phonetic information to achieve maximal future prediction. The preference of information can potentially stem from the use of GRUs, the VQ configuration, and the self-supervised, future prediction objective. More analyses are needed to disentangle the among these causes.

On the other hand, when the codebook size becomes large, the model falls back to regular APC and might suffer from overfitting, paying unnecessary attention to the spectral details that does not generalize for predicting future frames. Finally, even with a codebook size of 64, we still see gains over regular APC, showing the strong performance of VQ-APC in representation learning.

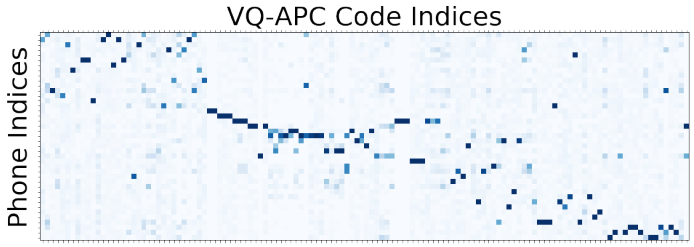
4.3.4 Relation of Learned Codebook to English Phones

To better measure the correspondence between the learned VQ codes and English phones, we compute co-occurrence statistics (at the frame level) across the 360-hour subset of LibriSpeech, the dataset we use to train the VQ-APC models. We compare three settings, {1}, {2}, and {3} with a codebook size of 128. The conditional probability $P(\text{phone}|\text{code})$, as shown in Figure 4-3, are estimated based on the co-occurrence statistics, i.e., via maximum likelihood. In each sub-figure, the rows and columns are ordered via spectral co-clustering with 15 clusters to group together phones that share similar sets of codes, and a diagonal segment would imply a high correspondence between phones and codes. Note that the phone labels of LibriSpeech are only used for analysis and never seen during training.

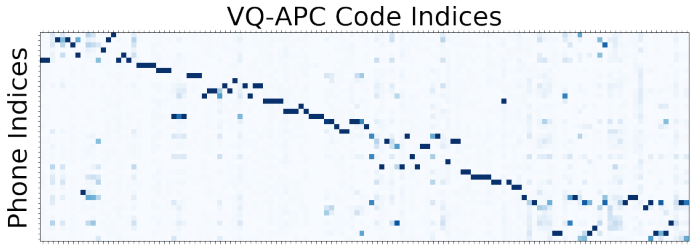
From Figure 4-3, we see that the correspondence between phones and VQ codes is stronger when quantized at higher layers, and is especially strong for {3}. Recall that probing tasks are useful for showing the presence of certain information, but have difficulty showing the absence of it. In contrast, given the co-occurrence statistics, mutual information can be estimated to support the absence of information. The normalized mutual information are 0.167, 0.285, and 0.406 for {1}, {2}, and {3},



(a) VQ-APC with configuration {1}.



(b) VQ-APC with configuration {2}.



(c) VQ-APC with configuration {3}.

Figure 4-3: From top to bottom, visualizations of the conditional probability matrix $P(\text{phone}|\text{code})$ for configurations {1}, {2}, and {3} with 128 codes, respectively. Each sub-figure is a 42×128 conditional probability matrix, where each row and column correspond to a phone and code index, respectively. Color scaling is saturated at probability 0.5 for better visualization.

respectively. In other words, not only can we conclude that the learned representations in {3} contain phonetic information, we can also readily conclude that {1} and {2} contain much less information for certain phones.

4.3.5 A Comparison with Other Self-Supervised Models

Finally, we compare VQ-APC with other self-supervised speech representation models, including Contrastive Predictive Coding (CPC) (Oord et al., 2018), Bidirectional Masked Reconstruction (Wang et al., 2020), and Mockingjay (Liu et al., 2020b). We briefly review these methods below, and show their results on phonetic and speaker

Table 4.2: Phonetic and speaker classification results of different self-supervised speech representation models. All features are fed to a linear logistic repression. For log Mel, we also include the results of using a 1- and 3-layer multi-layer perceptron, denoted as MLP-1 and MLP-3, respectively. We also note the neural architectures used by each model.

Model	Network	Phone error rate	Speaker error rate
log Mel	–	50.3	17.6
log Mel + MLP-1	–	43.1	12.3
log Mel + MLP-3	–	41.2	11.9
CPC	3-layer uni-GRU	34.1	9.7
APC	3-layer uni-GRU	33.3	8.5
VQ-APC	3-layer uni-GRU	28.4	5.5
Wang et al. (2020)	3-layer bi-GRU	32.4	6.2
Liu et al. (2020b)	3-layer Transformer	30.8	5.1

classification in Table 4.2. To stay as close to the original implementation as possible, we do not separate the discussion of models, such as the use RNNs or Transformers, and the self-supervised objectives.

CPC and APC share a similar methodology as both use an autoregressive model to learn representations through conditioning on past frames to predict information about a future frame. Their difference is that while APC tries to directly predict the future frame via regression, CPC aims to learn representations containing information that most discriminates the future frame from a set of negative samples. We mainly follow the original paper (Oord et al., 2018) for implementing CPC with some modifications described in Section 3.3 in the previous Chapter. These modifications are meant to minimize the architectural differences between APC and CPC while maintaining their training objectives.

Different from CPC and APC that are based on the idea of future prediction, Bidirectional Masked Reconstruction and Mockingjay are under the category of *masked prediction*. Inspired by the masked language modeling technique from BERT (Devlin et al., 2019), both approaches mask parts of the input signals, and predict them through conditioning on both past and future contexts with a bidirectional RNN

and Transformer encoder (Vaswani et al., 2017), respectively. For experiments, we mainly follow the implementations in the original papers (Wang et al., 2020; Liu et al., 2020b), except that the number of layers are reduced to match ours to minimize the architectural differences.

On phonetic classification, we see that VQ-APC (28.4) improves over APC (33.3), demonstrating the effectiveness of VQ layers. It also outperforms other self-supervised models despite using the same (vs. CPC) or smaller (vs. Bidirectional Masked Reconstruction and Mockingjay) network. On speaker classification, VQ-APC (5.5) again improves over APC (8.5), and is on par with the best model (Mockingjay, 5.1). On both tasks, all self-supervised models outperform log Mel regardless of the type of classifier it uses.

4.4 Chapter Summary

In this chapter, we have demonstrated that incorporating vector quantization (VQ) layers into an Autoregressive Predictive Coding model imposes a bottleneck, forcing the model to learn better representations, as measured by their performance on phonetic and speaker classification tasks. Extensive experiments have been conducted to compare different VQ configurations, to study the effect of varying codebook sizes, and to compare with other self-supervised speech representation models. We show evidence for the presence and absence of phonetic and speaker information in the learned representations, and also show the model's preference in retaining information when the model capacity is limited, in the hope to bridge the connection between the self-supervised objective and the properties of the learned representations. When the phonetic information is present, the learned VQ codes also correspond well with English phones.

Chapter 5

Improving the Generalization of Future Prediction

In the previous chapter, we proposed VQ-APC to study the connection between the APC objective and the properties of its learned representations, providing an explanation to why such self-supervised future prediction task is capable of learning speech representations that capture rich phonetic and speaker information. In this chapter, we shift our focus to improving APC so that it can learn even stronger speech representations.

Recall that the objective of APC is to train an autoregressive RNN/Transformer model to predict an unseen future frame given a context such as recent past frames. The basic hypothesis behind this self-supervised task is that the model is required to produce a good summarization of the past and encode such knowledge in the hidden states so as to accomplish the objective. In this chapter we extend this hypothesis and aim to enrich the information encoded in the hidden states by training the model to make more accurate future predictions. To accomplish the goal, we propose an auxiliary objective that serves as a regularization to improve generalization of the future frame prediction task. Experimental results on phonetic classification, speech recognition, and speech translation not only support the hypothesis, but also demonstrate the effectiveness of our approach in learning representations that contain richer phonetic content.

The content of this chapter was first published in Chung and Glass (2020b).

5.1 Proposed Method

5.1.1 Formulation

Given a context of a speech signal represented as a sequence of acoustic feature vectors $(x_1, x_2, \ldots, x_t)$, the objective of Autoregressive Predictive Coding (APC) is to use the context to infer a future frame x_{t+n} that is n steps ahead of x_t. Let $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ denote a full utterance, where T is the sequence length, APC incorporates an RNN to process each frame x_t sequentially and update its hidden state h_t accordingly. For $t = 1, \ldots, T - n$, the RNN produces an output $y_t = \mathbf{W} \cdot h_t$, where $\mathbf{W}$ is an affinity matrix that maps h_t back to the dimensionality of x_t. The model is trained by minimizing the frame-wise $\ell 1$ loss between the predicted sequence $(y_1, y_2, \ldots, y_{T-n})$ and the target sequence $(x_{1+n}, x_{2+n}, \ldots, x_T)$ (which cane be generated by simply right-shifting $\mathbf{x}$ by n steps):

$$L_f(\mathbf{x}) = \sum_{t=1}^{T-n} |x_{t+n} - y_t|. \tag{5.1}$$

When $n = 1$, one can view APC as an *acoustic* version of neural LM (NLM) (Mikolov et al., 2010) by viewing each acoustic frame as a token embedding, as they both use a recurrent encoder and aim to predict information about the future. A major difference between NLM and APC is that NLM infers tokens from a closed set, while APC predicts frames of real values. Once an APC model is trained, given an utterance $(x_1, x_2, \ldots, x_T)$, the last RNN layer $(h_1, h_2, \ldots, h_T)$ is taken as the extracted features.

Our goal is to make APC's prediction of x_{t+n} given h_t more accurate. In Section 5.2 we will show this leads to a representation that contains richer phonetic content.

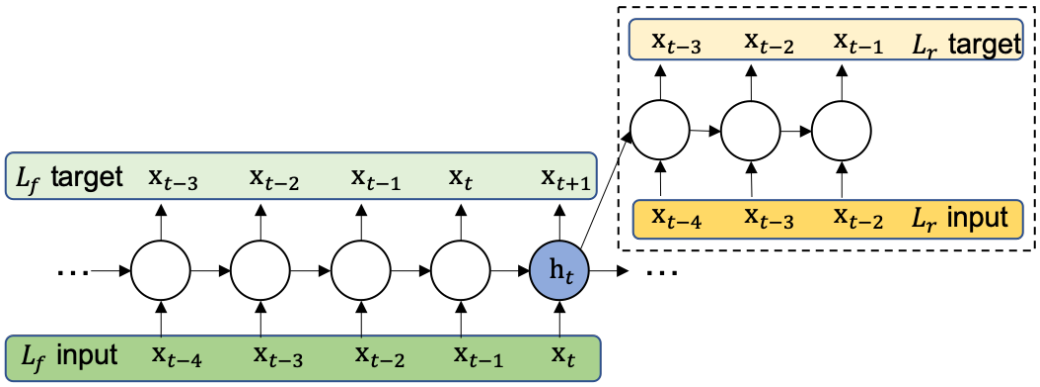


Figure 5-1: Overview of our method. L_f is the original APC objective that aims to predict x_{t+n} given a context $(x_1, x_2, \ldots, x_t)$ with an autoregressive RNN. Our method first samples an anchor position, assuming it is time step t. Next, we build an auxiliary loss L_r that computes L_f of a past sequence $(x_{t-s}, x_{t-s+1}, \ldots, x_{t-s+\ell-1})$ (see Section 5.1.2 for definitions of s and ℓ), using an auxiliary RNN (dotted line area). In this example, $(n, s, \ell) = (1, 4, 3)$. In practice, we can sample multiple anchor positions, and averaging over all of them gives us the final L_r.

5.1.2 Remembering More from the Past

An overview of our method is depicted in Figure 5-1. We propose an auxiliary loss L_r to improve the generalization of the main objective L_f (Equation 5.1).

The idea of L_r is to refresh the current hidden state h_t with the knowledge learned in the past. At time step t, we first sample a past sequence $\mathbf{p}_t = (x_{t-s}, x_{t-s+1}, \ldots, x_{t-s+\ell-1})$, where s is how far the start of this sequence is from t and ℓ is the length of $\mathbf{p}_t$. We then employ an auxiliary RNN, denoted as RNN_{aux}, to perform predictive coding defined in Equation 5.1 conditioning on h_t. Specifically, we initialize the hidden state of RNN_{aux} with h_t, and optimize it along with the corresponding $\mathbf{W}_{\text{aux}}$ using $L_f(\mathbf{p}_t)$, which equals to $\sum_{t'=t-s}^{t-s+\ell-1} |x_{t'+n} - y_{t'}|$. Such a process reminds h_t of what has been learned in $h_{t-s}, h_{t-s+1}, \ldots, h_{t-s+\ell-1}$.

For a training utterance $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, we select each frame with probability P as an anchor position. Assume we end up with M anchor positions: $a_1, a_2, \ldots, a_M$. Each a_m defines a sequence $\mathbf{p}_{a_m} = (x_{a_m-s}, x_{a_m-s+1}, \ldots, x_{a_m-s+\ell-1})$ before x_{a_m}, which we use to compute $L_f(\mathbf{p}_{a_m})$. Averaging over all anchor positions gives the final aux-

iliary loss L_r:

$$L_r(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} L_f(\mathbf{p}_{a_m}). \tag{5.2}$$

The final APC objective is a combination of Equations 5.1 and 5.2 with a balancing coefficient λ:

$$L_m(\mathbf{x}) = L_f(\mathbf{x}) + \lambda L_r(\mathbf{x}). \tag{5.3}$$

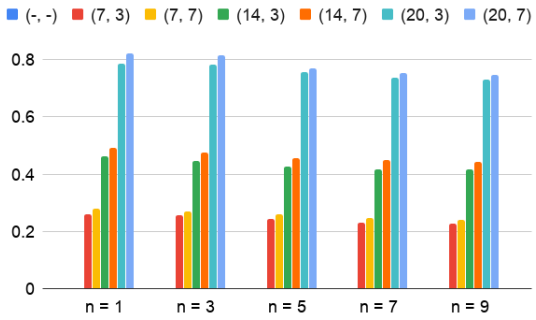
We re-sample the anchor positions for each $\mathbf{x}$ during each training iteration, while they all share the same RNN_{aux} and $\mathbf{W}_{\text{aux}}$.

5.2 Analysis Experiments

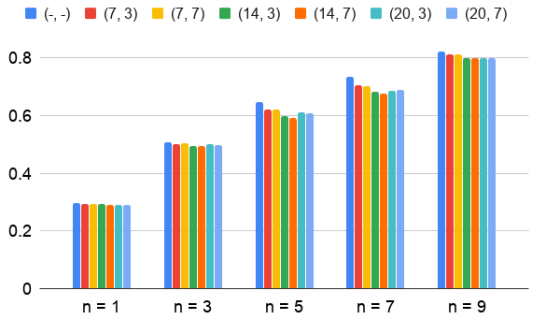
We demonstrate the effectiveness of L_r in helping optimize L_f, and investigate how the improvement is reflected in the learned representations.

5.2.1 Experimental Data and Setup

We follow the same setup as Chapter 3 and use the audio portion of the LibriSpeech (Panayotov et al., 2015) `train-clean-360` subset, which contains 360 hours of read speech produced by 921 speakers, for training APC. The input features are 80-dimensional log Mel spectrograms, i.e., $x_t \in \mathbb{R}^{80}$. Both RNN and RNN_{aux} are a 3-layer, 512-dim unidirectional GRU (Cho et al., 2014a) network with residual connections between two consecutive layers (Wu et al., 2016). Therefore, $\mathbf{W}, \mathbf{W}_{\text{aux}} \in \mathbb{R}^{512\times 80}$. λ is set to 0.1 and the sampling probability P is set to 0.15, that is, each frame has a 15% of chance to be selected as an anchor position. P and λ are selected based on the validation loss of L_f on a small data split. All models are trained for 100 epochs using Adam (Kingma and Ba, 2015) with a batch size of 32 and a learning rate of 10^{-3}.



(a) L_r (auxiliary objective, Equation 5.2)



(b) L_f (main objective, Equation 5.1)

Figure 5-2: Validation loss of L_r (left) and L_f (right) on LibriSpeech `dev-clean` when training APC using different (n, s, ℓ) combinations. Each bar of the same color represents one (s, ℓ) combination. We use $(-, -)$ to denote an APC optimized only with L_f. Bars are grouped by their n's with different (s, ℓ) combinations within each group.

5.2.2 Validating the Effectiveness of L_r

We first validate whether augmenting L_r improves L_f. As a recap, n is the number of time steps ahead of the current position t in L_f, and s and ℓ denote the start and length, respectively, of a past sequence before t to build L_r. We consider $(n, s, \ell) \in \{1, 3, 5, 7, 9\} \times \{7, 14, 20\} \times \{3, 7\}$. Note that each phone has an average duration of about 7 frames.

Figures 5-2a and 5-2b present L_r (before multiplying λ) and L_f of the considered APC variants on the LibriSpeech `dev-clean` subset, respectively. Each bar of the same color represents one (s, ℓ) combination. We use $(-, -)$ to denote an APC optimized only with L_f. Bars are grouped by their n's with different (s, ℓ) combinations within each group.

We start with analyzing Figure 5-2a. Note that L_r does not exist for $(-, -)$ and is set to 0 in the figure. We see that under the same n, the performance of L_r is mainly decided by how far (s) the past sequence is from the current position rather than the length (ℓ) to generate: when we keep ℓ fixed and increase s from 7 (red), 14 (green), to 20 (blue), we observe the loss surges as well. Moving our focus to Figure 5-2b, we have the following findings.

For a small n, the improvement in L_f brought by L_r is minor. By comparing $(-,-)$ with other bars, we see that when $n \leq 3$, which is smaller than half of the average phone duration (7 frames), adding L_r does not lower L_f by much. We speculate that when $n \leq 3$, x_{t+n} to be inferred is usually within the same phone as x_t, making the task not challenging enough to force the model to leverage more past information.

L_r becomes useful when n gets larger. We see that when n is close to or exceeds the average phone duration ($n \geq 5$), an evident reduction in L_f after adding L_r is observed, which validates the effectiveness of L_r in assisting with the optimization of L_f. When $n = 9$, the improvement is not as large as when $n = 5$ or 7. One possible explanation is that x_{t+9} has become almost independent from the previous context h_t and hence is less predictable.

By observing the validation loss, we have shown that L_r indeed helps generalize L_f.

5.2.3 Analysis of the Learned Representations

Next, we want to examine whether an improvement in L_f leads to a representation that encodes more useful information. Speech signals encompass a rich set of acoustic and linguistic properties. Here we will only focus on analyzing the phonetic content contained in a representation, and leave other properties such as speaker for future work.

We use phonetic classification on TIMIT (Garofolo et al., 1993) as the probing task to analyze the learned representations. The corpus contains 3696, 400, and 192 utterances in the train, validation, and test sets, respectively. For each $n \in \{1, 3, 5, 7, 9\}$, we pick the (s, ℓ) combination that has the lowest validation loss. As described in Section 5.1.1, we take the output of the last RNN layer as the extracted features, and provide them to a linear logistic regression classifier that aims to correctly classify each frame into one of the 48 phone categories. During evaluation, we follow the protocol (Lee and Hon, 1989) and collapse the prediction to 39 categories. We report

Table 5.1: Phonetic classification results using different types of features as input to a linear logistic regression classifier. The classifier aims to correctly classify each frame into one of the 48 phone categories. Frame error rates (↓) are reported. Given a time shift $w \in \{0, \pm 5, \pm 10, \pm 15\}$, the classifier is asked to predict the phone identity of x_{t+w} given x_t.

Feature	Time shift						
	-15	-10	-5	0	+5	+10	+15
log Mel	83.3	80.3	67.6	49.9	65.5	77.9	82.7
APC trained with L_f (Equation 5.1)							
$n = 1$	56.1	45.8	36.1	33.7	56.5	73.7	81.6
$n = 3$	50.8	41.8	34.8	33.4	56.0	73.5	81.1
$n = 5$	48.7	38.2	32.5	31.9	54.8	73.0	80.5
$n = 7$	44.6	38.6	32.9	32.1	56.3	73.8	80.4
$n = 9$	51.0	41.8	35.7	36.9	58.4	74.6	81.0
APC trained with L_m (Equation 5.3)							
$n = 1$	50.6	42.2	35.1	33.1	54.4	73.4	81.4
$n = 3$	46.4	38.0	34.1	32.4	54.1	71.4	80.5
$n = 5$	41.8	35.1	29.8	28.1	49.6	64.6	76.8
$n = 7$	39.8	33.8	28.7	27.8	46.8	60.6	74.4
$n = 9$	42.3	35.3	30.3	29.7	50.0	63.3	76.6

frame error rate (FER) on the test set, which indicates how much phonetic content is contained in the representations. We also conduct experiments for the task of predicting x_{t-w} and x_{t+w} given x_t for $w \in \{5, 10, 15\}$. This examines how contextualized h_t is, that is, how much information about the past and future is encoded in the current feature h_t. We simply shift the labels in the dataset by $\{\pm 5, \pm 10, \pm 15\}$ and retrain the classifier. We keep the pre-trained APC RNN fixed for all runs. Results are shown in Table 5.1.

We emphasize that our hyperparameters are chosen based on L_f and are never selected based on their performance on any downstream task, including phonetic classification, speech recognition, and speech translation to be presented next. Tuning hyperparameters towards a downstream task defeats the purpose of unsupervised learning.

Phonetic classification We first study the standard phonetic classification results, shown in the column where time shift is 0. We see that APC features, regardless of the objective (L_f or L_m), achieve lower FER than log Mel features, showing that the phonetic information contained in the surface features has been transformed into a more accessible form (defined as how linearly separable they are). Additionally, we see that APC features learned by L_m outperform those learned by L_f across all n. For $n \geq 5$ where there is a noticeable improvement in future prediction after adding L_r as shown in Figure 5-2b, their improvement in phonetic classification is also larger than when $n \leq 3$. Such an outcome suggests that APC models that are better at predicting the future do learn representations that contain richer phonetic content. It is also interesting that when using L_f, the best result occurs at $n = 5$ (31.9); while with L_m, it is when $n = 7$ that achieves the lowest FER (27.8).

Predicting the past or future We see that it is harder to predict the nearby phone identities from a log Mel frame, and the FER gets higher further away from the center frame. An APC feature h_t contains more information about its past than its future. The result matches our intuition as the RNN generates h_t conditioning on h_i for $i < t$ and thus their information are naturally encoded in h_t. Furthermore, we observe a consistent improvement in both directions by changing L_f to L_m across all n and time shifts. This confirms the use of L_r, which requires the current hidden state h_t to recall what has been learned in previous hidden states, so more information about the past is encoded in h_t. The improvement also suggests that an RNN can forget the past information when training only with L_f, and adding L_r alleviates such problem.

5.3 Speech Recognition and Speech Translation Results

The series of phonetic classification experiments in the previous section are meant for analyzing the phonetic properties of a representation. Finally, we apply the represen-

Table 5.2: Automatic speech recognition (ASR) and speech translation (ST) results using different types of features as input to a seq2seq with attention model. Word error rates (WER, ↓) and BLEU scores (↑) are reported for the two tasks, respectively.

Feature	ASR (WER ↓)	ST (BLEU ↑)
log Mel	18.3	12.9
APC w/ L_f	15.2	13.8
APC w/ L_m	14.2	14.5

tations learned by L_m to automatic speech recognition (ASR) and speech translation (ST) and show their superiority over those learned by L_f.

We follow the exact same setup in (Chung and Glass, 2020a). For ASR, we use the Wall Street Journal corpus (Paul and Baker, 1992), use `si284` for training, and report the word error rate (WER) on `dev93`. For ST, we use the LibriSpeech En-Fr corpus (Kocabiyikoglu et al., 2018), which aims to translate an English speech to a French text, and report the BLEU score (Papineni et al., 2002). For both tasks, the downstream model is an end-to-end, sequence-to-sequence RNN with attention (Chorowski et al., 2015). We compare different input features to the same model. Results, shown in Table 5.2, demonstrate that the improvement in predictive coding brought by L_r not only provides representations that contain richer phonetic content, but are also useful in real-world speech applications.

5.4 Chapter Summary

In this chapter, we have improved the generalization of Autoregressive Predictive Coding (APC) by multi-target training of future prediction L_f and past memory reconstruction L_r, where the latter serves as a regularization. Through phonetic classification, we find the representations learned with our approach contain richer phonetic content than the original representations, and achieve better performance on speech recognition and speech translation.

From Chapter 3 to Chapter 5, we carried out a series of studies on APC trying to understand how, why, and when it works. We started with proposing the initial

algorithm in Chapter 3. In Chapter 4 we developed VQ-APC in order to help us explore the connection between the self-supervised objective and the properties of its learned representations. Finally, in Chapter 5 we designed an auxiliary objective to improve APC's generalization with the goal of making APC a better representation learning algorithm. APC and CPC are two of the pioneering works in self-supervised speech representation learning, exploiting the same learning methodology, which is future prediction. In the next Chapter, we will introduce w2v-BERT, which makes use of a different learning methodology: predicting the masked from the unmasked.

Chapter 6

Self-Supervised Objective: Predicting the Masked from the Unmasked

Motivated by the success of masked language modeling (MLM) in pre-training natural language processing models, in this chapter we propose w2v-BERT, which explores MLM for self-supervised speech representation learning. w2v-BERT is a framework that combines contrastive learning and MLM, where the former trains the model to discretize input continuous speech signals into a finite set of discriminative speech tokens, and the latter trains the model to learn contextualized speech representations via solving a masked prediction task consuming the discretized tokens. In contrast to existing MLM-based speech pre-training frameworks such as HuBERT, which relies on an iterative re-clustering and re-training process, or vq-wav2vec, which concatenates two separately trained modules, w2v-BERT can be optimized in an end-to-end fashion by solving the two self-supervised tasks (the contrastive task and the MLM task) simultaneously. Our experiments show that w2v-BERT achieves competitive results compared to current state-of-the-art pre-trained models on the LibriSpeech benchmarks when using the Libri-Light 60k corpus as the unsupervised data. In particular, when compared to published models such as Conformer-based wav2vec 2.0 and HuBERT, our model shows 5% to 10% relative Word Error Rate reduction on the test-clean and test-other subsets. When applied to the Google's Voice Search traffic dataset, w2v-BERT outperforms the Conformer-based wav2vec 2.0 by more

than 30% relatively.

This work was done during a summer internship at Google, and was first published in Chung et al. (2021b).

6.1 Introduction

In this chapter, we propose a pre-training framework called w2v-BERT, which combines the core methodologies from two recent frameworks for self-supervised pre-training of speech and language respectively: wav2vec 2.0 (Baevski et al., 2020b) and BERT (Devlin et al., 2019). The idea of w2v-BERT is to use the contrastive task defined in wav2vec 2.0 to obtain an inventory of a finite set of discriminative, discretized speech units, and then use them as target in a masked prediction task in a way that is similar to masked language modeling (MLM) proposed in BERT for learning contextualized speech representations. Although the masked prediction task requires to consume tokens that are to be learned by solving the contrastive task first, we show that in practice the two objectives can be optimized simultaneously. Figure 6-1 illustrates the w2v-BERT pre-training framework.

In this chapter, we make the following contributions:

- We propose w2v-BERT that directly optimizes a contrastive loss and a masked prediction loss simultaneously for end-to-end self-supervised speech representation learning.
- We show that w2v-BERT yields state-of-the-art performance on the well-benchmarked LibriSpeech task.
- We show that w2v-BERT greatly improves a real-world recognition task (voice search) over Conformer-based wav2vec 2.0.
- We provide an analysis that empirically confirms the necessity of contrastive learning for enabling masked prediction in our framework. We also show in our voice search experiments that mask prediction is very useful for alleviating the problem of "easy negative samples" in contrastive learning.

The rest of the chapter is organized as follows. We begin with discussing the differences between w2v-BERT and some of the most relevant unsupervised speech pre-training frameworks from the literature in Section 6.2. Then, in Section 6.3 we present the w2v-BERT pre-training framework, including the model architecture and training objectives. Section 6.4 describes the experimental setup, followed by our results and analysis in Section 6.5 where we apply pre-trained w2v-BERT models to LibriSpeech and voice search ASR.

6.2 Related Work

We consider our work most related to HuBERT (Hsu et al., 2021), vq-wav2vec (Baevski et al., 2020a), and DiscreteBERT (Baevski et al., 2019): w2v-BERT and these methods all try to first transform continuous speech signals into discretized units so as to exploit masked language modeling (MLM) (Devlin et al., 2019) for learning contextualized speech representations. Despite sharing this same high-level philosophy for learning speech representations, there are two key differences between w2v-BERT and other methods.

The most noticeable difference is that w2v-BERT's speech discretizing module and its main contextualized representation learning module can be trained end-to-end. This is in contrast to vq-wav2vec and DiscreteBERT, which involve a two-stage process where the speech discretizing module needs to be obtained in advance and is kept frozen during the training of the representation learning module. In vq-wav2vec and DiscreteBERT, a problematic token ID assignment would negatively affect the subsequent learning module and it is hard for the learning module to recover the errors made by the discretizer. Observing such drawback, HuBERT greatly improves vq-wav2vec and DiscreteBERT by allowing refinement on the ID assignment via iterating between k-means clustering and re-training its representation learning module. However, the fact that HuBERT iterates between the two stages also means it involves more heuristic design choices, for example, the gradually increasing number of clusters in different iterations. End-to-end methods such as w2v-BERT alleviate the

need of coordinating multiple stages. One potential risk for end-to-end approaches compared to k-means clustering is codebook collapse. In w2v-BERT, we find the contrastive learning objective effectively avoids codebook collapse and thus enables masked prediction training.

In addition, unlike other methods that use transformer layers (Vaswani et al., 2017) as building blocks, w2v-BERT adopts Conformer layers (Gulati et al., 2020) for constructing the network. As demonstrated in Gulati et al. (2020), Conformer layers, which combine convolution neural networks (CNNs) and transformers to model both local and global dependencies of audio sequences, are likely a better option for modeling speech than transformer layers and CNNs. That being said, using a potentially more powerful building block is not the only factor that makes w2v-BERT outperform other methods, as the effectiveness of the pre-training framework itself is also validated in our experiments where w2v-BERT outperforms w2v-Conformer (Zhang et al., 2020b), which is also built with Conformer layers.

w2v-BERT is also related to wav2vec 2.0 (Baevski et al., 2020b). Same as w2v-BERT, wav2vec 2.0 is end-to-end where the discretizer is jointly trained with its representation learning module. However, wav2vec 2.0 only employs contrastive learning, whose resulting ASR performance lags behind that of combining contrastive learning and masked prediction.

6.3 Method

In this section we present each component in w2v-BERT, starting with its model architecture.

6.3.1 Model Architecture

Our model architecture for pre-training is composed of a feature encoder that extracts latent speech representations from raw acoustic inputs, a module for solving wav2vec 2.0's contrastive task (Baevski et al., 2020b) to obtain a set of discretized speech tokens, and a module for solving a masked prediction task (Devlin et al., 2019)

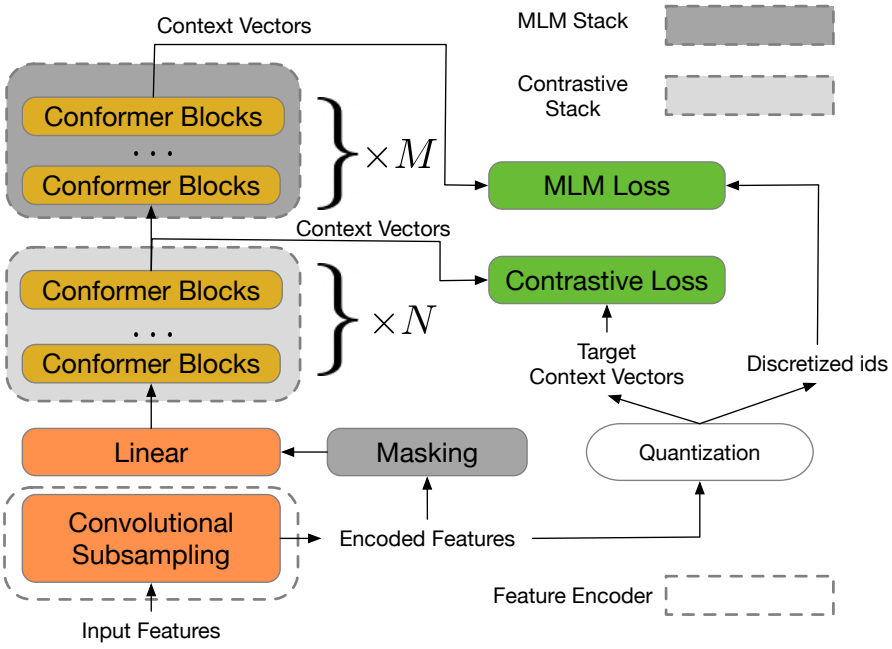


Figure 6-1: Illustration of the w2v-BERT pre-training framework. w2v-BERT is composed of a feature encoder, a contrastive module, and a masked language modeling (MLM) module, where the latter two are both a stack of Conformer blocks. N and M denote the number of Conformer blocks in the two modules, respectively.

for learning contextualized speech representations.

Feature encoder The feature encoder acts as a convolutional subsampling block that consists of two 2D-convolution layers, both with strides $(2, 2)$, resulting in a 4x reduction in the acoustic input's sequence length. Given, for example, a log-mel spectrogram as input, the feature encoder extracts latent speech representations that will be taken as input by the subsequent contrastive module.

Contrastive module The module contains a linear projection layer followed by a stack of Conformer blocks (Gulati et al., 2020), each of which is a series of multi-headed self attention (Vaswani et al., 2017), depth-wise convolution and feed-forward layers.

The goal of the contrastive module is to discretize the feature encoder output into a finite set of representative speech units. For this purpose, the contrastive module involves a quantization mechanism. The output of the feature encoder, on one hand, is fed into the linear projection layer followed by the stack of Conformer

blocks *after masking* to produce context vectors, and on the other hand, is passed to the quantizer *without masking* to yield quantized vectors and their assigned token IDs. The quantized vectors are used in conjunction with the context vectors that correspond to the masked positions to solve the contrastive task defined in wav2vec 2.0 to optimize the contrastive module; the assigned token IDs will be later used by the subsequent masked prediction module as prediction target.

Masked prediction module The masked prediction module is a stack of Conformer blocks where each block has an identical configuration to those from the contrastive module. The module directly takes in the context vectors produced by the contrastive module and extracts high-level contextualized speech representations.

6.3.2 Pre-Training Methods

During pre-training only unlabeled speech data is used.

Contrastive loss The contrastive loss is used to train the contrastive module along with the quantizer, such that the former yields adequate context vectors that will be taken as input by the subsequent masked prediction module, and the latter produces discriminative discretized speech tokens that will be used by the masked prediction module as prediction target. We adopt the contrastive task defined in wav2vec 2.0 and follow its quantization mechanism.

Once the feature encoder has transformed the raw acoustic input into latent speech representations, we randomly select some time steps to mask. Unlike wav2vec 2.0 where the masked positions' latent vectors are replaced with a shared learnable feature vector, we simply replace them with random vectors. The masked feature encoder output is fed into the contrastive module for producing context vectors. In parallel, the feature encoder output is also passed to the quantizer without masking to yield its quantized vectors. For a context vector c_t corresponding to a masked time step t, the model is asked to identify its true quantized vector q_t from a set of K distractors $\{\tilde{q}_1, \tilde{q}_2, ..., \tilde{q}_K\}$ that are also quantized vectors uniformly sampled from other masked time steps of the same utterance. We denote the loss as $\mathcal{L}_w$, and further

augment it with a codebook diversity loss $\mathcal{L}_d$ to encourage a uniform usage of codes. Therefore, the final contrastive loss is defined as:

$$\mathcal{L}_c = \mathcal{L}_w + \alpha \cdot \mathcal{L}_d, \tag{6.1}$$

where $\alpha = 0.1$ following Baevski et al. (2020b).

Masked prediction loss The context vectors produced by the contrastive module are directly passed to the masked prediction module for producing the final context vectors that are to be used to complete a masked prediction task. A softmax layer is appended on top of the module's last Conformer block. If a context vector at the final layer corresponds to a masked position, the softmax layer will take the context vector as input and attempt to predict its corresponding token ID, which is assigned earlier in the contrastive module by the quantizer. We denote the cross-entropy loss for this masked prediction task as $\mathcal{L}_m$.

w2v-BERT is trained to solve the two self-supervised tasks at the same time. The final training loss to be minimized is:

$$\mathcal{L}_p = \beta \cdot \mathcal{L}_c + \gamma \cdot \mathcal{L}_m. \tag{6.2}$$

In our experiments, we simply set both β and γ to 1.

6.3.3 Fine-Tuning Methods

During fine-tuning we have access to labeled data. We apply our pre-trained w2v-BERT to two tasks: LibriSpeech and voice search.

The ASR network is a sequence transducer (Graves, 2012) that consists of a pre-trained w2v-BERT model and a LSTM (Hochreiter and Schmidhuber, 1997) decoder. We insert a linear layer with Swish activation (Ramachandran et al., 2018) and batch normalization (Ioffe and Szegedy, 2015) between the pre-trained w2v-BERT model and the LSTM decoder as the projection block.

Table 6.1: Configurations for w2v-BERT models.

Model	# Params (B)	# Contrastive Layers	# Masked Layers	Model Dimension	# Attention Heads
w2v-BERT XL	0.6	12	12	1024	8
w2v-BERT XXL	1.0	12	30	1024	8

Table 6.2: (Continued) Configurations for w2v-BERT models.

Model	Conv. Layer Kernel Size	Relative Attention	Codebook Size	Code Dimension
w2v-BERT XL	5	No	1024	1024
w2v-BERT XXL	5	No	1024	1024

6.4 Experimental Setup

Apart from the pre-training method, the rest of the experimental pipeline follows the exact same setup as in Zhang et al. (2020b).

6.4.1 Data

We use the Libri-Light unlab-60k subset (Kahn et al., 2020b), which contains about 60,000 hours of unannotated speech audio, for pre-training w2v-BERT models. For our main results, we use the LibriSpeech 960hr subset (Panayotov et al., 2015) as the supervised data, and use the 100hr subset for ablation studies. We report word error rates (WERs) on the dev-clean, dev-other, test-clean, and test-other evaluation subsets. 80-dimensional log-mel filter bank coefficients are used as acoustic inputs to our model. For transcript tokenization, we use a 1024-token WordPiece model (Schuster and Nakajima, 2012) that is constructed from the transcripts of the LibriSpeech training set (or the 100hr subset when the models are fine-tuned on it).

6.4.2 Pre-Training Details

Masking For masking the feature encoder output, we randomly sample the starting positions to be masked with a probability of 0.065 and mask the subsequent 10 time

steps (same as Baevski et al. (2020b); Zhang et al. (2020b)). The masked spans may overlap.

Optimization We pre-train two versions of w2v-BERT models, one has about 0.6 billion parameters and the other has about 1 billion parameters, denoted as w2v-BERT XL and w2v-BERT XXL, respectively. The two variants share the same model configuration that is summarized in Table 6.1 and Table 6.2, and their only difference is the number of Conformer blocks. Specifically, w2v-BERT XL's contrastive module consists of 12 Conformer blocks and the masked prediction module is composed of another 12. w2v-BERT XXL, while having the same amount of Conformer blocks in its contrastive module, enlarges its masked prediction module to 30 Conformer blocks. For w2v-BERT XL, we train it with a batch size of 2048 using the Adam optimizer (Kingma and Ba, 2015) with a transformer learning rate schedule as described in section 5.3 of Vaswani et al. (2017). The peak learning rate is 2e-3 and the warm-up steps are 25k. For w2v-BERT-XXL, we train it with the Adafactor optimizer (Shazeer and Stern, 2018) with $\beta_1 = 0.9$ and $\beta_2 = 0.98$, with the learning rate schedule remaining the same.

6.4.3 Fine-Tuning Details

Optimization For both w2v-BERT XL and w2v-BERT-XXL, we take their pre-trained checkpoints at 400k steps, and fine-tune them on the supervised data with a batch size of 256. The decoder for both models are a two-layer LSTM with a hidden dimension of 640. We employ separate optimizers and learning rate schedules for optimizing the pre-trained model and the decoder, given the fact that the former has been pre-trained while the latter needs to be trained from scratch. For w2v-BERT XL, both the pre-trained model and the decoder are optimized with an Adam optimizer with a transformer learning schedule. The difference is that for the pre-trained component we use a peak learning rate of 3e-4 with 5k warm-up steps, while for the decoder we use a peak learning rate of 1e-3 and 1.5k warm-up steps. For w2v-BERT-XXL, an Adafactor optimizer that has the same configuration as in pre-

training is used, and the learning rate schedules for the encoder and decoder are the same as the XL variant.

Self-training, data augmentation, and LM fusion In addition to self-supervised pre-training, in the fine-tuning stage we also employ a number of practical techniques that further improve models' performance on ASR. These techniques include SpecAugment (Park et al., 2019, 2020a) for data augmentation, Noisy Student Training (Park et al., 2020b) for self-training, and language model fusion for decoding. When any of the techniques are used, we follow the exact same setup as in Zhang et al. (2020b). We refer the readers to the paper for the details on these techniques.

6.5 Results and Discussion

6.5.1 Main Results

In Table 6.3 (which should be read together with its references in Table 6.4), we present our results on the four LibriSpeech evaluation sets using the 960hr subset as the supervised data. We compare w2v-BERT to a number of state-of-the-art self-supervised representation learning methods from the literature such as HuBERT (Hsu et al., 2021) and wav2vec 2.0 (Baevski et al., 2020b) under different semi-supervised settings, including whether self-training is employed during the fine-tuning stage and whether a language model is incorporated during inference time. We also include the model size of the ASR network used by each method, denoted as acoustic model (AM) Size and language model (LM) Size. Results missing from the literature (e.g., results of HuBERT without self-training and LM) are indicated with a "−" in the table. From Table 6.3 we have the following two key conclusions.

Without self-training and LM, w2v-BERT already either outperforms or matches other models with LM. We see that with just pre-training when neither self-training nor LM is used, w2v-BERT XL achieves a WER of 1.5/2.9 (test/test-other), which already either outperforms or matches other models with LM, and outperforms their counterparts without LM by a larger margin. w2v-BERT XXL

Table 6.3: WERs(%) when using the LibriSpeech 960hr subset as supervised data (the table should be read together with Table 6.4). We compare models trained without any unlabeled data (Trained from Scratch), trained using Noisy Student Training (NST) without any pre-training (Self-training Only), fine-tuned from a pre-trained model only using supervised data (Pre-training Only), and the models obtained by combining pre-training and self-training (Pre-training + Self-training). We also include the best results of several methods that we can find from the literature. The lowest WER(s) under different settings are marked in bold.

Method	**No LM**				**With LM**			
	dev	**dev-other**	**test**	**test-other**	**dev**	**dev-other**	**test**	**test-other**
Trained from Scratch								
Conformer L	1.9	4.4	2.1	4.3	–	–	1.9	3.9
Self-training Only								
Conformer L with NST	1.6	3.3	1.7	3.5	1.6	3.1	1.7	3.3
Pre-training Only								
wav2vec 2.0	2.1	4.5	2.2	4.5	1.6	3.0	1.8	3.3
HuBERT Large	–	–	–	–	1.5	3.0	1.9	3.3
HuBERT X-Large	–	–	–	–	1.5	**2.5**	1.8	2.9
w2v-Conformer XL	1.7	3.5	1.7	3.5	1.6	3.2	**1.5**	3.2
w2v-Conformer XXL	1.6	3.2	1.6	3.3	1.5	3.0	**1.5**	3.1
w2v-BERT XL	**1.5**	2.9	**1.5**	2.9	**1.4**	2.8	**1.5**	2.8
w2v-BERT XXL	**1.5**	**2.7**	**1.5**	**2.8**	**1.4**	2.6	**1.5**	**2.7**
Pre-training + Self-training								
wav2vec 2.0	**1.3**	3.1	1.7	3.5	**1.1**	2.7	1.5	3.1
w2v-Conformer XXL	**1.3**	2.7	1.5	2.8	1.3	2.6	**1.4**	2.7
w2v-Conformer XXL+	**1.3**	2.7	1.5	2.7	1.3	2.6	**1.4**	2.6
w2v-BERT XL	**1.3**	2.6	**1.4**	2.7	1.3	2.6	**1.4**	2.6
w2v-BERT XXL	1.4	**2.4**	**1.4**	**2.5**	1.3	**2.4**	**1.4**	**2.5**

further increases the gap on the more challenging dev-other and test-other subsets. Noticeably, compared to wav2vec 2.0, w2v-BERT-XXL shows a relative WER reduction of 28%, 42%, 32%, and 38% on the four evaluation subsets respectively without LM, and 13%/ 13%/ 17%/ 18% when LM is employed.

We want to highlight that although w2v-BERT XL (0.6B) and w2v-BERT XXL (1.0B) have a larger pre-trained model size than wav2vec 2.0 (0.3B), the latter also incorporates a much larger LM (> 0.4B) during self-training and decoding according to Xu et al. (2021); Baevski et al. (2020b). When considering the sum of the two components, wav2vec 2.0 (> 0.7B) actually features a similar (if not bigger) model size as w2v-BERT XL (0.7B).

Contrastive learning combined with masked language modeling is more effective than contrastive learning alone. w2v-Conformer (Zhang et al., 2020b)

Table 6.4: References for Table 6.3. For each method, the corresponding reference is where the numbers are quoted from. AM/LM Size denotes the number of parameters in the acoustic/language model. *The reason why we do not include Conformer XL and Conformer XXL is that, according to Zhang et al. (2020b), simply enlarging Conformer L produces worse results when the model is trained from scratch. †Calculated based on the LM configuration provided in Xu et al. (2021); Baevski et al. (2020b); $>$ because some information such as the token embedding size is not given therefore not included.

Method	Unlabeled Data (hrs)	AM Size (B)	LM Size (B)
Trained from Scratch			
Conformer L (Zhang et al., 2020b)*	N/A	0.1	0.1
Self-training Only			
Conformer L with NST (Zhang et al., 2020b)	60k	0.1	0.1
Pre-training Only			
wav2vec 2.0 (Xu et al., 2021)	60k	0.3	$> 0.4^{\dagger}$
HuBERT Large (Hsu et al., 2021)	60k	0.3	–
HuBERT X-Large (Hsu et al., 2021)	60k	1.0	–
w2v-Conformer XL (Zhang et al., 2020b)	60k	0.6	0.1
w2v-Conformer XXL (Zhang et al., 2020b)	60k	1.0	0.1
w2v-BERT XL (Ours)	60k	0.6	0.1
w2v-BERT XXL (Ours)	60k	1.0	0.1
Pre-training + Self-training			
wav2vec 2.0 (Xu et al., 2021)	60k	0.3	> 0.4
w2v-Conformer XXL (Zhang et al., 2020b)	60k	1.0	0.1
w2v-Conformer XXL+ (Zhang et al., 2020b)	60k	1.1	0.1
w2v-BERT XL (Ours)	60k	0.6	0.1
w2v-BERT XXL (Ours)	60k	1.0	0.1

and w2v-BERT only differ in their pre-training method and have all other aspects in common such as their model size and fine-tuning pipeline. This allows a truly apple-to-apple comparison between the two pre-training methods for their effectiveness in representation learning. Below we briefly describe their differences in pre-training.

w2v-Conformer adopted wav2vec 2.0's contrastive task as the sole pre-training objective, but replaced the quantization module with a linear layer as the authors did not find quantization helpful for improving downstream ASR performance. w2v-BERT, on the other hand, adopts wav2vec 2.0's contrastive task not just for learning contextualized speech representations, but mainly for the purpose of obtaining a codebook that can represent every segment of continuous speech as an discriminative discrete

token, such that we can exploit MLM for learning powerful speech representations. As will be demonstrated in our analysis (Section 2.4), the contrastive loss is essential for making MLM to work.

By comparing the Pre-training Only results of w2v-Conformer and w2v-BERT, we see that w2v-BERT XL, despite having fewer model parameters, already outperforms w2v-Conformer-XXL—the previous state of the art—especially on the dev-other and test-other subsets. When self-training is applied, w2v-BERT XL still either outperforms or matches w2v-Conformer-XXL's results. w2v-BERT-XXL, which is of the same model size as w2v-Conformer XXL, outperforms w2v-Conformer XXL on even more evaluation subsets. These results demonstrate the superiority of the proposed w2v-BERT over existing pre-training frameworks.

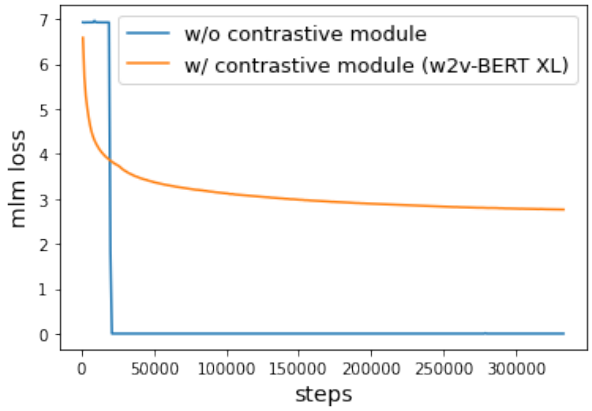
6.5.2 Analysis: On the Necessity of Contrastive Module

The goal of our analysis is to understand the roles of contrastive learning as well as its learned codebook in the w2v-BERT pre-training framework.

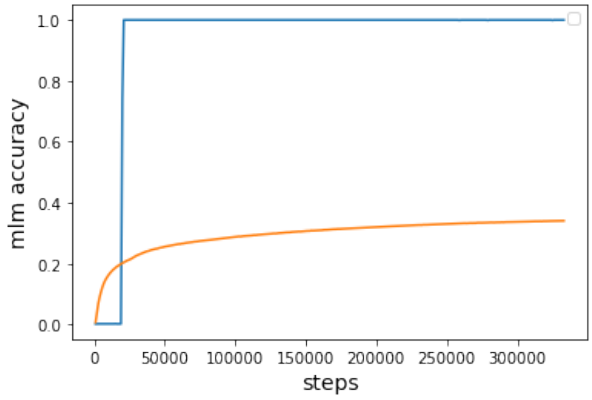
The first natural question is whether the contrastive module is an essential component of the framework, or is that a masked prediction module alone can already derive a suitable codebook for its own MLM purpose.

Without the contrastive module, the feature encoder output is directly fed to the masked prediction module. Intuitively, the masked prediction module then gets a full control over the quantizer (which may originally be viewed as part of the contrastive module) and decide its own prediction target. In order to maximize the prediction performance, the masked prediction module can "cheat" by coming up with a trivial solution where it asks the quantizer to cooperate with it by quantizing all feature encoder's output frames that correspond to the masked positions to the same code vector, in other words, always assigning the same target ID for the masked prediction module to predict. The module thus perfectly solves the masked prediction task without learning any useful representation.

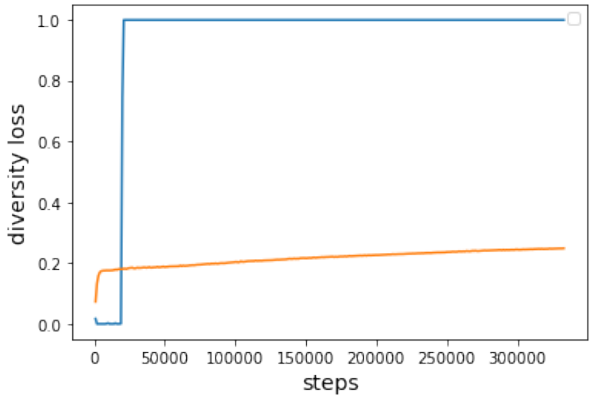
To verify our intuition, we train a series of w2v-BERT models without the contrastive module. These variants all have the same capacity as w2v-BERT XL, that



(a) MLM training loss



(b) MLM training accuracy



(c) Training diversity loss

Figure 6-2: Training curves of w2v-BERT models with and without contrastive module. From top to bottom: MLM training loss, MLM training accuracy, training diversity loss. The blue curve represents the w2v-BERT model without contrastive module, and the orange curve represents w2v-BERT XL (with contrastive module). We show results for the first 300k steps.

is, they are all constructed with 24 Conformer layers. We try different values of $\alpha = 0.1, 0.3, 0.5$, and 0.7 in Equation 6.1 for increasing encouragement of uniform usage of codes. Nevertheless, we find all models are untrainable and quickly collapse regardless of the value of α. In Figure 6-2 we show the training curves of a w2v-BERT model without contrastive module with α set to 0.5. We include the curves of the successfully trained w2v-BERT XL for comparison. The plots include the models' masked prediction loss (Figure 6-2a), masked prediction accuracy (Figure 6-2b), and diversity loss (Figure 6-2c) during pre-training.

We find that the training curves of w2v-BERT without contrastive module (in blue) strongly align with our intuition: the masked prediction loss quickly decreases to close to 0 at the early stage of training (Figure 6-2a) where the model reaches 100% prediction accuracy (Figure 6-2b). Meanwhile, as shown in Figure 6-2c, the diversity loss quickly increases to close to 1, where in our implementation this indicates an extremely low entropy of softmax distribution over the codebook entries, suggesting code collapse. Comparing the curves of w2v-BERT models with and without contrastive module, we hypothesize that the contrastive loss guides the entries in the codebook to be discriminative, thus preventing the masked prediction module from deriving a trivial solution just to maximize masked prediction performance.

6.5.3 Analysis: On the Impact of Contrastive Module's Capacity

After confirming the necessity of contrastive module, next we are interested in investigating the impact of its capacity on downstream ASR performance.

We train a series w2v-BERT models with different numbers of Conformer layers in their contrastive module. To rule out the factor of masked prediction module's capacity, we keep the total number of Conformer layers in the two modules fixed at 24. We use C_n to denote each variant, where n is the number of Conformer layers in the contrastive module. For instance, C_4 has 4 Conformer layers in its contrastive module and 20 in its masked prediction module. Here we consider $n = 2, 4, 6, 8, 10, 12$,

Table 6.5: WERs (%) when using the LibriSpeech 100hr subset as supervised data. For all methods, both self-training and LM fusion are not used. References are where the numbers are quoted from.

Method	**dev**	**dev-other**	**test**	**test-other**
Baseline				
wav2vec 2.0 (Baevski et al., 2020b)	3.3	6.5	3.1	6.3
w2v-Conformer XL (Zhang et al., 2020b)	2.5	4.7	2.6	4.9
w2v-BERT XXL (Ours)	2.3	4.0	2.3	4.3
w2v-BERT w/ 24 layers				
C_2	2.4	5.1	2.5	5.1
C_4	2.5	4.6	2.5	5.1
C_6	2.5	4.2	2.4	4.7
C_8	2.3	4.3	2.4	4.6
C_{10}	2.4	4.5	2.5	4.8
C_{12} (w2v-BERT XL)	2.4	4.4	2.5	4.6
C_{24}	2.4	4.9	2.5	5.0

and 24, where C_{24} is an extreme case where the two modules are completely overlapped with each other and hence the contrastive and MLM tasks will be both tackled at the last (24-th) layer. Note that C_{12} is essentially w2v-BERT XL.

We use the LibriSpeech 100hr subset as the supervised data for this experiment, and both self-training and LM fusion are *not* used when training the ASR network. Results are shown in Table 6.5. We include the results of some pre-training methods from the literature that also do not incorporate self-training and LM.

From Table 6.5 we can roughly observe a performance sweet spot on all four evaluation subsets when we increase the number of layers in the contrastive module. From C_2 to C_8, the WERs are mostly decreasing, meaning that enlarging the contrastive module is helpful for learning better representations. The fact that the performance continues to improve while the masked prediction module shrinks (and hence becomes less expressive) as we deepen the contrastive module further suggests the importance of making the contrastive module sufficiently large.

Starting from C_8, however, the WERs stop decreasing as we deepen the contrastive module. We hypothesize that this is because the masked prediction module has now become too small to learn representations useful for the MLM task. Such reasoning is

Table 6.6: Results on voice search data. Baseline Conformer model is 100M parameters. All the other models are 600M parameters, marked as XL.

Method	Unlabeled Data (Domain)	**Test (VS)**
Conformer	N/A	10.7
w2v-Conformer-XL	34.3k (Voice search)	10.8
w2v-Conformer-XL-tuned	34.3k (Voice search)	8.9
w2v-BERT XL (Ours)	34.3k (Voice search)	**6.2**

supported by the fact that enlarging the masked prediction module while keeping the contrastive module the same size can still improve the performance (w2v-BERT XL vs. w2v-BERT XXL).

Last but not least, we see that w2v-BERT always outperforms wav2vec 2.0 regardless of its layer configuration. It also either outperforms or matches w2v-Conformer XL's performance when its contrastive module has enough capacity (i.e., when $n > 4$).

6.5.4 Results on Voice Search Traffic

So far we have shown w2v-BERT pre-trained on read speech audio can achieve great performance on the well-benchmarked LibriSpeech task. To validate the effectiveness of w2v-BERT on real-world audio traffic, we apply it to Google's Voice Search traffic. Our train and test sets are derived from Li et al. (2021). We use 34.3k hours of English audio for pre-training, and randomly pick 1k hours as the fine-tuning data, which is anonymized and human-transcribed. The test set contains around 12k Voice Search utterances with duration less than 5.5s long. The testing utterances are anonymized and human-transcribed, and are representative of Google's Voice Search traffic.

The traffic data is more challenging to be used for pre-training than read speech audio in two folds: (1) It is noisier and contains more silences that make negative sampling for contrastive learning less effective. (2) The average length of the traffic audio (5 seconds) is much shorter than that of read speech audio. These factors make the context learned from the audio segments much less effective.

As shown in Table 6.6, if we take the same training script as w2v-Conformer-XL,

the model tends to cheat on negative samples due to the large portion of non-speech and shorter context. To make contrastive learning more effective, we have to use a less aggressive subsampling: instead of using a 4 times convolutional stride, we stack 3 frames as target to encourage the model to learn better context. However, by taking an identical architecture and using the same training receipt, our w2v-BERT XL significantly improves the tuned contrastive baseline by relative 30%.

6.6 Chapter Summary

In this chapter, we have proposed w2v-BERT for self-supervised speech representation learning. w2v-BERT is composed of a contrastive module for discretizing continuous speech and a masked prediction module that performs masked language modeling with the discretized speech. The two modules can be jointly optimized. We pre-trained w2v-BERT on 60k hours of unlabeled speech data from the Libri-Light corpus, and showed it either outperforms or matches state-of-the-art systems such as w2v-Conformer, HuBERT, and wav2vec 2.0. The gain also transfers to a more challenging dataset that reflects real-world audio traffic. We also provided an analysis on the importance of the contrastive module for enabling effective masked language modeling.

Chapter 7

Similarity Analysis of Self-Supervised Speech Representations

Self-supervised speech representation learning has recently been a prosperous research topic. Many algorithms have been proposed for learning useful representations from large-scale unlabeled data, and their applications to a wide range of speech tasks have also been investigated. However, there has been little research focusing on understanding the properties of existing approaches. In this work, we aim to provide a comparative study of some of the most representative self-supervised algorithms. Specifically, we quantify the similarities between different self-supervised representations using existing similarity measures. We also design probing tasks to study the correlation between the models' pre-training loss and the amount of specific speech information contained in their learned representations. In addition to showing how various self-supervised models behave differently given the same input, our study also finds that the training objective has a higher impact on representation similarity than architectural choices such as building blocks (RNN/Transformer/CNN) and directionality (uni/bidirectional). Our results also suggest that there exists a strong correlation between pre-training loss and downstream performance for some self-supervised algorithms.

The content of this chapter was first published in Chung et al. (2021a).

7.1 Motivation

Despite the recent progress in self-supervised speech representation learning, most of the effort is made to develop new algorithms or adapt existing methods to particular tasks, and only a few studies focus on reviewing existing approaches. In this work, we aim to provide a comparative study on some of the most representative self-supervised algorithms: contrastive predictive coding (CPC), autoregressive predictive coding (APC), and masked predictive coding (MPC). Our analysis focuses on the following two aspects. First, we hope to understand the similarity of representations learned by different self-supervised algorithms. To carry out this study, we adopt two similarity measures for quantifying the similarity of two given representations (to be more specific, two sequences of vectors). Although such a similarity analysis approach cannot discern absolute facts about the representations, it allows us to compare representations without subscribing to any specific type of information, and helps us answer questions like: Given the same input, how similar are different self-supervised representations? Which modeling choices, e.g., building blocks (RNN/Transformer/CNN) and directionality (uni/bidirectional), have a higher impact on representation similarity? How much does a model change when it is trained on more data?

Our second area of investigation examines, for each self-supervised algorithm, how well its pre-training loss correlates with downstream performance. Our approach is to use phonetic and speaker classification as probing tasks to measure the amount of phonetic and speaker information contained in the representations as a function of pre-training loss. This study could be useful for model selection if there exists a strong correlation between the pre-training loss and the probing task performance.

Only a few studies have focused on analyzing self-supervised models. Chung et al. (2020) propose to incorporate vector quantization layers to restrict model capacity during pre-training so as to uncover a model's preference in preserving speech information for achieving a maximal self-supervised objective. Blandón and Räsänen (2020) study the correlation between the self-supervised loss of APC and CPC and their performance on a phoneme discrimination task, which has the same goal as our

second study. However, neither of these two works investigated the similarity between different self-supervised representations. For the correlation study, we also consider more self-supervised models with diverse modeling choices as compared to Blandón and Räsänen (2020).

Our analysis yields the following insights:

- The objective has a higher impact on representation similarity than model architecture.
- Under the same objective, a model's directionality (uni/bidirectional) affects representation similarity more than its building blocks (RNN/Transformer/CNN).
- Both APC and MPC both have a stronger correlations between pre-training loss and phonetic and speaker classification performance than does CPC.
- While all models benefit from increasing the size of unlabeled training data, CPC is found to make use of these additional data more efficiently than APC and MPC.

The rest of the chapter is organized as follows. We start with introducing our methods for studying the two aspects of our investigation in Section 7.2. Then, in Section 7.3, we describe our experimental setup, including the implementations of the considered self-supervised models, and datasets for pre-training and probing. Experimental results and analysis are presented in Section 7.4, followed by chapter summary in Section 7.5.

7.2 Analysis Methods

We are interested in two aspects of self-supervised speech representation learning: (1) the similarities between representations learned by various models, and (2) how well their self-supervised pre-training loss correlates with downstream performance. We describe our methods for analyzing these two aspects in Sections 7.2.1 and 7.2.2, respectively.

7.2.1 Measuring Representation Similarity

Consider a pre-trained self-supervised model M. For an acoustic feature sequence (in our case, a log Mel spectrogram) $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^{80}$, from a dataset D, the model M transforms $\mathbf{x}$ into a higher-level representation $M(\mathbf{x}) = (\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_T)$, where $\mathbf{m}_t \in \mathbb{R}^{512}$. Given two representations extracted by two self-supervised models $M^{(1)}$ and $M^{(2)}$, a similarity measure outputs $\text{sim}(M^{(1)}(\mathbf{x}), M^{(2)}(\mathbf{x})) \in \mathbb{R}$ that quantifies their similarity. Note that this approach does not require D to be annotated.

Existing similarity measures are proposed to capture different similarity notions. Some focus on capturing the localization of information of two representations, which is usually done by comparing the behaviors of two individual elements $\mathbf{m}_i^{(1)}$ and $\mathbf{m}_j^{(2)}$ from $M^{(1)}(\mathbf{x})$ and $M^{(2)}(\mathbf{x})$, respectively (Bau et al., 2019). Other measures emphasize distributivity of information and find correlations between two representations $M^{(1)}(\mathbf{x})$ and $M^{(2)}(\mathbf{x})$ directly (Kriegeskorte et al., 2008; Kornblith et al., 2019; Raghu et al., 2017; Andrew et al., 2013): if two representations behave similarly over all of their elements, their similarity will be high even if no two individual elements have similar behaviors. In this work we focus on the latter case and adopt linear centered kernel alignment (`lincka`; Kornblith et al. 2019) and singular vector canonical correlation analysis (`svcca`; Raghu et al. 2017) as our similarity measures. We choose these two since they are found to be comparable or better than other measures in prior studies for analyzing contextual word representation models (Wu et al., 2020).

7.2.2 Measuring Phonetic and Speaker Information

We consider phonetic and speaker classification for measuring the amount of accessible phonetic and speaker content contained in a representation. Given a self-supervised model M pre-trained on an unlabeled dataset D_1, we use M to extract features $M(\mathbf{x}) = (\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_T)$, where $\mathbf{m}_t \in \mathbb{R}^{512}$ for another dataset D_2, and train a supervised linear classifier using the extracted features as input.[1]

[1]We adopt the setting where D_1 and D_2 have different distributions to simultaneously examine the richness and robustness against domain shift of a representation. We believe this is a more

For phonetic classification, the goal is to correctly predict the phone identity of each frame in an input utterance. For speaker classification, the extracted features of the utterance are first averaged before being fed to the classifier, and the goal is to correctly predict the speaker identity of the utterance. The frame-level phone error rate and utterance-level speaker error rate on the test set of D_2 indicate the amount of phonetic and speaker content contained in the representation.

7.3 Experimental Setup

7.3.1 Self-Supervised Models

In this work we consider some of the most representative self-supervised models for comparison, including contrastive predictive coding (CPC) (Oord et al., 2018), autoregressive predictive coding (APC) (Chung et al., 2019a), and masked predictive coding (MPC) (Liu et al., 2020b; Wang et al., 2020).

While there are additional models that have successfully been applied to speech applications, most of them are more or less an improvement or extension of the above models. For example, Riviere et al. (2020) improve CPC by modifying its batch normalization mechanism and replacing the linear prediction head with a 1-layer Transformer network. Kawakami et al. (2020b) modify CPC to make it bidirectional. wav2vec (Schneider et al., 2019) is essentially CPC with a fully convolutional architecture and a proposal distribution dedicated for speech recognition. DeCoAR proposed by Ling et al. (2020) can be viewed as a bidirectional version of APC. Chung and Glass (2020b) propose an auxiliary loss serving as a regularizer to help APC generalize better. Liu et al. (2021) apply SpecAugment (Park et al., 2019) to improve MPC's masking techniques. Jiang et al. (2021) combine APC and MPC to form a unified pre-training objective. We leave the explorations of these extensions for future work. Below we briefly review the considered models: CPC, APC, and MPC.

realistic setting than assuming D_1 and D_2 have the same distribution. Our setting is also closer to that in the literature of NLP pre-training models.

CPC & APC Contrastive predictive coding (CPC) and autoregressive predictive coding (APC) share a similar methodology as both use an autoregressive model to learn representations through conditioning on the past context to make predictions of future information. Their main difference lies in the manner in which they optimize the autoregressive model: while APC attempts to predict a future frame via L1 regression, CPC incorporates a proposal distribution for drawing negative samples, and learns representations containing information that most discriminates the future frame from the negative samples using a loss called InfoNCE, which is based on noise-contrastive estimation (Gutmann and Hyvärinen, 2010). We mainly follow the original papers (Oord et al., 2018; Chung et al., 2019a) for implementing the models with small modifications described in Chung et al. (2019a).

Since the objectives of APC and CPC are based on the notion of future prediction, bidirectional architectures are not applicable. A simple method for making these models have access to context from both directions is to separately train a forward and backward APC/CPC model and concatenate their output representations as the final representations (similar to how ELMo (Peters et al., 2018b) is trained for learning contextualized word embeddings). This method has been explored for APC and CPC in Ling et al. (2020) and Kawakami et al. (2020b), respectively.

MPC Inspired by the masked language modeling technique from BERT (Devlin et al., 2019), masked predictive coding (MPC) directly trains a bidirectional architecture by first masking parts of the input signals and then predicting them through conditioning on context from both directions. Similar to APC, MPC is optimized by minimizing the frame-wise L1 distance between the predicted output and the original input before masking. Transformer encoder (Liu et al., 2020b) and bidirectional RNN (Wang et al., 2020) have both been used to implement MPC.

To account for multiple factors in model design (objective, RNN/Transformer/CNN, uni/bidirectional), we consider the implementations of APC, MPC, and CPC as listed in Table 7.1.

Table 7.1: Information about various implementations of APC, MPC, and CPC to be compared in this work. All RNN and Transformer models have a hidden size of 512 (256 for forward and 256 for backward if bidirectional). For CPC, `cpc-mixed_spk-rnn` draws negative samples across speakers, while `cpc-within_spk-rnn` and `cpc-within_spk-cnn` draw negative samples from the same utterance as the target future frame.

Notation	Objective	Building block	Directionality
`apc-fw-rnn`	APC	3-layer GRU	Unidirectional
`apc-fw+bw-rnn`	APC	3-layer GRU	Bidirectional
`apc-fw-trf`	APC	3-layer Transformer decoder	Unidirectional
`apc-fw+bw-trf`	APC	3-layer Transformer decoder	Bidirectional
`mpc-birnn`	MPC	3-layer GRU	Bidirectional
`mpc-trf`	MPC	3-layer Transformer encoder	Bidirectional
`cpc-mixed_spk-rnn`	CPC	3-layer GRU	Unidirectional
`cpc-within_spk-rnn`	CPC	3-layer GRU	Unidirectional
`cpc-within_spk-cnn`	CPC	Same as Schneider et al. (2019)	-

7.3.2 Pre-Training Datasets

We use the LibriSpeech corpus (Panayotov et al., 2015), which contains 960 hours of read speech produced by 2,338 speakers, for pre-training all considered self-supervised models. We also use the `unlab-6k` subset from the Libri-Light corpus (Kahn et al., 2020b), which contains about 6k hours of speech audio produced by 1,742 speakers, for additional experiments in Section 7.4.3. We use 80-dimensional log Mel spectrograms as input acoustic features, i.e., $\mathbf{x}_t \in \mathbb{R}^{80}$. All models are trained for 10 epochs using Adam with a batch size of 32 and an initial learning rate of 10^{-3}. During pre-training, only the speech portion from the dataset is used.

7.3.3 Probing Datasets

Representation similarity measures For calculating representation similarity with `lincka` and `svcca` (described in Section 7.2.1), we use the `si284` subset from the Wall Street Journal corpus (WSJ) (Paul and Baker, 1992) and the train set from the TIMIT corpus (Garofolo et al., 1993).

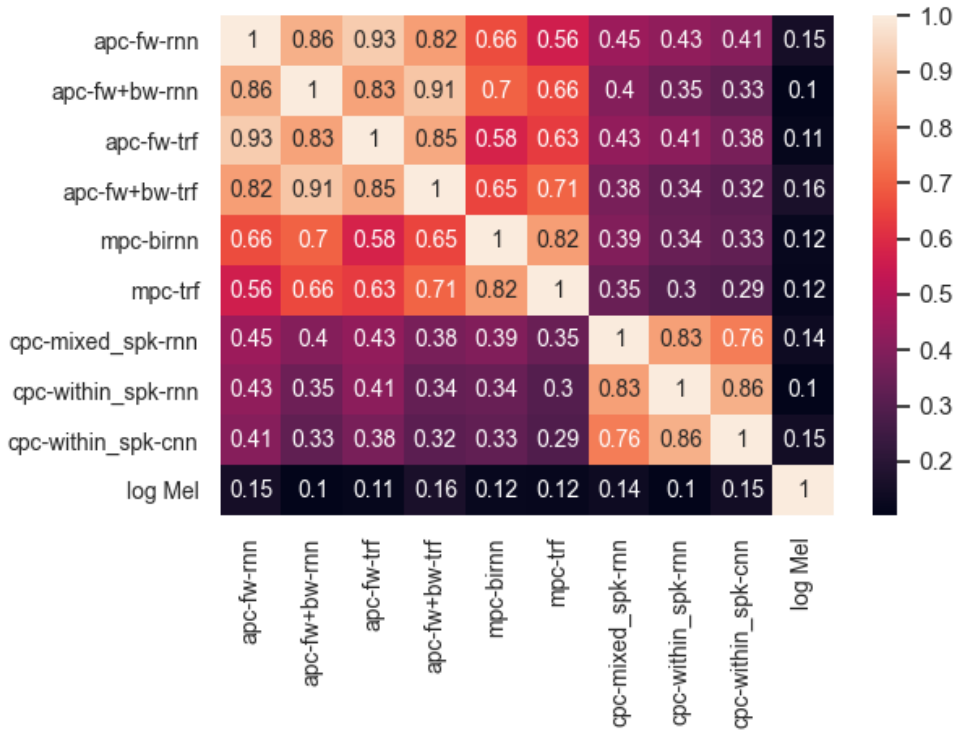


Figure 7-1: Similarity heatmap of various self-supervised representations on WSJ according to `lincka`. Values of similarity are also annotated.

Phonetic and speaker classification We carry out both classification tasks on WSJ. For phonetic classification, there are a total of 42 phone categories, and we follow the standard split of WSJ, using 90% of `si284` for training, 10% for validation, and reporting frame-level phone error rate on `dev93`. The phone alignments are generated with a speaker adapted GMM-HMM model. For speaker classification, we follow Chung and Glass (2020a) and consider a 259-class classification task where each class corresponds to an unique speaker, using 80% of `si284` for training, the other 10% for validation, and reporting utterance-level speaker error rate on the rest 10%. We note that speaker classification is not a typical task for WSJ, and only serves as a sanity check for the presence of speaker information. For both tasks, the classifier is a linear logistic regression trained for 10 epochs using SGD with a batch size of 32 and a fixed learning rate of 10^{-4}. All reported error rates are an average of 5 runs, of which variances are negligibly small and not included.

7.4 Results and Analysis

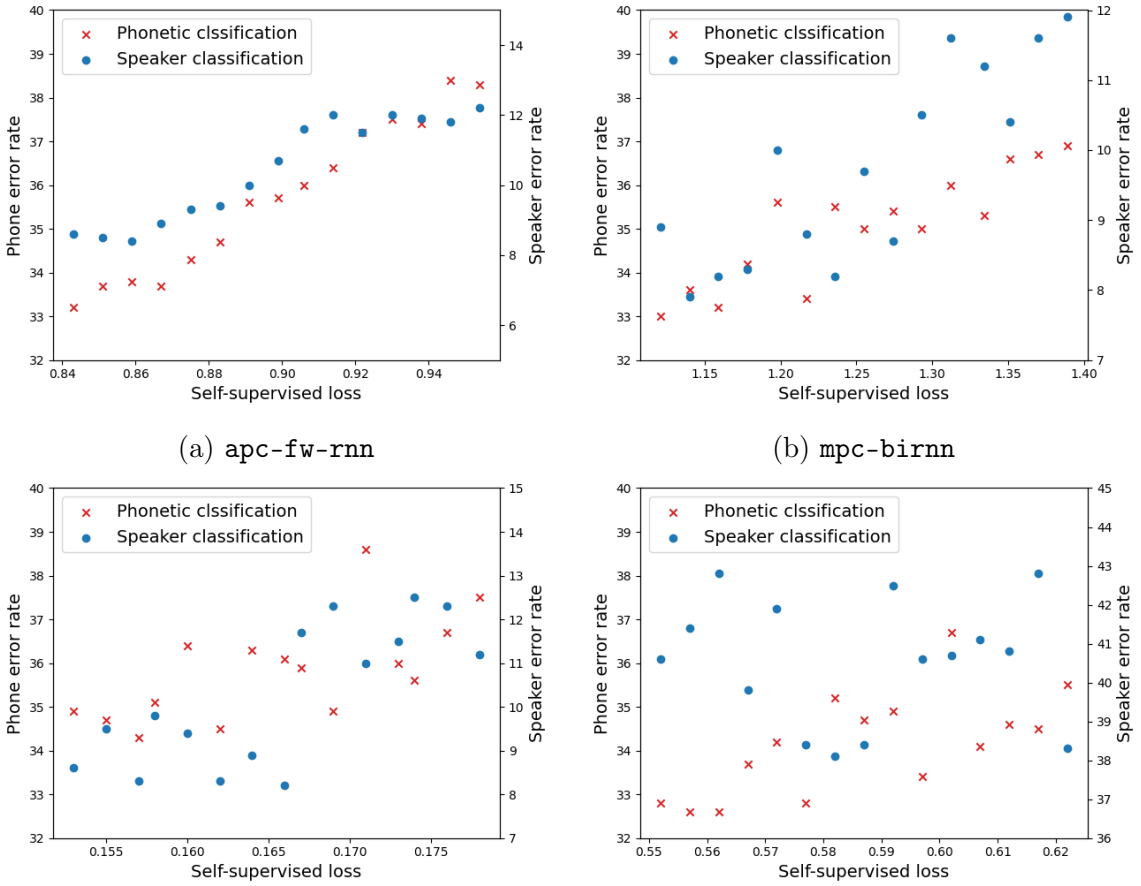
7.4.1 Similarities between Different Self-Supervised Representations

Figures 7-1 and A-1 (in Appendix A) show heatmaps of similarities between representations learned by various self-supervised models according to similarity measures `lincka` and `svcca` on two probing datasets WSJ and TIMIT. Brighter colors indicate higher similarity between two representations. For all heatmaps we also include the similarity between each self-supervised representation and the surface feature, i.e., log Mel spectrogram. We find all heatmaps exhibiting consistent patterns regardless of the probing dataset and similarity measure, and all self-supervised representations are very different from the surface feature (in our case, the log Mel spectrogram). The heatmaps reveal the following insights.

Objective affects similarity more than architecture. The most evident pattern from the heatmaps is that there is always a greater similarity within an objective than across objectives, indicated by the bright block diagonal. For example, `apc-fw-rnn` is always more similar to `apc-fw+bw-rnn`, `apc-fw-trf`, and `apc-fw+bw-trf` than to any MPC and CPC variants, even when `apc-fw-rnn` and `cpc-mixed_spk-rnn` / `cpc-within_spk-rnn` share the same building block and directionality. This conclusion also holds for the MPC- and CPC-family. Representations learned by generative-based objectives, i.e., variants of APC and MPC, are also more similar to one other than to the CPC variants.

Directionality affects similarity more than building block. When the objective is the same, we find that model's directionality (uni/bidirectional) has a higher impact on representation similarity than its building block (RNN/Transformer/CNN).[2] For instance, the similarity between `apc-fw-rnn` and `apc-fw-trf`, which are both unidirectional while the former uses RNNs and the latter uses Transformers, is higher

[2]Due to its nature of methodology, MPC is always bidirectional. Hence we refer to the cases within the APC- and CPC-family for this observation.



(a) `apc-fw-rnn` (b) `mpc-birnn`

(c) `cpc-mixed_spk-rnn` (d) `cpc-within_spk-rnn`

Figure 7-2: Scatter plots of various self-supervised representations' performance on phonetic and speaker classification as a function of their pre-training loss. For each figure, the x-axis is the pre-training loss, and the y-axis on the left is the corresponding phone error rate and on the right the speaker error rate.

than that between `apc-fw-rnn` and `apc-fw+bw-rnn`, which both use RNNs while the former is unidirectional and the latter is bidirectional. Furthermore, as may be expected, making APC bidirectional reduces its difference with MPC, which is indicated by the fact that `mpc-birnn` is more similar to `apc-fw+bw-rnn` than to `apc-fw-rnn`, and `mpc-trf` is more similar to `apc-fw+bw-trf` than to `apc-fw-trf`.

Source of negative samples affects similarity more than architecture. When focusing on the CPC-family, we find that proposal distribution where the negative samples are drawn—which could also be regarded as the objective—is more impactful on representation similarity than building block. This is indicated by

the fact that `cpc-within_spk-rnn` is more similar to `cpc-within_spk-cnn` than to `cpc-mixed_spk-rnn`, where the two models in the former case share the same proposal distribution but use different building blocks, and the two models in the latter case share the same building block but incorporate different proposal distributions.

7.4.2 Correlation between Self-Supervised Loss and Phonetic & Speaker Classification Performance

Experiments so far have only revealed the similarities between different self-supervised representations. In this study, we further uncover the correlation between self-supervised loss during pre-training and the amount of phonetic and speaker information contained in the representations, measured by their performance on phonetic and speaker classification defined in Section 7.2.2. We only consider `apc-fw-rnn`, `mpc-birnn`, `cpc-mixed_spk-rnn`, and `cpc-within_spk-rnn` in this experiment for a comparison only in terms of their objectives (except `mpc-birnn`, which has to be bidirectional). Figure 7-2 displays the scatter plots of phone and speaker error rates as a function of self-supervised loss for the four considered models.

Table 7.2: Pearson correlation coefficients between the self-supervised loss and the phone and speaker error rates. $*$ denotes statistical significance at $\rho < 0.05$.

Model	Phone	Speaker
`apc-fw-rnn`	0.989^*	0.950^*
`mpc-birnn`	0.885^*	0.847^*
`cpc-mixed_spk-rnn`	0.643^*	0.762^*
`cpc-within_spk-rnn`	0.675^*	-0.071

For each model, we only consider loss after 10k steps until the end of training (10 epochs, which is about 88k steps). Within this interval, we take 15 data points—each corresponding to a model checkpoint—with equally-sized chunk, and sort them with ascending order according to their loss values. Next, for each of these 15 checkpoints, we run the probing tasks and report the corresponding phone and speaker error rates. We also calculate the Pearson correlation coefficients r between loss value and both phone and speaker error rates, as listed in Table 7.2.

Overall, generative-based objectives (APC and MPC) are found to have a stronger positive correlation between the self-supervised loss and both their phonetic and speaker classification performance than contrastive-based objectives. In particular, `apc-fw-rnn` features the strongest correlation among the four considered self-supervised models. Our finding aligns with Blandón and Räsänen (2020), where the autoregressive loss of APC is found to be more correlated with the ABX-score of a phone discrimination task than the InfoNCE loss of CPC.

It is noteworthy that the loss of `cpc-within_spk-rnn` has almost no correlation with speaker classification performance. This result seems natural since the model always draws negative samples from the same utterance as the positive sample, so speaker information is never found to be useful for distinguishing them and thus not learned by the representation. On the other hand, the proposal distribution of `cpc-mixed_spk-rnn` allows the model to learn from negative samples coming from both the same and different utterances as the positive sample, meaning that both phonetic and speaker information could be relevant for discriminating them. Therefore, we find the loss of `cpc-mixed_spk-rnn` is still correlated with the speaker error rate to some degree.

We emphasize that our findings here are not meant to claim any self-supervised approach to be the best, but aim to provide some results for other researchers for future reference. For example, APC and MPC's strong correlation between their self-supervised loss and phonetic and speaker classification performance could be useful for model selection even during the pre-training stage, since a lower pre-training loss would indicate a richer phonetic and speaker representation. CPC, though exhibiting a smaller correlation between its self-supervised loss and phonetic and speaker classification performance, could still be extremely powerful when the downstream task is known and thus the pre-training proposal distribution can be determined aforehand, as shown by its recent impressive performance on semi-supervised speech recognition (Baevski et al., 2020b).

7.4.3 Effect of Increasing Unlabeled Data for Pre-Training

One of the biggest advantages of self-supervised learning is its capability to leverage very large-scale unlabeled data for representation learning. Here we train `apc-fw-rnn`, `mpc-birnn`, `cpc-mixed_spk-rnn`, and `cpc-within_spk-rnn` on 2k, 4k, and 6k hours of speech audio, all sampled from the `unlab-6k` subset of the Libri-Light corpus, and calculate the similarities between each of these variants and their counterpart trained on the original 960 hours LibriSpeech audio according to `lincka`. Results are shown in Table 7.3.

Table 7.3: Representation similarity between self-supervised models pre-trained on ∼1k hours of audio and their counterparts pre-trained on increasing amounts of audio according to `lincka`.

Model	Hours of pre-training audio		
	∼2k	∼4k	∼6k
`apc-fw-rnn`	0.957	0.935	0.923
`mpc-birnn`	0.940	0.939	0.925
`cpc-mixed_spk-rnn`	0.911	0.883	0.837
`cpc-within_spk-rnn`	0.920	0.896	0.861

Table 7.4: Phonetic and speaker classification results of self-supervised models pre-trained on different amounts of unlabeled data (in hours). Phone and speaker error rates are reported.

Model	Phone error rate				Speaker error rate			
	∼1k	∼2k	∼4k	∼6k	∼1k	∼2k	∼4k	∼6k
`apc-fw-rnn`	33.2	32.5	32.3	31.9	8.6	8.4	8.2	8.1
`mpc-birnn`	33.0	32.2	32.1	31.8	8.9	8.1	8.0	7.8
`cpc-mixed_spk-rnn`	34.9	33.7	33.2	33.0	8.6	7.9	7.5	6.8
`cpc-within_spk-rnn`	32.8	29.8	28.5	28.1	40.6	38.7	42.2	40.5

As may be expected, for all self-supervised models, their representations become more dissimilar when more unlabeled data are used for training. Interestingly, we find that CPC's representations change more than those of APC and MPC when increasing the data size. For instance, the similarity "only" drops from 0.957 to 0.923

for `apc-fw-rnn` when increasing the data size from 2k hours to 6k hours, while for `cpc-mixed_spk-rnn`, the similarity drops from 0.911 to 0.837.

Changes in representation similarity can be attributed to encoding details of speech other than phonetic and speaker information that might be unnecessary, such as background noises. To confirm whether such changes in representation similarity correspond to an actual richer phonetic and speaker representation, we again use phonetic and speaker classification performance to quantify the amount of phonetic and speaker information contained in the representation. Results are reported in Table 7.4.

Encouragingly (and probably unsurprisingly), we observe that most self-supervised models' performance on both tasks are improved when being trained on more data. The only exception is `cpc-within_spk-rnn` on speaker classification, which is expected as speaker information is never found relevant for discriminating positive and negative samples during its training. However, its performance on phonetic classification obtains the largest gain among all considered self-supervised models, with phone error rate decreasing from 32.8 to 28.1. Concerning `cpc-mixed_spk-rnn`, in addition to showing improvement on both tasks, the drop of its speaker error rate from 8.6 to 6.8 is also the largest among all models. Intuitively, having more data means that CPC models are provided with more comparisons of negative and positive samples to learn from, and our results seem to suggest that this is a more effective way for learning representations when large amounts of unlabeled data are available, as opposed to attempting to reconstruct details of the speech signals as APC and MPC models do. That being said, both generative- and contrastive-based objectives also benefit from having more unlabeled training data.

7.5 Chapter Summary

We have analyzed representations learned by contrastive predictive coding (CPC), autoregressive predictive coding (APC), and masked predictive coding (MPC) through the lens of similarity analysis. Extensive experiments have been conducted to study

the impact of different modeling choices for training self-supervised models, the effect of the size of unlabeled training data, and how well the self-supervised loss correlates with phonetic and speaker classification performance. We have found that the self-supervised objective has a much higher impact on representation similarity than architectural choices such as building blocks (RNN/Transformer/CNN) and directionality (uni/bidirectional). We have also observed that APC has the strongest correlation between its self-supervised loss and phonetic and speaker classification performance, which is useful for model selection. Finally, while all self-supervised models benefit from having more training data, CPC is found to learn from the additional data more efficiently than APC and MPC.

Chapter 8

Conclusions

8.1 Thesis Summary

This thesis has chronicled the story of our investigations into the use of self-supervised learning for improving spoken language technology. In Chapter 3, we drew inspirations from recent language model pre-training algorithms and proposed Autoregressive Predictive Coding (APC), which defines a future frame prediction task that enables autoregressive neural models to learn representations capturing high-level phonetic and speaker information from unlabeled speech data. Those positive preliminary results of APC encouraged us to dive deeper into understanding this self-supervised objective to seek for any possible improvements, and such efforts were detailed in Chapter 4 and Chapter 5. In Chapter 4, we conducted careful analysis on a spectrum of APC models with decreasing model capacity. Such analysis allowed us to scrutinize the information constituents of the representations learned by APC. Our findings bridged the connection between APC and its capability of learning rich phonetic and speaker information. After understanding why APC, the future frame prediction task, can learn high-level speech representations, in Chapter 5 we made the model to learn even stronger representations by improving its ability to predict the future. We demonstrated the effectiveness of APC as a self-supervised pre-training objective by showing its superiority over several other objectives on speech recognition, speech translation, and speaker classification tasks. In parallel to APC, which

makes use of the "future prediction" learning methodology, in Chapter 6 we proposed w2v-BERT, which adopted another proved powerful self-supervised learning methodology: "predicting the masked from the unmasked." w2v-BERT showed extremely strong performance on both a well-benchmarked speech recognition dataset and a more challenging dataset that reflects real-world audio traffic, and is the current state-of-the-art pre-training framework for speech processing. After exploiting two distinct learning methodologies to design self-supervised pre-training algorithms, we shifted our interest to understanding what modeling choices when training the networks make one self-supervised representation different from the other, and such efforts were documented in Chapter 7.

8.2 Thesis Contributions

We reiterate a summary of the contributions made by this thesis here:

1. **Introduction of one of the earliest successful self-supervised speech representation learning frameworks.** We exploit the idea of "future prediction" and propose a simple yet effective self-supervised objective called Autoregressive Predictive Coding (APC) for training deep neural networks. The designed future frame prediction task is able to leverage unlabeled speech data to learn representations that make high-level properties of speech utterances such as their phonetic contents and speaker characteristics more accessible (defined as linear separability) to downstream tasks. APC is one of the earliest works that showed the superiority of self-supervised representations over traditional hand-crafted acoustic features such as mel-frequency cepstral coefficients (MFCC) and log mel spectrograms, indicating the potential of using self-supervised learning for boosting spoken language technology performance.

2. **Introduction of one of the current state-of-the-art self-supervised speech representation learning frameworks.** We exploit the idea of "predicting the masked from the unmasked" and propose w2v-BERT, which is one

of the current state-of-the-art frameworks for pre-training very deep neural networks for speech applications. We train a speech discretizer (through optimizing a contrastive loss) for representing continuous speech signals as discriminative tokens, and use them to train a BERT-like model. In contrast to existing frameworks such as vq-wav2vec and HuBERT that also make use of the "predicting the masked from the unmasked" methodology, in w2v-BERT the discretizer and the context network can be optimized in an end-to-end fashion, avoiding the need of coordination between multiple training stages that could often involve brittle modeling choices. We demonstrate the effectiveness of w2v-BERT by showing its superiority over the state of the arts, including HuBERT and wav2vec 2.0, on both a well-benchmarked speech recognition dataset and a Google-collected voice search dataset.

3. **Introduction of an analysis method capable of bridging connections between self-supervised objectives and properties of the representations they learn.** We explore the use of vector quantization for controlling the amount of information flow inside deep neural networks to obtain a spectrum of models trained with the same self-supervised objective but with decreasing model capacity. We apply this analysis method to study APC, and diagnose the preferences of APC in preserving information while its model capacity becomes constrained. Our analysis results provide an explanation to why APC can learn representations that capture high-level phonetic and speaker information. The analysis method is general and can be applied to analyzing other self-supervised objectives as well.

4. **Demonstration of several shared natures of different self-supervised models.** When analyzing our own and other existing self-supervised models, we find that there exist several properties that most of those models share in common regardless of their differences in training objectives and neural network architectures. One of such properties is the ability of implicit discovery of an inventory of meaningful acoustic units. We find that there usually exist some

layers in the self-supervised models where representations have considerably high mutual information with English phones (when the models are trained on an English corpus), even though the models are not explicitly trained towards discovering them. Another properties shared by most self-supervised models is that different levels of speech information are captured in different layers, although the information distribution could vary model to model. For instance, in APC, the lower layers tend to be more discriminative for speakers, while the upper layers provide more phonetic content. Being aware of this insight is useful for selecting proper layers to extract representations from for the best performance on the tasks of interest.

5. **Identification of the order of importance of modeling factors for training self-supervised models that impact their representational similarity.** We compare a collection of self-supervised models with diverse modeling choices during their training, and use measures such as canonical correlation analysis (CCA) to quantify their pairwise similarities. We consider three modeling factors: training objectives, model directionality (i.e., whether the model is unidirectional or bidirectional), and neural network building blocks (CNN/RNN/Transformer), and show that the three factors have different weights in making one self-supervised representation different from another. Specifically, we find that training objective has the highest impact on representational similarity among all the factors; under the same training objective, a model's directionality affects representational similarity more than its neural network building blocks.

8.3 Future Directions

Several future directions arise from the work presented in this thesis. Here we briefly discuss a few of them.

8.3.1 Self-Supervised Pre-Training for Speech Generation Models

As is the case with modern ASR systems, a prevailing trend for tackling speech generation problems such as text-to-speech synthesis (TTS) and speech-to-speech translation (S2S) is to model them with a single deep neural network (Wang et al., 2017; Shen et al., 2018; Jia et al., 2019, 2021). The task of TTS is to synthesize a speech waveform for a given text, and can be thought of as the inverse problem of ASR. The task of S2S is an even more challenging one that aims to translate an utterance from one language to another, where both the input and output are in the form of speech waveforms. The speech generation models are required to have strong speech understanding and sometimes even linguistic knowledge about the underlying spoken utterances in order to produce high-quality natural speech. In this thesis, we have shown that our self-supervised pre-training frameworks are capable of initializing deep neural networks with strong acoustic representations that perform well when they are fine-tuned on ASR and speech-to-text translation. There is no reason why we cannot apply similar pre-training techniques to improve speech generation models.

There already exist some preliminary works in this direction. A simple method that has been explored by Chung et al. (2019b) was to pre-train the decoder of an end-to-end TTS model (Wang et al., 2017) with an APC-like self-supervised objective. They showed that when the decoder was pre-trained with large quantities of untranscribed audio, the amount of parallel text-audio pairs needed for training the TTS models in order to generate intelligible speech was greatly reduced. In Lee et al. (2021), the discrete speech units learned by a self-supervised model (HuBERT) (Hsu et al., 2021) have been applied to training end-to-end S2S models. Beyond the existing works, the research of applying self-supervised techniques to speech generation problems remains highly exploratory.

8.3.2 Self-Supervised Multimodal Speech Representation Learning

Recently there is a convergence of neural network architectures (Vaswani et al., 2017) and self-supervised pre-training objectives across different modalities: in text (BERT & RoBERTa) (Devlin et al., 2019; Liu et al., 2019), speech (w2v-BERT) (Chung et al., 2021b), and vision (BEiT) (Bao et al., 2022), training large Transformer models with masked language modeling-like objectives has become the dominant pre-training paradigm. Such convergence makes building a single model that can learn cross-modal speech representations a natural and promising next step. Previous works on multimodal pre-training of speech and text (Chung et al., 2021c; Lai et al., 2021; Chuang et al., 2020) as well as those of speech and vision (Akbari et al., 2021) still highly relied on the use of parallel data, which are harder to scale up than unpaired data, for supervised learning of cross-modal alignments. The fact that different modalities are now sharing similar neural architectures and self-supervised pre-training objectives could potentially alleviate the models' reliance on parallel data. A preliminary work in this direction is Bapna et al. (2021), while there still exists a large space for improvement.

8.3.3 Self-Supervised Multilingual Speech Representation Learning

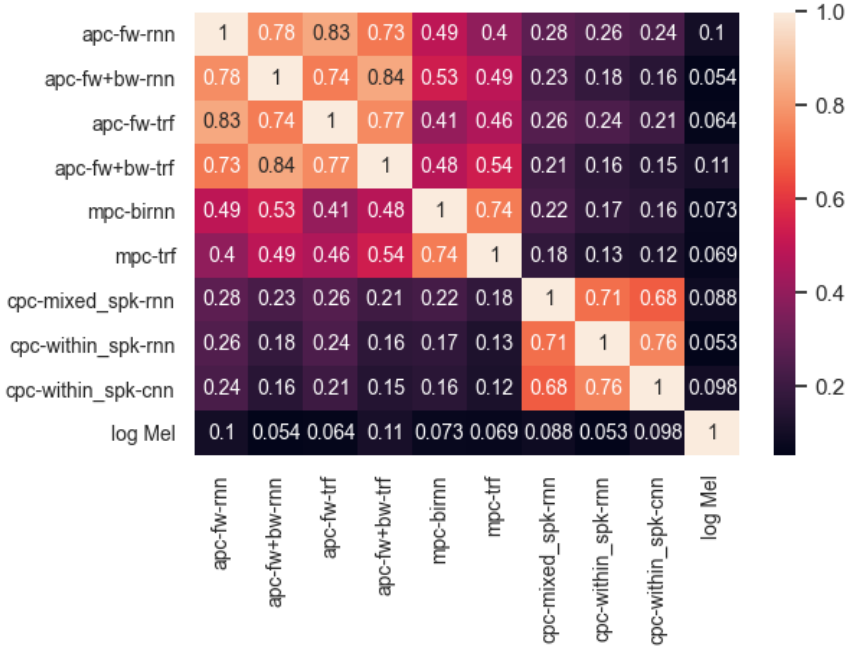
In text, multilingual representation learning models like mBERT (Devlin et al., 2019), XLM-R (Conneau and Lample, 2019), and mT5 (Xue et al., 2021) have shown the benefit of cross-lingual transfer for improving the representations of low-resource languages: on public benchmarks such as XTREME (Hu et al., 2020), multilingual models often greatly outperform monolingual models. The success of cross-lingual transfer learning in text has motivated researchers to develop analogous frameworks for learning multilingual speech representations. Although models like XLS-R (Babu et al., 2021), which is essentially wav2vec 2.0 (Baevski et al., 2020b) extended to a multilingual setting, is indeed able to improve speech representations of low-resource

languages over their monolingual counterparts, it comes at the cost of sacrificing the performance on high-resource languages. How to improve representations of low-resource languages while at least maintaining the quality of representations of high-resource languages remains an ongoing research topic. A straightforward way worthwhile trying is to replace wav2vec 2.0 with a more advanced pre-training framework such as w2v-BERT.

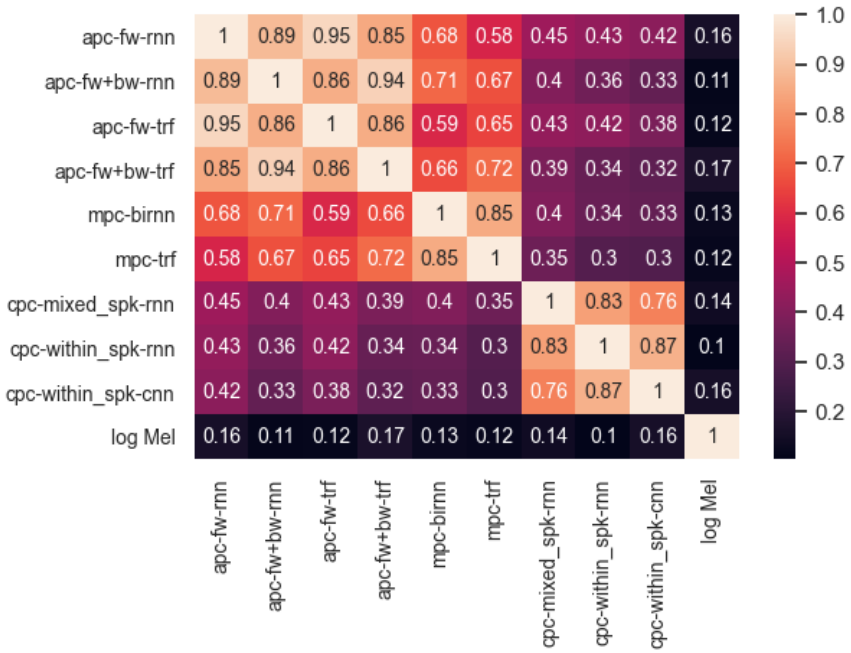
Appendix A

Additional Similarity Heatmaps

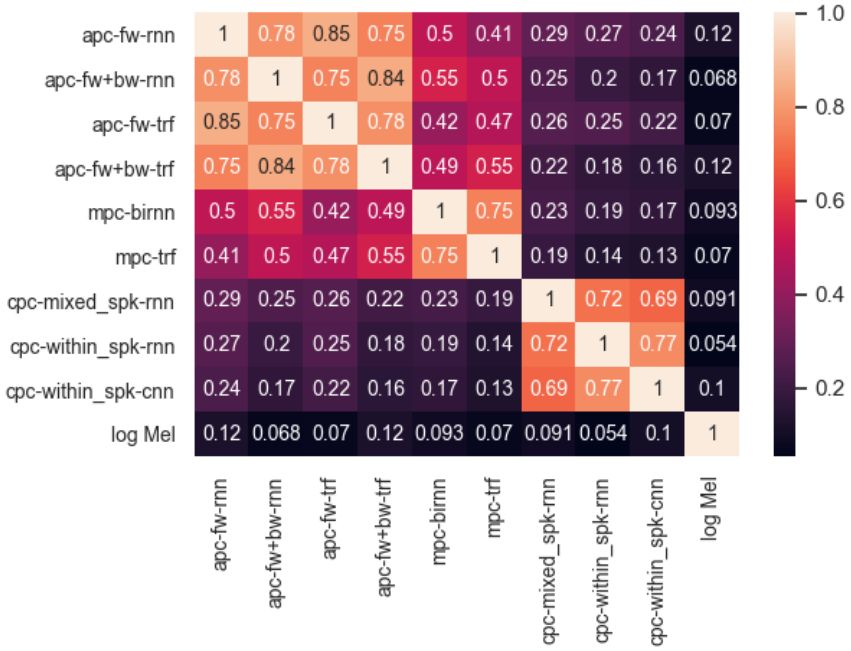
Figure A-1 displays additional similarity heatmaps for Chapter 7.



(a) `svcca` on WSJ



(b) `lincka` on TIMIT



(c) `svcca` on TIMIT

Figure A-1: Similarity heatmaps of various self-supervised representations on different probing datasets with different similarity measures.

Bibliography

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., and Penn, G. (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In *ICASSP*.

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., and Goldberg, Y. (2017). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *ICLR*.

Akbari, H., Yuan, L., Qian, R., Chuang, W.-H., Chang, S.-F., Cui, Y., and Gong, B. (2021). Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. In *NeurIPS*.

Alwassel, H., Mahajan, D., Korbar, B., Torresani, L., Ghanem, B., and Tran, D. (2020). Self-supervised learning by cross-modal audio-video clustering. In *NeurIPS*.

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. (2013). Deep canonical correlation analysis. In *ICML*.

Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., and Saunshi, N. (2019). A theoretical analysis of contrastive unsupervised representation learning. In *ICML*.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., and Auli, M. (2021). XLS-R: Self-supervised cross-lingual speech representation learning at scale. *arXiv preprint arXiv:2111.09296*.

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

Baevski, A., Schneider, S., and Auli, M. (2020a). vq-wav2vec: Self-supervised learning of discrete speech representations. In *ICLR*.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020b). wav2vec 2.0: A framework for self-supervised learning of speech representations. In *NeurIPS*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In *ICASSP*.

Bai, J., Wang, W., Zhou, Y., and Xiong, C. (2021). Representation learning for sequence data with deep autoencoding predictive components. In *ICLR*.

Baker, J. (1975). The DRAGON system–an overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29.

Bao, H., Dong, L., and Wei, F. (2022). BEiT: BERT pre-training of image transformers. In *ICLR*.

Bapna, A., Chung, Y.-A., Wu, N., Gulati, A., Jia, Y., Clark, J. H., Johnson, M., Riesa, J., Conneau, A., and Zhang, Y. (2021). SLAM: A unified encoder for speech and language modeling via speech-text joint pre-training. *arXiv preprint arXiv:2110.10329*.

Bau, A., Belinkov, Y., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. (2019). Identifying and controlling important neurons in neural machine tra nslation. In *ICLR*.

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., and Glass, J. (2017). What do neural machine translation models learn about morphology? In *ACL*.

Belinkov, Y. and Glass, J. (2017). Analyzing hidden representations in end-to-end automatic speech recognition systems. In *NIPS*.

Belinkov, Y. and Glass, J. (2019). Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Bilmes, J. A. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute*, 4(510):126.

Blandón, M. A. C. and Räsänen, O. (2020). Analysis of predictive coding models for phonemic representation learning in small datasets. In *ICML Workshop on Self-Supervision in Audio and Speech*.

Bouchacourt, D. and Baroni, M. (2018). How agents see things: On visual representations in an emergent language game. In *EMNLP*.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*.

Chen, S., Wang, C., Chen, Z., Wu, Y., Liu, S., Chen, Z., Li, J., Kanda, N., Yoshioka, T., Xiao, X., Wu, J., Zhou, L., Ren, S., Qian, Y., Qian, Y., Wu, J., Zeng, M., Yu, X., and Wei, F. (2021). WavLM: Large-scale self-supervised pre-training for full stack speech processing. *arXiv preprint arXiv:2110.13900*.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. (2020). Big self-supervised models are strong semi-supervised learners. In *NeurIPS*.

Chi, P.-H., Chung, P.-H., Wu, T.-H., Hsieh, C.-C., Chen, Y.-H., Li, S.-W., and Lee, H.-Y. (2021). Audio albert: A lite bert for self-supervised learning of audio representation. In *SLT*.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches. In *SSST*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Chorowski, J., Bahdanau, D., Cho, K., and Bengio, Y. (2014). End-to-end continuous speech recognition using attention-based recurrent NN: First results. In *NIPS Workshop on Deep Learning and Representation Learning*.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In *NIPS*.

Chorowski, J., Weiss, R. J., Bengio, S., and Van Den Oord, A. (2019). Unsupervised speech representation learning using wavenet autoencoders. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2041–2053.

Chowdhury, S. A., Durrani, N., and Ali, A. (2021). What do end-to-end speech models learn about speaker, language and channel information? a layer-wise and neuron-level analysis. *arXiv preprint arXiv:2107.00439*.

Chuang, Y.-S., Liu, C.-L., Lee, H.-Y., and Lee, L.-S. (2020). SpeechBERT: An audio-and-text jointly learned language model for end-to-end spoken question answering. In *Interspeech*.

Chung, Y.-A., Belinkov, Y., and Glass, J. (2021a). Similarity analysis of self-supervised speech representations. In *ICASSP*.

Chung, Y.-A. and Glass, J. (2018). Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech. In *Interspeech*.

Chung, Y.-A. and Glass, J. (2020a). Generative pre-training for speech with autoregressive predictive coding. In *ICASSP*.

Chung, Y.-A. and Glass, J. (2020b). Improved speech representations with multi-target autoregressive predictive coding. In *ACL*.

Chung, Y.-A., Hsu, W.-N., Tang, H., and Glass, J. (2019a). An unsupervised autoregressive model for speech representation learning. In *Interspeech*.

Chung, Y.-A., Tang, H., and Glass, J. (2020). Vector-quantized autoregressive predictive coding. In *Interspeech*.

Chung, Y.-A., Wang, Y., Hsu, W.-N., Zhang, Y., and Skerry-Ryan, R. (2019b). Semi-supervised training for improving data efficiency in end-to-end speech synthesis. In *ICASSP*.

Chung, Y.-A., Wu, C.-C., Shen, C.-H., Lee, H.-Y., and Lee, L.-S. (2016). Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. In *Interspeech*.

Chung, Y.-A., Zhang, Y., Han, W., Chiu, C.-C., Qin, J., Pang, R., and Wu, Y. (2021b). w2v-BERT: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *ASRU*.

Chung, Y.-A., Zhu, C., and Zeng, M. (2021c). SPLAT: Speech-language joint pre-training for spoken language understanding. In *NAACL-HLT*.

Collobert, R., Puhrsch, C., and Synnaeve, G. (2016). Wav2letter: An end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*.

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. (2018). What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Conneau, A. and Lample, G. (2019). Cross-lingual language model pretraining. In *NeurIPS*.

Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2011). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366.

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Doersch, C., Gupta, A., and Efros, A. (2015). Unsupervised visual representation learning by context prediction. In *ICCV*.

Dong, L., Xu, S., and Xu, B. (2018). Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *ICASSP*.

Ettinger, A., Elgohary, A., and Resnik, P. (2016). Probing for semantic evidence of composition by means of simple classification tasks. In *RepEval*.

Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., and Dahlgren, N. (1993). DARPA TIMIT acoustic-phonetic continuous speech corpus. Technical Report NISTIR 4930, NIST.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Gong, Y., Lai, C.-I. J., Chung, Y.-A., and Glass, J. (2022). SSAST: Self-supervised audio spectrogram transformer. In *AAAI*.

Goyal, P., Mahajan, D., Gupta, A., and Misra, I. (2019). Scaling and benchmarking self-supervised visual representation learning. In *ICCV*.

Graves, A. (2012). Sequence transduction with recurrent neural networks. In *ICML Workshop on Representation Learning*.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *ICML*.

Graves, A., Mohamed, A.-r., and Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. In *ICASSP*.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., and Valko, M. (2020). Bootstrap your own latent-a new approach to self-supervised learning. In *NeurIPS*.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. (2020). Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech*.

Guo, P., Boyer, F., Chang, X., Hayashi, T., Higuchi, Y., Inaguma, H., Kamo, N., Li, C., Garcia-Romero, D., Shi, J., Shi, J., Watanabe, S., Wei, K., Zhang, W., and Zhang, Y. (2021). Recent developments on espnet toolkit boosted by conformer. In *ICASSP*.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Harwath, D., Hsu, W.-N., and Glass, J. (2020). Learning hierarchical discrete linguistic units from visually-grounded speech. In *ICLR*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *ACL*.

Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Hsu, W.-N., Zhang, Y., and Glass, J. (2017a). Learning latent representations for speech generation and transformation. In *Interspeech*.

Hsu, W.-N., Zhang, Y., and Glass, J. (2017b). Unsupervised learning of disentangled and interpretable representations from sequential data. In *NIPS*.

Hu, J., Ruder, S., Siddhant, A., Neubig, G., Firat, O., and Johnson, M. (2020). XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *ICML*.

Hupkes, D., Veldhoen, S., and Zuidema, W. (2018). Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparametrization with gumble-softmax. In *ICLR*.

Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

Jia, Y., Ramanovich, M. T., Remez, T., and Pomerantz, R. (2021). Translatotron 2: Robust direct speech-to-speech translation. *arXiv preprint arXiv:2107.08661*.

Jia, Y., Weiss, R. J., Biadsy, F., Macherey, W., Johnson, M., Chen, Z., and Wu, Y. (2019). Direct speech-to-speech translation with a sequence-to-sequence model. In *Interspeech*.

Jiang, D., Li, W., Zhang, R., Cao, M., Luo, N., Han, Y., Zou, W., Han, K., and Li, X. (2021). A further study of unsupervised pretraining for transformer based speech recognition. In *ICASSP*.

Kahn, J., Lee, A., and Hannun, A. (2020a). Self-training for end-to-end speech recognition. In *ICASSP*.

Kahn, J., Rivière, M., Zheng, W., Kharitonov, E., Xu, Q., Mazaré, P.-E., Karadayi, J., Liptchinsky, V., Collobert, R., Fuegen, C., Likhomanenko, T., Synnaeve, G., Joulin, A., Mohamed, A., and Dupoux, E. (2020b). Libri-light: A benchmark for ASR with limited or no supervision. In *ICASSP*.

Kamper, H., Wang, W., and Livescu, K. (2016). Deep convolutional acoustic word embeddings using word-pair side information. In *ICASSP*.

Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X., Watanabe, S., Yoshimura, T., and Zhang, W. (2019). A comparative study on Transformer vs RNN in speech applications. In *ASRU*.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

Kawakami, K., Wang, L., Dyer, C., Blunsom, P., and Oord, A. v. d. (2020a). Learning robust and multilingual speech representations. In *EMNLP*.

Kawakami, K., Wang, L., Dyer, C., Blunsom, P., and Oord, A. v. d. (2020b). Unsupervised learning of efficient and robust speech representations.

Kim, K., Lee, K., Gowda, D., Park, J., Kim, S., Jin, S., Lee, Y.-Y., Yeo, J., Kim, D., Jung, S., Lee, J., Han, M., and Kim, C. (2019). Attention based on-device streaming speech recognition with large speech corpus. In *ASRU*.

Kim, S., Hori, T., and Watanabe, S. (2017). Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *ICASSP*.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *ICASSP*.

Kocabiyikoglu, A., Besacier, L., and Kraif, O. (2018). Augmenting LibriSpeech with French translations: A multimodal corpus for direct speech translation evaluation. In *LREC*.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. E. (2019). Similarity of neural network representations revisited. In *ICML*.

Kriegeskorte, N., Mur, M., and Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2:4.

Lai, C.-I., Chuang, Y.-S., Lee, H.-Y., Li, S.-W., and Glass, J. (2021). Semi-supervised spoken language understanding via self-supervised speech and language model pre-training. In *ICASSP*.

Larsson, G., Maire, M., and Shakhnarovich, G. (2016). Learning representations for automatic colorization. In *ECCV*.

Larsson, G., Maire, M., and Shakhnarovich, G. (2017). Colorization as a proxy task for visual understanding. In *CVPR*.

Lee, A., Chen, P.-J., Wang, C., Gu, J., Ma, X., Polyak, A., Adi, Y., He, Q., Tang, Y., Pino, J., and Hsu, W.-N. (2021). Direct speech-to-speech translation with discrete units. *arXiv preprint arXiv:2107.05604*.

Lee, K.-F. and Hon, H.-W. (1989). Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Li, B., Pang, R., Sainath, T. N., Gulati, A., Zhang, Y., Qin, J., Haghani, P., Huang, W. R., Ma, M., and Bai, J. (2021). Scaling end-to-end models for large-scale multilingual ASR. In *ASRU*.

Li, B., Sainath, T. N., Pang, R., and Wu, Z. (2019). Semi-supervised training for end-to-end models via weak distillation. In *ICASSP*.

Li, C.-Y., Yuan, P.-C., and Lee, H.-Y. (2020a). What does a network layer hear? analyzing hidden representations of end-to-end ASR through speech synthesis. In *ICASSP*.

Li, J., Wu, Y., Gaur, Y., Wang, C., Zhao, R., and Liu, S. (2020b). On the comparison of popular end-to-end models for large scale speech recognition. In *Interspeech*.

Li, J., Zhao, R., Meng, Z., Liu, Y., Wei, W., Parthasarathy, S., Mazalov, V., Wang, Z., He, L., Zhao, S., and Gong, Y. (2020c). Developing RNN-T models surpassing high-performance hybrid models with customization capability. In *Interspeech*.

Ling, S. and Liu, Y. (2020). DeCoAR 2.0: Deep contextualized acoustic representations with vector quantization. *arXiv preprint arXiv:2012.06659*.

Ling, S., Liu, Y., Salazar, J., and Kirchhoff, K. (2020). Deep contextualized acoustic representations for semi-supervised speech recognition. In *ICASSP*.

Liu, A., Tu, T., Lee, H.-Y., and Lee, L.-S. (2020a). Towards unsupervised speech recognition and synthesis with quantized speech representation learning. In *ICASSP*.

Liu, A. T., Li, S.-W., and Lee, H.-Y. (2021). TERA: Self-supervised learning of transformer encoder representation for speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2351–2366.

Liu, A. T., Yang, S.-W., Chi, P.-H., Hsu, P.-C., and Lee, H.-Y. (2020b). Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders. In *ICASSP*.

Liu, P., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. In *ICLR*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A ConvNet for the 2020s. *arXiv preprint arXiv:2201.03545*.

Manning, C. and Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT Press.

Mermelstein, P. (1976). Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, 116:374–388.

Miao, Y., Gowayyed, M., and Metze, F. (2015). EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *ASRU*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*.

Milde, B. and Biemann, C. (2018). Unspeech: Unsupervised speech context embeddings. In *Interspeech*.

Mohamed, A.-r., Dahl, G. E., and Hinton, G. E. (2011). Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22.

Morcos, A., Raghu, M., and Bengio, S. (2018). Insights on representational similarity in neural networks with canonical correlation. In *NeurIPS*.

Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*.

Novotney, S. and Schwartz, R. (2009). Analysis of low-resource acoustic model self-training. In *Interspeech*.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Oord, A. v. d., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. In *NIPS*.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). LibriSpeech: An ASR corpus based on public domain audio books. In *ICASSP*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *ACL*.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. In *Interspeech*.

Park, D. S., Zhang, Y., Chiu, C.-C., Chen, Y., Li, B., Chan, W., Le, Q. V., and Wu, Y. (2020a). SpecAugment on large scale datasets. In *ICASSP*.

Park, D. S., Zhang, Y., Jia, Y., Han, W., Chiu, C.-C., Li, B., Wu, Y., and Le, Q. V. (2020b). Improved noisy student training for automatic speech recognition. In *Interspeech*.

Parthasarathi, S. H. K. and Strom, N. (2019). Lessons from building acoustic models with a million hours of speech. In *ICASSP*.

Pasad, A., Chou, J.-C., and Livescu, K. (2021). Layer-wise analysis of a self-supervised speech representation model. In *ASRU*.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *CVPR*.

Paul, D. and Baker, J. (1992). The design for the wall street journal-based CSR corpus. In *Workshop on Speech and Natural Language*.

Peters, M., Neumann, M., Zettlemoyer, L., and Yih, W.-T. (2018a). Dissecting contextual word embeddings: Architecture and representation. In *EMNLP*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations. In *NAACL-HLT*.

Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice-Hall, Inc.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. Technical report, OpenAI.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. (2017). SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NIPS*.

Ramachandran, P., Zoph, B., and Le, Q. V. (2018). Searching for activation functions. In *ICLR Workshop Track*.

Riloff, E. and Wiebe, J. (2003). Learning extraction patterns for subjective expressions. In *EMNLP*.

Riviere, M., Joulin, A., Mazaré, P.-E., and Dupoux, E. (2020). Unsupervised pretraining transfers well across languages. In *ICASSP*.

Sainath, T. N., He, Y., Li, B., Narayanan, A., Pang, R., Bruguier, A., Chang, S.-Y., Li, W., Alvarez, R., Chen, Z., Chiu, C.-C., Garcia, D., Gruenstein, A., Hu, K., Jin, M., Kannan, A., Liang, Q., McGraw, I., Peyser, C., Prabhavalkar, R., Pundak, G., Rybach, D., Shangguan, Y., Sheth, Y., Strohman, T., Visontai, M., Wu, Y., Zhang, Y., and Zhao, D. (2020). A streaming on-device end-to-end model surpassing server-side conventional model quality and latency. In *ICASSP*.

Sak, H., Senior, A., and Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.

Sak, H., Shannon, M., Rao, K., and Beaufays, F. (2017). Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping. In *Interspeech*.

Saphra, N. and Lopez, A. (2019). Understanding learning dynamics of language models with svcca. In *NAACL-HLT*.

Schatz, T., Peddinti, V., Bach, F., Jansen, A., Hermansky, H., and Dupoux, E. (2013). Evaluating speech features with the minimal-pair ABX task: Analysis of the classical MFC/PLP pipeline. In *Interspeech*.

Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. In *Interspeech*.

Schroeder, M. and Atal, B. (1985). Code-excided linear prediction (CELP): high-quality speech at very low bit rates. In *ICASSP*.

Schuster, M. and Nakajima, K. (2012). Japanese and Korean voice search. In *ICASSP*.

Scudder, H. (1965). Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

Seide, F., Li, G., Chen, X., and Yu, D. (2011). Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *ASRU*.

Settle, S. and Livescu, K. (2016). Discriminative acoustic word embeddings: Recurrent neural network-based approaches. In *SLT*.

Shazeer, N. and Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., Saurous, R. A., Agiomvrgiannakis, Y., and Wu, Y. (2018). Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. In *ICASSP*.

Song, X., Wang, G., Wu, Z., Huang, Y., Su, D., Yu, D., and Meng, H. (2020). Speech-XLNet: Unsupervised acoustic model pretraining for self-attention networks. In *Interspeech*.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*.

Synnaeve, G., Xu, Q., Kahn, J., Likhomanenko, T., Grave, E., Pratap, V., Sriram, A., Liptchinsky, V., and Collobert, R. (2020). End-to-end ASR: from supervised to semi-supervised learning with modern architectures. In *ICML SAS Workshop*.

Tang, H. and Glass, J. (2018). On training recurrent networks with truncated back-propagation through time in speech recognition. In *SLT*.

Tang, H., Lu, L., Kong, L., Gimpel, K., Livescu, K., Dyer, C., Smith, N. A., and Renals, S. (2017). End-to-end neural segmental models for speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1254–1264.

Tishby, N., Pereira, F., and Bialek, W. (1999). The information bottleneck method. *arXiv preprint arXiv:physics/0004057*.

Ueno, S., Inaguma, H., Mimura, M., and Kawahara, T. (2018). Acoustic-to-word attention-based model complemented with character-level CTC-based model. In *ICASSP*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *NIPS*.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep graph infomax. In *ICLR*.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(12):3371–3408.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

Wang, W., Tang, Q., and Livescu, K. (2020). Unsupervised pre-training of bidirectional speech encoders via masked reconstruction. In *ICASSP*.

Wang, X. and Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *ICCV*.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. (2017). Tacotron: Towards end-to-end speech synthesis. In *Interspeech*.

Watanabe, S., Hori, T., Kim, S., Hershey, J. R., and Hayashi, T. (2017). Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

Wu, J. M., Belinkov, Y., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. (2020). Similarity analysis of contextual word representation models. In *ACL*.

Wu, Y., Schuster, M., Chen, Z., Le, Q., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xu, D., Xiao, J., Zhao, Z., Shao, J., Xie, D., and Zhuang, Y. (2019). Self-supervised spatiotemporal learning via video clip order prediction. In *CVPR*.

Xu, Q., Baevski, A., Likhomanenko, T., Tomasello, P., Conneau, A., Collobert, R., Synnaeve, G., and Auli, M. (2021). Self-training and pre-training are complementary for speech recognition. In *ICASSP*.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer. In *NAACL-HLT*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*.

Zavaliagkos, G. and Colthurst, T. (1998). Utilizing untranscribed training data to improve performance. In *DARPA Broadcast News Transcription and Understanding Workshop*.

Zeyer, A., Bahar, P., Irie, K., Schlüter, R., and Ney, H. (2019). A comparison of Transformer and LSTM encoder decoder models for ASR. In *ASRU*.

Zhang, Q., Lu, H., Sak, H., Tripathi, A., McDermott, E., Koo, S., and Kumar, S. (2020a). Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *ICASSP*.

Zhang, Y., Qin, J., Park, D. S., Han, W., Chiu, C.-C., Pang, R., Le, Q. V., and Wu, Y. (2020b). Pushing the limits of semi-supervised learning for automatic speech recognition. In *NeurIPS Workshop on Self-Supervised Learning for Speech and Audio Processing*.

Zhu, J. (2020). Probing the phonetic and phonological knowledge of tones in mandarin TTS models. In *Speech Prosody*.