creative commons

C O M M O N S D E E D

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사 학위논문

컨포머 기반 지식증류 기법을 사용한 발화 표현 학습 모델

Conformer-based Knowledge Distillation Speech Representation Learning Model

2023 년 2 월

서울대학교 대학원

협동과정 인공지능 전공

안 진 형

공학석사 학위논문

컨포머 기반 지식증류 기법을 사용한 발화 표현 학습 모델

Conformer-based Knowledge Distillation Speech Representation Learning Model

2023 년 2 월

서울대학교 대학원

협동과정 인공지능 전공

안 진 형

컨포머 기반 지식증류 기법을 사용한 발화 표현 학습 모델

Conformer-based Knowledge Distillation Speech Representation Learning Model

지도교수 이 교 구

이 논문을 공학석사 학위논문으로 제출함

2023 년 2 월

서울대학교 대학원

협동과정 인공지능 전공

안 진 형

안진형의 공학석사 학위 논문을 인준함

2023 년 2 월

위 원 장:	이 원 종	(인)
부위원장:	이 교 구	(인)
위 원:	서 봉 원	(인)

Abstract

Self-supervised learning (SSL) in speech involves training a network on large-scaled unlabeled speech corpora and using the learned representations from the network to perform downstream tasks. Commonly applied speech downstream tasks for SSL models are content related such as speech recognition and phoneme recognition. SSL models have achieved successful results in these tasks.

Despite their success, state of the art SSL models have limited usage for performing ASR in real-world situation. The major downside for using these models in real-word situation comes from their large parameter size. In order to effortlessly use SSL models, a resourceful computational environment where GPUs with enough memory space are available. The large parameter size further limits SSL models' application in on-device setting.

To overcome the shortcomings from the parameter size and the limited usage of SSL models in speech, we develop a parameter efficient using a distillation method and effectively reduce memory footprint with tolerable degradation in a downstream task performance. To be specific, we develop a parameter efficient network structure based on Conformer encoder instead of commonly used Transformer encoder. Our distillation method uses content informative teacher labels created by quantization of pre-trained teacher SSL model's representations. Furthermore, we have shown that our distillation method suits better for Conformer encoder instead of Transformer encoder.

To measure our model's representation performance, downstream tasks in speech processing universal performance benchmark (SUPERB) and ZeroSpeech2021 are used for evaluation. Our model shows par or better results than some of the existing SSL models in content related downstream tasks in SUPERB and outperformed the existing distillation model in phoneme recognition model. Moreover in Zerospeech2021, our model shows superior performance than the state of the art SSL models like ContentVec, HuBERR, and wav2vec2.0 in phonetic related tasks.

Furthermore, the techniques used for the distillation methods are investigated through ablation study. Our model trained by the proposed distillation method reduces the parameter size of the teacher model by 78% and increases the inference speed by 78% when only using CPUs and 99% when using single GPU.

Keywords: self-supervised learning, speech representation learning, knowledge distillation, model compression

Student Number: 2021-24076

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Self-supervised learning (SSL) in speech involves training a speech representation network on a speech data and using the learned representation to perform downstream tasks. A well-learned speech representation network could distinguish an utterance made of up different phonemes or sound units for a language, so these representations can be used as an input for a lightweight network to perform downstream tasks. The speech representations from SSL models have been applied to various speech related downstream tasks and achieve successful results.

To learn informative features from speech, SSL methods require a deep network with heavy memory usage and large-scaled unannoated speech corpora. Training a deep and heavy neural network on large-scaled speech data expect heavy computational resources and time. Once the training is over, the network becomes frozen or fine-tuned with a target dataset to perform downstream tasks.

The most common downstream task for SSL models in speech is automatic speech recognition (ASR), which transcribes text from speech. In fact, most of the SSL models report their representation performance by performing ASR on the test dataset of

LibriSpeech [4]. This is because ASR can be commonly found in people's daily lives. For example, Siri, Alexa, Bixby, and other virtual assistants implemented in smart devices receive order from speech and perform ASR before executing the received order.

Although state of the arts SSL models are reported to show powerful performance in ASR, the deep neural networks for performing ASR in real-life application are recurrent neural network (RNN) based models specifically trained for speech recognition. Such case shows a contradiction between the deep neural network model in the real-life application and academia. In academia, SSL representations with a small downstream deep neural network [5, 6, 2] are reported to show powerful performance and outperformed RNN based models in ASR. Yet, speech representations from SSL models are not used for ASR in the real-life situation.

Along with not using SSL models in real-life situation, SSL models are generally running on the remote server where resourceful computational power is available and not on-device setting with limited resources. Running a neural network model on-device means performing inference of a neural network only by using the resource of the device and not using computational resource outside. On-device neural network inference gives two major advantages over online inference. First is real-time application. On-device inference gives the results right away, as shown in Figure 1.2. It does not need to go through multiple steps, shown in Figure 1.1 to gives the inference results. Second is privacy. The input for the neural network, which usually contains private information such as a picture or speech of the user, stays within the device and does not need to be sent to the remote server which users cannot access or control.

Using SSL models in on-device setting gives an additional advantage. On-device SSL model can save memory space for additional neural networks to perform speech related tasks. Instead of training a deep and heavy network for each of the downstream task, a minimal network that takes representations of a SSL network as an input is needed .

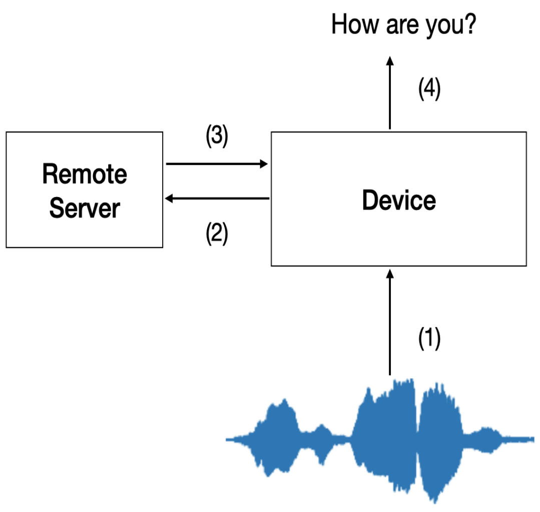

Figure 1.1: A flowchart of online inference process for a neural network to perform speech recognition task.(1) shows a device receives speech. (2) shows a device sends speech to the remote server (3) depicts a network in the remote server to perform speech recognition and sends results back to the device. (4) shows the device showing results received from the remote sever.

The motivation of the thesis is to develop a SSL model that can be used for real-life and on-device application. We define the major problem that causes limited application of SSL model in real-life and on-device setting is coming from the model parameter size. In order to reduce the number of parameters while maintaining the performance of SSL models, a knowledge distillation method is adopted to train a parameter efficient SSL model.

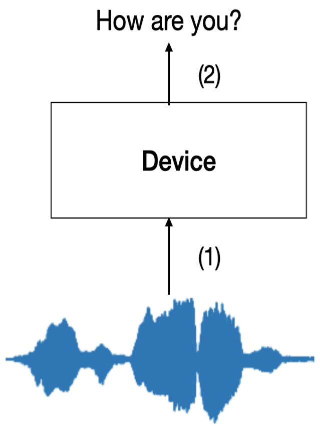

Figure 1.2: A flowchart of on-device inference process for a neural network. (1) shows a device receiving speech. (2) shows a device performs speech recognition and gives the results

1.2 Problem Definition

There are multiple reasons that prevent the self-supervised learning models to be used for ASR in real-life application, but the major reason we focus on is the model parameter size. Parameters are the entities that models learn by optimizing a loss function. Having a large number of parameter means more parameters to update in the training time and more parameters to calculate in the inference time. Having a large number of parameters also means taking a large memory space. Therefore, the state of the art SSL networks take a logn time to train, large memory space to locate and slow inference speed to provide representations for downstream tasks.

Most generally used SSL models in speech like wav2vec2.0 [6] and HuBERT [5] have nearly 95 million parameters for the base models and 317 million parameters for the large models. Although showing powerful performance in speech related downstream tasks, the state-of-the-art SSL networks can only be effortlessly used in the environment where GPUs with enough memory are available due to the large number of parameters. If the state-of-the-art SSL models in speech can reduce the number of parameters while maintaining their representation performance, their applicability in real-life can significantly increase. Because SSL model with the less number of parameters would take less memory space and faster inference time and become more capable to run in computationally less resourceful environment.

To be more specific, the large parameter size of SSL models majorly comes from Transformer encoder. Due to its powerful representation performance, using Transformer encoder for the SSL models becomes the standard approach in audio, vision, language and etc. If the alternate encoder that has similar representational performance but lighter in size is studied, then an efficient SSL model for speech can be developed.

1.3 Overview

To address the issues coming from the large number of parameters of the SSL models and open up the possibility of real-life and on-device application, we develop a parameter efficient SSL model that specifically focus on the content related speech downstream tasks. We focus on the content related speech downstream tasks, such as speech recognition and phoneme recognition because these tasks are most generally used to benchmark the performance of SSL models in speech and commonly found in the real-life application.

Our proposed model is trained by a distillation method. The pseudo-labels are created by discretizing speech representations of the pre-trained SSL teacher network, which we call teacher labels. By predicting teacher labels, a student model with the small number of parameters can learn the representation power of the teacher model.

A student model not only has to be parameter efficient but also needs to be able to learn well from the teacher labels. In order to learn speech content representation from the teacher labels, a student model adopts the structure of the state of the art ASR model [7]. The motivation for such model structure comes from previous researches [8, 9, 10, 11, 12] that treat pseudo-labels like contents of the speech. We propose a parameter efficient network structure that consists of Conformer [7] encoder instead of commonly used Transformer [13] encoder. Just like Conformer is originally built yo predict the contents from the speech, we built our model based on Conformer blocks to predict teacher labels from speech and capture meaningful content information.

After distillation is over, our student model is frozen and evaluated its representation performance using SUPERB [1] and ABX testing of ZeroSpeech2021 [14]. The contriubting techniques used for distillation method is investigated. The inference speed is evaluated by measuring the time taken for getting representation of the dev-set of LibriSpeech [4] using limited computational resource.

1.4 Contribution

Overall, our contributions of the thesis are the following:

- We investigate the performance of Conformer based speech representation model for our distillation training method. We reduced the number of parameters by 3 million just by changing the encoder than the existing model structure.
- We compared the distillation performance by predicting teacher labels created by discretizing pre-traiend SSL model's representations between Conformer and Transformer based model and showed the strength over Conformer based model than the Transformer based model in our method.
- The conformer based model is evaluated over wide range of downstream tasks and shows par to superior performance in content related speech downstream

tasks than existing SSL models and previous SSL distillation model in speech. Our model specifically shows strength in phonetically related downstream tasks

- The techniques involved in the distillation method and model structure are investigated through the ablation study. We analyze the major contributing factor that affect the performance of our model.
- Our proposed model reduces the number of parameters of the teacher model by 78% and inference time by 99% using single GPU and 78% using 4 CPUs.

Chapter 2

Background

This chapter describes the basic concepts and related work for helping to understand the thesis. Basic concepts focus on elaborating the general ideas that set the cornerstone of developing the proposed model. Self-supervised learning, Transformer, and knowledge distillation are introduced and explained for the basic concepts. Related work section shows the history of previous researches on the basic concepts and how these works related to the proposed model. Related work will explain researches about self-supervised learning in speech, knowledge distillation of self-supervised learning in speech, and two benchmarks used to evaluate the performance of the proposed model.

2.1 Basic Concept

There are three basic concepts that would enhance the understanding of the thesis, which are self-supervised learning (SSL) Transformer, and knowledge distillation. The proposed model is categorized under self-supervised learning (SSL), and the proposed training framework belongs to the knowledge distillation (KD). Lastly, Transformer is the essential encoder framework for providing representation of the speech, and the proposed model uses the convolution augmented version of Transformer to build parameter efficient self-supervised model structure.

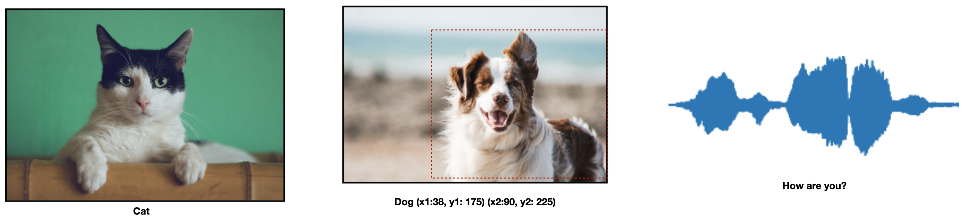

Figure 2.1: Data labeling examples for classification (left), object detection (center), and speech recognition (right).

2.1.1 Self-supervised Learning

Self-supervised learning (SSL) has been researched to capture hidden structure of raw data for the target downstream tasks. Before self-supervised learning method is proposed, the applications of deep neural network were mainly supervised learning tasks such as image classification, object detection, speech recognition, and etc. In order to perform these tasks, labeling for corresponding data is necessary, as shown in Figure 2.1. An image cannot be classified as a cat or a dog without the labeling of the image and speech cannot be recognized without knowing the contents of it. Getting labels for the data costs expensive human labors, and supervised learning tasks with small labeling data often leads to overfitting and becomes inapplicable to be solved by deep learning.

To avoid overfitting, transfer learning which reuses the weights of the pre-trained network for other supervised tasks, has been a popular approach. However, pre-training a network in a supervised fashion could not capture general representation of the data and limit its usage for specific tasks only. In order to overcome the shortcomings of a deep neural network trained with a lack of data and to allow fine-tuning for wide range of target tasks, a model that learns task-agonistic features of raw data is developed by self-supervised learning methods.

The general purpose of self-supervised learning is to capture meaningful representation. What defines ”meaningful” can be different according to the domain of its application. In computer vision, it is essential to represent how an image is related to one another regardless of position, color, and other nuisance factors. In order to satisfy this condition, a discriminative method using contrastive learning [15, 16, 17] has been adopted and shows state-of-the-art performance.

$$l_{i,j} = -log\frac{exp(sim(z_i, z_j)/\tau)}{\sum_{k=1\ [k\neq i]}^{N} exp(sim(z_i, z_k/\tau)} \tag{2.1}$$

An example of contrastive loss function is shown in equation 2.1, where z is the vector representation of an image, $sim()$ operation is measuring cosine similarity distance between two vectors, and τ is the smoothing temperature hyper-parameter. In numerator, z_i and z_j are positive pair whose representations come from the same image with different augmentation. In denominator, sum of z_i and z_k are negative pairs whose representation vectors come from different images. The summation in denominator is done over N mini-batch of images. Overall, the loss tries to shorten the distance between positive pair and to extend the distance between negative pairs. A diagram of how self-supervised learning is trained using the contrastive loss is shown in Figure 2.2

In natural language processing, a representation needs to catch context of a sentence or predicts a word given certain contents of a sentence. A mask prediction method [18, 19] which predicts masked part of the sentence or auto-regressive prediction method [20, 21] which predicts next word given previous words, show the state of the art performance. Specifically, BERT [18] gives masked input to the Transformer encoder to predict the masked part of the sentence, as shown in Figure 2.3. In speech, both discriminative and mask prediction methods are used to train SSL models

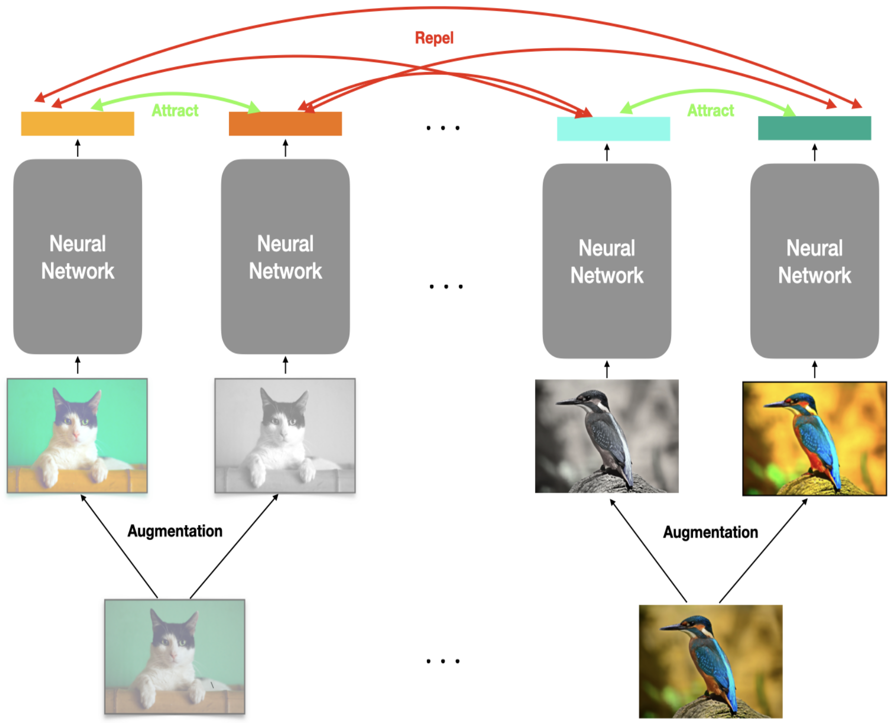

Figure 2.2: A diagram of how self-supervised learning model is trained with contrastive loss in computer vision. The vector representations of the same image with different augmentation are attracted to each other and vector representations of the different images are repelled using the contrastive loss.

because audio can be viewed as signal just like images in vision but also has sequential and contains discrete contents like a language.

2.1.2 Transformer

Transformer is first introduced in [13] to perform machine translation task in natural language processing. Then it becomes widely used in other domains such as computer vision and audio. The essential part of Transformer is the multi-head attention

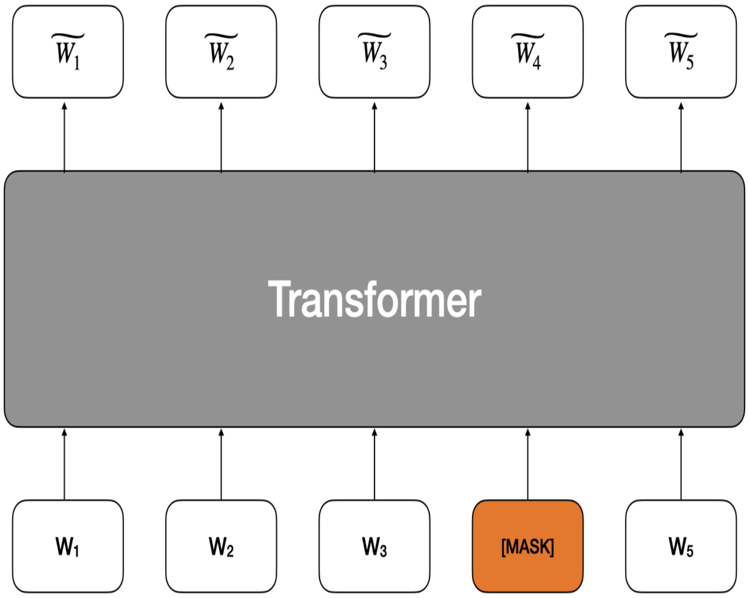

Figure 2.3: A diagram of how BERT, a SSL model in natural language processing, is trained. W stands for the text token input, $[mask]$ stands for the masking token, $\widetilde{W}$ stands for predicted text token. Text tokens and mask tokens are given as input to a Transformer to predict corresponding tokens.

module, which divides the dimension of the embedding vector n times and performs self-attention for each divided vector. The process of self-attention is summarized in equation 2.2

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

Q,K,V each stands for query, key, value, and these are obtained from the same embedding vectors. Query and key have d_k dimensions, or original embedding vector dimension divided by n. The query and transposed key K^T vectors are multiplied and then scaled by d_k. Then softmax of resulted output is calculated and multiplied by the value V. The self-attention mechanism shows how input vector interacts within each other and which part the "attention" is needed to be put on. Basically, the weights of attention applied to the value is calculated by softmax function of a matrix resulted from query and key transposed multiplication.

$$\begin{aligned} Multihead(Q,K,V) &= Concat(head_1, ..., head_h)W^o \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.3}$$

Multi-head attention uses n number of heads to perform self-attention at multiple aspects. Equation 2.3 shows how multi-head attention work. For the dimension of query, key, and value are divided into n parts, and each head uses self attention for the divided part. After self-attention is done for each sub-key, sub-query, and sub-value, the resulted outputs are concatenated and send it to the feed-forward module. The overall diagram of a Transformer encoder is shown in Figure 2.4.

2.1.3 Knowledge Distillation

Knowledge distillation is a method to reduce the model parameter size by using a teacher and student network. A teacher network, as the name suggests, is the model that shows promising results in the target downstream task. The student network, which usually has less number of parameters than the teacher network, learns the knowledge of the teacher model to perform the same downstream task. When the training is over, the student model is expected to perform like the teacher model with the less number of parameter.

The original idea of knowledge distillation in deep neural network comes from [22]. In this paper, a student network learns teacher network's knowledge for image

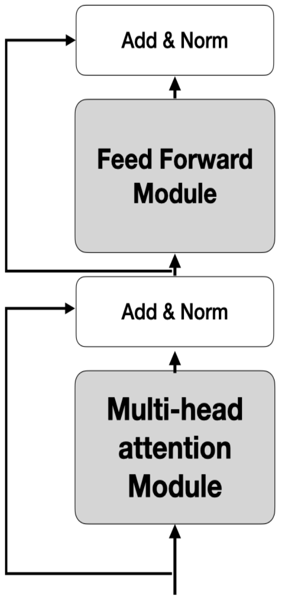

Figure 2.4: A diagram of a Transformer encoder. Transformer encoder consists of a multi-head self attenetion module and a feed forward module. Add & Norm stands for addition from the residual connection and normalization.

classification by using Kullback-Leibler (KL) divergence between teacher models' logits and student model's logits. KL divergence is reduced when the classification probability distribution between the teacher and student's model is similar. Therefore, the student model learns teacher models' classification probability distribution and can perform like teacher model in the downstream task.

$$KL(p||q) \simeq \frac{1}{N}\sum_{n=1}^{N} -lnq(x_n|\theta) + lnp(x_n) \tag{2.4}$$

Equation 2.4 shows the how KL divergence work like an object function in image classification. Denote $p(\cdot)$ as the probability distribution of the teacher model, and $q(\cdot)$

as the probability distribution of the student model, which is parameterized by θ. x denotes the image used for training, and N is the total number of images. For each image, the student model learns parameter to make the probability distribution similar to that of the teacher model. However, learning teacher model's performance does not always need to use KL divergence because the tasks solved by deep learning models are not always involving classification. Learning features can be done by closing the distance between teacher model's feature by L1 or L2 loss.

2.2 Related Work

In this section, previous researches related to the thesis will be introduced. Existing methods of training and developing self-supervised learning are described. Knowledge distillation of SSL models in speech is introduced to give an overview of previous distillation approaches. SUPERB and ZeroSpeech 2021 are used as evaluation metrics for SSL models in speech.

2.2.1 Self-supervised Learning in Speech

Self-supervised learning in speech can be divided into two categories: generative method and discriminative method. The models belong to both categories are used as baselines to compare the performance of the proposed model.

Generative Method A generative method in self-supervised learning in speech learns representation by predicting or reconstructing acoustic features, such as spectrogram or mel-spectrogram. APC [23] learns representation by auto-regressively predicting future frame from the past frames. MockingJay [24] predicts the current frame from the past and future frames using bidirectional Transformer. TERA [25] uses masked and time-altered acoustic feature as input for the Transformer encoder and learns speech representation by reconstructing the original acoustic feature. DeCoAR2.0 [26] learns speech representation by reconstructing acoustic feature from vector-quantized counterpart.

Discriminative Method A discriminative method in self-supervised learning in speech generally involves contrastive learning method to group features from the same time step together and put features from different different time step far from each other. CPC [27] first uses contrastive predictive coding method to learn representation of various modalities. wav2vec2.0 combines the idea of CPC and VQ-wav2vec [28] and proposes an end-to-end speech representation learning framework.

$$L_m = -log \frac{exp(sim(c_t, q_t)/\tau)}{\sum_{\widetilde{q}\ Q_t} exp(sim(c_t, \widetilde{q}/\tau)} \tag{2.5}$$

wav2vec2.0 applies contrastive loss on the masked input embedding, inspired by Bert [18]. As shown in equation 2.5, the cosine similarity between the context vector c_t at time step t in masked region and its vector quantized counterpart q_t are measured in numerator. In denominator, the cosine similarity between the context vector c_t and vector quantized part at other time steps besides t are calculated and summed. Therefore, the contrastive loss tries to group the context vector and the vector quantized counterpart at time t together and repel vector quantized part at other time steps within the mask region.

HuBERT [5] predicts pseudo-labels created by quantization of mel frequency cepstrum coefficient (MFCC) via k-means clustering to learn speech representation. MFCC is an acoustic feature based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. As shown in Figure 2.5, pseudo-labels z are created by quantization of MFCC feature from speech. An initial feature x is created from a convolutional neural network (CNN) encoder by taking speech as input. A portion of x is masked and goes through Transformer and corresponding pseudo-labels z at

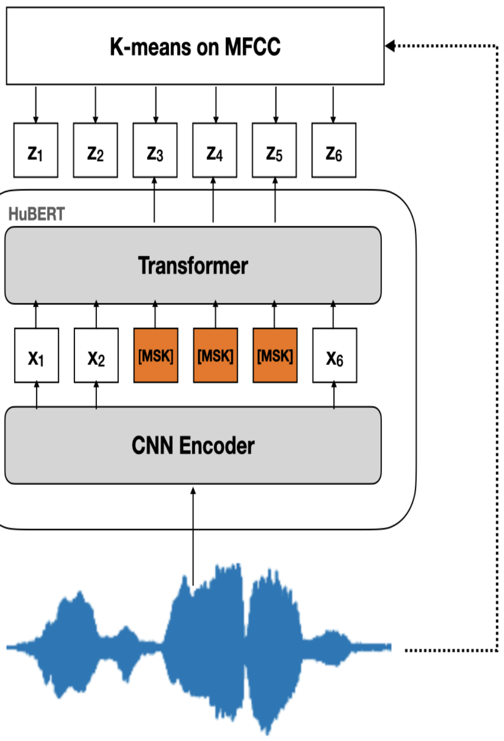

Figure 2.5: A diagram of how HuBERT is trained. z are the pseudo labels created by quantization of MFCC representations of speech, x are the initial features as speech goes through CNN encoder.

the same time steps of the masked feature are predicted. For pseudo-labels created by MFCC, 100 labels are used for quantization. After trained for several epochs, the pseudo labels are replaced by quantized representations of HuBERT's 6th Transformer encoder output and trained for epochs. The number of labels for HuBERT's representations was 500. Just like BERT [18], the mask prediction method is used to capture the context feature of a speech.

ContentVec Another discriminative method that is heavily related to the thesis is ContentVec [2]. ContentVec disentangles content from speaker information from HuBERT representations and achieved performance gain in content related downstream tasks. Contentvec attempts three different disentanglement in training. First is disentanglement in teacher pseudo-labels by performing voice conversion to utterances in LibriSpeech, obtaining HuBERT representations of all the utterances that went through voice conversion, and quantizing these represenations using k-means clustering. Second is disentanglement in model by perturbing utterances with the same method used in [29]. This method involves formant frequency shift, f0 shift, and equalizer. Then contrastive loss is used for HuBERT representations of the perturbed utterances. Lastly, the classifier that predicts teacher pseudo-labels are conditioned with speaker embedding, so the classifier tends to not look at the speaker information. Discretized ContentVec representations are used as teacher pseudo-labels for this thesis.

2.2.2 Knowledge Distillation of Self-supervised Learning in Speech

Shrinking Bigfoot Knowledge distillation of self-supervised learning in speech is a relatively new research topic. One of early attempts to distil self-supervised learning in speech is Shrinking Bigfoot [30]. Shrinking Bigfoot compresses wav2vec2.0 by adopting weight initialization method from DistilBert [31] and using the KL divergence loss between wav2vec2.0, the teacher model, and a student model. Along with KL divergence, a L2 regularization is used between CNN encoders' outputs of the teacher and a student model. The authors tested multiple student model structures such as having 6,8,10,12 Transformer layers. The authors further push the compression capacity of the student model by performing weight quantization, which reduces the number of bits used to represent weights and change the floating points weights to the integer weights. Weight quantization saves the memory usage and speeds up the inference time, but the models with quantize weights can only be used in CPUs because GPUs do not support integer calculation.

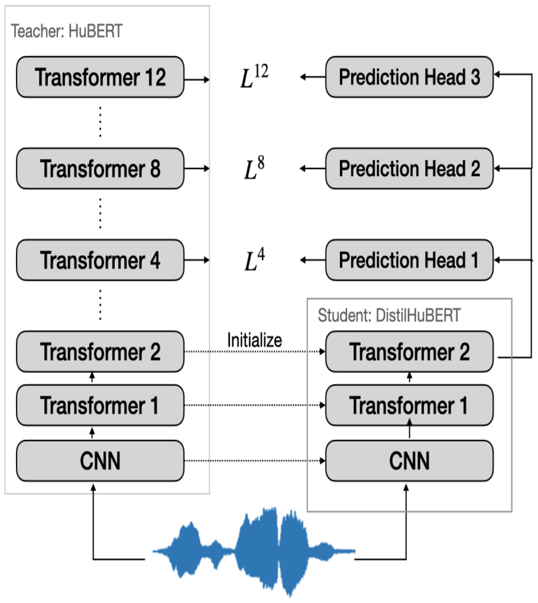

Figure 2.6: A diagram of how HuBERT is trained. z are the pseudo labels created by quantization of MFCC representations of speech, x are the initial features as speech goes through CNN encoder.

DistilHuBERT Another distillation approach is DistilHuBERT [3], which adopts DistilBert's weight initialization method and multi-task learning framework. DistilHuBERT uses 2 Transformer layers and linear prediction heads to predict 4th, 8th, and 12th layers of HuBERT to get a shared representation. The process of how DistilHuBERT is trained can be found in Figure 2.6.

Unlike Shrinking Bigfoot, DistilHubert uses the loss function that combines L1 loss and a loss that increases the cosine similarity between the predicted and HuBERT's representations. The overall loss used for DistilHuBERT is shown in Equation 2.6

$$L = \sum_{t=1}^{T}[\frac{1}{D}|h_t^l - \hat{h}_t^l| - \lambda log\sigma(sim(h_t^l, \hat{h}_t^l))] \tag{2.6}$$

In Equation 2.6, the original feature vector at time t with dimension D at the layer l is denoted as h_t^l and the predicted feature vector is denoted as $\hat{h_t^l}$. λ denotes the hyper-parameter of how much put on weights the second term, σ denotes the sigmoid function, and $sim()$ denotes the cosine similarity. As the result, DistilHuBERT could reduce the parameter size of the teacher network by 73% while achieving powerful performance in SUPERB benchmark. DistilHuBERT is used as the baseline to compare the performance between the SSL models trained by a distillation method.

2.2.3 Conformer-based Audio Representation Model

There has been previous attempts to capture audio representation using Conformer instead of Transformer. In [32], authors replace Transformer with Conformer in wav2vec2.0 architecture and use contrastive loss to capture audio representation. Unlike wav2vec2.0, authors use log-mel spectrogram as the input instead of raw audio. On the AudioSet benchmark, their audio representation acheived a new state-of-the-art mean average precision of 0.415 for self-supervsied learning model trained with audio data only.

2.2.4 SUPERB

Speech processing universal performance benchmark(SUPERB) [1] is introduced to evaluate the generalizability and re-usability of the self-supervised learning model in speech across the different downstream tasks. For evaluation, the pre-traiend self-supervised models are frozen and lightweight prediction heads are used to perform speech-related downstream tasks. Since lightweight neural networks are trained for the downstream tasks, the results for the downstream tasks depend on the how speech is well represented rather than the network capacity.

There are total of 11 downstream tasks in SUPERB, and these tasks are grouped by 4 categories: content, speaker, semantics, and paralinguistics. The tasks that we focus on are content and semantic tasks. The content and semantic related downstream tasks are most commonly used in people's daily lives. Moreover, the proposed distillation involves the teacher model that were specifically trained to capture content related information from speech.

The content related tasks that our proposed model is evaluated are phoneme recognition (PR), automatic speech recognition (ASR), and keyword spotting (KS). The semantic downstream tasks that our proposed model is evaluated are intent classification (IC) and slot filling (SF). The detailed descriptions of how each tasks are setup are as following:

$$ER = \frac{S + D + I}{S + D + C} \tag{2.7}$$

Phoneme recognition (PR) transcribes speech into phoneme, or sound unit of a language. The alignment modeling is introduced to avoid the potential inaccurate forced alignment between speech and phonemes. LibriSpeech [4] train-clean-100, dev-clean, test-clean subsets are used for training, validation, and testing. The ground truth phonemes are obtained by the LibriSpeech official g2p-model-5 and the conversion script in Kaldi librispeech s5 recipe. A frame-wise linear layer with CTC loss is used for performing training and testing. For evaluation metric, phoneme error rate (PER) is used. Phoneme error rate is calculated using the equation 2.7, where S is the substitution , D is the deletion, I is the insertion, and C is the correct phoneme unit.

Automatic speech recognition (ASR) transcribes speech into words. Just like PR, LibriSpeech train-clean-100, dev-clean, and test-clean subsets are used for training, validation, and testing. Word error rate (WER) is used for the evaluation metric. A vanilla 2-layer 1024-unit BLSTM trained by CTC loss on characters are used. Word error rate is calculated using the equation 2.7, where S is the substitution , D is the deletion, I is the insertion, and C is the number of correct words.

Keyword Spotting (KS) detects pre-registered keywords by predicting utterances into a set of words. Speech Commands dataset v1.0 [33] , which contains ten classes of keywords, a classes of keywords, a class for silence, and an unknown class to include the false positive, is used. The evaluation metric is classification accuracy. A linear classification head optimized by cross-entropy loss is used for performing keyword spotting.

Intent Classification (IC) classifies speech into intent classes to find out the intent of the speakers. Fluent Speech Commands [34] which has an utterance with its intention label pairs are used. Intention labels include action, object, and location. The evaluation metric is classification accuracy. A linear classification head optimized by cross-entropy loss is used for performing intent classifcation.

$$F1 = 2 * \frac{recall * precision}{recall + precision} \tag{2.8}$$

Slot Filling (SF) predicts a sequence of slot-types from a speech. For example, when a spoken word *Seoul*, a slot value, is given, *FromLocation*, a slot-type, needs to be predicted. Audio SNIPS [35] dataset is used, and US-accent speakers are selected for training, and others are selected for validation and testing. The evaluation metrics are slot-type F1 score and slot value concept error rate (CER). F1 score is calculated by using the Equation 2.8. Concept error rate is calculated using the Equation 2.7, where

S is the substitution , D is the deletion, I is the insertion, and C is the number of slot value. The special tokens to represent slot-type labels are used to wrap the slot values for transcription. The rest of the setting for model to perform SF is the same as ASR.

2.2.5 ZeroSpeech 2021

The Zero Resource Speech Benchmark 2021 [14] is introduced to benchmark the linguistic power of the self-supervised models representation without using labels. There are four categories to evaluate the self-supervised learning models which are phonetics, lexicon, syntax, and semantics. The proposed model will be evaluated by the phonetic criteria.

ABX testing is used to evaluating phonetic knowledge of the representation. ABX test measures the probability that the representation of the given sound is falling into the same category or the different category. The categories are triphones that only different in the middle phoneme such as 'aba' vs. 'apa'. Given a sound x, ABX test measures the probability that x belongs to category A ('aba') or B('apa'). The score can be computed within speaker, all the sounds are spoken by the same person, or across the speaker, sound categories are spoken by the same person and x is spoken by a different person.

$$\hat{e}(A,B) := \frac{1}{n_A(n_A-1)n_B} \sum_{a,x \in A; x \neq a} \sum_{b \in B} [1_{d(b,x)<d(a,x)} + \frac{1}{2} 1_{d(b,x)=d(a,x)}] \tag{2.9}$$

Equation 2.9 shows the asymmetric score computed asymmetric score with different tokens belong to category A (n_A denotes the cardinality of A) and to category B (n_B denotes the cardinality of B). d denotes the dynamic time wrapping (DTW) distance. For the dataset to perform ABX test, Libri-light [36] dev and test sets are used.

Chapter 3

Method

3.1 Introduction

The overall framework of our method is shown in Figure 3.1. Teacher labels of the corresponding utterances are created by quantizing speech representations of a pre-trained teacher model. A student model is trained by taking utterances as input and predicts their teacher labels using a linear prediction head. After training is over, a student model without the prediction head is used for performing downstream tasks.

3.2 Teacher Labels

For the teacher labels, pre-trained ContentVec [2] representations are used. We choose ContentVec because it disentangles speaker and content information from the HuBERT representation and achieves significant performance gain in content related speech downstream tasks. Through quantizing ContentVec representations, teacher labels are expected to contain content information of the speech.

The process of creating teacher labels Z is portrayed in Figure 3.1(a). Denote a speech utterance as $X = [x_1, ..., x_T]$, where x_t is the speech sample at frame t. The

Layer	Input Channel	Output Channel	kernel size	stride
1	1	512	10	5
2	512	512	3	2
3	512	512	3	2
4	512	512	3	2
5	512	512	3	2
6	512	512	2	2
7	512	512	2	2

Table 3.1: CNN encoder structure details

corresponding representation feature produced by a self-supervised learning(SSL) network is denoted as $R = g(x)$, where $g(\cdot)$ is the teacher network, and the output representation is denoted as $R = [r_1, ..., r_T]$, where r_t is the feature vector at frame t. Teacher labels are denoted as $Z = [z_1, ..., z_T]$, where a label $z_t \in [K]$ is assigned to kth cluster and is created by quantizing the continuous representations R, and this operation is denoted as $Z = h(R)$, where $h(\cdot)$ is k-means clustering.

3.3 Model Structure

This section describes the structure of the proposed model in details. As shown in Figure 3.1(b), the structure of our student model $f(\cdot)$ consists of a convolutional neural network (CNN) encoder and 2 Conformer [7] block encoders. The details of the CNN encoder, the Conformer weight initialization, and Conformer block encoder will be elaborated.

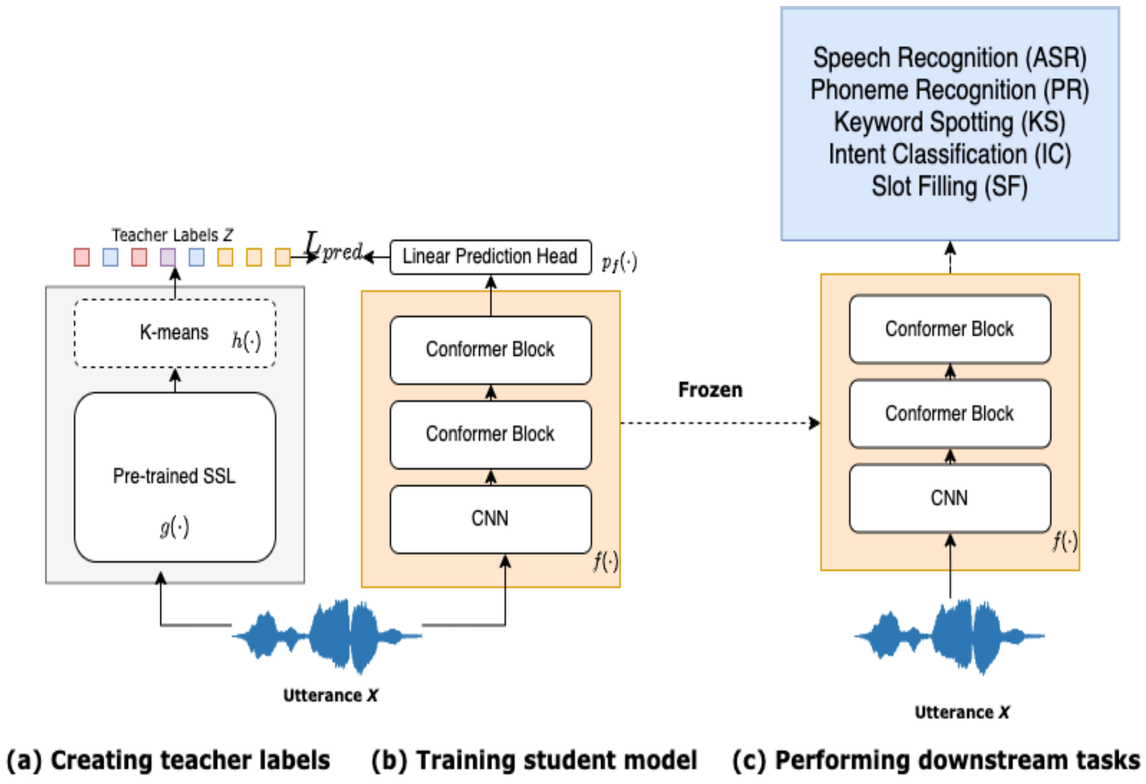

Figure 3.1: The overall framework of our proposed method. (a) Teacher labels are created by the quantization of pre-trained teacher SSL model's representations via k-means clustering. (b) Our proposed model consists of CNN encoder and 2 layers of Conformer blocks and learns speech representation by predicting teacher labels. (c) After pre-training is over, the frozen network without the linear prediction head is used to produce speech representation for content related downstream tasks.

3.3.1 CNN Encoder

The CNN encoder consists of 7 layers. The first layer takes a single channel raw wave audio as the input and uses 1D convolution with output channel of 512, kernel size of 10, and stride of 5 with no padding. The second to fourth layers uses 1D convolution which takes input channel of 512 and output channel of 512, kernel size 2 and stride 2 with no padding. The fifth and sixth layers use 1D convolution with 512 input channels, 512 output channels, 2 kernel size, and 2 strides with no pooling. The output feature of the CNN encoder is down-sampled by the factor of 320. For an audio with sample rate of 16k, each feature of the CNN encoder contains information of 320ms of the original audio. This down-sampling factor is kept for the output of the proposed model because Conformer block does not further down-sample the output feature of the CNN encoder. In each layer, the layer normalization [37] and GELU activation [38] are used for the output of the 1D convolution.

3.3.2 Weight Initialization

Inspired by previous distillation approaches in self-supervised learning[31, 3, 30], we initialized our model's CNN encoder with the weights of ContentVec's CNN encoder. The fact that the proposed model and ContentVec has the same CNN structure is exploited. Moreover, the weight initialization can be seen as the type of fine-tuning. Just like fine-tuning starts with the same weight of the pre-trained model for the target task, our model's CNN encoder is fine-tuned and offered good starting point for the training via distillation. The effect of weight initialization is reported in section 4.4.

3.3.3 Conformer Block

The essential difference between the proposed model and the previous SSL models in speech is the encoder after CNN encoder. Our proposed model uses Conformer instead of Transformer encoder. A Conformer block encoder is made up of a feed-forward module, a self-attention module, a convolution module, and another feed-forward module, as shown in Figure 3.2. The convolution module consists of a pointwise convolution, a gated lienar unit (GLU), a single 1D depthwise convolution and batch normalization, swish activation, and another pointwise convolution with dropout. Figure 3.3

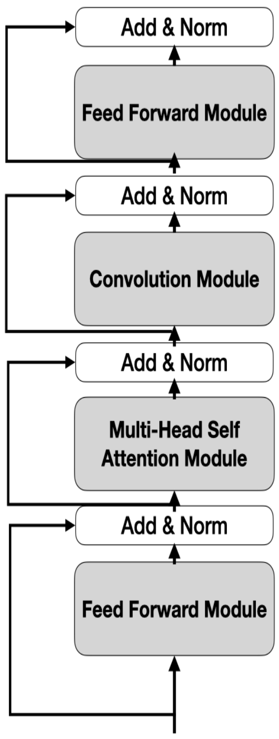

Figure 3.2: A diagram of a conformer encoder. Conformer block consists of feed forward module, multi-head self attention module, convolution module, and another feed-forward module. A convolution module and another feed-forward module is added to the original Transformer encoder structure. By having convolution module along with multi-head self attention module, conformer block can take account of global and local information.

shows the detailed structure of convolution module. By having a self-attention module and a convolution module together, Conformer can capture both global and local information and achieves better performance in automatic speech recognition (ASR) task than Transformer [13] based models. Just like predicting text tokens from speech in ASR, our model predicts teacher labels from speech and learns teacher models' representation.

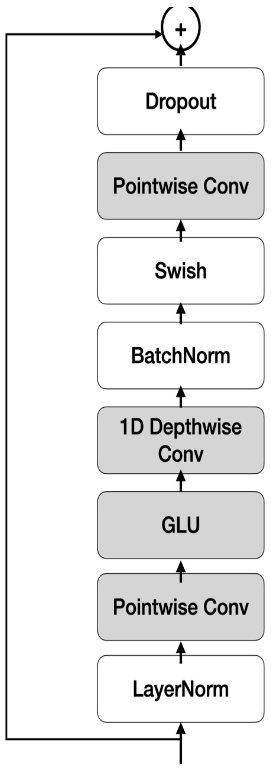

Figure 3.3: A diagram of a convolution module of Conformer encoder. The colored box consists of a neural network and empty box are operations with no trainable parameters.

To build a parameter efficient model, the channel dimension for the Conformer block is set to be 512, which follows the original setting of Conformer-L. Unlike the commonly used 768 channel dimensions, using 512 channel dimensions for Conformer encoder could reduce the number of parameters. Moreover, the CNN encoder's output feature has 512 channel dimensions as well, so there is no need to expand the output feature of CNN encoder to 768 dimensions before going through the representation encoder. No expansion in channel dimensions helps to further reduce the number of parameters. By reducing the channel dimensions and using Conformer blocks, our model could achieve less number of parameters than the model traiend by previous distillation approach [31].

3.4 Objective Function

Just like in HuBERT [5], our objective function, cross-entropy loss is computed over masked and unmasked time steps. Mask prediction helps to catch the context of the contents of the speech. The effect of mask prediction during the training will be reported in section 4.5 in details. We denote masked prediction loss as L_m and unmasked prediction loss, as L_u, which are defined as:

$$\begin{aligned} L_m &= \sum_{t \in M} logp_f(z_t|f(x_t)) \\ L_u &= \sum_{t \notin M} logp_f(z_t|f(x_t)) \end{aligned} \tag{3.1}$$

In Equation 3.1, z_t denotes the teacher labels at time step t, $f(\cdot)$ denotes the student model, x_t denotes the speech sample at time t. M denotes the time steps that are masked during the training. The overall prediction loss L_{pred} is computed as the weighted sum of L_m and L_u, which can be shown as

$$L_{pred} = \alpha L_m + (1 - \alpha) L_u \tag{3.2}$$

Chapter 4

Experiment

4.1 Implementation Details

As mentioned in section 3.2, our model consists of a 7-layer CNN encoder and 2 Conformer [7] blocks. Each Conformer block has 512 dimensions, 8 attention heads, 64 dimensions per head, and 31 kernel size for the convolution module. For the teacher labels, the output layer representations of pre-trained ContentVec [2] are used. The number of clusters for k-means clustering is 500. The α value for the prediction loss is set to be 0.8.

Our model is trained on six 12GB NVIDIA 2080Ti GPUs for 200k steps with a batch size of 48 utterances, taking about 27 hours. We use AdamW [39] optimizer with the initial learning rate of 2e-5, beta values of 0.9 and 0.98, weight decay factor of 1e-2, and epsilon value of 1e-6. The learning rate linearly increases to 2e-4 during the first 14000 steps and then linearly decreases to 0 by the end of the training step.

subset	hours	per-spkrs minutes	female spkrs	male spkrs	total spkrs
train-clean-100	100.6	25	125	126	251
train-clean-360	363.6	25	439	482	921
train-other-500	496.7	30	564	602	1166
dev-clean	5.4	8	20	20	40
dev-other	5.3	10	16	17	33
test-clean	5.4	8	20	20	40
test-other	5.1	10	17	16	33

Table 4.1: Detailed information of subset of LibiSpeech

4.1.1 Training Dataset

LibriSpeech LibriSpeech dataset [36] consists of train-clean-100, train-clean-360, train-other-500, dev-clean, dev-other, test-clean, and test-other. All the audio in the dataset are single channel recordings of audiobooks with the sample rate of 16k. The division between clean and other is measured by the word error rate (WER) of the speaker. Speakers with low WER are put into the ”clean” subset and speakers with high WER are put into the other subset. For the training of our model, we used train-clean-100, train-clean 360, and train-other 500 or commonly referred as 960-hour LibriSpeech training subset. The detailed information of each subset of LibriSpeech is shown in Table 4.1.

4.2 SUPERB

We evaluate our model on SUPERB [1], a benchmark dataset containing various supervised speech processing tasks. Our evaluation is done on the contents and semantic related tasks in SUPERB. These tasks include phoneme recognition (PR), automatic

Method	#Params Millions	PR PER ↓	ASR WER ↓	KS ACC ↑	IC ACC ↑	SF F1↑/ CER ↓
Generative Models						
APC [23]	4.11	41.9	21.2	91.0	74.69	70.4/50.8
TERA [25]	21.33	49.1	18.1	89.4	58.4	67.5/54.1
Mockingjay [24]	85.12	70.1	22.8	83.6	34.3	61.5/58.8
DeCoAR2.0 [26]	85.12	14.9	13.0	94.4	90.8	83.2/34.7
Discriminative Models						
CPC [27]	1.84	42.54	20.18	91.8	64.0	71.19/49.91
wav2vec [40]	32.54	31.5	15.8	95.5	84.9	76.3/43.7
wav2vec2.0 [6]	95.04	5.7	6.43	96.2	92.3	88.3/24.7
HuBERT [5]	94.68	5.4	6.42	96.3	98.3	88.5/25.2
ContentVec [2]	94.68	4.9	5.7	96.4	99.1	89.6/23.6
Distillation Models						
DistilHuBERT [3]	23.49	16.2	13.3	95.9	94.9	82.5/35.5
Our Model	20.42	15.3	13.5	95.8	93.7	80.1/36.9

Table 4.2: Evaluation results on content and semantic related tasks in SUPERB. The evaluation metrics for the benchmark are phoneme error rate (PER%), word error rate (WER%), accuracy (ACC%), F1 score (F1%), an concept error rate (CER%). The results for APC, TERA, Mockingjay, DeCoAR2.0, CPC, wav2vec, wav2vec2.0, HuBERT are from [1], and results for ContentVec and DistilHuBERT are from their original papers [2, 3].

speech recognition (ASR), keyword spotting (KS), intent classification (IC), and slot filling (SF). During the evaluation, our pre-trained model is frozen, and a minimal network is used to perform each downstream task to observe the representation performance.

There are ten baseline models to compare, which are contrastive predictive coding (CPC) [27], APC [23], Mockingjay [24], TERA [25], wav2vec [40], DeCoAR2.0 [26], wav2vec2.0 [6], HuBERT [5], ContentVec [2], and DistilHuBERT [3]. Among these model, APC, Mockingjay, TERA, and DeCoAR2.0 are trained by generative methods, CPC, wav2vec, wav2vec2.0, HuBERT, and ContentVec are trained by discriminative methods, and DistilHuBERt and our model are trained by distillation methods.

Our model ranked on average of 5.2 among eleven self-supervised learning models in speech listed in Table 4.2 and ranked third for having lowest number of parameters. Our model has the lowest number of parameters among the models that is made up of Transformer based encoders. When compared with the SSL model trained by distillation method [3], our model showed superior performance in phoneme recognition and comparable performance in automatic speech recognition and keyword spotting.

4.2.1 Parameter Size

The proposed model ranked third lowest number of parameters with the parameter size of 20.42 million among the existing SSL models that are used to compare the representation performance. Contrastive predictive coding (CPC) ranked the first for having lowest number of parameters with 1.84 million parameters. CPC has the simplest network structure which consists of 5 layers of CNN and single LSTM layer. APC, which consists of 3 gated recurrent unit (GRU), ranked second lowest with 4.11 million number of parameters. TERA, which has 3 Transformer layers, ranked fourth lowest with 21.33 million parameters followed by DistilHuBERT, which consists of 7 layers of CNN and 2 Transformer layers and has 23.49 million parameters. wav2vec, which has 19 CNN layers and 32.54 million parameters, ranked sixth and DeCoAR2.0, which has 12 Transformer layers and 85.12 million parameters, ranked sevnth. wav2vec2.0, HuBERT and ContentVec shared the same the structure which consists of 7 CNN layers and 12 Transformer layers, but wav2vec2.0 has 95.04 million parameters and is

Model	Network	#Params
Generative Models		
APC [23]	3-GRU	4.11
TERA [25]	3-Trans	21.33
Mockingjay [24]	12-Trans	85.12
DeCoAR2.0 [26]	12-Trans	85.12
Discriminative Models		
CPC [27]	5-Conv, 1-LSTM	1.84
wav2vec [40]	19-Conv	32.54
wav2vec2.0 [6]	7-Conv, 12-Trans	95.04
HuBERT [5]	7-Conv, 12-Trans	94.68
ContentVec [2]	7-Conv, 12-Trans	94.68
Distillation Models		
DistilHuBERT [3]	7-Conv, 2-Trans	23.49
Our Model	7-Conv, 2-Conf	20.42

Table 4.3: The network structure of SSL models in speech. GRU stands for gated recurrent unit, Conv stands for convolutional neural network, Trans stands for Transformer, and Conf stands for Conformer.

slightly larger than HuBERT and ContentVec which have 94.68 million parameters. The proposed model reduces the teacher models' parameter size by 78%.The details of network structure of SSl models can be found in Table 4.3.

4.2.2 Phoneme Recognition (PR)

For phoneme recognition, our proposed model ranked fifth with phoneme error rate (PER) of 15.3. There are four SSL models that showed better performance than the proposed model which are ContentVec, HuBERT, wav2vec2.0, and DeCoAR2.0. ContentVec ranked first with 4.9 PER, HuBERT ranked second with 5.4 PER, wav2vec2.0 ranked third with 5.7 PER and DeCoAR2.0 ranked fourth with 14.9 PER. Notice that the models that performed better than the proposed model have 85 to 95 million parameters which are 4.2 to 4.6 times larger in size than our model. When compared to DeCoAR2.0 which has 4.2 more parameters than our model, the PER difference was only 0.4. Our model did better than DistilHuBERT by 0.9 PER and other existing models like CPC, APC, TERA, Mockingjay, and wav2vec. The result for phoneme recognition in SUPERB can be found in third column of Table 4.2.

4.2.3 Automatic Speech Recognition (ASR)

For automatic speech recognition, our proposed model ranked sixth with word error rate (WER) of 13.5. Our model performed better than CPC, TERA, APC, Mockingjay, and wav2vec. There are five SSL models that performed better than our model which are ContentVec, HuBERT, wav2vec2.0, DeCoAR2.0, and DistilHuBERT. ContentVec ranked first with 4.9WER, HuBERT ranked second with 6.42 WER, wav2vec2.0 ranked third with 6.43 WER. Our model had close results with DeCoAR2.0 and DistilHuBERT. DeCoAR2.0, which has 4.2 times larger number of parameters than our model resulted with 13.0 WER. DisitlHuBERT has 3 million more parameters than our model and resulted with 13.3 WER. The automatic speech recognition results can be found in the fourth column of Table 4.2.

4.2.4 Keyword Spotting (KS)

For keyword spotting, our model ranked fifth with the classification accuracy of 95.8% among the SSL models listed in Table 4.2. Our model showed only 0.01% difference with DistilHuBERT, which ranked fourth in keyword spotting and had 3 million more parameters. Our model showed superior performance than CPC, TERA, APC, Mockingjay, wav2vec, and DeCoAR2.0. Except for CPC and APC, TERA, APC, Mockingjay, wav2vec, and DeCoAR2.0 had about 0.9 million to 65 million more parameters than our model. ContentVec ranked first with 96.4% accuracy, HuBERT ranked second with 96.3%, and wav2vec2.0 ranked third with 96.2%. The results for keyword spotting can be found in the fifth column of Table 4.2.

4.2.5 Intent Classification (IC)

For intent classification, our model ranked fourth among the existing SSL models listed in Table 4.2 with the classification accuracy of 93.7%. Our model performed better than CPC, TERA, APC, Mockingjay, wav2vec, DeCoAR2.0 and wav2vec2.0. It is noticeable that our model performed better than wav2vec2.0, which has 4.6 times more parameters and shows better performance in other downstream tasks in Table 4.2. ContentVec ranked first with 99.1% accuracy, HuBERT ranked second with 98.3% accuracy, and DistilHuBERT ranked third with 94.9% accuracy. The results for intent classification can be found in the sixth column of Table 4.2.

4.2.6 Slot Filling (SF)

For slot filling, our model ranked sixth among the existing SSL models listed in Table 4.2 with 80.1 F1 score and 36.9 concept error rate(CER). Our model shows superior results than CPC, TERA, APC, Mockingjay, wav2vec. ContentVec ranked first with 89.6 F1 score and 23.6 CER, HuBERT ranked second on F1 score and third on CER with 89.6 F1 and 25.2 CER, and wav2vec2.0 ranked third on F1 score and second on CER. DeCoAR2.0 ranked fourth with 83.2 F1 score and 34.7 CER followed by Distil-

Model	ABX(w)↓	ABX(a)↓
Our model	**3.58**	**4.23**
ContentVec	6.01	6.32
HuBERT	6.06	7.20
wav2vec2.0	8.70	10.34
DistilHuBERT	35.8	39.7

Table 4.4: ZeroSpeech2021 ABX testing results. The results for ContentVec, HuBERT, and wav2vec2.0 are from [2], and DistilHuBERT results are manually obtained by using the officially implemented model with the checkpoint used to report in the paper [3].

HuBERT which resulted with 82.5 F1 score and 35.5 CER. The results for slot filling can be found in seventh column of Table 4.2.

4.3 ZeroSpeech2021

For ZeroSpeech2021, ContentVec, HuBERT, wav2vec2.0, and DistilHuBERT are used to compare the performance. ContentVec, HuBERT, and wav2vec2.0 are chosen because they show state of the art performance in ASR tasks, and DistilHuBERT is selected to compare the performance of a SSL model trained using knowledge distillation method. Among four criteria in ZeroSpeech2021, ABX testing, which evaluates acoustic and phonetic representation power, is selected. The evaluation metric is error rate of a sound that is grouped into wrong category.

ABX(w) ABX(w) tests whether a triphone utterances spoken by the same person is categorized into the right category. Our model ranked first among five models with error rate of 3.58. The difference of error rate between the our model and the second best model [2] is 2.43. Notice that ContentVec and HuBERT showed difference in 0.05, and both of them performed far worse than our model. As shown in Table 4.2, both ContentVec and HuBERT has 4.6 times more parameters than our model. wav2vec2.0 resulted with 8.7, and DistilHuBERT ranked the last with error rate of 35.8. This is significant that our model, a student, showed better performance than the teacher model.

ABX(a) ABX(a) tests whether a triphone utterances spoken by a different person is categorized into the right category. Out model ranked first among five models with error rate of 4.23. ContentVec ranked second with 6.23 error rate, HuBERT ranked thrid with 7.20 and wav2vec2.0 ranked fourth with 10.34 error rate. The difference between ContentVec, the teacher model, and our model or the student, comes out to be 2.09. ContentVec, HuBERT, and wav2vec2.0 have 4.6 times more parameters than our model. DistilHuBERT which uses distillation method to train, ranked last with 39.7 error rate.

As shown in the Table 4.4, our model shows the best performance among five models. The main contribution for the result can be thought as the number of teacher labels. When training our model for distillation, the number of labels is 500. If each label can be thought as finite unit of sound in speech, then our model learns to distinguish the minor difference in sound. ContentVec and HuBERT uses 100 labels for learning, and wav2vec2.0 uses 320 labels. DistilHuBERT does not uses pseudo-label prediction for training but tries to learn continuous representation of HuBERT. Therefore, having finite pseudo-labels or number of quantization can be thought as the contributing factor to distinguish minor sound difference of speech.

4.4 Ablation Study

This section evaluates how each of our choice in the distillation technique contributes to the performance of our model. Four variations in technique are reported in Table 4.5, which are (1) when our model is trained with Transformer encoders instead of Conformer, (2) when our model is trained without weight initialization of the CNN encoder, (3) when our model is trained without masked prediction and (4) when our model is trained with HuBERT teacher labels. Just like in [3], we also report ASR, KS, and IC performance for ablation because these tasks represent recognition, detection, and semantics in content related downstream tasks.

	ASR	KS	IC
Method	WER↓	ACC↑	ACC↑
Our Model	13.5	95.8	93.7
w/ Transformer	18.7	94.9	89.2
w/o weight init.	14.0	95.3	93.7
w/o masking	13.3	94.9	92.7
w/ HuBERT labels	15.2	95.6	88.3

Table 4.5: Ablation study of the training techniques used

4.4.1 Transformer Encoder

When using two Transformer encoders with 768 dimensions instead of Conformer blocks are used, degradation in performance could be observed. The model structure becomes the same as DistilHuBERT, which means that the model tested for this ablation has 3 million more parameters. As the result, WER in ASR increases decreased by

5.2%, KS decreases by 0.9%, and IC decreases by 4.5%. Notice that performances of ASR and IC degraded significantly compared to the proposed model, despite having 3 million more parameters. This result shows that the proposed distillation method suits better for Conformer encoder than Transformer. The third row of Table 4.5 shows the ablation result for using Transformer encoder instead of Conformer.

4.4.2 Weight Initialization

When the weight initialization for CNN encoder is not used, WER for ASR increases by 0.5% and KS decreases by 0.5%. Any type of weight initialization is not done for CNN encoder such as Xavier normal or Kaiming normal initialization. This result shows that weight initialization offers the good starting point, and the idea of fine tuning can also be applied in this case. The fourth row of Table 4.5 shows the ablation result for not using weight initialization.

4.4.3 Mask Prediction

Training our model without mask prediction is done by using a regular cross-entropy loss for every teacher label. Using no mask prediction for the loss decreases the performance in KS and IC because these tasks related to the context of the contents of the speech. Since masked prediction is to capture context of the contents, this result fulfills the intention of using mask prediction in self-supervised learning.

Interestingly, WER for automatic speech recognition decreases by 0.2. This result shows tie performance to DistilHuBERT when using without masked prediction. Without mask prediction, the network focus on predicting the sound of the specific location rather than considering the context of the speech. Therefore, the model is suitable for direct transcription of the speech. The fifth row of Table 4.5 shows the ablation result for not using mask prediction for objective function.

4.4.4 HuBERT Teacher Labels

For creating HuBERT teacher labels, HuBERT's seventh layer representations are used. According to [2], seventh layer representation has lower speaker identification accuracy with comparable content information to sixth layer, which behaves similar to ContentVec. The number of labels for teacher labels created by HuBERT's 7th layer representation is 500 as well.

When our model is trained with the teacher labels created by HuBERT's representations, automatic speech recognition performance decreases by 1.7 %, keyword spotting performance decreases by 0.2 % and intent classification performance decreases by 5.4%. A noticeable difference can be observed for ASR and IC. This result shows that using ContentVec representations for the teacher labels contribute significantly for the distillation performance. The sixth row of Table 4.5 shows the ablation result for using HuBERT representations instead of ContentVec representations for teacher labels.

	ASR	KS	IC
Method	WER$\downarrow$	ACC$\uparrow$	ACC$\uparrow$
ContentVec	5.7	96.4	99.1
Our model(cont.)	24.6	80.5	30.2
Our model(500)	13.5	95.8	93.7

Table 4.6: Distillation analysis for the proposed model

4.5 Analysis of Distillation Method

This section analyzes the distillation method used for training our model. Our model is trained by a distillation method to predict the teacher labels that are created by quantization of continuous ContentVec's outer representation. The number of labels are set to 500 because both ContentVec and HuBERT uses 500 labels for training.

One can ask, what if our model learns continuous representation instead of quantized representation of ContentVec? The answer to that question is shown in Table 4.6. To learn continuous representation, the objective function used for DistilHuBERT is used. This objective function combines L1 regularization between predicted representation and the teacher representation with the loss that increases cosine similarity. The detailed loss is shown in Equation 2.6. When our model is trained to learn the outer representation of ContentVec, WER for ASR increases by 11.1, accuracy for keyword spotting decreases by 10.3% and the accuracy for intent classification decreases by 63%. The representation performance decreases for all three downstream tasks, and the performance decrease in intent classification is very noticeable.

From the result shown in Table 4.6, the conformer encoder is not well suitable for predicting continuous representation. This result shows interesting contradiction between the Conformer encoder and Transformer encoder. From table 4.5, Transformer encoder shows performance decrease when learning by predicting teacher labels. From these results, it can be assumed that Conformer better learns discretized representation and Transformer better learns continuous representation in our model structure.

4.6 Inference Speed

We evaluate the inference speed in two different computational environments. Specifically, we checked the inference speed of HuBERT, DistilHuBERT, and our model when using single NVIDIA 2080 Ti GPU or 4 CPUs. The inference time is measured

Model	# param. Millions	Inf (G) seconds	Inf (C) seconds
HuBERT	94.68	57.5	1703
DistilHuBERT	23.49	32.3	1038
Our Model w/ HuBERT	20.42	28.9	957

Table 4.7: Inference speed comparison. Inf (G) is the inference time measured using single GPU, and Inf (C) is the inference time measured using 4 CPUs. An average seconds of 3 runs is reported for each measurement. For the objective comparison, our model trained on using HuBERT teacher labels are used.

Model	MAC(G)
DistilHuBERT	3.17
Our Model	3.24

Table 4.8: The number of MAC for performing inference of a 1 second audio with sampler rate of 16k. The unit of MAC is giga or 10^9.

by getting representations for entire dev-clean subset of LibriSpeech [4]. Although the number of parameters is the same, our model trained with HuBERT teacher labels is used for fair comparison. When observing the inference time for using single GPU, we performed 10 dummy runs for warm-up before actually performing inference. No warm-up was done for performing inference using 4 CPUs. An average of 3 trials are measured. Table 4.7 shows the results of inference time measured in both computational environment for three models.

Moreover, the number of multiply-accumulate is also measured between our model and DistilHuBERT to objectively measure the inference speed. Only DistilHuBERT and our model are used to compare because these models are purposefully built to perform fast inference compared to HuBERT. The result for MAC is reported in Table

4.8.

Inference Time on GPU When inference time is measured using single GPU, our model took the average of 28.9 seconds which is 99% faster than HuBERT or the teacher model. Our model took less time than DistilHuBERT, which took 32.3 seconds. Our model shows fastest inference speed using single GPU among three models.

Inference Time on CPU When inference time is measured using 4 CPUs, our model took the average of 957 seconds, which is 78% faster than HuBERT that took the average of 1703 seconds. DistilHuBERT took the average of 1038 seconds and was slower than our model. Our model shows fastest inference speed when using 4 CPUs among three models.

Multiply Accumulate Multiply accumulate (MAC) counts the number of multiplication and addition to perform inference of a specific size of input. It can be be thought that having less number of MAC would lead to faster inference speed because of less number of multiplication or addition operations to complete inference. The input is 1 second single channel audio that has the sample rate of 16k. As reported in Table 4.8, our model had slightly more MAC than DistilHuBERT. The reason why our model has more MAC than DistilHuBERT is that our Conformer encoder has convolution module which requires more multiplication and addition calculation than DistilHuBERT's Transformer encoder which is made up of fully-connected layers.

Chapter 5

Conclusion

5.1 Overview

We propose a parameter efficient Conformer-based speech representation model. Our model is developed by a distillation method which predicts teacher labels created by quantizing representations from the pre-trained self-supervised learning model for speech. Our model uses Conformer encoder instead of Transformer encoder and could successfully reduce the number of parameters with tolerable degradation in performance. With the reduction of number of parameters, our model shows faster inference time compared to the teacher model and the model developed by existing distillation method.

Specifically, our model ranked an average of 5.2 among eleven SSL models in five content related downstream tasks in SUPERB. For intent classification, our model outperformed wav2ve2.0 which showed the state-of-the-art performance in automatic speech recognition. Among eleven models, our model ranked third lowest number of parameters and had three million less number of parameters than DistilHuBERT, which is trained by existing distillation approach. Our model showed superior to par performance in phoneme recognition, automatic speech recognition, and keyword spotting than DistilHuBERT.

Not just in SUPERB, our model outperformed the state of art SSL models such as ContentVec, HuBERT, and wav2vec2.0 in ABX testing of ZeroSpeech 2021. There are two criteria in ABX testing, and our model outperformed in both criteria. The SSL models such as ContentVec, HuBERT, and wav2vec2.0 have at least 4.6 times more parameters than our model.

When ablation study is conducted, the training methods used contribute significantly to enhance the performance of our model. Especially, using Conformer encoder instead of Transformer and creating teacher labels with ContentVec instead of HuBERT showed large improvement in ASR and intent classification performance. For distillation method, predicting discretized teacher model's representations was better distillation method for our model than predicting continuous representations.

For inference speed, our model reduced the inference speed by 99% when single GPU is used and by 78% when 4 CPUs used. Our model showed faster inference speed than DistilHuBERT. However when MAC is measured, our model had more number of MAC than DistilHuBERT.

5.2 Future Work And Limitation

Although our model could reduce the number of parameters with tolerable degradation, more research and experiments need to be conducted to use SSL models in real-life situation or in on-device setting. Just like in [30], weight quantization can be done to increase the inference speed. Or weight pruning method can be applied to reduce the number of weights of the models. The performance of the model after applying weight quantization and pruning needed to be experimented.

Moreover, our model is not tested for different languages. The downstream task performance needed to be seen when our model is fine-tuned with different language besides English. There are situations when non-native English speakers need assistant in speech-related downstream tasks as well.

To actually apply SSL models in real-life situation and on-device setting, the model needs to be more handy and robust to different settings. Not just reducing the number of parameters of existing models but further compressing the model is needed to be done. The test cases for real-life situation, not just the benchmark used in academia needs to be considered and researched for the future work.

Bibliography

[1] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, *et al.*, "Superb: Speech processing universal performance benchmark," *arXiv preprint arXiv:2105.01051*, 2021.

[2] K. Qian, Y. Zhang, H. Gao, J. Ni, C.-I. Lai, D. Cox, M. Hasegawa-Johnson, and S. Chang, "Contentvec: An improved self-supervised speech representation by disentangling speakers," in *International Conference on Machine Learning*, pp. 18003–18017, PMLR, 2022.

[3] H.-J. Chang, S.-w. Yang, and H.-y. Lee, "Distilhubert: Speech representation learning by layer-wise distillation of hidden-unit bert," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7087–7091, IEEE, 2022.

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

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

[6] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 12449–12460, 2020.

[7] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, *et al.*, “Conformer: Convolution-augmented transformer for speech recognition,” *arXiv preprint arXiv:2005.08100*, 2020.

[8] K. Lakhotia, E. Kharitonov, W.-N. Hsu, Y. Adi, A. Polyak, B. Bolte, T.-A. Nguyen, J. Copet, A. Baevski, A. Mohamed, *et al.*, “On generative spoken language modeling from raw audio,” *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1336–1354, 2021.

[9] T. A. Nguyen, E. Kharitonov, J. Copet, Y. Adi, W.-N. Hsu, A. Elkahky, P. Tomasello, R. Algayres, B. Sagot, A. Mohamed, *et al.*, “Generative spoken dialogue language modeling,” *arXiv preprint arXiv:2203.16502*, 2022.

[10] A. Polyak, Y. Adi, J. Copet, E. Kharitonov, K. Lakhotia, W.-N. Hsu, A. Mohamed, and E. Dupoux, “Speech resynthesis from discrete disentangled self-supervised representations,” *arXiv preprint arXiv:2104.00355*, 2021.

[11] F. Kreuk, A. Polyak, J. Copet, E. Kharitonov, T.-A. Nguyen, M. Rivière, W.-N. Hsu, A. Mohamed, E. Dupoux, and Y. Adi, “Textless speech emotion conversion using decomposed and discrete representations,” *arXiv preprint arXiv:2111.07402*, 2021.

[12] E. Kharitonov, A. Lee, A. Polyak, Y. Adi, J. Copet, K. Lakhotia, T.-A. Nguyen, M. Rivière, A. Mohamed, E. Dupoux, *et al.*, “Text-free prosody-aware generative spoken language modeling,” *arXiv preprint arXiv:2109.03264*, 2021.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

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

[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[16] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22243–22255, 2020.

[17] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

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

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot

learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[22] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

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

[24] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6419–6423, IEEE, 2020.

[25] A. T. Liu, S.-W. Li, and H.-y. Lee, "Tera: Self-supervised learning of transformer encoder representation for speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2351–2366, 2021.

[26] S. Ling and Y. Liu, "Decoar 2.0: Deep contextualized acoustic representations with vector quantization," *arXiv preprint arXiv:2012.06659*, 2020.

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

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

[29] H.-S. Choi, J. Lee, W. Kim, J. Lee, H. Heo, and K. Lee, "Neural analysis and synthesis: Reconstructing speech from self-supervised representations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16251–16265, 2021.

[30] Z. Peng, A. Budhkar, I. Tuil, J. Levy, P. Sobhani, R. Cohen, and J. Nassour, "Shrinking bigfoot: Reducing wav2vec 2.0 footprint," *arXiv preprint arXiv:2103.15760*, 2021.

[31] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[32] S. Srivastava, Y. Wang, A. Tjandra, A. Kumar, C. Liu, K. Singh, and Y. Saraf, "Conformer-based self-supervised learning for non-speech audio tasks," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8862–8866, IEEE, 2022.

[33] P. Warden, "Speech commands: A public dataset for single-word speech recognition," *Dataset available from http://download. tensorflow. org/data/speech_commands_v0*, vol. 1, 2017.

[34] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio, "Speech model pre-training for end-to-end spoken language understanding," *arXiv preprint arXiv:1904.03670*, 2019.

[35] C.-I. Lai, Y.-S. Chuang, H.-Y. Lee, S.-W. Li, and J. Glass, "Semi-supervised spoken language understanding via self-supervised speech and language model pretraining," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7468–7472, IEEE, 2021.

[36] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.-E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen, *et al.*, "Libri-light: A benchmark for

asr with limited or no supervision," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7669–7673, IEEE, 2020.

[37] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[38] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[39] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

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

초록

발화(speech)에 대한 자가지도학습모델(self-supervised learning model)은 수많은 발화 데이터를 사용해 훈련을 진행 후, 발화에 대한 특징(representation)을 사용하여 관련된 과제를 푸는데 큰 도움을 준다. 발화 관련된 과제 중에 가장 흔하게 사용되는 예로는 음성인식이나 음소 인식이 있으며, 최신 자기자도학습 모델들은 이 과제에서 매우 우수한 성적을 거두었다.

우수한 성적을 거두었음에도 불구하고 자가지도학습모델은 실제 우리 주변상황에서 음성인식을 위해 사용되는 일이 없다. 그 이유로는 굉장히 많은 양의 모델의 매개변수(parameter) 때문이다. 많은 양의 매개변수를 갖고 있는 모델은 실행하기 위해서는 굉장히 많은 컴퓨터 자원을 필요로 한다.

이러한 단점들을 보완하기 위해 매개변수 효율적인 자가지도 학습 모델의 개발에 대한 연구를 진행하였다. 모델 지식 증류기법(knowledge distillation)을 통해 기존의 최신 성능을 내는 자가지도학습모델을 컨포머(Conformer)를 기반으로한 매개변수가 적은모델을 학습 시켰다. 구체적으로 기존의 최신 성능을 내는 자가지도학습모델의 표현(representation)들의 이산화(discretization)을 통해서 의사 레이블(pseudo-label)을 만들어 학습을 진행하였다.

본 연구에서 개발된 모델은 SUPERB와 ZeroSpeech2021을 통해서 검증을 진행하였다. SUPERB 같은경우 기존의 자가지도학습 모델들과 유사하거나 더 좋은 성적을 거두었다. ZeroSpeech2021에서는 최신 성능의 ContentVec, HuBERT, 그리고 wav2vec2.0보다 훨씬 더 좋은 성적을 거두었다.

본 연구에서 개발된 모델지식증류 기법을 세분화 해서 분석을 진행하였고 어

떠한 요소가 성능 향상에 도움이 되었는지 연구하였다. 모델지식증류 기법을 통해 개발된 모델은 기존의 최신 모델의 매개변수 개수를 78% 줄였다. 또한 기존 최신 모델 대비 본 연구에서 개발된 모델은 GPU 하나를 사용해서 추론(inference)을 진행하였을 때 99% , 4개의 CPU를 사용하였을 때는 78%의 속도 향상을 보였다.

주요어: 자기지도 학습, 특징 학습, 모델 경량화, 모델 지식 증류기법

학번: 2021-24076

ACKNOWLEGEMENT

2021년 봄 처음 MARG에 들어왔을 때 모든 것이 낯설기만 하였습니다. 시간이 지나고 이교구 교수님 그리고 연구실의 선,후배 분들과 교류하는 시간이 많아지면서 낯선 감정보다 설레는 마음과 용기 그리고 좋은 영감을 더욱더 많이 받게 되었습니다. 2022년 초 부터 진행하던 연구 주제의 연구결과가 잘 안나와서 급하게 연구 주제를 바꾸게 되었습니다. 급하게 바꾼 연구 주제를 진행하느라 힘들고 지친 시간이 있었지만 교수님과 연구실 동료분들의 조언과 응원으로 잘 버티면서 연구를 마무리 할 수 있었습니다.

MARG의 좋은 환경은 한순간에 생겨났다고 생각하지 않습니다. 이교구 교수님의 노력과 수많은 선배님들의 연구와 노력이 지금의 MARG까지 이어오게 되었고 그 혜택을 지금의 제가 누리게 되었다고 생각합니다. 마그에서 배우고 경험한 것들을 선한 영향력으로 세상에 전파하고 싶은 마음을 담아 감사의 글을 적습니다.