Leveraging Pre-training Models for Speech Processing

I. Overview

Pre-training has proven to be crucial in advancing the state of speech, natural language processing (NLP), and computer vision (CV) research in recent years. The network is first trained via a pre-training task, leveraging ubiquitous unlabeled data, which is also known as self-supervised learning (SSL). The pre-training is usually application-agnostic, and the pre-trained models are transferred to multiple downstream applications. SUPERB [1]–[3], LeBenchmark [4], and NOSS [5] are the initial effort to evaluate the pre-trained models on their generalizability across various speech and audio processing tasks and find that the pre-trained models achieve outstanding performance on a wide range of tasks. The publicly available pre-trained models significantly benefit the small players. For example, with a pre-trained model, one only needs to train a two-layer LSTM as a downstream model to achieve 3% WER on Librispeech [1]. Without pre-training, a network with more than ten layers is usually required to achieve the same-level performance.

The project aims to expand the existing benchmarking to provide a comprehensive understanding of pre-trained networks and develop new techniques to leverage the pre-trained models better. We believe this project will broadly push the front of network pre-training technology in speech. We have explored the following research directions.

Greener Pre-trained Models. Despite the success of these vast models, they require large memory and high computational costs [6]. To what extent the large capacity is needed is an open question, but the high computation cost and memory usage pose a high technical barrier to the adoption, and, more importantly, training of these models. In this project, we aim to tackle the high computational cost of self-supervised learning by developing efficient pre-trained models regarding computation and memory footprint. We start by exploring various approaches of network compression, including iterative pruning, low-rank approximation, knowledge distillation, etc., to compress pre-trained networks. These technologies are helpful in NLP [7], but have not been fully explored in pretrained speech models. The results are in Section II and submitted to ICASSP 2023 [8]. We observe that in speech processing, the number of frames along the time axis is often the dominant factor in runtime. In Section III, we investigate subsampling techniques to reduce sequence length. The results have been published at SLT 2022 [9].

Model Robustness. The pre-training models show outstanding performance on various applications, but their failure modes are still unclear. Domain shifts caused by mismatches between training data and testing data usually occur in real-world scenarios. Are pre-training models robust to domain shift? In this project, we focus on the setting that the training data of downstream tasks contain clean speech while the testing data has distortions. There is some preliminary research about domain shift [10], [11] of pre-training models, but here we conducted a study focusing on compressed models, and we further study how to make the pre-trained models more robust. The results are shown in Section IV and have been published by SLT 2022 [12].

Visual-enhanced Pre-trained Models. It has been known that visual information improves speech representations [13]. Much effort was put into using paired images and spoken captions to help speech processing [14], and they are usually called visually grounded speech models (VGS). E.g., the recent Fast-Slow Transformer for Visually Grounding Speech (FaST-VGS and FaST-VGS+) succeeds in many speech processing tasks by utilizing transformers and cross-modal attention mechanisms to perform image-speech retrieval and semantic tasks [13], [15]. Moreover, VGS models trained with retrieval objectives can extract semantic and word-level information from speech [16], which is difficult to achieve by training solely with speech [17]. This project focuses on leveraging not only paired image-speech data but also an image-text pre-trained model to enhance speech SSL models. By taking advantage of paired image-speech and image-text data, we can bridge speech and text domains via images. The results are in Section V and published at SLT 2022 [18].

How to use the pre-trained models. So far, there are two main methods of using the speech pre-training models: (1) freezing the representation models and using them as feature extractors, and (2) fine-tuning the representation models with downstream tasks. Are there other ways to use them? This project looks for more efficient ways to leverage the pre-trained models in downstream tasks. In the NLP community, the adapter-based method [19], [20] and prompt/instruction learning [21] have achieved competitive performance compared with fine-tuning, but the exploration in speech is insufficient. In Section VI, we show the exploration results of using adapters in speech pre-training models, and the results were published in SLT 2022 [22]. On the other hand, in Section VII, we explore the possibilities of using prompt learning on speech pre-training models, and some preliminary results have been published at INTERSPEECH 2022 [23], while more complete results are submitted to ICASSP 2023 [24].

Pre-trained Models for Prosody. Here we expand the use of pre-trained techniques for modeling prosody. Previous work has studied the ability of pre-trained models to extract phonetic and speaker information [1], [4]. However, it is unclear whether they can extract prosodic information from speech, which is essential for interactive dialogue systems. The project extends the existing pre-training models to support prosody for pragmatic-related classification tasks, including turn-taking, micro-emotion, and prediction of response prosody. The results are presented in Section VIII and published in SLT 2022 [25][1].

Pre-trained Models for low-resource South African languages. We study how self-supervised pre-training and semi-supervised training can be used to leverage untranscribed audio data in underrepresented languages. In semi-supervised training, the initial acoustic model is initialized with a flat-start initialization or with cross-lingual transfer learning with spectral representations as inputs. We propose another initial acoustic model trained on learned multilingual speech representations. In particular, we explore how 200 hours of untranscribed South African soap operas data can help to improve the initial acoustic model trained only on 12.7 hours of manually transcribed code-switched speech between four South African languages and English [26]. Please refer to Section IX for the results. The results have been submitted to ICASSP 2023 [27].

Unsupervised Automatic Speech Recognition. During the workshop, we developed the ESPnet Unsupervised ASR Open-source Toolkit (EURO), an end-to-end open-source toolkit for unsupervised ASR (UASR). EURO adopts the state-of-the-art UASR learning method introduced by the Wav2vec-U [28], originally implemented at FAIRSEQ, which leverages self-supervised speech representations and adversarial training. In addition to wav2vec2, EURO extends the functionality and promotes reproducibility for UASR tasks by integrating S3PRL and k2, resulting in flexible frontends from many self-supervised models and various graph-based decoding strategies. EURO is implemented in ESPnet and follows its unified pipeline to provide UASR recipes with a complete setup. This improves the pipeline's efficiency and allows EURO to be easily applied to existing datasets in ESPnet. Extensive experiments on three mainstream self-supervised models demonstrate the toolkit's effectiveness and achieve state-of-the-art UASR performance on TIMIT and LibriSpeech datasets. EURO will be publicly available at `https://github.com/espnet/espnet`, aiming to promote this exciting and emerging research area based on UASR through open-source activity. For more information about EURO, please refer to Section X-A. The introduction of EURO has been written as a paper and submitted to ICASSP 2023 [29].

Usage Extension of Unsupervised ASR. Spoken language understanding (SLU) is a task aiming to extract high-level semantics from spoken utterances. Previous works have investigated the use of speech pre-trained models and textual pre-trained models, which have shown reasonable improvements to various SLU tasks [30]. However, because of the mismatched modalities between speech signals and text tokens, previous methods usually need complex designs of the frameworks. We proposes a simple yet efficient unsupervised paradigm that connects speech and textual pre-trained models, resulting in an unsupervised speech-to-semantic pre-trained model for various tasks in SLU. To be specific, we propose to use unsupervised ASR as a connector that bridges different modalities used in speech and textual pre-trained models. Our experiments show that unsupervised ASR itself can improve the representations from speech self-supervised models. More importantly, it is shown as an efficient connector between speech and textual pre-trained models, improving the performances of five different SLU tasks. Notably, on spoken question answering (SQA) [31], we reach the state-of-the-art result over the challenging NMSQA benchmark. The results are shown in Section X-B and submitted to ICASSP 2023 [32].

The 2022 Jelinek Memorial Summer Workshop on Speech and Language Technologies was an incredible opportunity for an international team of experts to come together and develop cutting-edge speech SSL technology. Hosted at the prestigious Johns Hopkins University, this six-week workshop was made possible thanks to the generous support of Amazon, Microsoft, and Google. The team was also grateful for the computational resources provided by the Taiwan Web Service (TWS) and the Maryland Advanced Research Computing Center (MARCC). For more information about the amazing team behind this project, please visit their webpage at `https://jsalt-2022-ssl.github.io/`.

II. SSL Model Parameter Compression

A. Introduction - Model Compression

Model compression as a concept is broad. Any approach that returns a small yet performant model contributes to the trade-off and should be considered in the opportunity cost of compression. Common compression techniques, such as iterative pruning [33]–[35], low-rank approximation [36]–[38], and knowledge distillation [39], have little in common, another sign that the concept of model compression is broad. In fact, training a small model from random initialization and neural architecture search can both be treated as model compression, though nothing is really compressed. In this report, we limit the study to iterative pruning, low-rank approximation, and knowledge distillation. We also provide a simple baseline, where the forward process of the large model is stopped early. Stopping the forward process early not only reduces computation, but also maintains satisfactory performance because low layers are often already useful for downstream tasks [17].

Iterative pruning, low-rank approximation, and knowledge distillation, have been explored extensively in the supervised setting, for example, on automatic speech recognition [40] and machine translation [41]. However, only recently are these techniques applied to self-supervised learning. Some study compression for a particular downstream task [42], while some aim to maintain the self-supervised loss [43]–[46]. Moreover, unlike regular tasks with a single objective, there are multiple measurements to compare. In addition to the common accuracy measures, some compare the number of parameters [44], while some compare wall-clock time [46].

In this report, we focus on studying to what degree we can maintain self-supervised loss when compressing and to what degree maintaining the self-supervised loss leads to generalization to different downstream tasks. We make several contributions. We design a clean training pipeline to train a 12-layer Transformer with a simplified HuBERT loss, focusing

[1]The paper received the Best Paper Award of SLT 2022.

on reproducibility. We compare iterative pruning, low-rank approximation, and knowledge distillation for compressing the pre-trained 12-layer Transformer. We also provide a comprehensive study of theoretical and practical speed-up of the compression techniques, measuring the wall-clock time, the number of parameters, and the number of multiply-accumulate operations (MACs) per one second speech, charting the landscape of compressing Transformer-based self-supervised models.

B. MelHuBERT

Since the loss is not accessible for the pre-trained HuBERT[2] we train a self-supervised model from random initialization. We make several changes to simplify the training pipeline, in line with the goal of model compression. Our model, MelHuBERT, is a regular 12-layer Transformer that directly takes Mel spectrograms as input, as opposed to wave samples in HuBERT. The 7 convolutional layers in HuBERT, accounting for 33% of MACs in HuBERT for producing 20-ms frames, are removed in MelHuBERT. We train two variants, MelHuBERT-10ms and MelHuBERT-20ms. MelHuBERT-10ms takes the typical Mel spectrograms at 10 ms frame period, while MelHuBERT-20ms splice every two frames to achieve a 20 ms frame period.

In addition to the slight modification in architecture, we also simplify the training objective. We first follow HuBERT, train a k-means model on Mel spectrograms, and quantize a sequence of frames into a sequence of cluster labels. Instead of using an ad-hoc training loss, we simply use cross entropy to predict the cluster labels for a masked input. The loss is only computed on the masked portion, similar to HuBERT.

C. Model Compression

In this section, we briefly review the specifics of model compression techniques used in this report, including weight pruning, head pruning, low-rank approximation, knowledge distillation. Except knowledge distillation, the others can be seen as instances of iterative pruning. Iterative pruning performs the following two steps iteratively.

- Prune a block of weights.
- Train the pruned network until the loss converges.

What constitutes a block and how a block is chosen can be customized. Pruning can be implemented by zeroing out the chosen weights, or by avoiding the computation entirely. We do not consider one-shot pruning as it typically achieves a worse compression [47].

1) Weight Pruning: Weight pruning is one of the earliest pruning algorithms [48]. The idea is based on the finding that the increase in loss due to small changes to the weights of neural networks can be quickly recovered by a small amount of training. Weight pruning usually is an instance of iterative pruning, which converges faster and with lower training loss comparing to one-shot pruning. In this report, we adopt one of the simplest criterion for pruning [34], [47], setting the weight to zero when the absolute value is below a threshold. The threshold is determined based on the amount of remaining weights to be pruned, and we will have a schedule for the amounts to be pruned in the experiments.

[2]https://github.com/facebookresearch/fairseq

We follow the usual implementation [34], using a binary mask to keep track of what weights are pruned. Weights that are pruned are never instantiated again. Since the mask does not exhibit any structure, speed-up of weight pruning can only come from sparse parallel computations or dedicated hardware [33]. In this report, we do not explore these options and only report theoretical speed-up.

2) Head Pruning: Head pruning is another iterative pruning technique [35], where the unit of a block is the set of key, query, and value weights for computing self-attention. Head pruning becomes an option when heads in Transformers are found not to be useful for particular tasks [49], [50]. Besides, pruning heads can lead to significant reduction in computation, considering the $O(T^2)$ computation for a sequence of length T to compute the self-attention.

Similar to weight pruning, we can specify a threshold on the L_1 norm of the weights, and prune heads based on the desired amount to be pruned. Another option [50] is to use a small data set $\{(x_1, y_1), \dots, (x_n, y_n)\}$ to compute a score

$$\sum_{i=1}^{n} \left\| V_k(x_i)^\top \frac{\partial \mathcal{L}}{\partial V_k}(V_k(x_i), y_i) \right\|_1 \tag{1}$$

for each head k, where $\mathcal{L}$ is the loss function, V_k is the attention multiplied by the value matrix, and $\frac{\partial \mathcal{L}}{\partial V_k}$ is its gradient. This quantity considers the L_1 of both the attention map and its gradient. This L_1 of the gradient is small when the attention map has little effect on the loss, suggesting the head is a good candidate to prune. Contrary to weight pruning, the computation of self-attention can be avoided entirely when pruned, so the approach can lead to speed-up without sparse operations or dedicated hardware.

3) Low-Rank Approximation: Low-rank approximation assumes that the weight matrix is low rank [36], and can be approximated by factorization, such as singular value decomposition [51]. Transformers have two feed-forward layers after self-attentions, and the dimension after the first feed-forward layer is typically large (e.g., 3072). Most of the computation is due to the large dimension of this hidden layer. Despite having the nonlinearity between the two feed-forward layers, we find that the two weight matrices are both low rank. As a result, instead of doing matrix factorization, we prune the 3072-dimensional hidden layers. This amounts to pruning the columns of the first feed-forward layer and the corresponding rows of the second feed-forward layer based on the sum of their weight magnitudes.

4) Knowledge Distillation: Knowledge distillation trains a neural network (a student) to match parts of another neural network (a teacher) [52]. Matching the output of a teacher is the most common, while matching the hidden layers are also a viable option [53], [54]. In terms of the losses, KL divergence is commonly used to match probability outputs, while L_2 is often used to match the hidden vectors [55]. In this report, since MelHuBERT uses cross entropy for training, we simply use KL divergence as the distillation loss. There are several positive results with knowledge distillation, compressing self-supervised models [44]–[46].

D. Experiments

Pre-training and the subsequent compression training are done on the 360-hour subset of LibriSpeech. For the downstream tasks, we conduct phone recognition on 100-hour subset of LibriSpeech, and speaker identification on VoxCeleb 1. All experiments follow default SUPERB settings [1], where the pre-trained models are frozen after pre-training and not fine-tuned[3]. The downstream tasks are allowed to use the weighted sum of the 12 layers produced by the Transformer, and the weights are learned together with the downstream models.

When measuring the rate of compression, we define density as the pruned amount over the total amount that can be pruned. For example, the density in head pruning is the amount of heads pruned over the total number of heads.

1) MelHuBERT: To train MelHuBERT-10ms and MelHuBERT-20ms, we first run k-means with 512 clusters on log Mel features extracted from the 360-hour subset of LibriSpeech. We then train a 12-layer Transformer to predict the cluster labels of each frame for MelHuBERT-10ms or every other frame for MelHuBERT-20ms. We choose a masking strategy that masks roughly the same amount of content, 7% of masking probability with a mask of 10 frames for MelHuBERT-10ms and 14% with a mask of 5 frames for MelHuBERT-20ms. To compare, HuBERT uses 8% of masking probability with a mask of 10 frames. Both MelHuBERT-10ms and MelHuBERT-20ms are trained on single RTX 3090 GPUs with Adam for 200 epochs, with an effective batch size of 32 and a learning rate of 10^{-4}. It requires about 220 hours to train MelHuBERT-10ms, and about 150 hours for MelHuBERT-20ms. Dropout of 10% is applied after the multiplication of matrices, such as query, key, and value matrices, and after FC1 and FC2.

Table I shows the downstream performance of MelHuBERT-10ms and MelHuBERT-20ms. We include a 6-layer LSTM trained with APC, with a time shift of 3 frames. We also include HuBERT for completeness, but it is not directly comparable due to an additional training stage and additional training data. MelHuBERT-20ms not only performs better than its 10 ms counterpart, but, as we will see soon, runs faster.

2) Weight Pruning: For weight pruning, we iteratively prune individual weights based on their L_1 for all weights and biases in linear layers of Transformers. We keep track the exponential moving average of the loss with a decay of 0.9998, and if the loss is within 0.001 compared to 15000 steps before, pruning is triggered. We use a pruning schedule that is aggressive when the network is dense, and mild when the network is sparse. In particular, we pruning 20% until 80% density, 10% until 50% density, 5% until 35% density, 2.5% until 30% density, 1% until 10% density, and 0.5% until 5% density. We use a batch size of 4 and a learning rate of 10^{-5}.

Results of weight pruning are shown in Figure 1 (a). We can maintain the pre-training loss until 55% density. For phone recognition, we can maintain the PER until 45% density. For speaker identification, the performance is in fact better until 30% density. When we reach 10% density, the performance only degrades by 1.7% and 3.7% absolute for phone recognition and speaker identification, respectively.

3) Head Pruning: For head pruning, recall that we have two approaches, one based on the L_1 of head weights, and the other based on the L_1 of gradients For the weight-based approach, we prune a fixed amount of heads for each layer, because higher layers tend to have larger L_1. If we compare heads across layers, an entire layer would be pruned before we prune others. For the gradient-based approach, we use a $1/4$ of the training data to compute the scores of each head. As opposed to pruning a fixed number for each layer, we normalize the scores of heads within each layer, and prune the heads by comparing them altogether. As for all iterative pruning approaches, we train the model for a fixed 25,000 steps after pruning, with learning rate 10^{-5} and a batch size of 4.

Results of head pruning is shown in Figure 2. We find that the gradient-based approach is better than the weight-based approach. Figure 1 (b) shows the gradient-based head pruning results comparing with other pruning techniques. We find that speaker identification is more susceptible to head pruning compared to phone recognition. In fact, head pruning improves phone recognition until density 50%. Overall, with a density of 25%, the performance only degrades by 0.9% and 3.4% absolute for phone recognition and speaker identification, respectively.

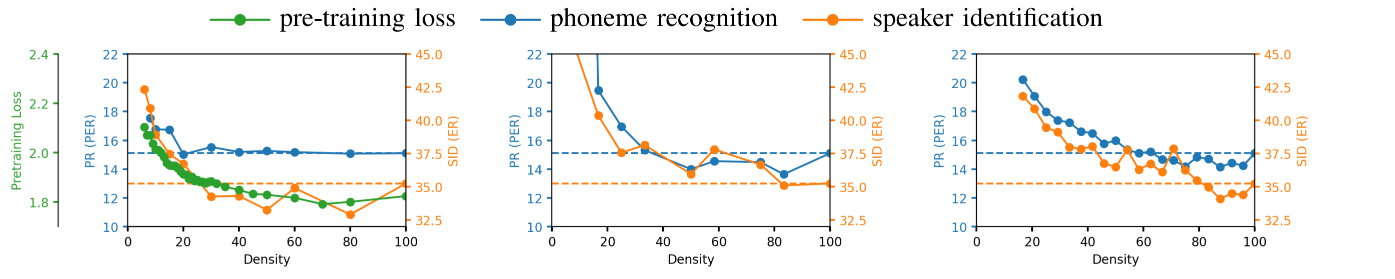
4) Low-Rank Approximation: For low-rank approximation, recall that we target the 3072-dimension output of FC1 layer. Reducing the dimension amounts to pruning the rows of FC1 and the columns of FC2 based on the sum of their L_1. We prune 128 dimensions every 25,000 steps. We use a learning rate of 10^{-5} and a batch size of 4. Results are shown in Figure 1 (c). We see trends similar to other pruning methods, and can obtain an absolute 1.5% and 2.6% drop in phone recognition and speaker identification until density 40%.

5) Knowledge Distillation: For knowledge distillation, we explore a 2-layer and a 6-layer student network. Following [44], we study several design decisions, including masking strategy, initializing with MelHuBERT layers, and the temperature when training with KL divergence. Our baseline in this case is simply taking the first 2 or the first 6 layers from a pre-trained MelHuBERT. Overall, our results are negative. The distillation loss continues to drop as we train, but representation produced by the student does not perform better on the downstream tasks. This is consistent with the finding of others [45] in that the representation fails to be useful for the downstream tasks when the network is overly shallow. As shown in Figure 3, taking the first few layers in MelHuBERT performs better than knowledge distillation. Taking the first few layers in MelHuBERT does not even require any additional training compared to knowledge distillation.

E. Discussion

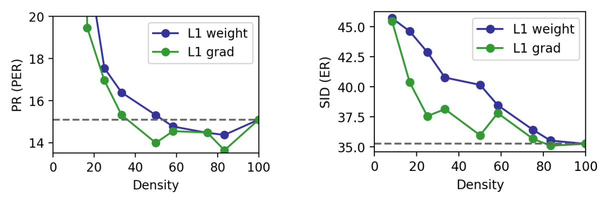
Results in Figure 1 are based on densities with respect to each individual techniques. The downstream performance can

[3]https://github.com/s3prl/s3prl



(a) weight pruning (b) head pruning (c) low-rank approximation

Fig. 1: Downstream performance of compressed MelHuBERT-10ms on phoneme recognition and speaker identification for weight pruning, head pruning, and low-rank approximation. The dashed lines are the performance of the unpruned MelHuBERT-10ms.



(a) phoneme recognition (b) speaker identification

Fig. 2: Phoneme recognition and speaker identification performance of weight-based and gradient-based head pruning on MelHuBERT-10ms. The dashed line is the performance of MelHuBERT-10ms.

TABLE I: Downstream performance of MelHuBERT-10ms and MelHuBERT-20sm compared to APC and HuBERT.

	PR	SID
APC 6-layer LSTM	32.0	44.7
MelHuBERT-10ms	15.1	35.2
(first 6 layers)	24.1	34.1
(first 2 layers)	84.9	41.1
MelHuBERT-20ms	13.0	33.7
(first 6 layers)	19.0	33.5
(first 2 layers)	46.3	40.3
HuBERT	5.4	18.5

be compared, but the benefits due to compression are difficult to compare and particularly difficult to be fair to all techniques. We decide to report different aspects to evaluate the benefits of compression, including reduction in runtime on GPUs, reduction in MACs per one second speech, and reduction in parameters. Reduction in runtime on GPUs immediately transfers to speed-up when GPUs are available. Reduction in MACs shows the (theoretical) speed-up when large amounts of parallelization are not available. Reduction in parameters brings benefit to generalization, as we have shown in Figure 1. A small amount of pruning almost always improves the downstream performance.

The analyses are shown in Figure 3 for MelHuBERT-10ms and MelHuBERT-20ms, respectively. We find that weight pruning brings significant reduction in parameters and MACs per one second speech, without affecting the performance of phoneme recognition much. However, weight pruning introduces sparse matrices and does not enjoy real-time speed-up on GPUs. Significant speed-up with weight pruning is possible when there is no option for parallelization, for example, on devices where there is only a single processor.

Head pruning is effective in maintaining performance while improving all metrics. It is less effective in reducing parameters, because the parameters of Transformers are concentrated in the FC layers. However, computation of self-attention is expensive, and we find a significant reduction in MACs per one second speech and runtime on GPUs.

Low-rank approximation is effective in reducing the number of parameters, as the parameters are concentrated at the FC layers in Transformers. The improvement in runtime is less pronounced compared to head pruning.

Finally, we use the first 2 and first 6 layers of MelHuBERT as baselines (shown in Table I). We find that removing the layers provides a strong trade-off across all metrics. Phonetic information is sufficiently accessible within the first 6 layers, while the speaker information is already accessible in the first 2 layers.

III. Sequence Compression

A. *Introduction - Sequence Compression*

In Section II, several model compression techniques have been explored, including weight pruning, head pruning, low-rank approximation, and knowledge distillation. However, in speech processing, the number of frames, along the time axis, is typically the dominating factor at runtime. For Transformers in particular, though quadratic memory consumption of self-attention is possible to avoid [57], most implementations still require quadratic memory and quadratic runtime. Sub-quadratic attention mechanisms are actively being developed [58]–[60], but the overhead typically makes these attention mechanisms less useful in practice [61]. Instead of reducing the runtime and memory complexity of self-attention, in this section, we study subsampling techniques to reduce the sequence length directly.

Subsampling (and sometimes concatenating) contiguous frames is a common technique for speeding up training and

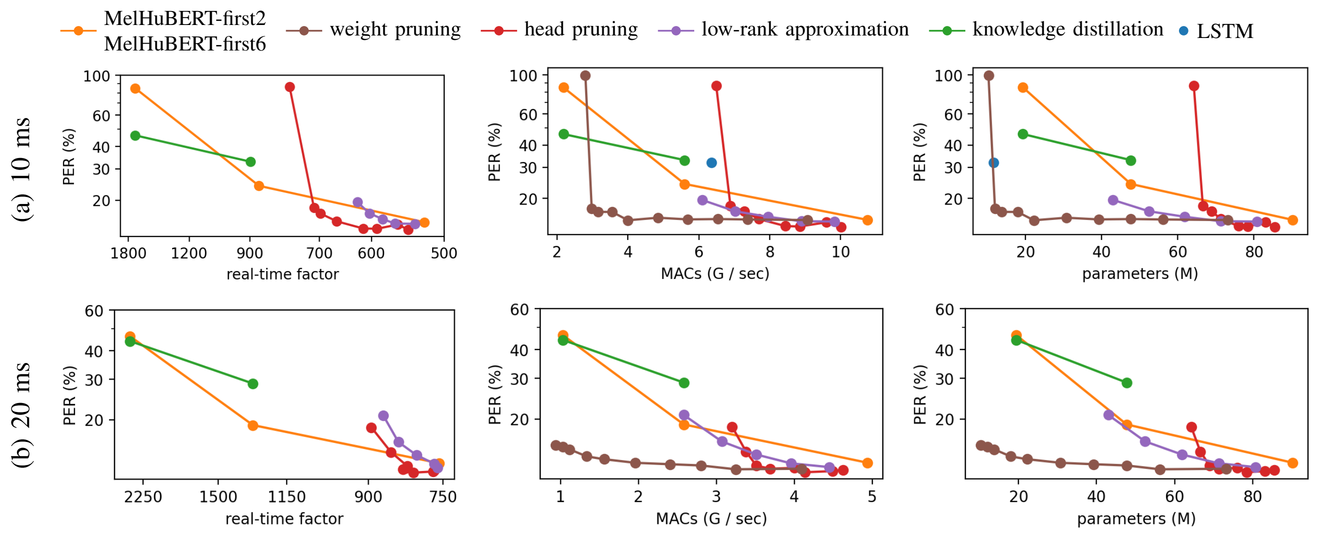


Fig. 3: Real-time factor, MACs per one second speech, and the number of parameters for various compression techniques on MelHuBERT-10ms (top) and MelHuBERT-20ms (bottom). Real-time factor of weight pruning is not shown, as implementing sparse operations is beyond the scope of this report. A 6-layer unidirectional LSTM trained with APC [56] (with real-time factor 99.8) is included for comparison. MelHuBERT-first2 and MelHuBERT-first6 are the first 2 and first 6 layers of MelHuBERT.

inference of automatic speech recognizers [62]–[64]. This approach can be as simple as concatenating every two frames [64] or dropping every even frames [65], [66]. Subsampling can also be achieved with pooling or convolution with a stride larger than 1 [67]–[70]. In this report, we term this family of approaches **fixed-length subsampling**. Fixed-length subsampling has been explored in self-supervised learning [71]–[73]. Both Wu et al. [71] and Vyas et al. [72] use fixed-length subsampling, paired with upsampling, during the optimization of self-supervised losses. Lee et al. [73] is the most similar to our work in that it also uses fixed-length subsampling to reduce the time resolution paired with upsampling in a knowledge distillation setting. The subsampled representation is then upsampled when evaluated on downstream tasks. Though this approach achieves the desired computational speed-up, how subsampling affects the learned representation largely remains unclear and is the central focus of this report. Another difference is that we directly use the subsampled representation in downstream, and explored a framework that works better without upsampling involved.

Since phones in speech come in varying duration, it is widely accepted that self-supervised models enforcing piecewise constant constraints would learn better representations [74]–[76]. Piecewise constant constraints can have various forms, but they mostly involve finding boundaries of segments and pooling frame representations into segment representations. We term this approach **variable-length subsampling**. In this work, we study both the computational speed-up and the impact on learned representations with variable-length subsampling. Our approach involves adding a layer of Continuous Integrate-and-Fire (CIF) [77] to learn subsampled representation while optimizing loss functions.

To push the boundary of small, yet competitive self-supervised models, we build our work on top of DistilHuBERT [44], a model consisting of two Transformer layers, trained with knowledge distillation from HuBERT [78]. We evaluate our approach on phone recognition, automatic speech recognition (ASR) with word pieces of different sizes, keyword spotting, intent classification, speaker identification, and emotion recognition. We find that phone recognition and ASR are the tasks most impacted by the subsampled representation. Matching the frame rate of downstream tasks gives the best performance. Variable-length subsampling has particularly strong performance for the settings with a low frame rate. For analysis, we also show that given the phonetic segmentation, variable-length sampling can be as good as, if not better than, the DistilHuBERT baseline while having a frame rate as low as 10 Hz.

B. *Compressing Sequences with Subsampling*

The model we study in this report is based on DistilHuBERT. For an input waveform s, there is a convolutional neural network (CNN) that converts the waveform s into a sequence of feature vectors $x_1, \dots, x_T$, or simply $x_{1:T} = \text{CNN}(s)$. we obtain a sequence of hidden vectors $u_1, \dots, u_T$ from the teacher model f_H (in this case HuBERT). Similarly, $v_{1:T} = f_\text{D}(x_{1:T})$, where f_D is the student model, DistilHuBERT. The student model is trained to minimize the distance of the representations $L(u_{1:T}, v_{1:T}; W) = \sum_{t=1}^{T} d(u_t, W v_t)$, where W is a trainable projection and the distance d can be cosine, dot product, Euclidean distance, or a combination of them. In DistilHuBERT, the distillation is applied to multiple layers of HuBERT via multitask learning. In particular, we minimize the sum of several loss terms $L(u_{1:T}^{(\ell)}, v_{1:T}; W^{(\ell)})$, where $u_{1:T}^{(\ell)}$ is the ℓ-th hidden layer of HuBERT and each loss has a learnable projection $W^{(\ell)}$ (also called a prediction head).

To subsample frames in time while maintaining the use of the loss from DistilHuBERT, we have two options, one with subsampling followed by upsampling, and another by subsampling the targets. Specifically, the first option minimizes

$$L\Big(u_{1:T}, \text{upsample}(f_{\text{D}}(\text{subsample}(x_{1:T})))\Big) \tag{2}$$

by having upsampling after the output of the student, while the second option optimizes

$$L\Big(\text{subsample}(u_{1:T}), f_{\text{D}}(\text{subsample}(x_{1:T}))\Big) \tag{3}$$

by subsampling the output of the teacher. Figure 4 shows the two options. For the rest of the sections, we will discuss the choices of subsampling and upsampling functions.

1) Fixed-Length Subsampling: For fixed-length subsampling, we adopt two commonly used approaches, convolution subsampling and average pooling with a stride larger than 1. We can control the frame rates by changing the strides. For convolution subsampling, the kernel size is set to the same as the stride size.

2) Variable-Length Subsampling: For variable-length subsampling, we follow a two-step approach. A segmentation (a sequence of boundaries) is first proposed; vectors within each segment are pooled (for example with weighted averaging) and passed to the subsequent layers. We use Continuous Integrate-and-Fire (CIF) [77] to produce segmentation proposals. At a high level, for an input sequence of length T, the CIF module takes input from the previous layer and produces a sequence $\alpha_1, \alpha_2, ..., \alpha_T$ of nonnegative numbers (using a combination of convolution, feedforward layer, and sigmoid function). Whenever $\sum_{i=1}^{t} \alpha_i$, i.e., the accumulation up to time t crosses an integer boundary, a segment boundary at time t is proposed (or fired). In other words, the sequence $\alpha_{1:T}$ controls both where and how many boundaries should be present. In fact, fixed-length subsampling can be seen as a special case with $\alpha_t = 1/F$ for all $t = 1, \dots, T$ where F can be set based on the desired subsampling rate.

The CIF module can be trained end to end without any supervision. In practice, however, the segmentation seldom corresponds well with any actual boundaries in speech. Below we discuss several options of boundary guidance to help CIF learn more meaningful boundaries.

Cardinality Guidance The cardinality guidance encourages the CIF module to produce the desired number of segments at training time; hence the name. Specifically, we add the term

$$L_{\text{card}}(\alpha) = \left(\frac{\sum_{i=1}^{T} \alpha_i - K}{T}\right)^2. \tag{4}$$

to our loss function, where K is the desired number of segments. The normalization T (absent in the original CIF formulation) is added to make sure the term is comparable across utterances of different lengths. Though the guidance is used at training time, at test time, we use the α's produced by the CIF module as is.

Segmentation Guidance In some cases where we have access to phonetic boundaries, such as through forced alignments or other unsupervised approaches, the segmentation can be used as a source of supervision for the CIF module.

Suppose a segmentation of K segments is provided as a sequence of boundaries in time indices $t_1, \dots, t_K$. To encourage the boundaries to be placed in accordance with the provided target, we introduce a segment-based loss

$$L_{\text{seg}}(\alpha) = \sum_{k=1}^{K} \left| \sum_{j=1}^{t_k} \alpha_j - k \right|. \tag{5}$$

The loss only focuses on where the boundaries are proposed. We also introduce a more stringent constraint, constructing a target $\alpha_i^{\text{sup}} = \frac{1}{t_{k+1} - t_k}$ for $t_k \le i < t_{k+1}$. We optimize

$$L_{\text{frame}}(\alpha, \alpha^{\text{sup}}) = \|\alpha - \alpha^{\text{sup}}\|_1 \tag{6}$$

to make sure the sequence α produced by the CIF module is close to the target α^{sup} at every frame; hence a frame-based loss.

Each of these losses can be added to our loss function with an interpolation factor. The segmentations for supervision are only used at training time, while at test time, we use the α's produced by the CIF module as is.

C. Experimental Setting

Our experiment is based on the original DistilHuBERT implementation with S3PRL [1] and fairseq [79]. The pre-training setting and hyperparameters, including learning rate schedule, are the same as the original DistilHuBERT implementation except that we introduce the subsampling and upsampling modules. We have three prediction heads, targeting the representations of the 4th, 8th, and 12th layers of a frozen HuBERT model. We use the 960-hour LibriSpeech dataset [80] for pre-training, and all models are trained for 200,000 updates with a batch size of 24.

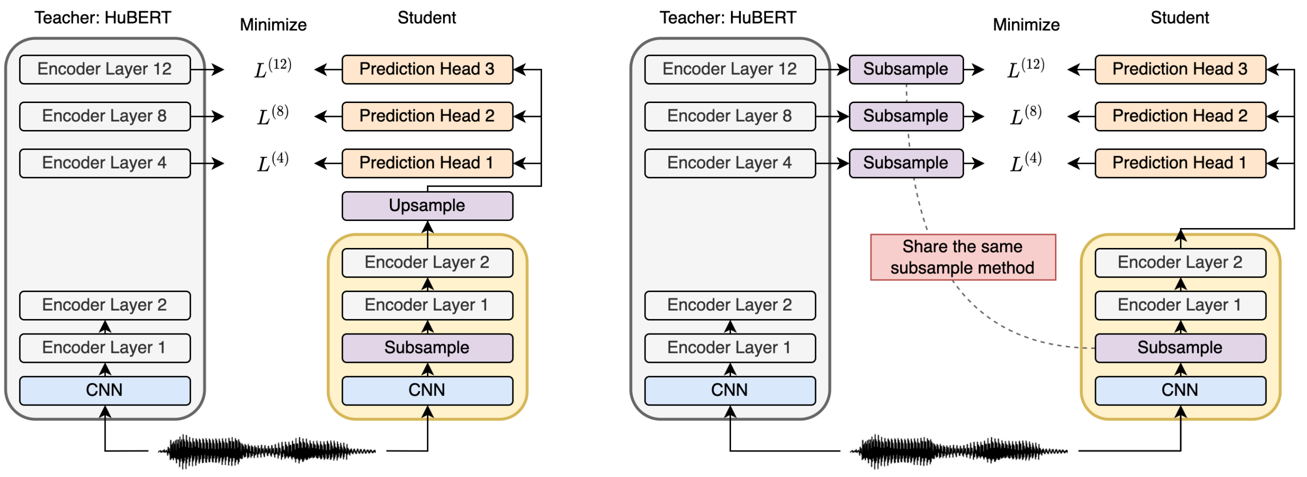
The CIF module for variable-length subsampling consists of a single one-dimensional 512-channel convolution with a stride of 1 and kernel width of 5, followed by a feedforward layer of 512 dimensions and an output dimension of 1.

The experiment is evaluated on a subset of the SUPERB benchmark[4], including phone recognition (PR), automatic speech recognition (ASR), keyword spotting (KS), intent classification (IC), speaker identification (SID), and emotion recognition (ER). We further evaluate on an additional task, automatic speech recognition with word pieces (ASR-5k). The ASR-5k model is trained with 5000 sentencepiece [81] targets (trained with byte-pair-encoding [82]). Both ASR and ASR-5k are evaluated without a language model (LM). To compare the performance with the DistilHuBERT paper, only the representation of the last layer (without the prediction heads) is used for downstream evaluation.

D. Preliminary Analysis

Before we train our models with subsampling, we explore a range of frame rates for several tasks. We concatenate contiguous hidden vectors produced by the vanilla DistilHuBERT to achieve a target frame rate. This approach preserves the information sent to the downstream tasks while only changing

[4]https://superbbenchmark.org



(a) With upsample (b) Subsample targets

Fig. 4: Two subsampling options. (a) A subsampling module is placed after the student's CNN layer, and an upsampling module is placed after the student's encoder output. (b) Identical subsampling modules are placed after the student's CNN layer and after the teacher's encoder output.

the frame rate. We also explore averaging contiguous hidden vectors as an alternative. Since KS, IC, SID, and ER are utterance-level tasks, subsampling the hidden vectors does not affect the performance, so we only focus on PR, ASR, and ASR-5k.

Results are shown in Table II, where two subsampling methods are tested: concatenating (cat) and average pooling (avg). Phone recognition starts to fail for frame rates lower than 12.5 Hz. ASR with characters completely fails for frame rates lower than 25 Hz, while ASR with word pieces can sustain subsampling to a frame rate as low as 6.25 Hz.

Figure 5 shows the average frame rates of different units, such as phones, characters, and word pieces. When we increase the number of word pieces learned, the size of the word pieces increases. In particular, when the number of word pieces is large, many of the word pieces are actual words, and the average frame rate decreases accordingly. The results in Table II are consistent with the average frame rates in Figure 5, as these tasks rely on a CTC layer that can only produce labels as many as it has frames for.

We also find that averaging is on par with concatenation, and decide to use averaging as the pooling method for the rest of the experiments.

E. Experiments

Unlike the previous section, here, we study models pretrained along with subsampling. Models trained with subsampling could potentially learn to represent the input speech differently from the vanilla DistilHuBERT.

1) Fixed-Length Subsampling: In this section, we explore the two options for fixed-length subsampling: subsampling paired with upsampling shown in Figure 4 (a), and subsampling of both the teacher's and the student's output shown in Figure 4 (b). For subsampling and upsampling pairs, we have convolution paired with deconvolution and averaging paired with repeating (duplicating frames to the desired frame rate).

TABLE II: Results of subsampling the output of DistilHuBERT. FP and FR represent the frame period and frame rate, respectively, after subsampling. The metrics include phone error rate (PER) and word error rate (WER).

Model	**FP**	**FR**	**PR**	**ASR**	**ASR-5k**
	ms	Hz	PER ↓	WER ↓	WER ↓
DistilHuBERT	20	50	16.27	13.37	12.86
cat 2	40	25	15.17	17.44	13.01
cat 4	80	12.5	31.03	>100	13.77
cat 8	160	6.25	82.32	>100	17.83
avg 2	40	25	15.57	16.45	12.95
avg 4	80	12.5	30.12	>100	14.06
avg 8	160	6.25	79.57	>100	18.86

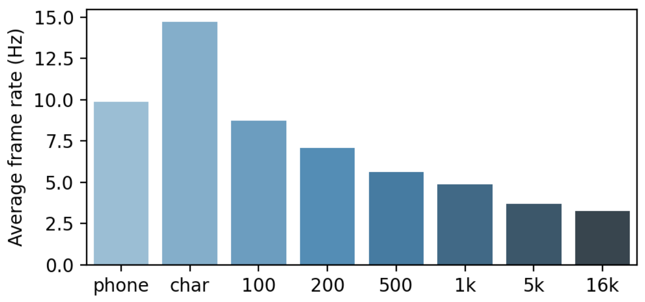


Fig. 5: Average frame rates for different units. Phones and characters are labeled as *phone* and *char*. The numbers 100, 200, 500, 1k, 5k, and 16k represent the number of word pieces learned with BPE. The average frame rates are computed on the LibriSpeech *dev-clean* subset.

In Figure 6 (Left), we show the pretraining losses for different subsampling and upsampling pairs, and in Table III (I), we have their respective downstream performance. In

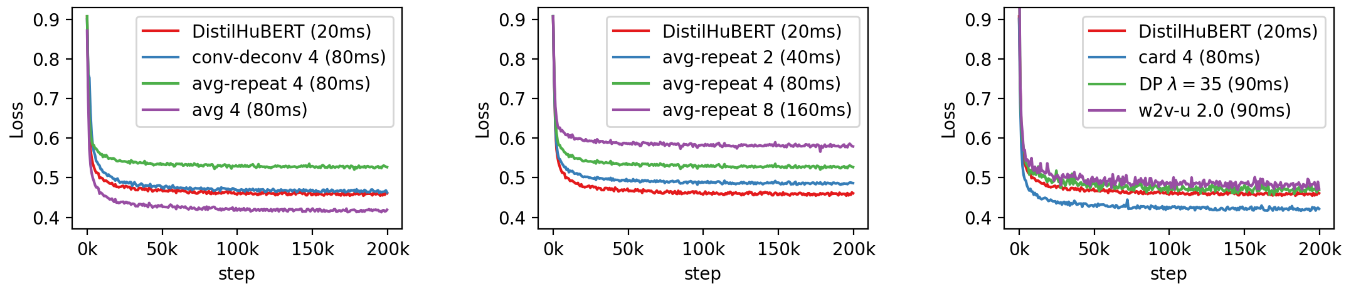


Fig. 6: Losses for pretraining with subsampling. *Left*: subsampling and upsampling pairs for pretraining compared to DistilHuBERT and subsampling both the teacher and the student. *Middle*: various frame rates of using averaging and repeating for subsampling and upsampling. *Right*: various types of boundary guidance.

Figure 6 (Left), we first note that convolution paired with deconvolution is able to recover the DistilHuBERT loss while averaging paired with repeating is worse. In Figure 6 (Middle), we also find that more aggressive sampling makes it more difficult to match the training loss of the vanilla DistilHuBERT. However, in Table III (I), the downstream performance for averaging paired with repeating is on par with convolution paired with deconvolution for the two frame rates we explored.

Due to this finding and the simplicity, we use averaging to conduct the experiments for subsampling both the teacher and the student. The training loss is shown in Figure 6 (Left) and the downstream performance is in Table III (II). We do find the training loss to be lower (perhaps due to having fewer frames), and the downstream performance is generally on par, if not better than, with upsampling. These results are also comparable to the ones in Table II, including the failed case for phone recognition at the frame rate of 6.25 Hz.

2) Variable-Length Subsampling: Based on the findings in the previous section, we decide to perform variable-length subsampling on both the teacher and the student. Recall that variable-length subsampling is achieved by producing a sequence α that decides where a boundary should be placed. Here, we choose a weighted sum of the vectors within a segment to produce a segment representation. In particular, suppose $\alpha_t, \dots, \alpha_{t'}$ are the α's in between two boundaries. The weights used for weighted averaging are $\epsilon, \alpha_t, \dots, \alpha_{t'} - \epsilon'$, where ϵ is the leftover weight that exceeds the integer from the previous boundary and ϵ' is the leftover weight that exceeds the integer boundary and gets carried over to the next boundary. In order to match the subsampling exactly for both the teacher and the student, we decide to reuse the α produced by the student on the teacher, as indicated in Figure 4 (b).

We first explore cardinality guidance, denote as *card*, where we set the number of segments to match the desired frame rate. For example, the number of segments K is set to $T/4$ if the desired frame rate is 12.5 Hz. We tune the weight of the added loss term L_{card} and find that a weight of 0.5 works the best. The downstream performance is shown in the first two rows of Table III (III). The results are generally worse than fixed-length subsampling. We suspect for the cardinality guidance is too weak for constraining the model, so we turn to the stronger segmentation guidance.

Below we explore two options, using smoothed HuBERT codes and unsupervised ASR, to obtain segmentation for guiding the CIF module.

Smoothed HuBERT Codes as Guidance The amount of repeated HuBERT codes tend to correlate well with the duration of phones [83], so we can make use of the codes produced by the teacher as a proxy to segmentation. Here we run k-means on the output of HuBERT (layer 6, following [83]) to obtain the codes. We then use the dynamic programming algorithm proposed by Kamper and van Niekerk [84] to obtain the segmentation. Their algorithm generally smooths the frequent changes of a HuBERT code sequence, while also providing a hyperparameter λ penalizing short segments. In other words, larger λ produces larger segments, leading to a lower frame rate. There is no clear correspondence between λ and frame rate (except that they are positively correlated), so we simply sweep λ to find the desired frame rates. Recall that there are a frame-based approach and a segment-based approach to incorporate segmentation guidance. We tune the weight of the two additional loss terms, and find that 0.005 and 0.25 work best for L_{seg} and L_{frame}, respectively. Despite the weight of L_{seg} being deceptively small, it is crucial to prevent CIF from producing degenerate solutions.

Results of this approach are listed in the second block of Table III (III). Note that we tune λ to find a frame rate closest to the desirable one. For readability, we round the frame rates to the desirable ones in Table III (III). This approach performs significantly better than others for the 12.5 Hz frame rate, while matching the result of others for the 25 Hz frame rate. It shows the great potential of variable-length subsampling and calls for better segmentation.

Unsupervised ASR as Guidance Another approach to obtain a segmentation of speech is through unsupervised ASR [28], [85]. Our pre-trained unsupervised ASR model is a simplified version of wav2vec-U 2.0 [85] without the auxiliary k-means cluster loss. The model is trained on audio from the 100-hour LibriSpeech and texts from the LibriSpeech language modeling text. More investigations about unsupervised ASR are in Section X in this report. Once the model is trained, it simply acts as a frame classifier, producing posterior probabilities of phones at a frame rate of 16.7 Hz (60 ms frame period). The segmentation can be obtained by finding the maximum of each frame as predictions and merging the

TABLE III: Downstream performance for various subsampling approaches. FP and FR denote the frame period and frame rate, respectively. The metrics include phone error rate (PER), word error rate (WER), and accuracy (Acc). For further results, please refer to the SUPERB website.

Model	**FP**	**FR**	**Params**	**MACs**	**MACs-C**	**PR**	**ASR-5k**	**KS**	**IC**	**SID**	**ER**
	ms	Hz	Millions	GMACs	GMACs	PER ↓	WER ↓	Acc ↑	Acc ↑	Acc ↑	Acc ↑
DistilHuBERT	20	50	23.49	758.9	207.1	16.27	12.86	95.98	94.99	73.54	63.02
(I) Fixed-Length - *with upsampling*											
conv-deconv 2	40	25	25.20	667.6	115.8	16.06	13.62	95.78	92.06	67.13	63.08
conv-deconv 4	80	12.5	26.90	610.3	58.5	32.04	16.15	95.75	90.67	62.65	61.62
avg-repeat 2	40	25	23.49	664.6	112.8	15.54	13.50	96.04	95.15	71.16	62.98
avg-repeat 4	80	6.25	23.49	607.3	55.5	30.69	16.14	95.78	93.33	68.56	61.13
(II) Fixed-Length - *subsampling targets*											
avg 2	40	25	23.49	664.6	112.8	15.43	13.31	95.59	94.31	72.19	63.17
avg 4	80	12.5	23.49	607.3	55.5	30.50	15.55	95.88	93.12	69.33	62.41
avg 8	160	6.25	23.49	579.5	27.7	79.98	24.90	95.26	90.85	68.46	60.79
(III) Variable-Length - *subsampling targets*											
card 2	40	25	24.81	675.7	123.9	17.23	14.24	94.97	88.74	70.44	62.04
card 4	80	12.5	24.81	620.7	68.9	38.51	16.88	95.13	90.51	70.27	61.31
DP $\lambda = 0$	40	25	24.81	680.6	128.8	15.53	14.19	95.65	94.54	71.42	62.67
DP $\lambda = 25$	80	12.5	24.81	623.3	71.5	21.94	15.33	95.72	94.62	71.05	63.19
DP $\lambda = 35$	90	11.1	24.81	614.6	62.8	31.73	16.66	95.72	94.41	69.46	62.38
DP $\lambda = 75$	160	6.25	24.81	593.4	41.6	78.41	27.04	95.39	89.16	66.83	61.64
w2v-u 2.0	90	11.1	24.81	616.6	64.8	23.37	15.71	95.62	94.65	71.70	61.74
(IV) Variable-Length - *subsampling targets*											
MFA	100	10	-	-	-	12.33	11.85	-	-	-	-

same contiguous predictions. Since voice activity detection is involved during training, we overwrite segments where the silence is detected. Once the boundaries are extracted, we add frame-based and segment-based losses with the same weights as in Section III-E2.

The downstream performance of using the segmentation of wav2vec-U 2.0 is in the last row of Table III (III). We also conduct a similar experiment with the smoothed HuBERT codes for comparison, matching the frame rate of the segmentation produced by the wav2vec-U 2.0 model. We find that this approach is particularly strong at phone recognition and ASR with word pieces, giving the best performance and frame rate tradeoff.

The training loss of various variable-length subsampling is shown in Figure 6 (Right). Though the downstream performance of using wav2vec-U 2.0 segmentation provides a better tradeoff, the loss is slightly higher than the approach of smoothing HuBERT codes.

3) Topline: Subsampling with Forced Alignments: Given the results from previous sections, an accurate segmentation could prove to be sufficient for an aggressive subsampling frame rate. As an analysis, we study forced alignments as a variable-length subsampling approach of its own. Forced alignments (of phones) are obtained from the Montreal Forced Aligner (MFA) [86]. Once the segmentation are obtained, we simply use average pooling to obtain a segment representation from the teacher and the student. We follow DistilHuBERT training without adding any other terms. For the downstream tasks, we also use forced alignments for subsampling. Note that the CIF module is not used in this setting, as segmentation is always provided.

The result is shown in Table XIII (IV) and serves as our topline result. We do not report downstream performance for all tasks as transcripts are not available for computing forced alignments on those datasets. This approach achieves an average frame rate of 10 Hz (i.e, the average frame rate of phones), and has the best phone recognition and ASR results. The result suggests that there is still room for better segmentations.

F. Discussion

To measure the impact of subsampling on runtime, we report average multiply-accumulate operations (MACs) on a subset of LibriSpeech *test-clean* consisting of utterances between 1 and 20 seconds. MACs are measured in inference mode without counting prediction heads and upsampling. MACs of various subsampling approaches are reported in Table III. The improvement in MACs is obscured by the MACs of the seven CNN layers, as the MACs of CNN dominate everything else. We therefore also report MACs without the CNN layers (denoted as MACs-C). The improvement is then clear and consistent with the reduction in frame rates.

IV. Generalization of SSL

A. Introduction

This section investigates the generalizability of SSL models. Domain shifts caused by mismatches between training data and testing data usually occur in real-world scenarios. A common factor that causes domain shifts is speech distortions. Here we focus on the setting that the training data of downstream tasks contain clean speech while the testing data has distortions. For many downstream speech processing tasks, since the cost of collecting labelled training data is expensive, the training dataset can not be diverse, and usually only has clean speech. However, there may be background noises during the testing phase in real-world applications, making SSL models vulnerable in performance as studied in [11].

We further find that distilled SSL models suffer from performance degradation even more than their teachers in distorted environments. To overcome the problem that distilled models are especially vulnerable to distorted speech, we propose to apply Cross-Distortion Mapping (CDM) during knowledge distillation to improve the generalizability of DistilHuBERT. The process of Cross-Distortion Mapping refers to a teacher-student learning framework with the teacher and student model having different distorted inputs. The results show that CDM improves the testing performance on downstream speech processing tasks under the setting with speech distortions, even when the distortion types are unseen during training. To further improve the robustness of the teacher model, we performed domain-adaptive pre-training on the teacher model by utilizing distorted pre-training data so that the student model would be able to have a more robust target to learn with. We also applied Domain Adversarial Training [87] (DAT) in the hope of generating more domain-invariant speech representations, and found out that DAT benefits the generalization of models in some cases[5].

B. Related work

There are several studies enhancing the robustness of SSL models. It has been found that pre-training some more steps with unlabeled target domain data [10] on Wav2vec2.0 mitigates the problem of domain shifts. Therefore an intuitive method to enhance the robustness of SSL models is to augment pre-trained data by adding distortions. Here we found that the proposed CDM method further improved the distilled SSL model learned from the domain-adapted teacher model pre-trained with distorted speech.

Besides domain-adaptive pre-training, DAT is applied to improve the generalizability of SSL models. Augmentation adversarial training [88] combines the concept of augmentation and DAT to generate representations invariant to augmentations. Some other studies also apply DAT to adapt models to different kinds of data and are also proved to benefit unseen domains, such as different accented speech [89], [90] or distorted speech [11], [91]. Based on our experimental results, DAT sometimes further improved the results when combined with our proposed CDM method.

Several studies have successfully improved the robustness of Wav2vec2.0 models. Having noisy waveforms as input, performing clean speech reconstruction with a reconstruction module along with pre-training [92] also improves noise robustness. Having both clean and noisy speech as input, a denoising approach [93] conducted by constructing clean quantized vectors serving as the target for the noisy representations successfully improved performance on noisy testing sets while preserving performance on the original clean testing set. The idea of [93] is similar to setup1 of CDM (will be elaborated in Section IV-C2b), where the student model learns to generate clean representations of the teacher given distorted speech inputs. However, we find that setup2 of CDM, where both student and teacher models are given distorted speech, is more effective in gaining robustness than setup1.

CDM is also similar to another series of work, Bootstrap Your Own Latent (BYOL) [94]. BYOL performs augmentation to data and trains an online-target framework from scratch. The online-target framework consists of an online network and target network with similar model architectures. Both networks receive the same input but with different augmentations and minimize the distance of output representations between the two. As BYOL, CDM includes the augmentation method of BYOL and the concept of denoising by viewing different distorted speech as different domains and minimizing the distance of speech belonging to different domains. Different from BYOL, we utilized a teacher-student model compression framework where a small student model is trained to mimic a large teacher model.

C. Methods

1) Knowledge distillation: Throughout this work, DistilHuBERT [44] is adopted to meet our needs for reducing model size. DistilHuBERT is trained with a teacher-student learning framework with knowledge distillation. As shown in Figure 7, the student network consists of a subnet $\mathbf{F}$ followed by some prediction heads $\mathbf{p}$, where $\mathbf{F}$ is constructed by reducing the number of transformer encoder layers of the HuBERT teacher model. Given an input speech utterance $\mathbf{x} \in \mathbb{R}^T$, where T is the number of timesteps of $\mathbf{x}$, a predicted hidden representation sequence $\hat{\mathbf{h}}^i$ is output by the student model as shown in Eq. (7).

$$\begin{aligned} \mathbf{z} &= \mathbf{F}(\mathbf{x}) \\ \hat{\mathbf{h}}^i &= \mathbf{p}_i(\mathbf{z}) \end{aligned} \tag{7}$$

In Eq. (7), $\mathbf{z}$ is the last hidden representation of the transformer layers in the student model, and serves as the input of the prediction heads $\mathbf{p}$. Prediction head $\mathbf{p}_i$ predicts the i^{th} hidden layer representation $\mathbf{h}^i$ of the teacher model. The overall objective of DistilHuBERT consists of a L_1 loss term and a cosine similarity loss term as shown in Eq. (8),

$$\begin{aligned} &\mathcal{L}_{distil} = \mathcal{L}_{L_1} + \mathcal{L}_{cos} \\ &= \sum_{i\in\{4,8,12\}} \sum_{t=1}^{T} \left[\frac{1}{D} \left\| \mathbf{h}_t^i - \hat{\mathbf{h}}_t^i \right\|_1 - \gamma \log \sigma(\cos(\mathbf{h}_t^i, \hat{\mathbf{h}}_t^i)) \right] \end{aligned} \tag{8}$$

where σ is the sigmoid function and $\cos(\cdot,\cdot)$ is the cosine similarity function, D is the feature dimension of the representations, and γ is a constant to scale the value of the cosine similarity loss. Though both loss terms may have similar goals,

[5]Code will be released at https://github.com/nobel861017/distort-robust-distilSSL.

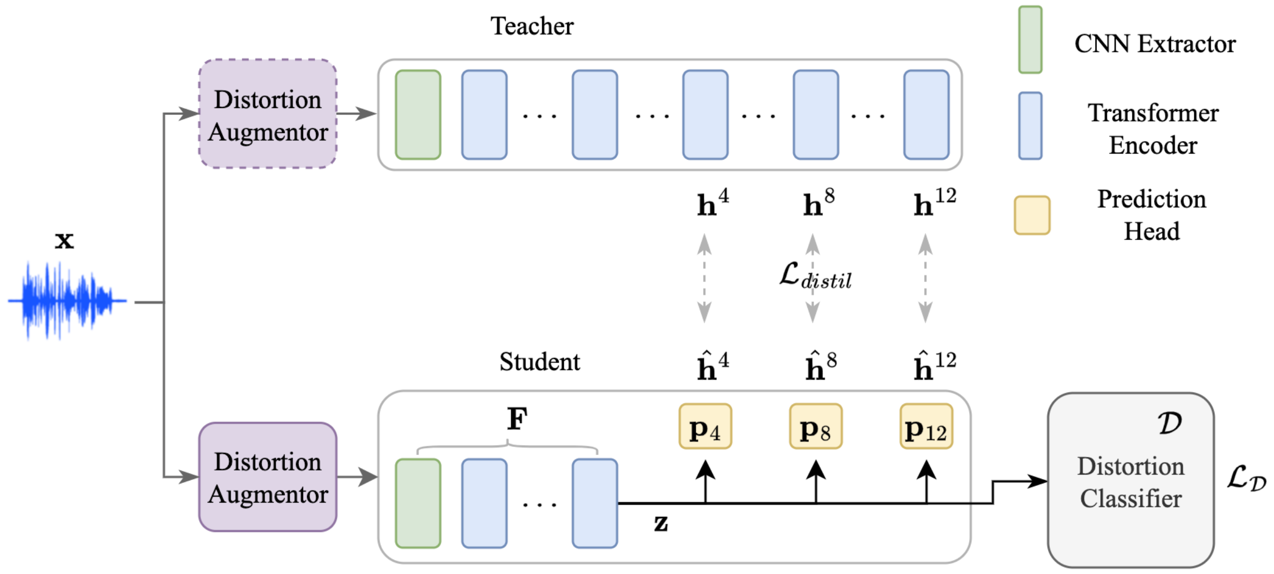


Fig. 7: Illustration of DistilHuBERT with a distortion classifier. The distortion augmentor follows the procedure mentioned in Section IV-D1. For setup1, there is no distortion augmentor for the teacher model. For setup2, both the teacher and student model have a distortion augmentor in front.

the original paper [44] reports that considering both terms results in better performance.

2) *Model generalization:*

a) *SSL model domain-adaptive pre-training:* Pre-training SSL models with target domain data is an intuitive way to adapt SSL models to another domain [10], [11], [92], [95]. This method is sometimes referred to as domain-adaptive pre-training or continual training, depending on its training setting and data configuration[6]. For some experiments in our work, we trained the pre-trained teacher model with distorted speech data for additional steps to enhance robustness. The distorted data is generated by adding distortions to the original pre-training data of the corresponding SSL model. We hope that in this way, the teacher model will be able to output more robust representations for the student model to learn with. To avoid any possible misunderstanding, we refer to this method as domain-adaptive pre-training in the further subsections. Note that this method differs from Domain Adversarial Training mentioned in subsection IV-C2c, though both are domain-adaptive methods.

b) *Cross-Distortion Mapping (CDM):* Cross-Distortion Mapping refers to the augmentation procedure for the teacher-student framework. The teacher and student model receive the same speech utterance augmented with different distortions as shown in Figure 7. We investigate two setups for augmentation.

- setup1: The first setup is to input speech utterances without distortions to the teacher model, while the student model takes distorted speech as input. This setup has the denoising concept since the distorted speech representations have clean speech representations as targets, and there are previous works [93], [96], [97] showing that this setup is beneficial for training models robust to noise.
- setup2: The second setup is to apply different distortions to speech similar to [94], [98], [99], resulting in the teacher and student model observing same speech utterances but with different distortions.

[6]Continual training is performed under a life-long learning scheme, and the data of previous tasks are usually unavailable, which is not the case in our work.

c) *Domain Adversarial Training (DAT):* DAT is performed by utilizing a distortion classifier $\mathcal{D}$ that takes the last hidden states $\mathbf{z}$ of the student model as input. The distortion classifier aims at identifying the distortion types of the distorted speech input by optimizing a multi-label cross-entropy loss (each speech utterance may be distorted with multiple distortions).

During the training process, the distortion classifier and the student model are trained in turn. First, the parameters of the distortion classifier θ_d is updated with gradient descent shown below,

$$\theta_d \leftarrow \theta_d - \alpha \frac{\partial \mathcal{L}_D}{\partial \theta_d} \tag{9}$$

where $\theta_{\mathcal{D}}$ is the parameters of the distortion classifier, $\mathcal{L}_{\mathcal{D}}$ is the cross-entropy loss, and α is the learning rate.

After training the distortion classifier, the parameters θ_s of the student model are updated through the process in the following,

$$\theta_s \leftarrow \theta_s - \beta \frac{\partial (\mathcal{L}_{distil} - \lambda \mathcal{L}_D)}{\partial \theta_s} \tag{10}$$

where θ_s is the parameters of the student model, β is the learning rate, and λ is a constant controlling the scale of $\mathcal{L}_{\mathcal{D}}$.

D. *Experimental setup*

1) *Data preparation:* The corpus used for knowledge distillation is LibriSpeech [80] 960-hour, which is same as the pre-training data of HuBERT-base in [78]. In our distorted

				KS (Acc% ↑)				**IC** (Acc% ↑)				**ER** (Acc% ↑)			
		da.	para.	clean	2-dist	fsd	dns	clean	2-dist	fsd	dns	clean	2-dist	fsd	dns
(T1)	HuBERT [78]	X	95M	96.30	89.81	90.94	77.60	98.34	89.09	91.93	74.11	64.92	56.72	60.05	52.08
(T1')	HuBERT	V	95M	96.53	94.77	94.00	82.83	98.37	96.20	96.78	85.00	65.88	62.82	63.89	56.70
(S1)	DistilHuBERT (Tr2) [44]	X	23M	95.98	87.57	88.70	75.07	94.99	70.29	72.50	48.30	63.13	55.09	57.05	49.76
(S1')	DistilHuBERT (Tr2)	V	23M	96.14	86.86	90.56	76.47	95.65	77.99	81.73	57.50	64.01	58.89	59.06	53.14
(S2)	DistilHuBERT (Tr2) setup1	X	23M	95.52	92.92	93.44	76.66	94.17	89.53	89.61	72.11	63.51	58.11	60.17	50.66
(S2')	DistilHuBERT (Tr2) setup1	V	23M	96.17	93.61	94.09	77.44	95.57	86.11	89.03	71.26	63.72	59.62	61.42	53.69
(S3)	DistilHuBERT (Tr2) setup2 (same)	X	23M	96.11	89.84	91.69	78.42	94.62	75.40	80.33	57.92	61.87	55.72	59.41	50.27
(S3')	DistilHuBERT (Tr2) setup2 (same)	V	23M	96.33	92.57	93.48	**80.04**	95.68	85.16	86.84	64.46	**64.25**	59.62	60.93	51.78
(S4)	DistilHuBERT (Tr2) setup2	X	23M	96.27	92.99	93.96	77.47	95.91	90.72	90.77	73.87	63.77	59.89	61.62	51.25
(S4')	DistilHuBERT (Tr2) setup2	V	23M	**96.53**	93.61	94.38	79.10	96.57	**92.25**	**92.67**	**78.41**	63.08	60.38	60.89	53.38
(S5)	DistilHuBERT (Tr2) setup1 + DAT	X	23M	95.94	93.80	93.83	79.36	96.02	90.35	91.09	74.61	63.41	59.34	60.58	53.29
(S5')	DistilHuBERT (Tr2) setup1 + DAT	V	23M	95.75	93.61	93.35	78.25	96.34	88.82	90.48	73.19	63.23	60.06	61.15	**53.71**
(S6)	DistilHuBERT (Tr2) setup2 + DAT	X	23M	96.17	93.77	93.90	78.45	96.49	91.35	92.09	75.51	63.44	59.36	**61.82**	51.01
(S6')	DistilHuBERT (Tr2) setup2 + DAT	V	23M	96.46	**94.03**	**94.55**	78.90	**96.75**	91.01	92.06	76.51	63.45	**61.15**	61.62	53.49
(S7)	DistilHuBERT (Tr1)	X	20M	94.90	86.34	87.47	71.44	92.35	60.43	64.25	38.65	62.45	52.65	57.19	49.23
(S7')	DistilHuBERT (Tr1) setup2	V	20M	96.46	92.79	93.44	75.33	94.96	84.66	86.29	64.36	62.93	58.56	59.53	50.43
(S8)	DistilHuBERT (Tr3)	X	34M	96.53	89.39	90.85	76.50	94.70	74.00	78.12	54.15	62.94	55.34	56.94	51.69
(S8')	DistilHuBERT (Tr3) setup2	V	34M	96.53	93.90	94.61	77.90	97.47	93.49	93.80	79.57	64.63	62.88	63.25	53.98

TABLE IV: Evaluation results for **KS**, **IC**, and **ER** in accuracy (Acc). By default, DistilHuBERT and has two transformer encoder layers (Tr2). Tr1 and Tr3 denote the number of transformer encoder layers (1 and 3) of DistilHuBERT, which are different from the default configuration (Tr2). The second column, da., specifies whether domain-adaptive pre-training is conducted to the teacher model. The third column, para., lists the number of parameters for each model. The terms “setup1” and “setup2” refer to the two setups of the CDM method mentioned in Section IV-C2b. The best performance on each test set throughout the twelve DistilHuBERT models in (S1)-(S6’) is marked in bold.

setting (denoted as 2-dist in Table IV and V), we consider clean speech and speech containing one or two distortions. Distorted speech is generated by applying either one of the additive distortions or one of the non-additive distortions, or both to speech.

Additive distortions are noises directly added to speech data at a specific speech-noise ratio (SNR) between 10 dB and 20 dB. We adopted additive noise from four widely-known noise datasets, Musan [100], WHAM! [101], FSD50k [102], and DNS[7] [103]. Apart from the recorded noise data of the aforementioned datasets, we also took advantage of Gaussian noise, a hand-crafted noise that follows the Gaussian distribution in the time domain. During testing, we evaluated models on four downstream speech processing tasks, **KS**, **IC**, **ER**, and **ASR**. Besides the original testing set configured by the SUPERB benchmark (denoted as clean in Table IV and V), we also tested the models under our distorted setting (2-dist). Furthermore, to evaluate the robustness of models to unseen distortions, the two noise datasets, FSD50k and DNS, are held out from the training phase among all the experiments and are only adopted during testing to create a domain mismatch scenario. Note that speech in the FSD50k-distorted testing set (denoted as fsd in Table IV and V) contains one background noise sampled from the FSD50k corpus, creating a single distortion setting. Speech in the DNS-distorted testing set (denoted as dns in Table IV and V) is constructed by adding one background noise and convolving a room impulse response [104] to speech.

For non-additive distortions, we chose some common sound effects, such as reverberation, pitch shift, and band rejection to apply to speech data. Adding non-additive distortions to speech does not require additional data and can be directly applied to waveforms. We followed the configurations and implementation details proposed in WavAugment[8] [105].

For the **ASR** task, we also report the performance on the test-other split of LibriSpeech and the real speech recordings of CHiME3 [106], no additional distortions are applied to these two testing sets.

2) Upstream models and training details: HuBERT-base is the teacher model used for knowledge distillation, and the pre-trained weights are initialized by the checkpoints released in Fairseq[9] [79]. The domain-adaptive pre-trained version of HuBERT-base follows the same procedure mentioned in [11], except for the generating process of distorted speech replaced by our procedure mentioned in subsection IV-D1.

For knowledge distillation, we train each model for 20k steps and adopt the checkpoint that yields the lowest distillation loss in (8) on the development set (dev-clean of LibriSpeech). Other hyperparameters such as learning rate, optimizers, and schedulers are the same as the original DistilHuBERT [44] training configuration.

For DAT, the distortion classifier is a mean pooling operation followed by a linear layer projecting the representations

[7]We follow the procedure in the original paper [103] to generate noisy data.

[8]https://github.com/facebookresearch/WavAugment

[9]https://github.com/facebookresearch/fairseq

				ASR (WER% ↓)											
		da.	para	clean	LM	other	LM	2-dist	LM	fsd	LM	dns	LM	CHiME3	LM
(T1)	HuBERT [78]	X	95M	6.42	4.79	15.67	12.14	11.28	8.59	9.36	7.20	22.72	18.87	24.74	20.28
(T1')	HuBERT	V	95M	6.75	4.91	16.01	12.44	8.31	6.17	7.78	5.86	16.31	12.85	21.98	17.51
(S1)	DistilHuBERT (Tr2) [44]	X	23M	13.37	9.21	35.65	27.78	30.65	24.32	24.27	18.37	54.46	47.52	46.62	37.68
(S1')	DistilHuBERT (Tr2)	V	23M	**13.14**	**9.08**	**33.97**	**26.63**	26.02	19.97	21.80	16.28	48.29	41.70	43.76	36.38
(S2)	DistilHuBERT (Tr2) setup1	X	23M	13.96	9.64	36.23	28.67	18.52	13.25	16.69	11.87	38.63	31.39	38.15	29.92
(S2')	DistilHuBERT (Tr2) setup1	V	23M	13.54	9.41	34.37	27.18	**17.58**	12.71	16.38	11.61	37.41	30.66	37.92	29.74
(S3)	DistilHuBERT (Tr2) setup2 (same)	X	23M	13.77	9.52	35.87	28.23	23.49	17.34	19.83	14.36	45.18	37.69	41.30	32.73
(S3')	DistilHuBERT (Tr2) setup2 (same)	V	23M	13.57	9.34	34.34	26.91	18.71	13.33	17.06	12.16	38.77	31.29	38.25	29.94
(S4)	DistilHuBERT (Tr2) setup2	X	23M	14.33	9.90	36.99	28.95	18.58	13.44	17.18	12.13	39.14	32.11	39.13	30.80
(S4')	DistilHuBERT (Tr2) setup2	V	23M	13.7	9.66	**33.97**	27.03	17.89	12.98	16.35	11.65	37.73	31.06	38.02	30.18
(S5)	DistilHuBERT (Tr2) setup1 + DAT	X	23M	13.64	9.40	34.53	27.27	17.86	12.66	16.45	11.72	37.56	30.53	37.63	30.17
(S5')	DistilHuBERT (Tr2) setup1 + DAT	V	23M	13.72	9.57	34.08	26.82	17.74	12.70	16.30	11.61	37.30	30.43	**37.21**	**29.31**
(S6)	DistilHuBERT (Tr2) setup2 + DAT	X	23M	14.46	10.02	37.37	29.41	18.96	13.69	17.52	12.30	39.03	31.77	39.26	31.01
(S6')	DistilHuBERT (Tr2) setup2 + DAT	V	23M	13.58	9.45	34.07	26.97	17.61	**12.63**	**16.16**	**11.60**	**37.11**	**30.33**	37.57	29.71
(S7)	DistilHuBERT (Tr1)	X	20M	14.70	10.03	39.73	31.73	35.97	29.33	28.12	21.41	62.91	57.22	52.48	43.44
(S7')	DistilHuBERT (Tr1) setup2	V	20M	15.20	10.41	38.37	30.61	20.37	14.64	18.69	13.40	44.26	37.07	42.80	34.33
(S8)	DistilHuBERT (Tr3)	X	34M	12.61	8.73	32.50	25.24	28.05	21.95	22.05	16.54	51.28	45.07	44.06	36.37
(S8')	DistilHuBERT (Tr3) setup2	V	34M	12.12	8.48	30.92	23.95	15.37	10.92	14.20	10.22	32.08	25.71	33.94	26.63

TABLE V: Evaluation results for **ASR** in word error rate (WER). Results of the test-clean set of LibriSpeech is abbreviated as clean, and the results of the test-other set of LibriSpeech is abbreviated as other. LM represents the results after language model rescoring. Notations are same as Table IV.

to a dimension equal to the number of distortion types. In our work, there are seven distortion types, including Musan, Gaussian, WHAM!, reverberation, pitch shift, band rejection, and clean. The distortion classifier is trained in a multi-label classification style. The λ value is set to $1\mathrm{e}{-2}$ for all of the DAT experiments.

3) Downstream models and training details: For the downstream speech processing models of the four tasks reported in the results, we follow the same model configurations of the SUPERB benchmark[10]. We adopt the last hidden states of the student model as the input of the downstream models. During downstream training, the batch sizes for training downstream tasks **KS**, **IC**, and **ASR** are set to 32, and 4 for **ER**. The learning rate for the optimizer is set to $1\mathrm{e}{-4}$ for **IC**, **ER**, and **ASR**, and $1\mathrm{e}{-3}$ for **KS**. Tasks **KS**, **IC**, and **ASR** are trained for 200k steps and task **ER** is trained for 30k steps.

E. Results

1) Baselines: HuBERT and DistilHuBERT: By comparing (T1) to (S1), and (T1') to (S1') in Table IV and Table V, it is obvious that HuBERT and DistilHuBERT have similar performance on the clean testing set for most of the speech tasks. However, both models suffer from performance degradation when distortions are introduced, especially for DistilHuBERT. This suggests that HuBERT is not robust, and the distillation process even worsens the generalizability of it.

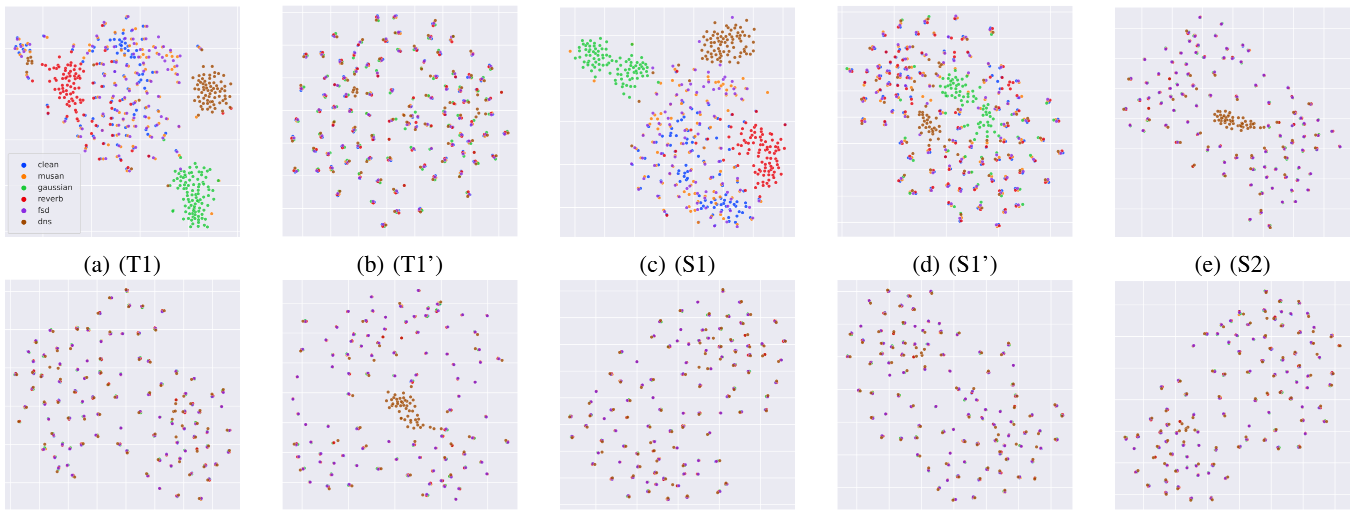
2) Different CDM settings: Setup1 forces the student to learn the clean representations of the teacher regardless of the distortions of speech. The results show that applying setup1 by forcing representations of all kinds of input speech (clean or distorted) to fall in the clean representation domain is effective ((S2)(S2') compared to (S1)(S1')). However, by comparing (S2) to (S4), and (S2') to (S4'), we observe that setup2 yields better performance than setup1 on almost every testing set for **KS**, **IC**, and **ER**. This indicates that learning representations of the same speech utterance but with different distortions improves the generalizability of the student model.

For setup2, we also experimented on the case where the teacher and student models take the same distorted speech as input ((S3) and (S3')) and found out that this setting yields low distillation loss but does not generalize as well as the original setup2 on each testing set. We also observed that (S3) and (S4) have large performance gaps on the FSD50k and DNS testing set for **IC**, **ER**, and **ASR**. This is because the student model (S3) is prone to output representations containing distorted information when the teacher and student have the same distorted speech inputs, causing the representations to be less domain-invariant.

3) Domain Adversarial Training: Setup1 with DAT not only forces the student to map its representations to the clean representations of the teacher, but also regulates the representations of the last hidden layer to be domain-invariant. The results show that regulating the representations of the last hidden layer is effective for both distorted and clean speech ((S5) (S5') compared to (S2) (S2')), and this setting (S5) even outperforms setup2 (S4) on most of the testing sets. We notice that whether using the domain-adaptive pre-trained HuBERT as the teacher model seems to have minor impacts on the average performances by comparing (S5) and (S5'), implying that DAT reduces the gap between different teacher models (the gap between (S5) and (S5') compared to the gap between (S2) and (S2')).

We constructed models (S6) and (S6') by applying DAT

[10]Details for training downstream speech models can be found at https://github.com/s3prl/s3prl/blob/main/s3prl/downstream/docs/superb.md



(a) (T1) (b) (T1') (c) (S1) (d) (S1') (e) (S2)

(f) (S2') (g) (S4) (h) (S4') (i) (S5) (j) (S5')

Fig. 8: Visualization of representations for models in Tables IV and V. Each visualization corresponds to an upstream model without any fine-tuning with downstream data. Colors blue, orange, green, red, purple, and brown represent the representations of clean speech and speech with Musan, Gaussian, reverberation, FSD50k and DNS noises, respectively. FSD50k and DNS noises are unseen distortions during pre-training.

to models (S4) and (S4'). By comparing (S6) with (S4), it showed that applying DAT to setup2 improved the performance for **IC**. By comparing (S6') with (S4'), we also found similar results for **ER** and **ASR**. However, we noticed that there were still some cases where models did not benefit from DAT by observing (S4) and (S6) having similar results for **KS**, **ER**, and **ASR**. Knowing that model (S4) does not use the domain-adaptive pre-trained HuBERT model during distillation, causing a performance gap between (S4) and (S4'), we hoped that applying DAT to (S4) could make up the gap. Unfortunately, this is not the case. DAT also seems to worsen the performances of some testing sets of **KS** and **IC**.

4) Different model sizes: We also trained different model configurations of DistilHuBERT by alternating the number of the transformer encoder layers of the student model. From models (S7)(S7')(S8)(S8'), by comparing the performance between clean testing sets and distorted testing sets (2-dist, fsd, and dns), we found out that smaller models are less robust to distortions. To ensure general usage of our proposed methods, we trained a smaller student model (S7') and a larger student model (S8') under the setting that yielded best performance (setup2). By comparing (S7) to (S7'), and (S8) to (S8'), we conclude that setup2 shows consistent results for student models of different sizes. This demonstrates that this setting is model-agnostic, and can be applied to different student architectures in the future.

F. Visualization

1) Visualization setup: To demonstrate the robustness of our proposed approaches, we visualized the last layer representations of the models with t-SNE [107] for the test-clean portion of LibriSpeech. We show the speech representations of six kinds of speech, including clean speech, speech with Musan noise, speech with Gaussian noise, speech with reverberation, and speech distorted with FSD50k and DNS noise by the following process. First, we distort all the speech utterances in the test-clean set with one kind of speech distortion and extract their representations from the last transformer layer of the upstream model. The representations are further averaged along the timestep dimension to produce a flat vector of length D. Then we divide the representations into 100 splits and average the representations in each split, resulting in 100 representations. Finally, we repeat this process for the six kinds of speech, resulting in 6 vectors for each split and 600 vectors in total for visualization.

2) Visualizing results: We show t-SNE visualizations as described in subsection IV-F1 to further understand the modeling capability of each upstream model. From Figure 8a, we observe that the original HuBERT model (T1) is not robust when the speech signal is subdued under different distortions. There are clear cluster assignments for each of the distorted speech configurations. The two most prominent clusters are the ones with Gaussian noise and DNS noise added to speech. This explains the large performance gap of model (T1) between the clean testing set and the testing set with DNS noise in Tables IV and V. A similar phenomenon can be seen for the model distilled from the original HuBERT model (S1) (see Figure 8c).

Figure 8b shows that model (T1') is robust to all the distortions. Notice that some figures in Figure 8 show multiple data points overlapped together. We verified that the representations representing speech with clean, Musan, Gaussian, reverberation, FSD50k, or DNS noise in the same split overlap together on the t-SNE visualization, meaning that, no matter if the speech signals belonging to a particular split have been distorted or not, the model will still place them into the same representational space. This supports our claim that model (T1') is more robust than the baseline teacher model (T1).

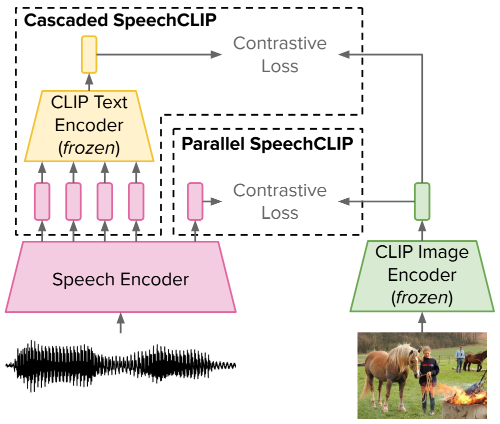


Fig. 9: An overview of the proposed SpeechCLIP model.

Having a robust teacher model proves to be crucial, as can be seen by comparing Figure 8c and 8d, where we show that a model distilled from a teacher that is not robust results in less robust student models and vice versa, explaining the large performance gap between models (S1) and (S1') in Table IV and Table V. Similar conclusions can be inferred for models in Figure 8e and 8f.

Finally, in Figure 8h, all the points in the same split with different distortions almost completely overlap with each other, showing the distortion-invariant capability of performing distillation with CDM. On the other hand, Figure 8i and 8j show that DAT is important when the teacher model is not robust, as supported by the analysis in subsection IV-E3 and the results for models (S5) and (S5') in Table IV and Table V. Yet, performing DAT does not always improve the robustness of the student when distilled from a robust teacher model. Hence, performing distillation with our CDM method is enough for achieving robustness under distorted settings when we have a robust teacher. These conclusions are supported by the results in Table IV and Table V where the difference in performance between model (S4') and (S6') are almost marginal. Our visualization provides insightful understandings of which pipeline to use depending on the characteristics of the teacher model involved during the distillation process.

V. Visually-Enhanced SSL Models

A. Introduction

This report uses paired image-speech data and an image-text pre-trained model to enhance speech SSL models. This report introduces SpeechCLIP , a novel framework to integrate speech SSL models with a pre-trained vision and language model as depicted in Figure 9. We use Contrastive Language-Image Pre-training (CLIP), a powerful model pre-trained to align parallel image-text data [108]. Then, a speech encoder initialized by a pre-trained speech SSL model is enhanced by aligning with CLIP using paired image-speech data. By aligning a speech encoder's and CLIP's image embedding spaces, the speech encoder is implicitly aligned with CLIP's text encoder, forcing it to capture more textual content.

We propose two SpeechCLIP architectures: parallel and cascaded. The parallel model is similar to WAV2CLIP [109]. However, our speech encoder uses a pre-trained speech SSL model and focuses on capturing local and global spoken contents. Meanwhile, WAV2CLIP extracts global features in general audio for classification and retrieval. Furthermore, AudioCLIP is an extension of WAV2CLIP since it is trained with paired image, audio, and text data [110]. The cascaded SpeechCLIP cascades CLIP's text encoder on top of the speech encoder, forcing the model to output subword embeddings. Eventually, the cascaded model captures spoken words in speech signals.

In this report, the proposed SpeechCLIP models achieve state-of-the-art image-speech retrieval on two standard spoken caption datasets with minimal fine-tuning. Moreover, we demonstrate SpeechCLIP 's capability of performing zero-shot speech-text retrieval and capturing keywords directly from speech. We also make our code available on Github[11].

B. Method

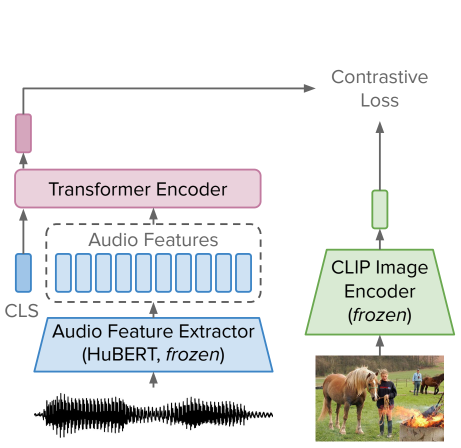
1) Preliminaries: Contrastive Language-Image Pre-training (CLIP): CLIP [108] uses contrastive learning to pre-train visual models from natural language supervision on an enormous scale, where the supervision comes from paired image-text data. Composing two encoders processing image and text separately, CLIP aims to align semantically similar images and text captions. CLIP can easily transfer across various computer vision tasks with little supervision.

In SpeechCLIP , pre-trained CLIP and HuBERT models are frozen and serve as feature extractors, as shown in Figure 10. The CLIP model extracts image and sentence embeddings to supervise SpeechCLIP . Following SUPERB [1], HuBERT 's CNN output and transformer encoder's hidden representations are weighted and summed by a set of learnable weights. The weights automatically assign importance to each hidden layer to minimize the overall objective function. Only the newly added components excluding HuBERT and CLIP are learnable during training, reducing the computational cost significantly, thus enabling a larger batch size for contrastive pre-training. In the following sections, we introduce two SpeechCLIP architectures: parallel and cascaded.

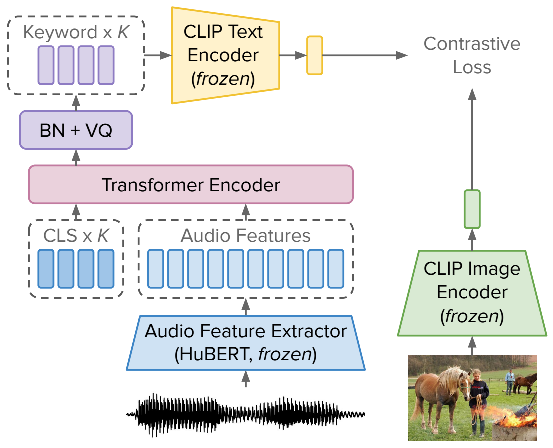
2) Parallel SpeechCLIP : Parallel SpeechCLIP is similar to CLIP, which aligns semantically related images and spoken captions, as shown in Figure 10a. Since the weighted sum of HuBERT 's output is a sequence of frame-level features, we add a learnable CLS token at the beginning of each sequence. The sequence is passed through a transformer encoder layer to obtain an utterance-level representation [111]. The representation is used to compute the cosine similarity with image embeddings in a mini-batch for calculating the contrastive loss. Cosine similarity scores are also used for retrieving speech and image samples. Following CLIP, the loss function has a learnable temperature for scaling the similarity scores.

By aligning speech and CLIP image encoders, parallel SpeechCLIP implicitly bridges speech and text representations since CLIP's image and text encoders are well-aligned. Therefore, it can perform both image-speech and speech-text retrieval. Still, this method is limited to summarizing utterances because it has no explicit constraints to capture

[11]https://github.com/atosystem/SpeechCLIP


Contrastive Loss
Transformer Encoder
Audio Features
CLS
Audio Feature Extractor (HuBERT, *frozen*)
CLIP Image Encoder (*frozen*)

(a) Parallel SpeechCLIP


Keyword x K
CLIP Text Encoder (*frozen*)
Contrastive Loss
BN + VQ
Transformer Encoder
CLS x K
Audio Features
Audio Feature Extractor (HuBERT, *frozen*)
CLIP Image Encoder (*frozen*)

(b) Cascaded SpeechCLIP

Fig. 10: An illustration of SpeechCLIP models. (a) A pre-trained HuBERT [78] extracts audio features. The features are concatenated with a learnable CLS token and fed into a transformer encoder layer to obtain a single vector representing the information of the entire sequence. The vector is then used to compute contrastive loss with the CLIP image encoder's output [108]. (b) Cascaded SpeechCLIP uses K CLS tokens to capture a small sequence of keywords from the audio signal. The keywords are batch-normalized and vector-quantized before passing to the CLIP text encoder. BN and VQ respectively denote batch normalization and vector quantization.

word-level content. Thus, the following section introduces a novel method addressing this issue.

3) Cascaded SpeechCLIP : To force the speech encoder to capture semantic information from speech, we propose cascaded SpeechCLIP by cascading speech encoder with CLIP's text encoder as shown in Figure 10b. Following parallel SpeechCLIP , the cascaded model is trained with contrastive loss, but the difference lies in the summarization process of utterances.

First, we add K learnable CLS tokens at the beginning of an audio feature sequence, where K is a hyper-parameter for the number of keywords obtained from an utterance. The sequence is fed into a transformer encoder and projected to the CLIP input embedding dimension. Next, the projected CLS tokens are batch-normalized to match the mean and variance of CLIP's subword embeddings. We apply vector quantization (VQ) to map the K normalized embeddings to CLIP's V subword embeddings. This operation produces keywords indicating the essential concepts in each utterance.

The VQ process is described as follows. We first compute the cosine similarity between the k^{th} normalized CLS embedding ($\boldsymbol{z}_k$) and the v^{th} subword embedding ($\boldsymbol{e}_v$) as

$$s_{kv} = \cos\left(\boldsymbol{z}_k, \boldsymbol{e}_v\right). \tag{11}$$

Next, we choose the subword embedding with the highest similarity from the vocabulary, which can be expressed as

$$\boldsymbol{e}_{v^\star}, \text{ where } v^\star = \underset{1\le v\le V}{\text{argmax}}\ s_{kv}. \tag{12}$$

Since $\boldsymbol{e}_{v^\star}$ is not differentiable, we compute another embedding by weighted summing all V subword embeddings as

$$\overline{\boldsymbol{h}}_k = [\boldsymbol{e}_1 \dots \boldsymbol{e}_V]\,\text{softmax}\left(\left[s_{k1}\dots s_{kV}\right]^\top / \tau\right), \tag{13}$$

where each embedding $\boldsymbol{e}_v$ is a column vector and τ is a hyper-parameter ($\tau = 0.1$). Combining (12) and (13), we apply straight-through gradient estimator [112] to obtain quantized keywords

TABLE VI: Model details. The number of parameters varies since they include parallel and cascaded models.

Model	Audio Encoder	CLIP Image Encoder	Trainable Params (M)	Total Params (M)
Base	HuBERT Base (95 M)	ViT-B/32 (250 M)	2.8 – 7.5	252 – 257
Large	HuBERT Large (316 M)	ViT-L/14 (422 M)	6.1 – 13.4	765 – 772

$$\boldsymbol{h}_k = \boldsymbol{e}_{v^\star} + \overline{\boldsymbol{h}}_k - \text{sg}\left(\overline{\boldsymbol{h}}_k\right), \tag{14}$$

where $\text{sg}(x) = x$ and $\frac{d}{dx}\text{sg}(x) = 0$ is the stop gradient operator. The K keywords are then fed into the CLIP text encoder for computing the contrastive objective.

Overall, the cascaded SpeechCLIP encourages the speech encoder to extract subwords because of the supervision from the CLIP text encoder. Hence, it is expected to capture more semantic and content information from speech.

C. Setup

1) Dataset: SpeechCLIP is pre-trained and evaluated with retrieval on Flickr8k Audio Captions Corpus [113] and SpokenCOCO dataset [114]. Each image in both datasets is paired with five spoken captions produced by humans uttering text captions. Flickr8k consists of 8k images and 46 hours of speech, while SpokenCOCO has 123k images and 742 hours of speech. Following FaST-VGS , we use the Karpathy split for SpokenCOCO [115].

2) Implementaion Details: We implemented SpeechCLIP in two sizes: Base and Large, a detailed comparison is shown in Table VI. Note that we omit the Base notation in the following sections. The hidden dimension of the transformer encoder is the same as that of the audio encoder. The feed-forward network in the cascaded model's transformer encoder is removed for better performance. Parallel and cascaded

TABLE VII: Recall scores for image-speech retrieval on Flickr8k and SpokenCOCO testing sets.

	Speech → Image			Image → Speech		
Method	R@1	R@5	R@10	R@1	R@5	R@10
	Flickr8k					
FaST-VGS$_{\text{CO}}$ [13]	26.6	56.4	68.8	36.2	66.1	76.5
FaST-VGS$_{\text{CTF}}$ [13]	29.3	58.6	71.0	37.9	68.5	79.9
MILAN [116]	33.2	62.7	73.9	49.6	79.2	87.5
Parallel	26.7	57.1	70.0	41.3	73.9	84.2
Cascaded	8.2	25.7	37.2	14.1	34.5	49.2
Parallel Large	**39.1**	**72.0**	**83.0**	**54.5**	**84.5**	**93.2**
Cascaded Large	14.7	41.2	55.1	21.8	52.0	67.7
	SpokenCOCO					
ResDAVEnet [117]	17.3	41.9	55.0	22.0	50.6	65.2
FaST-VGS$_{\text{CO}}$ [13]	31.8	62.5	75.0	42.5	73.7	84.9
FaST-VGS$_{\text{CTF}}$ [13]	**35.9**	66.3	77.9	48.8	78.2	87.0
Parallel Large	35.8	**66.5**	**78.0**	**50.6**	**80.9**	**89.1**
Cascaded Large	6.4	20.7	31.0	9.6	27.7	39.7

models have respectively eight and one attention head. We set K to 8 in all experiments. All models are trained with Adam optimizer with a weight decay of 10^{-6}, batch size of 256, and 50k steps in total. The learning rate linearly increases to 10^{-4} in the first 5k steps and linearly decreases to 10^{-8} afterward. All experiments are conducted on a 32GB V100 GPU except for pre-training on SpokenCOCO , which uses two. The largest model's pre-training lasts approximately two days.

D. Experiment

1) Image-Speech Retrieval: In this section, we evaluate SpeechCLIP on the image-speech retrieval task, showing how well models can align speech with CLIP image embeddings. As shown in Table VII, parallel SpeechCLIP models surpass almost all baseline methods, especially for the Large SpeechCLIP models. The parallel Base model on Flickr8k also shows competitive performance with the FaST-VGS $_{\text{CO}}$ model, indicating that utilizing powerful pre-trained models and a small set of learnable parameters is sufficient. Moreover, the cascaded models obtain the lowest recall scores because passing encoded speech through VQ and a CLIP text encoder loses information. Overall, the results show the benefits of integrating CLIP in VGS models even with minimal fine-tuning.

2) Zero-shot Speech-Text Retrieval: This section highlights parallel SpeechCLIP 's capability to perform zero-shot speech-text retrieval. Speech and text representations are respectively computed from a pre-trained parallel SpeechCLIP 's speech encoder and a CLIP text encoder. The representations are then used to calculate cosine similarity scores for retrieval. Although this problem has been studied for a while, prior studies require either paired speech-text training data [118], [119] or pretrained image tagger [120].

Additionally, two supervised parallel SpeechCLIP models respectively trained with paired spoken and text captions in Flickr8k and SpokenCOCO are considered as toplines. These models' CLIP image encoders are replaced with CLIP text encoders to align speech and text explicitly. When computing recall, we regard retrieving speech and text captions related to the same image as successful. Therefore, results only show whether models retrieve semantically related samples, not exact matching of speech and transcriptions.

TABLE VIII: Recall for speech-text retrieval on Flickr8k and SpokenCOCO . 'Sup." indicates the supervised version of parallel SpeechCLIP by replacing the image encoder with CLIP text encoder in parallel SpeechCLIP .

	Speech → Text			Text → Speech		
Method	R@1	R@5	R@10	R@1	R@5	R@10
	Flickr8k					
Random	0.10	0.50	0.99	0.10	0.50	0.99
Parallel Large	19.56	44.06	58.46	22.50	44.14	54.54
Parallel Large (Sup.)	97.06	99.24	99.46	97.88	99.76	99.90
	SpokenCOCO					
Random	0.02	0.10	0.20	0.02	0.10	0.20
Parallel Large	60.32	81.81	88.18	65.45	85.82	91.27
Parallel Large (Sup.)	95.02	99.46	99.78	95.35	99.68	99.93

TABLE IX: Keyword hit rates for cascaded SpeechCLIP . Avg denotes averaged hit rate. † and ‡ respectively denote models trained on Flickr8k and SpokenCOCO .

Model	kw1	kw2	kw3	kw4	kw5	kw6	kw7	kw8	Avg
Base†	57.0	25.6	20.2	5.0	20.0	26.5	10.5	16.6	22.7
Large†	56.5	19.6	20.5	37.5	21.7	34.6	26.4	44.7	32.7
Large‡	27.5	22.4	35.8	61.0	21.6	54.2	60.1	22.9	38.2

According to Table VIII, proposed SpeechCLIP models yield considerably better performance than random retrieval, showing that speech and text embedding spaces are well aligned. Specifically, parallel SpeechCLIP performs better on this task when trained on a larger dataset like SpokenCOCO . Although the performance gap between the proposed methods and the supervised toplines remains, we show that bridging speech and text with image is possible and promising.

We demonstrate that parallel SpeechCLIP retrieves noisy transcriptions for speech signals. These transcriptions can then be used for supervised or semi-supervised speech recognition model training. Furthermore, by replacing CLIP with Multilingual-CLIP[12], we can retrieve noisy transcriptions of different languages, thus performing speech translation.

3) Keyword Retrieval with Cascaded SpeechCLIP: Due to the unique design of cascaded SpeechCLIP , we investigate what and how well the speech encoder extracts keywords. For each encoded and normalized CLS token z_k, keywords are retrieved by finding subwords with the highest cosine similarities between z_k and the corresponding subword embeddings. Notice that previous works [120], [121] are also capable of retrieving semantically related keywords from speech. Nonetheless, they required pretrained image tagger and the size of keywords set is very limited. For SpeechCLIP , we can apply the same method to other pretrained langange models' vocabulary, technically. Also, our setting is quite

[12]https://github.com/FreddeFrallan/Multilingual-CLIP

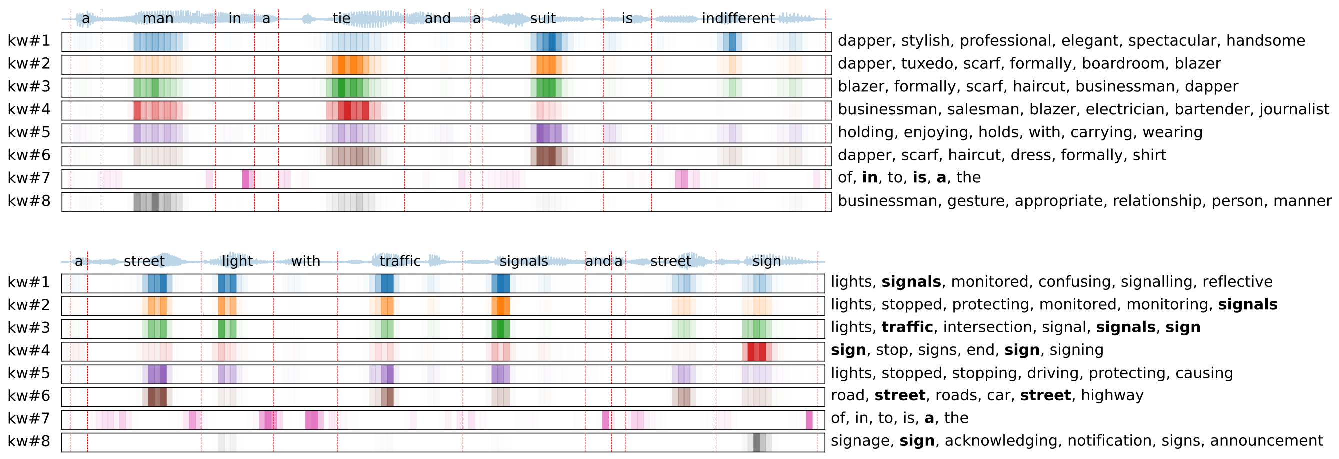


Fig. 11: Demonstration of a cascaded SpeechCLIP Large model retrieving words using its CLS tokens' outputs. The two utterances are from the SpokenCOCO test set. For each keyword in each sample, we show the transformer encoder's attention map over the whole sequence and the retrieved subwords on the right and sorted in decreasing cosine similarity. Subwords in boldface indicate they exist in the ground truth caption.

different from [122], where the 8 keywords are discovered from speech utterance without any text query in our work. Namely, SpeechCLIP can automatically summarize the speech by selecting 8 keywords. We offer quantitative and qualitative analyses in the following paragraphs.

We inspect how well keywords are retrieved from speech signals for the quantitative analysis. The evaluation metric is hit rate, which is the percentage of successful top-1 keyword retrieval of any word in the caption averaged over all testing samples. In Table IX, some CLS tokens frequently retrieve words in the ground truth captions, showing that the cascaded architecture can directly capture words from speech. Moreover, the first keyword's hit rate for models trained on Flickr8k is relatively high compared to other keywords. Probably because the first word in a sentence has a higher chance to be "a", which is also the top-1 commonly retrieved subword from the first keyword in Flickr8k . Another finding is that the Large model obtains a higher averaged keyword hit rate than the Base model on Flickr8k , which is consistent with the trend in Table VII. Hence, retrieving correct keywords is related to retrieving between speech and image samples. Although some CLS tokens obtain reasonable hit rates, one might question whether the retrieved words are meaningful instead of stopwords. Hence, we next analyze the results qualitatively to address this concern.

For the qualitative analysis, we offer two samples from the SpokenCOCO testing set in Figure 11, showing their attention maps in the transformer encoder and retrieved words for each CLS token. In the first example, although only a few retrieved keywords are in the ground truth caption, some semantically related words are found. For instance, attention maps of keywords 1, 2, and 6 focus on segments uttering "tie" and "suit." Meanwhile, they retrieve words related to clothes and appearance, e.g., "dapper", "tuxedo", and "scarf." A similar trend can be found in the second sample, showing that the cascaded objective makes the speech encoder captures semantic information. Moreover, looking at both examples, each keyword seems to have a particular purpose, e.g., the 8th keyword tends to retrieve specific nouns from utterances while the 7th retrieves prepositions. This observation leads us to investigate the properties of each keyword.

TABLE X: Top 10 successfully retrieved subwords for each keyword on SpokenCOCO test set using the cascaded Large model. The subwords are sorted in decreasing occurrence.

kw1	kw2	kw3	kw4	kw5	kw6	kw7	kw8
a	cat	bathroom	a	a	street	in	train
pizza	a	skateboard	of	tennis	bathroom	of	sign
the	room	room	in	with	kitchen	to	cake
giraffe	sheep	horse	man	eating	train	from	clock
bathroom	frisbee	elephant	woman	and	beach	for	is
skateboard	skis	motorcycle	dog	playing	bed	a	bus
living	bird	kitchen	train	the	bus	on	truck
gira	skateboard	clock	with	flying	grass	at	car
sheep	surf	tower	is	sitting	road	the	of
an	kite	bear	to	walking	room	–	signs

In Table X, we list the top 10 successfully and frequently retrieved subwords for each keyword in SpokenCOCO . Generally, commonly retrieved subwords are either stopwords like "a" and "of" or objects like "skateboard" and "street." In the first case, the phenomenon might be caused by the supervision from the CLIP text encoder because stopwords contain little information about speech signals but are sometimes crucial for maintaining the syntactic structures. Moreover, we find the frequently retrieved words for objects sometimes appear in SpokenCOCO 's captions but not very frequently. Hence, these words might be easier to be detected in speech, and the corresponding objects are more concrete to be found in images.

Additionally, we find that some keywords predict specific subword categories successfully. For instance, keyword 7 tends to output prepositions and articles, while keyword 5 mostly retrieves action words. As for the rest of the keywords, nouns are mostly retrieved. Particularly, for keyword 2, "frisbee", "skis", "skateboard", and "surf" are all related to outdoor activities. As for keyword 8, "train", "sign", "bus", "truck", "car", and

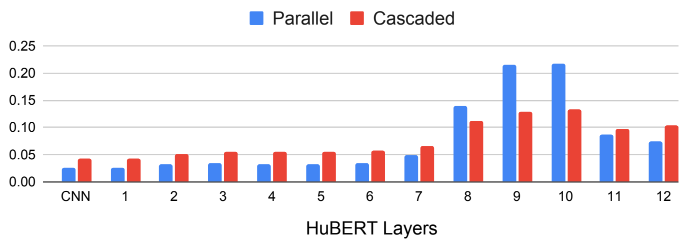


Fig. 12: Normalized weights for layer summarization of HuBERT in parallel and cascaded SpeechCLIP . CNN denotes the HuBERT CNN feature extractor.

TABLE XI: Recall scores on Flickr8k for ablation studies.

	Speech → Image			Image → Speech		
Method	R@1	R@5	R@10	R@1	R@5	R@10
	Batch Normalization					
Cascaded (w/ BN)	8.2	25.7	37.2	14.1	34.5	49.2
Cascaded (w/o BN)	1.1	4.7	8.4	1.4	5.7	9.4
	Keyword Num					
Cascaded ($K = 8$)	8.2	25.7	37.2	14.1	34.5	49.2
Cascaded ($K = 4$)	3.5	13.2	21.1	5.2	17.5	27.4
Cascaded ($K = 2$)	2.1	8.4	14.4	2.7	10.6	17.6

"signs" are all related to traffic. This section demonstrates the cascaded SpeechCLIP for retrieving semantically related keywords from speech signals.

4) Layer Importance in SpeechCLIP Speech Encoder: In this section, we show which HuBERT hidden layers are crucial for SpeechCLIP to perform well in various tasks discussed earlier. Hence, we visualize the learned weights in the weighted sum mechanism mentioned in Section V-B1 in Figure 12. Both parallel and cascaded SpeechCLIP utilize the roughly the 8th to the 10th layers in HuBERT , inferring that HuBERT 's top layers capture rich content and semantic information. This result is consistent with prior works investigating the importance of different hidden layers in speech SSL models [17], [28], [44], i.e., the top hidden layers contain word meaning and content information. However, the cascaded model's weights distribute more evenly over the layers than parallel SpeechCLIP , showing that the model architecture design affects the utilization of HuBERT 's layers.

5) Ablation Studies:

a) Batch Normalization in Cascaded SpeechCLIP : Here, we demonstrate the importance of batch normalization in the cascaded SpeechCLIP . We compare cascaded SpeechCLIP with its variant without using batch normalization, as shown in the first two rows of Table XI. Removing batch normalization degrades retrieval performance significantly, showing the significance of mean and variance matching described in Section V-B3.

b) Number of Keywords in Cascaded SpeechCLIP : This section discusses the impact of the number of keywords in cascaded SpeechCLIP . We report retrieval results on Flickr8k using different amounts of keywords in Table XI. Results show that reducing keywords degrades retrieval performance, indicating that using fewer keywords is incapable of passing information from the speech encoder to the CLIP text encoder. Furthermore, the number of subword tokens in a Flickr8k utterance is 11.3 ± 4.1, and some tokens carry less information like stopwords. Therefore, we suggest 8 is a reasonable number for K to obtain good performance with cascaded SpeechCLIP . Although dynamically assigning K for utterances of different lengths is more appropriate, we leave this approach for future investigation.

VI. Adapter

A. Introduction

In the presence of various downstream tasks, fine-tuning pre-trained models for each downstream task is parameter-inefficient since massively self-supervised pre-trained models are notoriously deep, requiring millions or even billions of parameters. Due to this reason, adapting the SSL speech model by fine-tuning requires large storage space. For example, HuBERT X-Large [78] contains 964M parameters. This results in requiring large storage space for each complete set of tuned parameters per downstream task. Furthermore, overwriting the pre-trained model parameters may not be the best way of utilizing the pre-trained knowledge from the SSL model.

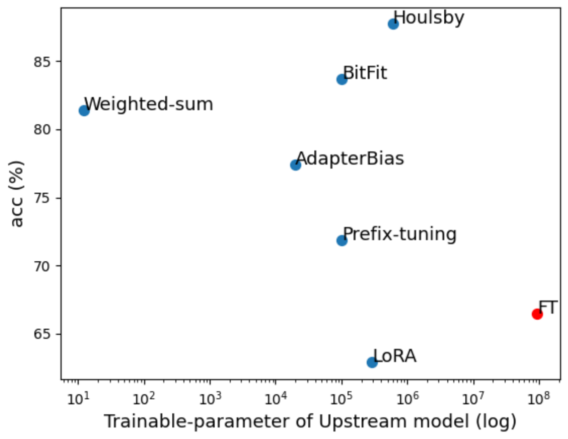


Fig. 13: The trade-off between accuracy and number of trained task-specific parameters, for several efficient tuning methods and fine-tuning. The x-axis represents trainable parameter of the upstream model, while the y-axis represents the accuracy of Speaker Identification task (SID). The red point is fine-tuning (FT), and the blue points are the efficient methods.

To overcome these shortcomings, researchers then utilize the SSL speech model by only using the frozen representation [1]. In NLP, efficient tuning techniques have been proposed for leveraging SSL models. One of the most popular efficient methods is adapters [20], which introduce extra tunable weights and freeze the original parameters of the pre-trained language model (PLM). Adapters have demonstrated comparable performance with fully fine-tuning the entire model while being parameter-efficient. More recently, the prompting technique has shown to be surprisingly effective on PLM [123]. Both methods shows that "freezing" pre-trained models is appealing, especially as model size continues to increase. Rather than requiring a separate copy of the model for each downstream task, a single generalized upstream model can simultaneously transfer to many different tasks. Adapters have been shown to work well for machine translation [124], cross-lingual transfer [125], as well as transfer learning in automatic speech recognition (ASR) [126]. However, these efficient tuning methods are not systematically studied with SSL speech models.

In order to utilize efficient tuning methods to the field of SSL speech representation, in this work, we explore the effectiveness of efficient tuning methods for self-supervised speech models on the SUPERB benchmark [1]. We apply different efficient tuning methods, including adapter tuning and prompt tuning, on SSL speech models with different training objectives. We propose an adapter framework for multiple downstream speech processing tasks, including the recognition tasks, classification, as well as speaker tasks. To investigate the effectiveness of these efficient methods, we conduct experiment on 3 SSL models with different training objectives: HuBERT, Wav2vec2 [127], and DeCoAR2 [128]. The main concept of our work is shown in Figure 14. To our best knowledge, this is the first comprehensive investigation of various efficient tuning methods on different speech tasks. We show that the performance parity can be achieved with over 90% parameter reduction. Furthermore, we show the pros and cons of various efficient tuning techniques, e.g., the Houlsby adapter [20] is the most efficient in the trade of between performance and the number of parameters, and weighted sum is a very suitable efficient method to use in SSL speech tasks.

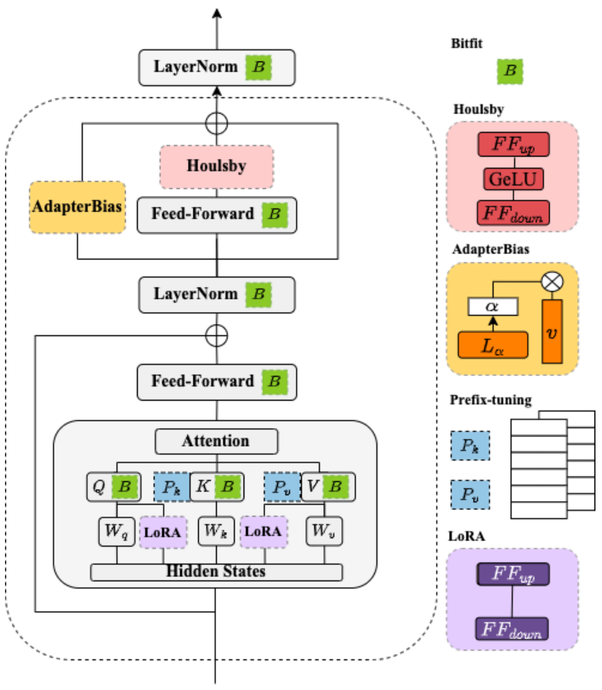


Fig. 14: Illustration of the transformer architecture and parameter-efficient tuning methods. The blocks with dashed borderlines are the added parameters by the efficient method. W_q, W_k, W_v represents the weights of query, key and value, respectively.

B. Efficient tuning for self-supervised speech models

We propose a framework to consistently evaluate the efficient tuning methods for SSL speech models. The framework is designed based on three aspects of the experiment: generalizability, coverage, and comparability.

1) Generalizability: For the purpose of examining the generalizability of the efficient tuning methods in SSL speech models, this framework includes multiple downstream speech processing tasks, involving the recognition tasks, classification tasks, as well as speaker tasks. For recognition tasks, we examine automatic speech recognition (ASR) and phoneme recognition (PR); classification tasks include keyword spotting (KS), slot filling (SF), and intent classification (IC); and for the speaker tasks, we have speaker identification (SID) and speaker diarization (SD). As for the upstream model, we conduct experiments with different training objectives SSL models: HuBERT, Wav2vec2, and DeCoAR2. The former two models are discriminative models, while DeCoAR2 is a generative model.

2) Efficient tuning approaches: As for coverage, we implement mainstream efficient tuning methods in NLP, and conduct experiments to understand different efficient methods, as well as their integration with SSL model.

The structure of our framework is shown in Figure 14. In our experiments, we apply adapters at the place where they originally added in NLP. Based on different tasks, we apply different downstream models (i.e. LSTM module, a linear classifier) on top of the transformer network. A set of adapters and the downstream model are trained per task, and the rest of the network remains frozen.

a) Houlsby adapter: Houlsby adapters [20] are small bottleneck modules consisting of a down-projection (FF_{down}), a non-linearity ($GeLU$), and an up-projection (FF_{up}), with a skip connection. Here, we add Houlsby adapters to the second feed-forward layers of transformer layers. The fully connected layers are initialized as a near identity function.

b) LoRA: LoRA [129] reduces the number of trainable parameters by learning pairs of rank-decomposition matrices (FF_{down}, FF_{up}) while freezing the original weights. This reduces the number of parameters for large language models when adapted to specific tasks. In our work, LoRA is added to the attention modules of transformer layers.

c) AdapterBias: AdapterBias [130] adds frame-dependent biases to the representation shifts by using a vector (v) and a linear layer (L_α). v represents the task-specific shift, and L_α produces the weights (α) for input frames. Thus, with the vector and the weights, AdapterBias can add a frame-dependent shift to the transformer layer. We add AdapterBias module to the second feed-forward layers of transformer layers.

d) BitFit: Instead of adding additional parameters for adaptation, Bitfit [19] tunes the bias term of each module. In our method, we tune the weight of all modules in the upstream model, such as HuBERT, Wav2vec2, and DeCoAR2.

e) Prefix tuning: For prompt tuning [123] in our unified efficient tuning settings, we use prefix tuning, which could be considered as a variant of adapter [131]. l trainable prefix vectors were prepended to the multi-head attention modules of all transformer layers. To be more specific, the original key (K) and value (V) are concatenated with trainable prefix vectors $P_k, P_v \in R^{l \times d}$, where d is the model dimension. During training, only the prefix vectors and the downstream model are updated, while the upstream model remains fixed.

f) Weighted sum: In the framework of [1], they weighted the sum of multiple hidden states from the upstream model as the final representation. In our framework, we regard the weighted-sum technique as an efficient method.

3) Comparability: For the purpose of the comparability of our proposed framework, we design our downstream model to be similar to the SUPERB benchmark, so that our approach is reproducible and comparable. The configuration setting and the hyper-parameter search is consistent with the SUPERB benchmark so that the efficient tuning methods could be evaluated from the aspect of performance, parameter efficiency, as well as stability, and understand the pros and cons of each method for SSL speech processing tasks.

Inspired by the SUPERB benchmark, we design our framework to keep the downstream models and their fine-tuning simple, while ensuring the performance across pre-trained models with different efficient tuning methods is comparable. PR, KS, SID, and IC are simple tasks that are solvable with linear downstream models. Hence, we use a frame-wise linear transformation for PR with CTC loss [132]; mean-pooling followed by a linear transformation with cross-entropy loss for utterance-level tasks (KS, SID, and IC). For ASR, a vanilla 2-layer 1024-unit BLSTM is adopted and optimized by CTC loss on characters. The trained model is decoded with LibriSpeech [80]. Regarding SF, slot-type labels are represented as special tokens to wrap the slot values in transcriptions. Similar to the SUPERB benchmark, SF is also re-formulated as an ASR problem. As for SD, we train SD with permutation-invariant training (PIT) loss, which is also used in the SUPERB benchmark.

C. Experiment

1) Performance on the SUPERB benchmark: We explore different efficient methods in the SUPERB benchmark. Note that 'FT' represents fine-tuning. The 'Baseline' here means that we tune the downstream model only. The tasks we have examined can be categorized into three: recognition task, classification task, and speaker task. The result is shown in Table XII. In general, most efficient methods perform better than Baseline and FT. For the classification tasks (i.e. KS, IC), Baseline already yields good performance. Thus, the improvement in using efficient methods is not apparent. For recognition and speaker tasks (i.e. ASR, PR, SD, SID), the advantage of using efficient methods can be seen. Especially in SID, Houlsby improves 23% accuracy compared to Baseline. On average, Houlsby yields high and stable performance among all efficient methods since it has the biggest trainable parameter. For LoRA, it performs worst among efficient methods and even worse than Baseline in some tasks (i.e. PR, SD, SID). One thing worth mentioning is that Weighted-sum is a powerful and efficient method for speech tasks, where it gets comparable performances in the SUPERB benchmark by just adding a few trainable parameters to the upstream model.

2) Upstream models with different training objectives: We also examine the generalization ability of these efficient methods with upstream SSL speech models with different training objectives. We use three different training objective models as upstream models: HuBERT, DeCoAR2, and Wav2vec2. As shown in Table XIII, efficient methods all gain comparable performance when applied to different upstream models. For example, in SD, Houlsby performs best when using HuBERT, DeCoAR2, and Wav2vec2; in KS, BitFit performs best. Moreover, the improvement of utilizing efficient methods depends on the upstream model. If the upstream model already yields strong performance in Baseline, the performance gain of using efficient methods becomes less. In contrast, if Baseline does not get a strong performance, the improvement of using efficient methods is more significant. For ASR, we can observe that Houlsby adapter improves 1.21% word error rate (WER) than Baseline when the upstream model is HuBERT. However, when the upstream model is DeCoAR2, using Houlsby adapter improves 10.43% WER.

D. Low-resource Adaptation

In NLP, adapters are shown to have advantages over fine-tuning when adapting to low-resource datasets [19], [130], [133]. To see if this property also holds when applied in speech tasks, we trained different efficient methods in the low-resource settings. All methods were trained with 1-hour and 10-hour datasets generated by Libri-Light and tested on the testing set of LibriSpeech. We conducted experiments on recognition tasks, including ASR and PR. As shown in Fig 15, the efficient methods perform better than fine-tuning in the low-resource settings. We observed a similar tendency in speech tasks. As the training data becomes smaller, tuning the majority of the parameters may result in a higher risk of overfitting the training data. Using adapter methods helps overcome this issue. Also, we found that LoRA failed to achieve comparable performance in the low resource settings as it cannot perform well in speech tasks generally. For PR, fine-tuning performs better than Houlsby adapter in 100-hour training data. However, as the size of training data decreases, the benefit of efficient tuning methods started to emerge. As shown in Fig 3, in 10-hour and 1-hour, Houlsby adapter started to perform better than fine-tuning.

E. Analysis

In this part, we explore the benefit of efficient tuning methods beyond parameter-efficiency from two aspects: stability and learning rate robustness.

1) The stability of low-resource adaptation: Here we use the Libri-Light tool to split different low-resource data from LibriSpeech with different random seeds. For each efficient method, we run three random seeds and compute the mean and standard deviation. From Table XIV, we can find that efficient methods have more tolerant than FT when the training data

Method	Params	ASR	PR	SD	SID	SF	IC	KS
FT	94.7M	6.35	**2.45**	9.32	66.48	84.87	99.10	95.87
Baseline	0	7.09	7.74	7.05	64.78	86.25	96.39	95.32
Houlsby	0.60M	5.88	3.00	**4.00**	**87.71**	85.87	**99.60**	97.17
AdapterBias	0.02M	**5.54**	4.19	5.48	77.38	86.60	99.50	97.30
BitFit	0.10M	9.34	4.23	5.13	83.68	**87.40**	99.50	**97.33**
LoRA	0.29M	6.94	8.74	7.39	62.90	86.25	96.57	96.59
Prefix	0.10M	6.56	4.18	8.17	71.87	85.85	99.31	97.05
Weighted-sum	12	6.42	5.41	5.88	81.42	88.53	98.34	96.30

TABLE XII: Performance of different efficient methods in the SUPERB benchmark. The second column represents additional trainable parameter used in upstream model. Note that except for the "Weight-sum" method, other methods directly use the last layer representation of upstream model as the input of the downstream model.

	ASR			SD			KS		
Method	HuBERT	DeCoAR2	Wav2vec2	HuBERT	DeCoAR2	Wav2vec2	HuBERT	DeCoAR2	Wav2vec2
FT	6.35	25.46	6.01	9.32	12.67	12.38	95.81	27.36	97.50
Baseline	7.09	39.06	10.79	7.05	9.14	8.07	95.32	91.69	91.95
Houlsby	5.88	28.63	5.99	**4.00**	**5.76**	**4.23**	97.17	96.07	96.75
AdapterBias	**5.54**	29.89	**5.96**	5.48	6.95	6.16	97.30	96.23	91.56
BitFit	9.34	29.40	6.01	5.13	7.05	5.29	**97.33**	**96.72**	**96.85**
LoRA	6.94	39.52	11.32	7.39	8.78	7.92	96.59	25.38	92.80
Prefix	6.56	**13.48**	6.54	8.17	10.05	10.01	97.05	88.7	96.82
Weighted-sum	6.42	36.26	6.43	5.88	5.88	6.08	96.30	94.48	96.23

TABLE XIII: Performance of different upstream models. We used three different objective self supervise speech models: HuBERT, DeCoAR2, and Wav2vec2.

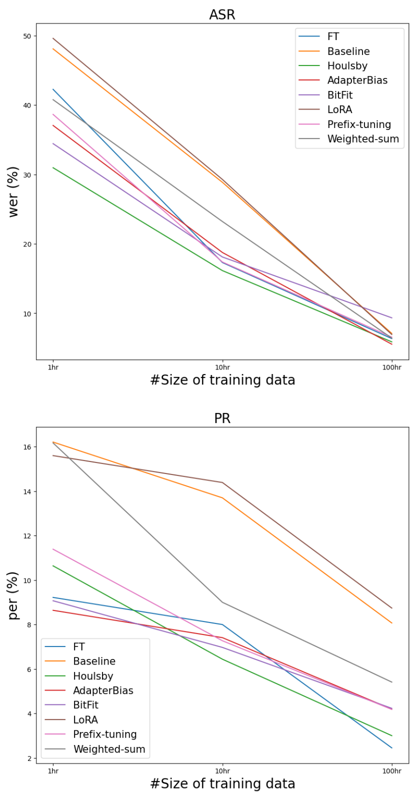


Fig. 15: Performance of different efficient methods in low-resource adaptation. We train with 1 hour data and 10 hour data from librilight and test models on LibriSpeech.

	ASR		PR	
Method	1hr	10hr	1hr	10hr
FT	35.38±**6.02**	15.30±1.73	15.60±**5.56**	6.15±0.98
Baseline	51.54±3.04	24.51±3.85	16.90±0.63	12.93±0.72
Houlsby	31.67±0.94	16.67±4.20	11.26±0.62	6.44±0.13
AdapterBias	37.97±1.45	19.35±5.92	8.83±0.32	6.96±0.39
BitFit	35.47±1.39	15.17±2.54	8.89±0.34]	6.93±0.09
LoRA	51.27±1.62	29.47±**8.20**	15.96±0.32	14.85±**1.64**
Prefix	39.00±1.17	17.71±1.04	11.75±0.35	7.25±0.08
Weighted-sum	43.84±2.15	23.22±7.77	13.58±2.24	9.04±0.08

TABLE XIV: Performance of different low-resource data in efficient methods. We train with three random seeds and report the mean and standard deviation.

Method	$5{\times}10^{-6}$	$5{\times}10^{-5}$	$5{\times}10^{-4}$	$5{\times}10^{-3}$
FT	3.03±0.1	2.81±0.4	100±0	100±0
Houlsby	6.09±0.49	3.24±0.14	2.81±0.03	3.06±0.03
AdapterBias	7.54±0.06	4.52±0.01	3.79±0.02	3.72±0.02

TABLE XV: Performance of different methods with different learning rates. The downstream task is PR. We run 5 different random seeds and report the mean and standard deviation.

becomes less. Compared with ASR and PR, ASR has a bigger standard deviation than PR. The reason may be that we use a more complex downstream model (2 layers of LSTM) in ASR. Training with low-resource data would make the complex model more unstable than a simple downstream model (i.e. a linear layer) used in PR.

2) Learning rate robustness of efficient tuning methods: This part evaluates the tolerance of the learning rate in different methods. Here we pick fine-tuning (FT), Houlsby adapter, and AdapterBias since Houlsby adapter has the biggest trainable parameters and AdapterBias has the lowest parameters. In Table XV, we train on PR and learning rates ranging from $5{\times}10^{-6}$ to $5{\times}10^{-2}$. We can observe that FT has less tolerance than efficient methods. FT does not work on larger learning rates, while efficient methods receive more stable performance among a large range of learning rates. Comparing with Houslby adapter and AdapterBias, AdapterBias has smaller standard deviation than Houlsby adapter since AdapterBias has less trainable parameters than those of Houlsby adapter. Thus, with less trainable parameters, the model would not overfit to training data.

3) Discussions: In this section, we discuss the strength and limitation of efficient tuning methods in speech processing tasks, as well as their behavioral difference from NLP.

a) Performance analysis of adapter methods: From the experimental results, we found that Houlsby adapter performs the best among all efficient tuning methods. This is different from NLP, as in NLP, the overall performance gain of Houlsby adapter is not that significant [131]. In the SUPERB benchmark, Houlsby adapter outperforms other efficient methods in 3 out of 7 tasks.

LoRA is an effective adapter in NLP, achieving comparable performance with other adapters [129]. However, it performs worst in the SUPERB benchmark. We guess that the position added adapters play a crucial role. Both Houlsby adapter and AdapterBias are added behind the second feed-forward layer, while LoRA is added in the attention module. Therefore, in SUPERB benchmark, adding adapters in the feed-forward layer is more effective than adding adapters in the attention module.

In NLP, prefix-tuning achieves comparable performance with adapter methods [131]. Nonetheless, prefix-tuning does not perform better than adapter methods in the SUPERB benchmark. One reason may be the initialization of prefix-tuning significantly affects the performance in speech tasks. The embedding is discrete in NLP tasks, while in speech tasks, each frame representation is continuous. Thus, we initialize the prefix with the average of the hidden states of the first batch of data. However, it is still worth designing a suitable initialization of prompt in the future.

In addition, weighted-sum is not a common technique in NLP. Nevertheless, weighted-sum improves a huge performance in the SUPERB benchmark. In the work [17], they find that output from each layer of speech SSL model contain information related to different tasks. Therefore, weighted-sum leverages information from different layers and receives high performance in speech tasks.

b) Performance analysis of different types of tasks: In NLP, most efficient methods work well on classification tasks, but do not perform as well in generative tasks. In the SUPERB benchmark, utilizing efficient methods achieves good performance in general on not only classification tasks (i.e. IC, KS), but also generative tasks, such as ASR. However, there are some tasks (i.e. PR, SF) where efficient methods do not work very well. In the future, it is worth designing a suitable adapter for speech and considering more challenging tasks, such as Out-of-domain Automatic Speech Recognition Tasks (OOD-ASR).

VII. Prompt

A. Introduction

In the Natural Language Processing (NLP) field, prompting methods have gained researchers' attention [21]. The methods scale up pre-trained language models (LMs) at serving multiple downstream tasks in a unified and efficient way. For each downstream task, prompting methods aim to find task-specific templates or a limited number of parameters that steer LMs to generate results for the task without modifying LM's parameters. For example, in a sentiment classification task for movie review, we can design a prompt "[X] The movie is __". The LM takes a sentence to be classified and fits it into the template at [X]. By generating a sentiment word from a pre-defined set of tokens (e.g. great, neutral, bad) that one-to-one mapped to classification labels, we transform the sentiment classification task into a generation problem. Alternatively, prompts are not necessary to be readable by humans. Researchers proposed prompt tuning methods that learn continuous prompts [21], [123], [134], [135] in models' embedding space. Studies have shown that prompt methods can reformulate most NLP tasks as generation problems and yield competitive performance [21].

The prompting paradigm is appealing as the number of downstream tasks to be served increases. Rather than requiring a specialized downstream model for each task, a single generalist model can simultaneously serve many different tasks in one inference batch. Since parameters of tuned prompts are usually several orders smaller than parameters of LMs [136], the prompting paradigm significantly improves memory and computation efficiency. Furthermore, there is a unified inference process with the original pre-trained LM for all downstream tasks in the paradigm. Hence, less human labor is required in model authoring for each task. Despite the success in NLP, there is little research on the prompting paradigm in the speech community.

To bring the benefit of the prompting paradigm to the speech processing field, we propose a prompt tuning framework for multiple downstream speech processing tasks, including Keyword Spotting (KS), Intent Classification (IC), Automatic Speech Recognition (ASR), and Slot Filling (SF). The framework unifies training and inference for multiple tasks by leveraging the generation capability of the pre-trained LM. To our best knowledge, our work is the first study in the prompting paradigm that achieves competitive performance in various speech processing tasks.

We utilize **Generative Spoken Language Model (GSLM)** [137] as our backbone LM and apply prompting on top of it. GSLM is used, for GSLM is the first generative speech LM pre-trained on a large-scale speech dataset, and it has a large model capacity to generate meaningful output. Experiment shows that the proposed framework achieves competitive accuracy (Acc) in single-label and multi-label speech classification tasks. While the framework demonstrates the potential of prompting GSLM, we also identify the limitations when performing challenging sequence generation tasks and discuss the potential research directions in the paper. We hope by exploring and analyzing the novel prompting paradigm for speech processing, this work can inspire the speech community to explore more on this paradigm.

B. Related Works

1) SSL Speech Representations: Learning speech representations with SSL objectives has become a vital research topic

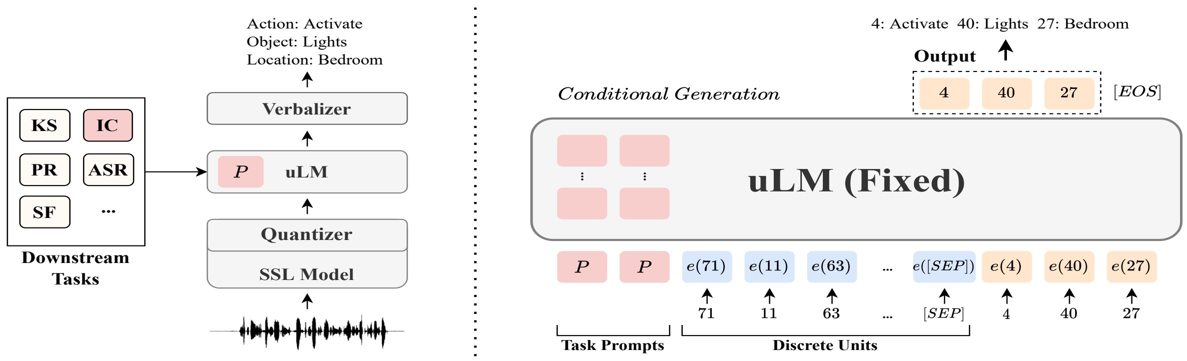


Fig. 16: (a) The overview of the proposed framework. Task-specific prompts are applied to the unit language model (uLM) to generate predictions. (b) The uLM performs generation conditioned on the discrete unit sequence and the task prompts.

in the speech community. For example, CPC [138] learns by predicting future features in a contrastive manner. HuBERT [**?**] maximizes the similarity between output representations and clusters of acoustic features during pre-training. To leverage SSL representations, a common way is to build specialized downstream models on top of SSL representations and fine-tune the entire models or only the downstream ones for supervised downstream tasks. Based on this, SUPERB [1] benchmarks speech SSL techniques with a wide variety of downstream tasks. This work explores an alternative paradigm: we add a fixed, pre-trained LM on top of SSL representations and prompt the LM to generate predictions directly.

2) Generative Spoken Language Model: Researchers proposed **Generative Spoken Language Model (GSLM)** [137] to model the rich expressiveness of spoken language based on discovered units of raw audio without any text or labels. GSLM first leverages the representation learning ability of SSL speech models and encodes raw speech into a sequence of discrete units by an SSL model and the K-means clustering algorithm. A generative **unit language model (uLM)**, the core component of GSLM, is then trained to perform speech generation on top of discrete units. The technique shows competitive performance to generate novel speech unconditionally or conditioned on a speech segment. GSLM can be considered the speech version of GPT-3 [139] and is the first generative speech LM trained on a large speech corpus. The work opens the door to applying prompt tuning methods for speech processing tasks.

3) Prompting and Reprogramming: Prompting refers to techniques for finding task-specific instructions or templates that steer a pre-trained LM without modifying its parameters [21]. By concatenating examples and a task description to the original sentence as the input, GPT-3 [139] performs *in-context learning* to directly generate labels of the sentence. Although in-context learning yields competitive results, it requires heavy hand-crafted prompt engineering, and it is difficult to scale to smaller pre-trained models [140]. To avoid hand-crafted prompt engineering, automatically generating templates [141], [142] is another research direction.

Under the premise of fixing pre-trained LMs' parameters, researchers further explore *prompt tuning*, where continuous prompts are learned in the model's embedding space. For example, [134], [136] learn continuous prompts in LMs' input embedding space. We refer to this kind of methods as *input prompt tuning*. Another similar technique to input prompt tuning is *model reprogramming* [143], where an input transformation function is learned to reprogram a pre-trained model to perform a target task. [144], [145] have explored reprogramming acoustic models while focusing on single-label classification tasks with a supervised model. In this work, the proposed framework can perform various speech processing tasks and is not limited to input transformation. Alternatively, Prefix-Tuning [123] and P-Tuning v2 [135] performs *deep prompt tuning*, in which prefix prompts are further prepended at the input of model's hidden layer. We mainly utilize deep prompt tuning in this work. Meanwhile, we also investigate applying prompts only at the input of the LM for comparing different prompting techniques.

C. Method

We propose a prompting framework to adapt GSLM to a given downstream task by conditioning the **uLM** on task-specific prompts. Figure 16 illustrates the framework. An utterance is first encoded into discrete units by an SSL speech model and a K-means quantizer. The uLM then takes the sequence of units as input and prepends it with task-specific prompts. We then perform conditional generation with the uLM to output units that will be mapped to task labels with a pre-defined verbalizer. In the following, we describe details in the framework, including applying prompts at uLM, controlling the output of conditional generation, and the label mapping with a verbalizer.

D. Prompt Tuning

A causal uLM $\mathcal{M}$ takes a discrete unit sequence $\boldsymbol{u}_x$ as input and autoregressively outputs a sequence $\boldsymbol{u}_y = \mathcal{M}(\boldsymbol{u})$ until an end-of-sentence token "$[EOS]$" is produced.

In prompt tuning, the parameters of the pre-trained uLM $\mathcal{M}$ are fixed. Given $\mathcal{M}$ and a downstream task, a set of trainable task-specific prompt vectors $\mathcal{P}$ is optimized during adaptation with supervision from the task. The number of trainable parameters for each task is denoted as $|\mathcal{P}|$. We adopt deep prompt tuning similar to [123], [135] in our framework. Given an utterance, the SSL model and the quantizer first encode it into a sequence of discrete units $\boldsymbol{u}_x = [u_1, u_2, \cdots, u_T], u_i \in \mathcal{U}$, where T is the unit sequence length, and $\mathcal{U}$ is the unit space of the uLM[13]. Trainable continuous prompts are then applied to (a) the input of the uLM in its embedding space and (b) the input of the attention mechanism in each Transformer block.

(a) Prompts at the input of the uLM

Given an unit sequence $\boldsymbol{u}_x$ as the original input, the input embedding layer of the uLM $\boldsymbol{e}(\cdot) : \mathcal{R} \mapsto \mathcal{R}^d$ first transforms it into a sequence of embedding vectors: $\boldsymbol{e}(\boldsymbol{u}_x) = [\boldsymbol{e}(u_1), \boldsymbol{e}(u_2), \cdots, \boldsymbol{e}(u_T)]$. The sequence is then prepended with continuous prompts and fed into the uLM:

$$[\boldsymbol{p}_1^I, \boldsymbol{p}_2^I, \cdots, \boldsymbol{p}_l^I, \boldsymbol{e}(\boldsymbol{u}_x)] \tag{15}$$

where $\boldsymbol{p}^I$ are trainable vectors in $\mathcal{P}$, and l is the prompt length.

(b) Prompts at the input of the attention mechanism

Solely applying prompts to the input embedding may not be powerful enough to steer a pre-trained LM [135]. Therefore, we also apply prompts to the input of the self-attention mechanism [111] in every Transformer block of the uLM. Given a Transformer block that takes the embedding $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]$ as input, we manipulate the key K and value V in the attention function $Attn(Q, K, V)$ [111]:

$$K = Concat(\boldsymbol{p}^K, \boldsymbol{x}_{l+1:T})W^K \tag{16}$$

$$V = Concat(\boldsymbol{p}^V, \boldsymbol{x}_{l+1:T})W^V \tag{17}$$

where $\boldsymbol{p}^K$ and $\boldsymbol{p}^V$ are trainable vectors in $\mathcal{P}$. That is, we replace the first l vectors of $\boldsymbol{x}$ with the trainable prompt vectors.

E. Conditional Generation and Verbalizer

To leverage the generation capability of the uLM for inference in downstream tasks, we reformulate all tasks into conditional generation problems. The uLM generates an output sequence $\boldsymbol{u}_y$ conditioned on the input unit sequence $\boldsymbol{u}_x$ and task-specific prompts $\mathcal{P}$. Let $\boldsymbol{y} = (y_1, ...y_n)$, $y \in \mathcal{Y}$, be a sequence of n task labels and $\mathcal{Y}$ is the label space of a task. The task label length $|\boldsymbol{y}|$ is flexible depending on the task. For example, in classification tasks, $\boldsymbol{y}$ can be a single label or multiple labels. In recognition tasks, $\boldsymbol{y}$ can be a character sequence.

To connect the LM's output with the labels of downstream tasks, a *verbalizer* [140], [146] is introduced in the paradigm of prompting LMs. The verbalizer is a one-to-one mapping $v : \mathcal{Y} \mapsto \mathcal{U}$ that maps from the task label space to vocabulary of the language model. In the case of the uLM, the vocabulary is the units $\mathcal{U}$. With the help of the verbalizer, the output units can be mapped back to task labels. For example, in Intent Classification, the output units $\boldsymbol{u}_y = [4, 40, 27]$ can be interpreted as intent labels ["Active", "Lights", "Bedroom"].

The trainable prompts are then optimized with a loss function $\mathcal{L}$, which is cross-entropy for every task in this work:

$$\mathcal{P} = \arg\min_{\mathcal{P}} \mathcal{L}(\mathcal{M}(\mathcal{P}, \boldsymbol{u}_x), v(\boldsymbol{y})) \tag{18}$$

[13]Following GSLM [137], unit deduplication is also applied universally. (e.g. the unit sequence 71 11 11 63 63 63 becomes 71 11 63.)

TABLE XVI: Summary of downstream tasks used in the work. **SLU** denotes Spoken Language Understanding. **CLS**: Classification. **SG**: Sequence Generation. $\overline{|\boldsymbol{y}|}$: average label length in the task.

Task		Type	N_{class}	$\overline{\|\boldsymbol{y}\|}$	Dataset
KS	Detection	CLS	12	1	[147]
IC	SLU	CLS	24	3	[148]
ASR	Recognition	SG	29	173	[80]
SF	Recogition + SLU	SG	69	54	[149]

F. Experiment Setup

1) Tasks and Datasets: We evaluate the proposed framework on various speech processing tasks, including speech classification tasks: Keyword Spotting (KS) and Intent Classification (IC); sequence generation tasks: ASR and Slot Filling (SF). Table XVI gives a brief summary for each task. We follow the same dataset and data splits as in SUPERB [1].

2) Implementation Details: **uLM** We use the checkpoints of the pre-trained uLMs corresponding to HuBERT and CPC representation with 100 clusters on *fairseq*[14] [79]. The uLM is a causal LM consisting of 12 Transformer decoder layers with 151M parameters. All the parameters are fixed during prompt tuning.

Verbalizer We utilize a simple frequency-based algorithm to implement the verbalizer, in which no model estimation [146], [150] is involved to simplify the pipeline. For N classes in the task (c.f. Table XVI), we map those N classes into N unique units by the following steps: (1) Find and sort top-N frequent units in the input of the training data, denoted as $[u_1, u_2, ...u_N]$. (2) Find and sort top-N frequent classes in the ground truth of the training data, denoted as $[c_1, c_2, ...c_N]$. (3) Define the verbalizer v as an one-to-one function: $v(c_i) = u_i$. We find that applying the frequency-based verbalizer improves the performance by a small margin compared to random assignment. We do not show the experiment result of random assignment due to space limitations.

Prompt Length We find that the optimal prompt length varies between tasks. For speech classification tasks, we used as fewer prompts as possible while keeping the performance competitive. Regarding sequence generation tasks, we use prompt length $l = 180$, where 4.5M parameters, which equals 3% parameters of the uLM, are trainable.

G. Results

1) Speech Classification Tasks: Table XVII shows the result of the proposed prompt tuning (PT) framework in multiple tasks. For comparison, we also list the performance of fine-tuning the uLM (FT-LM), where the same framework is adopted but with the entire uLM trainable. We also list the performance of fine-tuning the specialized downstream models (FT-DM) as in SUPERB [1] as a strong baseline. SUPERB utilizes linear models as downstream models for KS and IC, and 2-layer Bi-LSTMs for ASR and SF. As shown in Table XVIIa, in speech classification tasks, prompt tuning

[14]https://github.com/pytorch/fairseq/tree/main/examples/textless_nlp/gslm

achieves competitive performance with fewer trainable parameters. Notably, in IC, a multi-label classification task, prompt tuning outperforms fine-tuning the entire uLM or downstream models. The advantage might be that a sequence generation model is suitable for learning the correlation between labels [151] [15].

TABLE XVII: Performance of prompt tuning in various speech processing tasks. Fine-tuning baselines are listed for comparison. **PT**: Prompt Tuning. **FT-LM**: Fine-Tuning the pre-trained uLM. **FT-DM**: Fine-Tuning the Downstream Model as in SUPERB.
#: Number of trainable parameters.

(a) Performance on speech classification tasks.

Scenarios	KS		IC	
	Acc ↑	#	Acc ↑	#
HuBERT-PT	95.16	0.08M	**98.40**	0.15M
FT-LM	94.03	151M	97.63	151M
FT-DM	**96.30**	0.2M	98.34	0.2M
CPC-PT	**93.54**	0.05M	**97.57**	0.05M
FT-LM	93.48	151M	95.62	151M
FT-DM	91.88	0.07M	64.09	0.07M

(b) Performance on sequence generation tasks.

Scenarios	ASR		SF		#
	WER ↓	CER ↓	F1 ↑	CER ↓	
HuBERT-PT	34.17	26.14	66.90	59.47	4.5M
FT-LM	26.19	16.80	80.58	40.15	151M
FT-DM	**6.42**	**1.48**	**88.53**	**25.20**	43M
CPC-PT	59.41	37.12	65.25	60.84	4.5M
FT-LM	35.61	17.90	**79.34**	**42.64**	151M
FT-DM	**20.18**	**5.25**	71.19	49.91	42.5M

2) Sequence Generation and Curse of Long Sequences: We further push the limit of the prompting paradigm to perform challenging sequence generation tasks: ASR and SF. As shown in Table XVIIb, we find that even fine-tuning the uLM (FT-LM) is not comparable to the performance of fine-tuning the specialized downstream models (FT-DM), where CTC loss and Bi-LSTMs are adopted. To better understand the gap and possible mitigations of proposed prompt tuning, we study the correlation between the label length $|\boldsymbol{y}|$ (i.e., sequence to be generated) and the character error rates (CERs) of HuBERT-PT and HuBERT-FT-LM in ASR. Figure 17 shows that the performance drops significantly when it comes to long sequences.

We surmise that the performance drop results from that the uLM is a causal, decoder-only model, which may be unsuitable for recognizing long sequences. Similar phenomena have also been observed in the NLP field. Due to the limitation of the unidirectional attention mechanism, generative models need more parameters and pre-trained data to work on Natural Language Understanding (NLU) tasks [152]. In a more complex task, text summarization, GPT-2 is also suffered from long sequences [123]. Although GPT-3 [139] shows competitive performance by performing NLU tasks as a generation problem, a much larger model (175B parameters) is also required. The uLM has only 151M parameters and therefore falls behind the fine-tuning of specialized downstream models

[15]For an example in IC, object "lights" can be "activated" but cannot be "decreased."

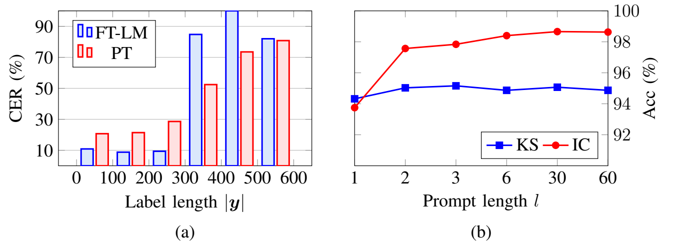


Fig. 17: (a) CER of different label length $|\boldsymbol{y}|$ intervals when using HuBERT. (b) Accuracy of HuBERT-PT with different prompt length on KS and IC.

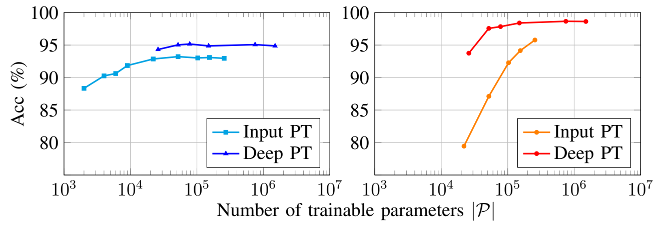


Fig. 18: Comparison of input prompt tuning (Input PT) and deep prompt tuning (Deep PT). Left: on Keyword Spotting. Right: on Intent Classification.

in challenging sequence generation tasks, where label lengths are way longer than those in [123] (c.f. Table XVI). We will continue the discussion of the limitations of existing pre-trained generative models and possible research directions in Section VII-G5.

3) Prompt Length: We vary the prompt length l in HuBERT-PT to study the effect of the number of trainable parameters. In Figrue 17b, the result shows that as the prompt length increases, there is a trend of increasing performance. It is worth noting that it still achieves a reasonable accuracy with prompt length only equal to 2, where 52K trainable parameters are introduced.

4) Input Prompt tuning: When the inner parameters of pre-trained LMs are not accessible, only input prompt tuning can be applied. Thus, we further study the IC and KS performance of our approach when prompts are only used at the input of the uLM. Figure 18 shows that although deep prompt tuning (i.e. applying prompts further at LMs' hidden layer described in section VII-D(b)) consistently outperforms input prompt tuning, the latter can also achieve competitive performance with sufficient trainable parameters.

5) Discussion and Future Works: Unlike prompt tuning in NLP, the meaning of the uLM's vocabulary is not obvious. In NLP, it is usually simple to identify how to define the verbalizer [146], and often the verbalizer is even an identity function when the prediction target is the vocabulary itself [140]. This paper leverages a heuristic, frequency-based approach to define the verbalizer. How to better identify the mapping between discovered units and task labels is critical for performance and remains unsolved.

Although the experiments show that the proposed framework achieves competitive results in speech classification tasks, we are restricted by the nature of the uLM when performing challenging sequence generation tasks. In NLP, prompting on text classification tasks has also achieved remarkable results [21], [140], [146], [153]. However, to solve more difficult text generation tasks (e.g. summarization, translation), larger and more powerful pre-trained LMs including Prefix LMs (e.g. UniLMs [154], [155]) and Encoder-Decoder LMs (e.g. T5 [142], BART [156]) are often introduced [21], [123], [135], [157]. For speech processing tasks, the problems might be even more difficult since the model is expected to perform recognition (KS, ASR), understanding (IC), or both at the same time (SF), while there are few LMs available for speech. We hope this work motivates the speech community to invest in diverse and effective speech LMs.

VIII. SSL for Prosody

A. Introduction

Self-supervised Learning (SSL) has revolutionized research in many areas of artificial intelligence, including speech processing. SSL pre-trained speech models have shown remarkable performance and generalizability across a wide range of tasks [1], [2], [158]. However, we do not currently have a good understanding of what knowledge these models capture nor of the limits of their power. This is true in particular for the prosodic aspects of speech.

The speech signal contains not only lexical but prosodic information. Broadly speaking, the latter has three realms of function: paralinguistic, phonological, and pragmatic. Paralinguistic functions, such as marking speaker identity and expressing emotion, are largely conveyed by prosodic settings that are stable over the span of many utterances, and are often evident from any sample of just a few syllables. Phonological functions, notably marking the identity of syllables and words with tones and stress patterns, are largely conveyed by prosodic features whose temporal occurrence is tightly linked to the units they mark. The utilities of SSL models for these two realms have been demonstrated, by many recent studies using tasks from the SUPERB [1] benchmark, among others.

However, for the third realm, the realm of pragmatic function, the question of the utility of SSL models has remained open. Functions in this realm, include managing turn-taking, marking topic structure and information structure, and expressing engagement, stance, attitude, and intent. These pragmatic functions are especially important in dialog, and we expect that future dialog systems will need more prosodic competence, in order to enable more satisfying user experiences and to support interaction in novel genres and situations [159]. In many cases, these functions are expressed using multistream temporal configurations of low-level prosodic features, where these configurations can last from a few hundred milliseconds to several seconds [160], [161], and may be only loosely aligned with the lexical content. As these configurations are fundamentally different from the forms of prosody in the other two realms, it is an open question whether SSL models are also useful for this realm.

Accordingly our research question is whether pre-trained models have utility for prosody-conveyed pragmatic functions. We investigate this in four ways. First, we assemble a set of prosody-intensive tasks and measure how well pre-trained models support them. Second, we use pitch and energy reconstruction pseudo-tasks to measure how well these models represent prosodic information. Third, we evaluate the utility of these models for the prediction of future pitch and energy. Fourth, we probe the pre-trained models to see in which layers prosodic information is likely represented.

Our contributions are: 1) The finding that pre-trained SSL models indeed can provide value for prosody-intensive tasks, often reaching state-of-the-art performance. 2) Results for 15 recent SSL models span different model architectures and pre-training objectives. 3) Analysis of the representation of prosody in SSL models, including layerwise analysis. 4) An open-source evaluation framework, SUPERB-prosody, examines the prosodic prowess of SSL models.

B. Related work

The most directly relevant study [162] aimed primarily to evaluate a model for producing de-identified representations of speech, but includes three aspects that are very relevant to our research question. First, for evaluation purposes, several “spoken language understanding” tasks were selected, of which three of these were both pragmatics-relevant and prosody-intensive. Second, six pre-trained models were tested against this task set, showing various levels of performance. Third, probing their own model, VQP, provided evidence that it was encoding, to some extent, several prosodic features. Taken together these results suggest that the answer to our research question is yes, but the case is not settled for two reasons. First, neither their performance results nor their probing results were compared against non-pre-trained baselines, leaving open the question of whether the pre-trained models were in fact providing any benefit. Second, many aspects of their methods are unclear, and no code is available to enable replication. Thus, we need further investigation, and an open and transparent evaluation framework for SSL models. Very recent work has found that SSL models are helpful for predicting some perceptions of speaking style, but require additional downstream sequence-modeling layers for best performance [163].

Evaluation benchmarks have been critical in supporting and evaluating the rise of SSL in the speech field. NOSS [5] is a benchmark for non-semantic downstream tasks. SUPERB [1] broadly examines SSL models for content, speaker, semantic and paralinguistic aspects, demonstrating that SSL models generalize across diverse downstream tasks. SUPERB-SG [2] enhances the SUPERB benchmark with more challenging semantic and generative tasks. SLUE [30], another recent benchmark, targets spoken language understanding tasks. However, up to now, the speech community lacks an evaluation benchmark/framework to measure the prosodic utility of SSL models.

Analysis of speech SSL models has recently received significant attention, with previous works examining various aspects [17], [162], [164], [165]. [17], [165] mainly focus on analyzing lexical information in SSL models. [164] investigate acoustic, syntactic, and semantic characteristics, but they only

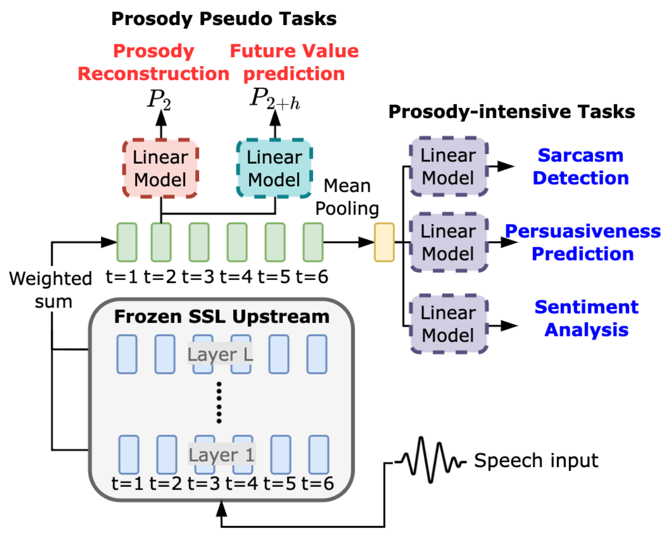


Fig. 19: Diagram of `SUPERB-Prosody` framework. We extract the hidden representations from a frozen SSL model, and lightweight linear models are used for each downstream task. P_i means the value of the rule-based prosodic feature at time frame i.

experiment on two SSL models and probe only by utterance-wise regression tasks. Accordingly, more focused analysis is needed, especially regarding prosodic information.

C. *The SUPERB-Prosody Framework*

This section introduces our tasks — classification, prosody reconstruction, and future value prediction task — and our evaluation framework, including the upstream/downstream setup.

1) Classification tasks: To evaluate the pragmatic and prosody-related abilities of pretrained models, we need a set of "prosody-intensive" tasks, that is, tasks where it is known that prosodic abilities are useful. We chose three well-curated, open-source, utterance-classification datasets, involving sentiment, sarcasm, and persuasiveness. To briefly describe each task:

Sentiment Analysis (SA) involves detecting the degree of positive or negative feeling in an utterance. While sentiment and emotion are often conflated, and similar methods may work for both tasks, sentiment is less visceral and of greater practical importance. Specifically we chose the MOSEI [174] corpus, in which each utterance is labeled from -3 to 3, representing the degree of sentiment. Following previous works, we experiment with two settings: binary classification, with the dataset split by the labels in $[-3, 0)$ and $(0, 3]$, and seven-category classification. The evaluation metric is accuracy.

Sarcasm Detection (SarD) is perhaps the most obviously prosody-intensive task, as a mismatch between the lexical content and the prosodic message is frequently the major marker of sarcasm. We chose the MUStARD [175] corpus, in which each utterance has a label of 0 or 1, for sarcasm or non-sarcasm. We follow the speaker-dependent setup in the original paper [175], using five-fold cross-validation for evaluation. The evaluation metric is the F1 score.

Persuasiveness Prediction (PP) is the task of detecting whether a presentation is likely to be convincing to others. Correlates include how pleasant the speaker's voice is and their perceived confidence and dominance, all of which involve prosody. We chose the POM [176] corpus. The labels are 0 or 1, for persuasive or non-persuasive. The evaluation metric is accuracy.

2) Prosody reconstruction: Prosody Reconstruction (ProP) is a pseudo-task designed to test whether SSL models embed specific prosody features in their hidden representation. As our target, we chose the two most commonly used prosody features, pitch and energy. Given the SSL features, we use a lightweight linear downstream model to predict each. The pitch is represented in log scale, using as targets the values computed by pYAAPT[16]. For energy, we use the librosa toolkit[17], again using a log scale. As data, we use LibriTTS [177], a multi-speaker text-to-speech dataset, and for both features, the evaluation metric is the Mean Square Error (MSE) of the differences. No loss is computed for unvoiced frames, that is, frames where pYAAPT detects no pitch.

3) Future value prediction: Future Value Prediction (FVP) is designed to test whether the output from SSL models can predict future prosody. The task setting is similar to the prosody reconstruction task, using pitch and energy features as the prediction targets and using LibriTTS dataset with MSE objective. The task is, given the information from $t = 1$ to the current frame i, to predict the value at $i + h$, where h is the prediction horizon. Four prediction horizons h, namely 0.12, 0.24, 0.50, and 1.00 seconds, are used. Because most SSL speech models are not causal (due to the inclusion of either self-attention or bi-directional connections), we only test causal SSL models or attention-based SSL models for which we can apply an attention mask to avoid cheating with future information[18]. The evaluation metrics are MSE.

4) Evaluation framework: As suggested by Figure 19, our framework consists of (1) an upstream SSL speech model and (2) a linear downstream model for probing. Following the procedure of `SUPERB` [1], the parameters of the upstream models are fixed for all the downstream tasks. For each frame of the input, we extract the representations $\mathbf{x_i}$ from each hidden layer i of the upstream model, and we aggregate those hidden representations per utterance into $\mathbf{y} = \sum_i^L w_i \mathbf{x_i}$ where the w_i are trained per task. The resulting $\mathbf{y}$, a two-dimensional matrix (time $\times$ aggregated features), is the input for the downstream model.

Upstream models – Speech SSL models: The SSL models tested are summarized in Table XVIII. These are a diverse collection, including modified CPC [170], APC [56], VQ-APC [166], NPC [167], TERA [169], vq-wav2vec [172], DistilHuBERT [44], HuBERT [78], wav2vec [171], wav2vec 2.0 [127], and WavLM [173]. The selected SSL models span

[16]http://bjbschmitt.github.io/AMFM_decompy/pYAAPT.html

[17]https://github.com/librosa/librosa

[18]We exclude WavLM for this task because it uses a gated relative position bias, so simply modifying the attention mask would cause a model mismatch between training and inference.

Model	Network	#Params	Stride	Input	Corpus	Pretraining
FBANK	-	0	10ms	waveform	-	-
APC [56]	3-GRU	4.11M	10ms	FBANK	LS 360 hr	F-G
VQ-APC [166]	3-GRU	4.63M	10ms	FBANK	LS 360 hr	F-G + VQ
NPC [167]	4-Conv, 4-Masked Conv	19.38M	10ms	FBANK	LS 360 hr	M-G + VQ
Mockingjay [168]	12-Trans	85.12M	10ms	FBANK	LS 360 hr	time M-G
TERA [169]	3-Trans	21.33M	10ms	FBANK	LS 960 hr	time/freq M-G
modified CPC [170]	5-Conv, 1-LSTM	1.84M	10ms	waveform	LL 60k hr	F-C
wav2vec [171]	19-Conv	32.54M	10ms	waveform	LS 960 hr	F-C
vq-wav2vec [172]	20-Conv	34.15M	10ms	waveform	LS 960 hr	F-C + VQ
DistilHuBERT [44]	7-Conv 2-Trans	23.49M	20ms	waveform	LS 960 hr	KD
wav2vec 2.0 Base [127]	7-Conv 12-Trans	95.04M	20ms	waveform	LS 960 hr	M-C + VQ
wav2vec 2.0 Large [127]	7-Conv 24-Trans	317.38M	20ms	waveform	LL 60k hr	M-C + VQ
HuBERT Base [78]	7-Conv 12-Trans	94.68M	20ms	waveform	LS 960 hr	M-P + VQ
HuBERT Large [78]	7-Conv 24-Trans	316.61M	20ms	waveform	LL 60k hr	M-P + VQ
WavLM Base [173]	7-Conv 12-Trans	94.68M	20ms	waveform	LL 60k hr	M-P + VQ
WavLM Large [173]	7-Conv 24-Trans	316.62M	20ms	waveform	Mix 94k hr	M-P + VQ

TABLE XVIII: SSL models examined. #Params includes parameters for both pre-training and inference. LS = LibriSpeech and LS = LibriLight. For the pre-training methods, VQ = vector quantization, F = future, M = masked, G = generation, C = contrastive discrimination, P = token prediction/classification, and KD = knowledge distillation.

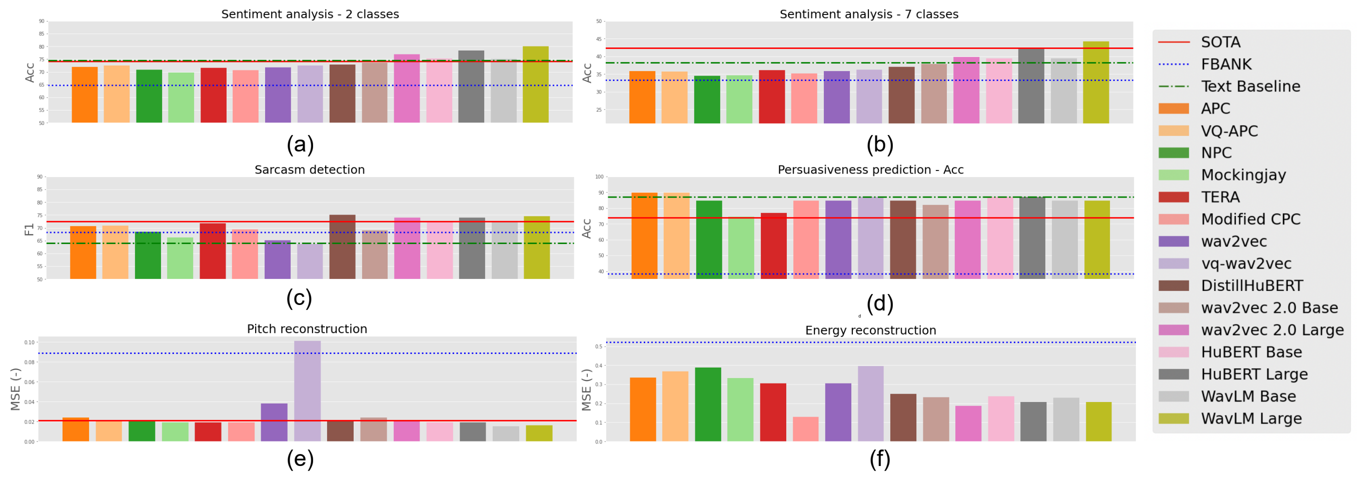


Fig. 20: Results for Sentiment Analysis (SA), Sarcasm Detection (SarD), Persuasiveness Prediction (PP), and Prosody Reconstruction (PR). State-of-the-art (SOTA) values are from [174] for SA, [178] for SarD, [179] for PP, and REAPER [180] for pitch reconstruction. Text-only baselines are only for the prosody-intensive downstream tasks, and thus not available for PR.

different network architectures and pre-training objectives.

Downstream models – linear probing model: For the classification tasks, SA, SarD, and PP, the representation $\mathbf{y}$ is mean-pooled along the time axis, forming a dense vector of dimension (time × aggregated features) to (aggregated features). This vector is then fed to a simple linear model to project from model dimension to 1. The training objective is Cross-Entropy Minimization.

In ProR and FVP, the goal is to predict the fine-grained prosodic information. We use the *frame-level representation* from each time step of $\mathbf{y}$ as the input, and the downstream model is a linear model, projecting from model dimension to 1. MSE minimization is the training objective. We try multiple learning rates for each task ($[1e^{-2}$, $1e^{-3}$, $1e^{-4}$, $1e^{-5}$, $1e^{-6}]$), and report the best performance. The training step is 3000 for SarD and 50000 for the other tasks.

Baselines: As a baseline feature set, we use "FBANK," the 80-dimensional Log Mel Filterbank features with delta and delta-delta features (240 dimensions in total), chosen because this is known to work well for many speech tasks. For the classification tasks, FBANK features are average-pooled across each utterance. For pitch reconstruction, we also compare the performance of another high-quality off-the-shelf pitch extractor, Talkin's REAPER. For future value prediction (FVP) we design a baseline, FBANK + RNN, which feeds filterbank features from $t = 1$ to i into a one-layer uni-directional Recurrent Neural Network (RNN) with 128 hidden size to predict the value at $t = i + h$.

As an additional point of comparison, we also explore the text-only performance for each classification task. Since all datasets contain ground truth speech transcription, we take these transcriptions as the input data. As the NLP model we use the pre-trained RoBERTa [181] for SarD and SA, and Longformer [59] for PP[19]. We follow the typical method to fine-tune the pre-trained NLP model: extract the sentence embedding from the [CLS] token's embedding, and feed it to a linear classification model. No parameters are frozen during fine-tuning. We vary the learning rate ($[1e^{-3}$, $1e^{-4}$, $1e^{-5}$, and $1e^{-6}]$) and report the best performance.

D. Main Results

1) SSL models perform well on prosody-intensive tasks: The experimental results for SA, SarD, and PP are shown in Figure 20. In Figure 20 (a) and (b), we can see that all SSL models yield better SA performance than the baseline FBANK features in both 2-label and 7-label evaluations. The large SSL models, wav2vec 2.0, HuBERT, and WavLM, even improve on the state-of-the-art (SOTA) performance [174] in the 2-label setup. In the 7-label setup, WavLM Large outperforms audio-only SOTA performance. Around one-third of the SSL models show better performance than the text-only baseline, confirming the value of acoustic information for SA.

Figure 20 (c) shows the SarD result. Although Mockingjay, wav2vec, and vq-wav2vec show inferior performance to the baseline FBANK, the other SSL models do well, with Distil-HuBERT, HuBERT, and WavLM improving on the previous audio-only SOTA [178]. In SarD, all acoustic models, both SSL models and FBANK, outperform the text-only baseline, confirming that SarD requires acoustic information beyond content for prediction.

Lastly, for the PP results, Figure 20 (d), shows that all SSL models yield better performance than the FBANK features and the previous audio-only SOTA [179]. APC, VQ-APC, vq-wav2vec, and HuBERT obtains superior or equal accuracy to the text-only baseline.

2) SSL models encode prosodic information: While the results above suggest that the SSL models are encoding prosodic information, there is also direct evidence from the ProR and FVP tasks, as seen in Figure 20 and Table XIX.

For PR, the two features, pitch and energy, have slightly different results. In pitch reconstruction, Figure 20 (e), all the SSL models perform better than baseline FBANK except for vq-wav2vec. Several SSL models surpass REAPER performance, with the WavLM the best. As for energy reconstruction, Figure 20 (f) shows that all SSL models greatly improved over baseline FBANK. Although generation-based SSL models perform well on pitch reconstruction, they did relatively worse on energy reconstruction. On the other hand, the SSL models pre-trained by masked contrastive discrimination/token prediction show strong performance on both pitch and energy reconstruction. Overall, we observe that SSL models indeed encode prosodic information.

FVP is more challenging than ProR since it requires the model to capture both global and local prosodic information for successful future prediction. From Table XIX, we see, unsurprisingly, that the larger prediction horizons make prediction harder. For pitch, HuBERT Large gets the best performance, outperforming other SSL models and baseline FBANK+RNN at all four horizons. Although the pre-training objectives of APC, modified CPC, and wav2vec involve future generation/discrimination, they still result in inferior performance to wav2vec 2.0 and HuBERT. As for future energy prediction, only wav2vec 2.0 and HuBERT consistently outperform baseline FBANK. In general, we observe that some SSL models are not good at FVP, but wav2vec 2.0 and HuBERT outperform FBANK by a large margin. This result suggests that wav2vec 2.0 and HuBERT do have the capability to encode and summarize relevant prosodic information.

E. Further Analysis

1) Layerwise contribution analysis: In order to estimate the contribution of each layer, we consider two factors. First, we use the weight w_i from each layer (through the weighted-sum mechanism) to the downstream model. Second, because typical feature magnitudes may vary, we also consider the values of the hidden representation $\mathbf{x}$ in each layer, as measured by the L2-norm of feature values for the testing data. After getting the whole L2-norm features for each layer for each sample, we take the mean across samples to get the feature magnitude estimation. The contribution for each layer is then defined as: $c_i = ||\mathbf{x_i}||_2 \times w_i$, where i means the layer number.

The results are shown in Figure 21. From Figure 21(a), we can see that for most SSL models, the contribution is strongest in the first few layers for both pitch and energy reconstruction. This shows that SSL tends to best represent prosodic information in the front. One exception is Mockingjay, where the largest contribution is located in the last layer. Because Mockingjay's pre-training objective is to reconstruct frame-wise features, it is unsurprising that the later layers contain a good representation of low-level prosody.

However, for the classification tasks SarD and PP, Figure 21 (c) and (d) shows that the distribution of layer contributions is smooth, suggesting that both tasks need information across multiple layers. As previous work has shown that later layers represent more content information [17], [173], this suggests that both prosodic and content information are needed for SarD and PP. As for SA, in Figure 21(e), we observe a high contribution value in the latter layers for the high-performing models HuBERT and WavLM. This suggests that SA might only require content information to perform well.

2) Feature integration: To further examine whether the encodings in the first few layers bring the most benefit, we designed a new experiment for three prosody-intensive tasks using wav2vec 2.0 and HuBERT. We compare two settings, concatenation of 1) the features from the first two layers and the best layer we discovered[20], and 2) the best layer with its two neighbor layers. These concatenated features are passed to the downstream model. The downstream model size of the two settings is the same, so we can compare the performance fairly. If the first setting is better than the second setting, this would indicate that early-layer (low-level) information indeed most benefits the final results.

[19]Longformer is based on RoBERTa, but can accept up to 4096 tokens (versus 512 tokens for RoBERTa), and is used here because the transcriptions of PP are too long for RoBERTa.

[20]The best layer is determined by the contribution analysis above.

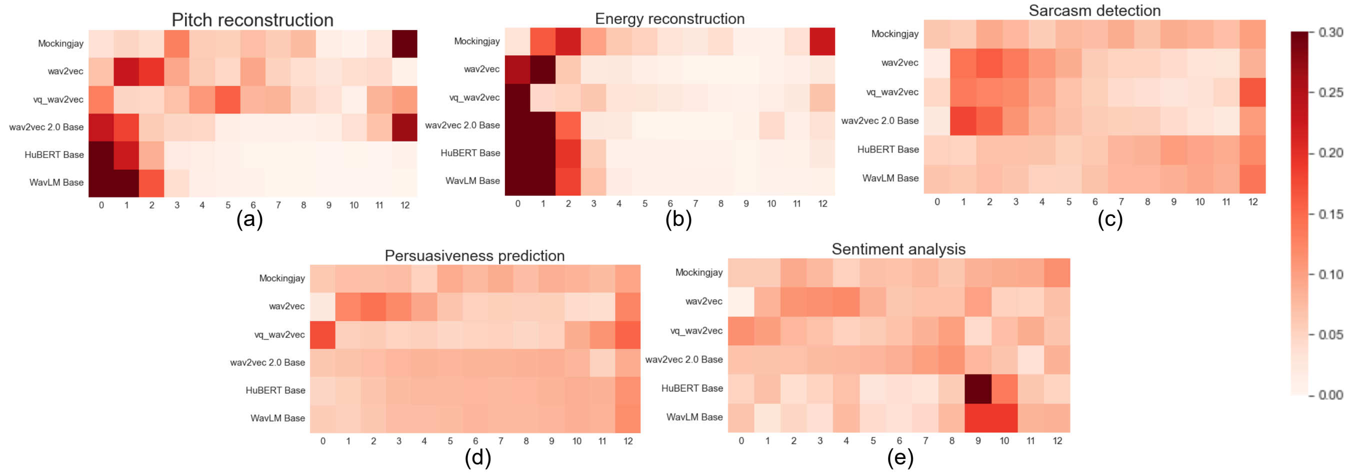


Fig. 21: The contribution analysis for each task. The darker the color, the higher the contribution. We only include models which have 12 layers of representations.

Model	Pitch w/ Prediction Horizon (s)↓				Energy w/ Prediction Horizon (s)↓			
	0.12	0.24	0.50	1.00	0.12	0.24	0.50	1.00
FBANK + RNN	0.049	0.104	0.142	0.157	0.52	1.42	2.06	2.46
APC	0.033	0.043	0.052	0.053	0.91	1.46	1.98	2.37
modified CPC	0.038	0.051	0.062	0.065	0.79	1.37	1.94	2.47
wav2vec	0.053	0.064	0.075	0.075	0.68	1.15	1.70	2.31
Mockingjay	0.069	0.077	0.081	0.099	0.48	1.92	2.17	2.43
wav2vec 2.0 Base	0.038	0.047	0.047	0.049	0.44	0.85	1.24	1.49
wav2vec 2.0 Large	0.035	0.039	0.045	0.046	0.43	0.80	1.23	**1.31**
HuBERT Base	0.029	0.036	0.041	0.042	**0.39**	**0.70**	**1.12**	1.42
HuBERT Large	**0.025**	**0.027**	**0.028**	**0.037**	0.41	0.77	1.21	1.36

TABLE XIX: The MSE loss for FVP. Lower values indicate better prediction, of future pitch or energy.

	SarD F1↑		PP Acc↑		SA Acc↑	
Layer selection	(0,1,<u>12</u>)	(10,11,<u>12</u>)	(0,1,<u>12</u>)	(10,11,<u>12</u>)	(0,1,<u>8</u>)	(7,<u>8</u>,9)
wav2vec 2.0 Base	**70.6**	66.3	**81.3**	81.3	73.6	**74.0**
HuBERT Base	**72.0**	70.0	**85.8**	83.8	75.2	**76.2**

TABLE XX: Experimental results for layer-limited feature integration for SarD, PP, and SA. The best layer, underlined, is chosen based on the contribution analysis.

Method	ZH-p↓	PO-p↓	ZH-e↓	PO-e↓
FBANK	0.050	0.096	0.849	0.477
vq-wav2vec	0.022	0.097	0.490	0.339
wav2vec 2.0 Base	0.012	0.039	0.338	0.160
HuBERT Base	0.009	0.042	**0.232**	**0.149**
WavLM Base	**0.008**	**0.019**	0.233	0.192

TABLE XXI: MSE for prosody reconstruction, showing cross-lingual transferability. p = pitch, e = energy.

The results are shown in Table XX. In SarD and PP, we see the integration of the first two layers yields better performance, which means the low-level information improves the modeling of SarD and PP. Yet the use of low-level information does not improve SA performance, suggesting that SA may rely on content rather than just low-level information.

3) Cross-lingual transferability: Further, we did a preliminary investigation of whether SSL models may have cross-lingual transferability for prosodic information, using the ProR task as in previous experiments: specifically, attempting pitch and energy reconstruction. While the pitch of one frame is primarily a physical phenomenon, in context there may be language dependencies. All the SSL models being pre-trained in English, we try this for Mandarin and Polish, using data from AISHELL-3 [182] multilingual LibriSpeech (MLS) [183], respectively.

Table XXI summarizes the results. We note that WavLM is good at pitch reconstruction, and HuBERT is superior at energy reconstruction.

IX. SSL for Low-Resource South African Languages

A. Introduction

Automatic speech recognition (ASR) systems have recently achieved great accuracy in well-resourced languages [127], [184]. This accuracy has been achieved thanks to the modelling improvements and thousands of hours of manually transcribed speech. However, ASR performance in low-resource languages is still lacking due to limited amounts of transcribed data. (*i*) Cross-lingual transfer, (*ii*) self-supervised pre-training and (*iii*) semi-supervised training have been proposed to solve this problem in the past [184]–[186]. (*i*) In cross-lingual transfer, we first train an acoustic model for a set of well-resourced languages and then transfer the parameters of the acoustic model to the new low-resource language. Finally, we train the final acoustic model by fine-tuning a subset of the parameters on a small amount of available transcribed data [185]. (*ii*) In self-supervised pre-training, the acoustic model parameters are pre-trained on thousands of hours of untranscribed data with a self-supervised loss. These parameters are subsequently fine-tuned on the transcribed training data with a standard supervised loss, for example, a Connectionist Temporal Classification (CTC) loss [132], or a lattice-free maximum mutual information (LF-MMI) loss [187]. Popular examples of pre-trained self-supervised models include wav2vec 2.0 model [127] and its multilingual version XLSR-53 model [184]. (*iii*) In semi-supervised training, we start by training a seed model on the transcribed training data. Then, we use this acoustic model with a language model to produce pseudo-labels for the unlabelled data. These pseudo-labels are then used as training targets for fine-tuning the acoustic model [186]. Overall, cross-lingual transfer and self-supervised pre-training can be used to train a robust seed acoustic model to improve the impact of semi-supervised training [188], [189]. The limiting factor of semi-supervised training is the quality of the language model used to produce the pseudo-labels [190]. A good language model typically needs to be trained on large amounts of text data; in our experience, at least several hundred million words are needed to train a strong general language model. Unfortunately, such large amounts of text are not available online for many low-resource languages [191]. Furthermore, the language modelling in these low-resource languages is further exacerbated by the presence of code-switching [192].

In this work, we study how self-supervised training and semi-supervised training can be used to leverage untranscribed audio data in underrepresented languages. In semi-supervised training, the initial acoustic model is initialized with a flat-start initialization or with cross-lingual transfer learning with spectral representations as inputs. We propose another initial acoustic model which is trained on learned multilingual speech representations. In particular, we explore how 200 hours of untranscribed South African soap operas data can help to improve the initial acoustic model trained only on 12.7 hours of manually transcribed code-switched speech between four South African languages and English [26]. We show that continued pre-training of self-supervised XLSR-53 model [184] on the 200 hours of untranscribed speech data is the best strategy for improving the initial model when we have access only to a weak language model. Furthermore, when we combine the continued self-supervised pre-training with semi-supervised training, we gain additional slight improvement without a strong language model.

B. Related Work

Many approaches have been proposed to tackle the training data scarcity for low-resource languages [184]–[186]. In this section, we summarise two of them; self-supervised pre-training and semi-supervised training. In addition, because of the very specific nature of the South African linguistic landscape, we briefly review literature about ASR for South African languages.

In speech processing, models trained with self-supervised learning (SSL) recognize latent speech representations from waveforms with a contrastive loss, such as InfoNCE [138]. Thanks to SSL, only unlabelled data is needed to train a model. These pretrained models work well on a variety of downstream tasks when fine-tuning them with the addition of a projection layer on top of them.

For English, wav2vec2.0 model [127] has been proposed and has an architecture composed of a convolutional network followed by a Transformer network [111] and finally a quantization module. The model learns by predicting the best sequence of acoustic units from a set of codebooks with a contrastive loss. Furthermore, an auxiliary diversity loss makes sure that all code words in the codebooks are used. Alternatively, HuBERT models [78] separate the creation of the codebook (built with the k-means clustering algorithm) from the neural network parameters' learning (via masked predictions). For speech recognition, this fine-tuned model is trained with the supervised CTC loss function [132]. In low-resource speech recognition, pre-training models based on wav2vec2.0 model with a set of various languages as training data, as it is done for XLSR-53 and XLS-R models, proves to be preferable than using a pre-trained English model [184], [193], [194]. Additionally, using continued pre-training with unlabelled in-domain data to further train the pre-trained model can improve the results once the fine-tuning is done [195], [196]. This procedure of continual learning (CL) [197] alleviates the domain mismatch between the previous datasets and the current dataset while retaining knowledge from the previous training datasets.

Semi-supervised training (SST) uses a seed acoustic model trained on small amounts of manually transcribed data to produce pseudo-labels for the unlabelled data [186]. Traditional approaches for semi-supervised training use one-best transcripts as pseudo-labels. However, these transcripts might contain transcription errors negatively affecting the acoustic model. Therefore, it is necessary to perform some form of confidence filtering to discard utterances with noisy transcripts [198]. Another option is to use lattices as pseudo-labels [199]. This approach circumvents the issue of erroneous one-best transcript by representing possible alternative transcriptions and their corresponding uncertainties within the lattice. Popular way of performing lattice-based semi-supervised training is to use lattice-free maximum mutual information criterion [187], [199]. It was also shown that semi-supervised training is possible even with a very weak seed acoustic model if provided with a strong language model [190]. In the past, semi-supervised training have been successfully used for many low-resource languages including building ASR

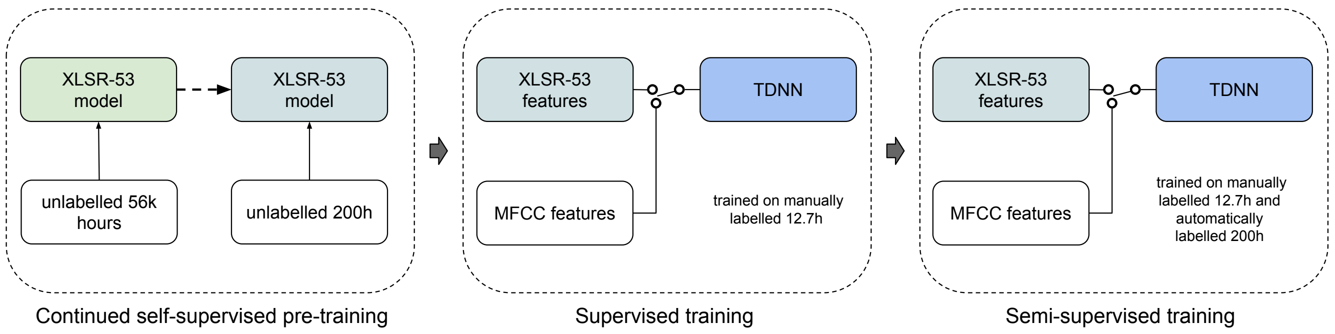


Fig. 22: Diagram of the combination of self-supervised and semi-supervised trainings for 12.7 hours of labelled South African data and 200 hours of unlabelled data from other South African soap operas. XLSR-53 is further trained with the 200 hours of unlabelled data (continued self-supervised pre-training) and used as features extractor to train a TDNN. The TDNN model can also be trained with MFCC features. The obtained TDNN model is the seed model for the subsequent semi-supervised training with both labelled and unlabelled data.

for six languages spoken in India during the MUCS 2021 challenge [200]. This was possible because of having access to large text corpus for all these languages extracted from CommonCrawl.[21]

ASR systems for South African languages from soap operas have been proposed, with a main focus on dealing with the lack of in-domain data for these languages [201]–[203]. An effective way to work with under-resource languages is to use training data from other related languages. For the acoustic models of the Soap Operas data, multilingual training with close-related languages has indeed been shown to be effective for transfer learning [201]. Semi-supervised learning has also been applied to take advantage of some available unlabelled data but the code-switching in the transcriptions data makes it hard to have a strong language model when only few code-switching texts are available compared to the monolingual texts. Consequently, studies mainly address how to segment unlabelled data, either manually [202] or automatically [203]. Both types of segmentation have equally succeeded to improve the accuracy of the speech recognition of the South African languages, with the use of a five-lingual language model. Following these works, we focus on building a five-lingual model and taking advantage of some unlabelled data for self- and semi-supervised training.

C. Improving the Acoustic Model with Unlabelled Data

The performance of ASR systems in low-resource languages is limited by the small amount of manually transcribed data. In this section, we describe how we collected untranscribed speech and how we used it to improved the performance of ASR systems using self-supervised training and semi-supervised training. The whole pipeline is illustrated in Figure 22. For efficiency reasons, we used continued self-supervised pre-training to continue training a multilingual self-supervised pre-trained model on our untranscribed speech data. This allowed us to use multilingual speech features to train a better seed acoustic model for the semi-supervised training.

1) Dataset: We conducted experiments on the South African Soap Operas dataset [26]. This dataset contains 14.3 hours of labelled speech with people alternating between four Banto languages and English, in language-balanced sets, making it a code-switched dataset We used the official training (12.7 hours) and test splits (1.3 hours). The represented pairs of languages are: English-Sesotho (eng-sot), English-Setswana (eng-tsn), English-isiXhosa (eng-xho) and English-isiZulu (eng-zul).

We also collected unlabelled speech. In our previous data collection pipeline [190], [200] we used the most common n-grams in YouTube queries to identify videos for collecting unlabelled data for low-resource languages. Since this method can be noisy and collect music videos or videos from other languages we filtered these videos using mean confidence and speaking rate thresholds. Even with filtering, this data collection pipeline proved to be too noisy for South African languages, therefore we opted for a different selection scheme. We identified South African soap operas on Wikipedia and we downloaded trailers for these soap operas. This method ensured that the crawled data is in-domain and contains relevant languages together with other South African languages used in the soap operas. In total we were able to collect 200 hours of raw recordings which were segmented with WebRTC VAD[22] for further processing.

2) ASR model training: Our multilingual five-lingual (four Bantu languages + English) South African acoustic models used either 40-dimensional MFCC features or 1024-dimensional XLSR-53 features as inputs. Both types of models were trained using Kaldi toolkit [**?**] and had the same alignments obtained with a standard GMM model. We used a CNN-TDNN architecture with 16.7M parameters for the acoustic model using MFCC features and a TDNN-F architecture with 23.7M parameters for the acoustic model using XLSR-53 features. The difference in sizes between models comes from using LDA on features of more dimensionality. We did not use convolutional layers with the XLSR-53 features because the XLSR-53 itself acts as a convolutional front-end. Both types

[21]https://www.statmt.org/ngrams/

[22]https://github.com/wiseman/py-webrtcvad

of models were trained with the LF-MMI criterion [187] for six epochs on speed-perturbed training data. The pronunciation dictionaries for the four South African languages and English were built using the NCHLT dictionaries and corresponding grapheme-to-phoneme sets of rules [204]. Merging the five dictionaries, the final dictionary has 88 phones, including 12 phones shared by all five languages. The vocabulary size is 18.8k tokens. Furthermore, we used a 3-gram language model trained only on the 155k tokens of available training transcripts.

To improve the performance of the model using MFCC features, we transferred parameters from a model trained either on the NCHLT dataset [204], which contains read speech from 11 official South African languages, or on the MGB dataset [205], which contains British English broadcast speech. To deal with the mismatch in acoustic units, we replaced the final layer of the pre-trained model and retrain the whole model. Based on our experience with fine-tuning hybrid models, we used 10-times smaller learning rate than during training with flat-start initialization, for all layers except for the final layer.

3) Continued self-supervised pre-training: To adress the lack of a strong language model, we used the collected 200 hours of South African speech unlabelled data for self-supervised training. However, since training the self-supervised models with a flat-start initialization is computationally expensive, we only performed continued pre-training of a self-supervised model using these 200 hours with the contrastive loss of wav2vec2.0 [127]. As pretrained model, we chose to use the multilingual model XLSR-53, which is based on wav2vec2.0 LARGE architecture and is trained on 53 languages for a total of 56k hours. Instead of using the pre-trained model directly as an acoustic model, we used it as a multilingual bottleneck feature extractor, extracting the representations from the last layer with the S3PRL toolkit [1]; a standard hybrid TDNN model [187] was then trained on these features.

We experimented with various pre-trained self-supervised models and extracted the features from different layers and we found that the last layer of XLSR-53 worked best in our scenario.[23] We up-sampled the extracted features to the standard Kaldi frame rate of 100 frames per second. We performed the continued self-supervised pre-training of the XLSR-53 model with the fairseq toolkit [79] using the 200 hours of unlabelled data. We kept the hyperparameters identical to the ones used to train wav2vec2.0 LARGE [127] on Librivox. We continued pre-training for 8k iterations equivalent to 67 epochs, with a batch size of at most 1.4M tokens and we used gradient accumulation to simulate training on a bigger batch size using only two Tesla V100 GPUs.

4) Semi-supervised training: In semi-supervised training, we first (*i*) trained a seed acoustic model on the small amounts of manually labelled data. This acoustic seed model was subsequently used with a language model trained on available text corpora to (*ii*) produce pseudo-labels for the unlabelled speech. In this report, we trained the seed acoustic model on the training part of the South African Soap Operas dataset [26]. The semi-supervised training was done using the lattice-free maximum mutual information (LF-MMI) training criterion [187] and followed the semi-supervised training approach proposed in [199]. We used the semi-supervised training pipeline described in [190], [200]. We performed semi-supervised training on a combination of the manually transcribed training data and the unlabelled 200 hours. We decoded this data with seed models trained on the 12.7 hours of the labelled data with cross-lingual transfer from MGB or trained on the 12.7 hours of labelled data with features from XLSR-53 with continued pre-training on the unlabelled 200 hours. We used the decoded lattices as pseudo-labels for semi-supervised training. Note that, we filtered the unlabelled data with the minimum mean recording confidence threshold of 0.8 and the minimum speaking rate threshold 1.25 words per second prior to the semi-supervised training as in [200]. Similar to the supervised training, we trained the models for six epochs using LF-MMI.

D. Results

1) Baselines acoustic models: In the first set of experiments, we assessed how well cross-lingual transfer works for South African languages. We compared training the acoustic model with flat-start initialization, denoted as (1) in Table XXII, using cross-lingual transfer from a model trained on the NCHLT dataset (2) and using cross-lingual transfer from a model trained on the MGB challenge dataset (3). Our results demonstrated that a domain-match and data diversity in the MGB dataset (3) is more important than training on additional data for the target languages (2) with overall word error rates (WER) of 50.8% and 52.0% respectively. This is because the NCHLT data contains clean read speech, which is acoustically very different from the speech in the Soap Operas dataset. In addition to the noticeable background noise in the Soap Operas dataset, the speech characteristics are also very different, for example the average speaking rate in the NCHLT dataset is three times slower than in the Soap Operas dataset. Subsequently, we compared the model trained with cross-lingual model from MGB (3) with a model using the multilingual self-supervised hidden representations from XLSR-53 as input features (5). We found that these two approaches achieved similar WER of 50.8% and 50.9%. The benefit of (3) is that it is trained on very well matched data and (5) benefits from being pre-trained on large amounts of multilingual data. We hypothesize that (3) would improve if cross-lingual transfer was done from a model trained on a diverse multilingual dataset, not only on British English dataset.

2) Semi-supervised training vs self-supervised pre-training: In the next set of experiments, we showed how 200 hours of unlabelled data can help improve the performance. We found that semi-supervised training with (3) as a seed model achieves worse results (4), WER of 49.3%, than continued self-supervised pre-training of the multilingual model XLSR-53 (6), WER 46.5%. It is worth noting that it was crucial to combine the labelled Soap Operas training data with the

[23]Note that XLS-R [193] achieved better results than XLSR-53 in our preliminary experiments. However due to its size we were not able to continue pre-training it and therefore we left it for future work.

		eng-sot	ent-tsn	eng-xho	eng-zul	all
(1)	CNN-TDNN baseline	55.9	46.6	63.9	56.6	54.7
(2)	CNN-TDNN with cross-lingual transfer from NCHLT	52.8	44.5	60.4	54.1	52.0
(3)	CNN-TDNN with cross-lingual transfer from MGB	50.1	43.8	60.8	52.8	50.8
(4)	+ semi-supervised training on 200h of unlabelled data	48.3	42.6	59.2	51.1	49.3
(5)	TDNN-F using XLSR-53 bottleneck features	49.8	45.1	61.7	51.6	50.9
(6)	+ continued self-supervised pre-training on 200h of unlabelled data	45.5	40.9	56.2	47.6	46.5
(7)	+ semi-supervised-training on 200h of unlabelled data	44.3	38.5	56.5	46.4	45.2

TABLE XXII: Word Error Rate (WER) on the test set of the South-African soap opera dataset. The results are split by language pairs: English-Sesotho (eng-sot), English-Setswana (eng-tsn), English-isiXhosa (eng-xho) and English-isiZulu (eng-zul).

200 hours of unlabelled data to make semi-supervised training work. Without the labelled Soap Operas data the semi-supervised model was worse than the seed model. When we combined the continued self-supervised pre-training with semi-supervised training we achieved further gains (7), WER of 45.2%. In total, we reduced the WER by 17% relative compared to the baseline model trained with flat-start initialization (1).

To overcome the code-switched aspect of the Soap Operas dataset, we tried using a language model with South African monolingual texts crawled from the web and MGB [205] transcriptions but the resulting WER were worse than using the in-domain language model trained only on the transcriptions of the Soap Operas dataset. This shows that the language model domain match is very important, especially since this in-domain language follows clear transcription conventions consistent with the test transcription. This explains why the in-domain language model is stronger because not all South African languages have standardised orthography and the crawled online texts might follow different spelling rules, which negatively affects the WER. Code-switching is also a mostly unwritten phenomenon, which makes language modelling even more difficult. Furthermore, the performance of the model could be improved by fine-tuning the model for each language pair individually, but we chose not to do it since we were interested in building a unified five-lingual model. Previous works on this South African data indeed demonstrated that the semi-supervised training batch by batch yields improvement over training in a single pass, as does training bilingual acoustic models instead of a five-lingual model [202]. Improvement also comes from adding generated texts and automatic transcriptions to build a strong LM. However, this type of bilingual training with a strong LM is less practical because it requires to obtain unlabelled data with a specific type of code-switching to train the model.

X. SSL with Unsupervised ASR

A. Unsupervised ASR Open-source Toolkit

1) Introduction: Over the past decade, end-to-end (E2E) supervised ASR has achieved outstanding improvements. These achievements keep pushing the limit of ASR performance in terms of word error rate (WER). However, training the state-of-the-art model still heavily relies on a reasonable amount of annotated speech [206]–[208]. Unfortunately, labeled data is quite limited for most of the 7000 languages worldwide [209]. The fast development of self-supervised learning (SSL) could mitigate the issue to some extent by leveraging unlabeled data. This paradigm first learns the speech representations from raw audio and then fine-tunes the model on limited transcribed speech data [78], [127], [173], resulting in reducing the need for annotated speech.

However, as there is still a need for transcribed data for downstream model training, it is difficult to directly apply the system to all languages, especially those endangered languages that are extremely difficult to obtain data [210], [211]. Unsupervised ASR could be one possible direction to solve the problem where the model can be trained with more accessible unpaired speech and text data.

Wav2vec-U is the state-of-the-art UASR framework. It utilizes both SSL (i.e., wav2vec2.0) and adversarial training for the UASR task [28]. The framework is shown working not only in English, where the wav2vec2.0 [127] is trained on but also in several other mainstream languages [80], [183], [212], as well as low-resource languages [213]. This finding reveals the possibility of utilizing unsupervised learning for more languages.

The authors of wav2vec-U have released their code in FAIRSEQ [79], which greatly improves reproducibility. The implementation mainly consists of 3 steps: data preparation, generative adversarial training (GAN) [214], and iterative self-training [215] followed by Kaldi LM-decoding [216]. Along with this reproducibility direction, we develop an unsupervised ASR toolkit named ESPnet Unsupervised ASR Open-source toolkit (EURO). EURO complements the original FAIRSEQ implementation with more efficient multi-processing data preparation, flexible choices over different SSLs, and large numbers of ASR tasks through ESPnet [217]. EURO also integrates a weighted finite-state transducers (WFST) decoder using the k2 [218] toolkit for word-level recognition. K2 is the updated version of the popular ASR toolkit Kaldi [216]. It seamlessly integrates WFST and neural models implemented in PyTorch [219] by supporting automatic differentiation for finite state automaton (FSA) and finite state transducer (FST), which are commonly used in ASR as a natural representation of the model's architecture [220]. In EURO, k2 provides a compact WFST structured and efficient algorithm for decoding. With these advantages, EURO can considerably benefit the UASR study for the speech community, together with the FAIRSEQ UASR implementation.

This report first introduces the toolkit and its features. Then, we conduct experiments that explore mainstream self-supervised models as speech feature extractors for UASR

TABLE XXIII: Framework comparison of EURO with wav2vec-U

Features	Details	Fairseq	EURO
Model	Frontend	1 SSL	27 SSLs in [1]
Efficiency	Prepare	single-thread	multi-process
	Train	multi-GPU	multi-GPU
	Decode	single-thread	multi-process
Decoding	Prefix	by FlashLight [221]	self-implemented
	WFST	by PyKaldi [222]	by k2

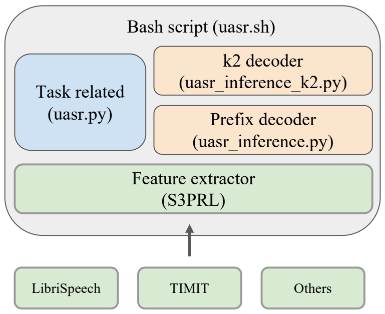
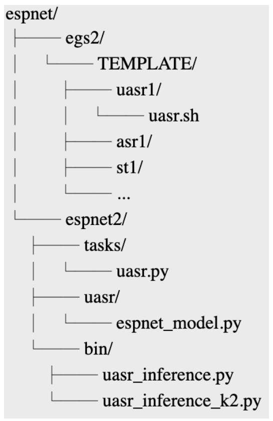

Bash script (uasr.sh)
Task related (uasr.py)
k2 decoder (uasr_inference_k2.py)
Prefix decoder (uasr_inference.py)
Feature extractor (S3PRL)
LibriSpeech
TIMIT
Others

Fig. 23: Architecture of EURO



```
espnet/
├── egs2/
│   └── TEMPLATE/
│       ├── uasr1/
│       │   └── uasr.sh
│       ├── asr1/
│       ├── st1/
│       └── ...
└── espnet2/
    ├── tasks/
    │   └── uasr.py
    ├── uasr/
    │   └── espnet_model.py
    └── bin/
        ├── uasr_inference.py
        └── uasr_inference_k2.py
```

Fig. 24: Directory of EURO

in different languages. Finally, we provide details of the hyperparameters of our experiments.

2) Related works: This section briefly compares the framework of EURO to wav2vec-U. As summarized in Table XXIII, EURO provides more flexible choices of SSL model as the acoustic feature extractor by integrating with the S3PRL toolkit [1]. With the comprehensive pipeline in the template, EURO enjoys a fast adoption to various datasets with a limited data preparation effort (less than 20 lines of code for a minimum runnable solution). Meanwhile, all the data preparation stages in EURO are designed to enable computing in parallel, which greatly minimized the preprocessing time compared to wav2vec-U. Besides of WFST decoder introduced in Sec. X-A1, EURO provides a self-implemented decoder to eliminate external dependencies.

3) Functionalities of EURO: Figure 23 shows the architecture of EURO. Similar to other ESPnet tasks, EURO includes two major components: a Python library of network training/inference and a collection of recipes for running complete experiments for a number of datasets. The library is built upon PyTorch, while the recipes offer all-in-one style scripts that follow the data format style in Kaldi [216] and ESPnet [217]. In addition, a WFST decoder is included to perform word-level recognition.

As discussed in Sec. X-A2, we follow previous works in using adversarial training to achieve UASR in EURO. To be specific, we extend the Wav2vec-U framework into our implementation.

Given a spoken utterance $\mathbf{X} \in \mathcal{D}_{\text{speech}}$, we first extract the speech representation by using a speech SSL model as the feature extractor $f(\cdot)$, resulting in a sequence of hidden representations $\mathbf{H}$. Then, the sequence $\mathbf{H}$ is passed into a preprocessor $m(\cdot)$ to form a segmented feature sequence $\mathbf{S}$, which is used for the generative adversarial network (GAN). The preprocessor $m(\cdot)$ includes three steps, including adjacent clustering pooling, principle component analysis (PCA) dimension reduction, and mean pooling. The adjacent clustering pooling utilizes the K-Means cluster IDs from the input feature $\mathbf{H}$ as guidance to merge the adjacent feature frames.

The network is mainly trained with a GAN-based loss and some auxiliary supporting losses. Given the segmented feature $\mathbf{S}$ and an unpaired phonemicized text sequence $\mathbf{Y}_u \in \mathcal{D}_{\text{text}}$, the framework includes a generator $\mathcal{G}$ and a discriminator $\mathcal{C}$ as the classic GAN framework. The generator $\mathcal{G}$, which serves as the ASR model, transcribes $\mathbf{S}$ into a phoneme sequence $\mathbf{P}$ and the discriminator $\mathcal{C}$ tries to distinguish $\mathbf{Y}_u$ from the generated phoneme sequence $\mathbf{P}$. The GAN-based loss $\mathcal{L}_{\text{GAN}}$ is as follows:

$$\mathcal{L}_{\text{GAN}} = \min_{\mathcal{G}} \max_{\mathcal{C}} \mathbb{E}_{\mathbf{Y}_u}[\log \mathcal{C}(\mathbf{Y}_u)] - \mathbb{E}_{\mathbf{H}}[\log(1 - \mathcal{C}(\mathcal{G}(\mathbf{S})))]. \tag{19}$$

To stabilize the training, three auxiliary losses are also proposed, including (a) a gradient penalty loss $\mathcal{L}_{\text{gp}}$ to sample the mixing rate of real and fake input for different steps:

$$\mathcal{L}_{\text{gp}} = \mathop{\mathbb{E}}_{\mathbf{S},\mathbf{Y}_u,\alpha\sim U(0,1)}[(||\nabla\mathcal{C}(\alpha\mathcal{G}(\mathbf{S}) + (1-\alpha)\mathbf{Y}_u)|| - 1)^2], \tag{20}$$

where α is the mixing weight sampled from a uniform distribution [223]. (b) a smoothness penalty to penalize inconsistent phoneme prediction between adjacent segments:

$$\mathcal{L}_{\text{sp}} = \sum_{(p_n,p_{n+1})\in\mathcal{G}(\mathbf{S})} ||p_n - p_{n+1}||^2, \tag{21}$$

where $p_n \in \mathbf{P}$ is the generator output distribution at the n's segment. (c) a phoneme diversity loss to prevent the generator $\mathcal{G}$ from generating the same phoneme all the time:

$$\mathcal{L}_{\text{pd}} = -\sum_n \text{Entropy}_{\mathcal{G}}(\mathcal{G}(\mathbf{S})), \tag{22}$$

that is defined by the entropy of the average generator output distribution over every frame. The final loss is defined as

$$\mathcal{L} = \mathcal{L}_{\text{GAN}} + \lambda\mathcal{L}_{\text{gp}} + \gamma\mathcal{L}_{\text{sp}} + \eta\mathcal{L}_{\text{pd}}, \tag{23}$$

where λ, γ, η are the weights for each term.

One of the major benefits of EURO compared to wav2vec-U is its tight integration with S3PRL, a toolkit for speech/audio self-supervised models, to avoid manually managing and switching between different SSL models. Born initially with the official implementation of TERA [169], S3PRL has extended further to support various SSL models as a general toolkit. Taking benefits from the SUPERB benchmark, S3PRL has kept up-to-date with the latest SSL models in the speech and audio domain. Based on the integration with S3PRL, by simply changing **one line** of the configuration, EURO can utilize up to **27** speech and audio SSLs with more than **70** of their variants.[24]

[24]The number is recorded in Oct. 2022.

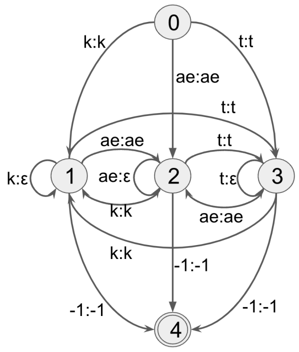


Fig. 25: H topology of phone set {k, ae, t}. It merges duplicated adjacent phones in the input sequence, e.g., (k, k, ae, t, t) → (k, ae, t). The arc with -1 is a special arc defined in k2 pointing to the final state.

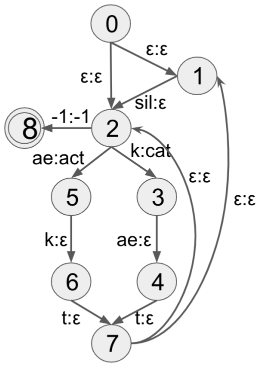


Fig. 26: L topology of lexicon {cat: k, ae, t; act: ae, k, t} in k2. It maps the phone sequence to the corresponding word allowing optional silence token (sil) between words. e.g., (sil, k, ae, t, sil) → cat.

EURO offers two methods for decoding, including a self-implemented prefix beam search method and a graph-based search method using k2.

The prefix beam search utilizes the same decoding process as the CTC prefix decoding [224], [225], but without the blank symbols. Similar to other ESPnet tasks, the decoding can be integrated with phoneme-level language models (LM) from both n-gram LMs and neural LMs.

As briefly introduced in Sec. X-A1, the graph-based search employs WFST for decoding. Different from the prefix beam search method, the graph-based search can utilize word-level LMs in the search graph, and can also be extended to word recognition. The search graph T is a composition of three functional graphs: an alignment graph H, a lexicon graph L, and a grammar graph G:

$$T = H \circ L \circ G, \tag{24}$$

where $\circ$ is WFST composition. Specifically, H merges duplicated adjacent phones; L maps sequences of phonemes to sequences of responding words; G is an n-gram word LM. Figure 25 and Figure 26 show an example of H and L implemented in k2. The lattice is generated during decoding [226], which represents the set of most likely hypothesis transcripts structured in a directed graph and can be easily integrated with neural LMs by performing lattice scoring [227], [228]. The best hypothesis is obtained by searching the best path in the lattice.

EURO follows the unified directory organization of ESPnet as shown in Figure 24. Similar to other tasks (e.g., ASR, speech translation (ST)), the recipe `usar.sh` and its related bash scripts are stored under `egs2/TEMPLATE/uasr/`. The model `espnet_model.py` and task `uasr.py` are stored at `espnet2/uasr/` and `espnet2/tasks/`, respectively. Decoding scripts `uasr_inference.py` and `uasr_inference_k2.py` are placed under `espnet2/bin/`.

The recipe in EURO follows the ESPnet2 style of task, which provides the template `uasr.sh`. The stages are defined as follows:

Stage 1-5*: Data preparation.* The initial data format starts from the Kaldi style [216], but the `text` for transcription is supposed to be unpaired with the speech data. Then, we offer two optional data preprocessing stages: speed perturbation and voice activity detection (VAD). The VAD results can be applied in silence removal as it is shown to be important for some speech corpus for wav2vec-U. After the preprocessing, all speech data is converted into a standard format, by resampling, segmentation, silence removal, and dumping from pipe-style formats.

Stage 6-7*: Text tokenization and token list generation.* Texts are converted to phoneme tokens using the graphemes to phonemes toolkit g2p-en. Tokens are collected from the training text and formed into a corresponding token list for modeling. For UASR, the unpaired text $\mathbf{Y}_u$ is fed into training in a random fashion. For efficiency purposes, we initialize a randomized text loader with the tokenized text, which is especially useful for large text data.

Stage 8-11*: LM preparation.* These stages train and evaluate LMs based on the unpaired text $\mathcal{D}_{\text{text}}$. The LMs include both neural-based LMs and N-gram LM.

Stage 12*: WFST graph construction.* This stage creates the WFST decoding graph for the k2 decoder.

Stage 13*: UASR statistics collection.* In this stage, EURO collects necessary input statistics for batching and mean-variance normalization (MVN). Optionally, the feature from frontends (i.e., S3PRL module in EURO) can be extracted at the stage to support efficient training in further stages.

Stage 14*: UASR feature preprocessing.* This stage applies the preprocessing steps discussed in Sec. X-A3, including adjacent clustering pooling, PCA, and mean pooling. The resulting segments are used for UASR training.

Stage 15*: UASR training.* This stage conducts the adversarial training as discussed in Sec. X-A3.

Stage 16*: UASR decoding.* EURO has two decoding schemes as introduced in Sec. X-A3. This stage supports both decoding methods.

Stage 17*: UASR evaluation.* The evaluation in this stage utilizes the NIST *sclite* toolkit to compute the phone error rate (PER) or the word error rate (if applicable).

Stage 18-20*: Model packing and uploading.* This stage automatically packs the trained model checkpoint for easier sharing of the pre-trained model. EURO also supports uploading models to Huggingface for model sharing.

4) Experiments: **TIMIT**: The TIMIT dataset is a popular benchmark for the UASR task [212]. It contains 6300 sentences (5.4 hours) of reading speech. All sentences are manually transcribed to phonemes with time alignment. We use the standard split of the train (3696 sentences), dev (400 sentences), and test (192 sentences) sets for our experiments.

Librispeech: The LibriSpeech dataset is a common benchmark for the ASR task [80]. It contains 960 hours of reading

TABLE XXIV: PER (%) on TIMIT of different SSL models. SE-ODM [229] only reports results on the test set. The best result is highlighted in **bold**. Experimental details can be found in Sec. X-A4.

Framework	SSL model	Layer	LM	TIMIT	
				dev	test
SE-ODM [229]	-	-	5-gram	-	36.5
wav2vec-U [28]	wav2vec 2.0	15	4-gram	17.0	17.8
EURO	wav2vec 2.0	15	4-gram	18.5	19.8
	HuBERT	15	4-gram	14.9	16.4
	WavLM	14	4-gram	**14.3**	**14.6**

TABLE XXV: PER/WER (%) on Librispeech of different SSL models. PER is from the prefix beam search decoder and WER is from the k2 WFST decoder. The best result is highlighted in **bold**. Experimental details can be found in Sec. X-A4.

Framework	SSL model	Layer	LM	Librispeech			
				dev clean	dev other	test clean	test other
wav2vec-U	wav2vec 2.0	15	4-gram	18.9/31.7	22.4/35.4	18.4/30.7	23.0/36.1
EURO	wav2vec 2.0	15	4-gram	16.2/26.5	**19.3**/29.8	15.7/25.6	**19.8**/30.7
	HuBERT	15	4-gram	**15.2**/23.1	20.7/**29.3**	**15.1**/**22.8**	21.1/**29.8**
	WavLM	15	4-gram	18.0/**23.0**	21.2/31.0	16.6/22.9	21.4/31.0

speech automatically derived from the audiobooks of LibriVox. This corpus is split into 3 training sets (100 and 360 hours of clean speech, and 500 hours of other speech), 2 dev sets (each has 5 hours), and test sets (each has 5 hours).

We explore three SSL models for UASR in EURO, wav2vec 2.0 (wav2vec2-large-ll60k), HuBERT (hubert-large-ll60k), and WavLM (wavlm-large).[25] The three models have the same architecture that consists of 24 layers of Transformer encoders [111] with a similar number of parameters. Among them, wav2vec 2.0 and HuBERT are pre-trained on 60,000 hours of Libri-Light. In addition to Libri-Light [230], Wavlm uses 10,000 hours of Gigaspeech [231] and 24,000 hours of VoxPopuli [232] for pre-training.

TIMIT: We test EURO on TIMIT to confirm that the toolkit works properly. We use the text from the same training set for unsupervised training. To make it comparable with wav2vec-U, we adopt the same setup for EURO wav2vec 2.0. More specifically, we use the model wav2vec2-large-ll60k and the features are extracted from the 15th layer of the model. For Hubert and WavLM, we explore the performance of features from different layers and report the best PER results.

Table XXIV shows the results on TIMIT. Wav2vec-U serves as the baseline and achieves 17.0% and 17.8% PER on dev and test sets, respectively. EURO with the same setup gets 18.5% and 19.8% PER on the dev and test set which is slightly worse than wav2vec-U. While our model is not heavily tuned under this setup, the results are comparable with wav2vec-U. For Hubert (hubert-large-ll60k), EURO performs best on features extracted from the 15th layer. The PERs on the dev set and test set are 14.9% and 16.4%. Compared with the wav2vec-U baseline and EURO wav2vec 2.0, it provides a relative improvement of 10% and 20%, respectively. WavLM (wavlm-large) provides further improvement. The model trained using features extracted from 14th layer of WavLM achieves 17% relative improvement compared with baseline and 25% relative improvement compared with EURO wav2vec 2.0 in terms of PER. It reduces the PER to 14.3% on the dev set and 14.6% on the test set.

[25]Corresponding models can be found in https://s3prl.github.io/s3prl/tutorial/upstream_collection.html

Librispeech: For Librispeech, because of the limitation of computing resources, we use 100 hours of clean speech for UASR training. Unlike TIMIT, we use the text from the whole training set (960 hours) excluding the overlap part with the training speech. The text is phonemicized using G2P phonemizer [233]. In total, around 25m sentences are used for UASR training.

We measure both the PER and WER of UASR systems using different SSL models for this dataset. Wav2vec-U uses a sophisticated decoder to convert phoneme sequences to word sequences and it may not be easily applied to other datasets. To make a fair comparison, we train the wav2vec-U model using the same data and load the model into EURO. We use the same prefix beam search decoder for PER and k2 WFST decoder for WER. Table XXV show the results of LibriSpeech's standard dev and test sets. All EURO models outperform the baseline wav2vec-U. For phone recognition, the HuBERT model performs best on clean sets. It achieves PER 15.2 on the dev clean set and PER 15.1 on the test clean set. Wav2vec 2.0 performs best on more difficult sets. It achieves PER 19.3 and 19.8 on dev other and test other sets, respectively. For word recognition, Hubert gets the best WER of 29.3, 22.8, and 29.8 on dev other, test clean and test other sets. WavLM performs best on dev clean and achieves 23.0 WER which is slightly better than HuBERT.

For training, we set $\lambda = 1.5$, $\gamma = 0.5$, and $\eta = 2.0$ for TIMIT and $\lambda = 2.0, \gamma = 1.0, \eta = 4.0$ for LibriSpeech datasets. For (phone-level) prefix beam search decoding, we set beam size $= 2$ and tunes the weight n-gram language model in $[0, 0.9]$. For (word-level) graph-based WFST decoding, we set search beam size $= 30$, output beam size $= 15$, min active states $= 14,000$, and max active stats $= 56,000$. More details can be found in the configuration file under `egs2/TEMPLATE/uasr/conf/`.

B. Usage Extension of Unsupervised ASR

1) Introduction: Transfer learning has been known as one of the major methodologies in the deep learning era. It has demonstrated its success in speech and natural language [127], [234]. In transfer learning, self-supervised learning (SSL) has been a large branch of studies that focuses on leveraging large amounts of unlabelled data. According to previous literature, SSL can learn to extract contextual representations that greatly improve the performances of various downstream tasks [1].

Many real-world tasks involve transforming one modality to another (e.g., image caption, speech recognition/understanding, songwriting, etc.) [235]–[237]. Usually, there is a need to understand both modalities in those tasks. Given the advances in pre-trained models, several previous works have shown significant improvements by simply applying the pre-trained models [238]–[240]. In their applications, pre-trained models over source modalities are easier to adopt.

Spoken language understanding (SLU) is an example of modalities' transfer, which aims to infer the semantic meaning from spoken utterances. Conventional SLU methods usually adopt a pipeline that consists of an automatic speech recognition (ASR) module and a natural language understanding (NLU) module [241]. However, in recent days, more researchers have started to focus on end-to-end modeling as it could avoid potential error propagation between the two modules. Because end-to-end SLU involves both NLP and speech processing knowledge, previous studies also revealed that SSL in each modality could improve end-to-end SLU performances. Some works have investigated the joint pre-training with both speech and text modalities [242], [243]. While for most other works, researchers adopted speech SSL as a feature extractor, shown to receive improvements in both performance and efficiency [1], [240], [244]–[247]. On the other hand, textual SSLs are difficult to integrate into end-to-end SLU because of the mismatched modalities. Therefore, existing works usually need specific designs to utilize those textual pre-trained models, such as deliberation modules [248]–[250], two-stage decoding [251], K-means clustering for "tokenization" [252]. All approaches could get decent performances on some SLU downstream tasks. However, most of them complicate the training/inference procedure and need further fine-tuning with supervision when applied to downstream tasks.

To achieve a simple way of connecting SSLs between two modalities for SLU, this report proposes a novel usage of unsupervised ASR (UASR) to bridge SSLs between two modalities in a fully **unsupervised** manner. We start with a base framework that applies UASR to augment the speech SSL model by considering unpaired textual information. Following that, we introduce the method that uses UASR to bridge speech and textual pre-trained models. In this work, we focus on connecting wav2vec2 [127] and a phoneme variant of ByT5 [253] with wav2vec-U 2.0 [85]. Our experiments validate the advantages of our proposed methods in various tasks and settings. Notably, we also reach the state-of-the-art performance on the challenging NMSQA benchmark on spoken question answering [252].

2) Background: **SSL as a feature extractor**: The SSL task is to find better speech representation for downstream tasks. Previous SSL-related works usually utilize a speech-only corpus $\mathcal{D}_{\text{speech}}$. The encoder of the SSL model (i.e., $f(\cdot)$) is primarily used, after training with designed self-supervised objectives. Given a speech signal $\mathbf{X} \in \mathcal{D}_{\text{speech}}$, the encoder $f(\cdot)$ generates a hidden representation $\mathbf{H} = (h_1, h_2, ..., h_T)$, where T is the number of frames of the representation. As illustrated in Figure 27 (a), representation $\mathbf{H}$ is used as features for an additional model, when applying to downstream tasks.

UASR: Unsupervised ASR focuses on utilizing unparallel speech and text to realize speech recognition systems. Before the application of SSLs, previous methods have investigated the direction by learning acoustic-text alignment, adversarial networks, and quantized auto-encoding [229], [254], [255].

Wav2vec-U 2.0: Extended from its previous version [28], Wav2vec-U 2.0 is the state-of-the-art model in UASR, which is trained in an end-to-end manner [85]. In the method, SSL (i.e., wav2vec2) is employed as a feature extractor to extract hidden representation $\mathbf{H}$. The later setup follows generative adversarial training. The generator $\mathcal{G}$ takes the segmented acoustic features to generate pseudo phonetic sequences, while the discriminator $\mathcal{C}$ focuses on distinguishing the pseudo phonetic sequence from generator $\mathcal{G}(\mathbf{H})$ and the real phonetic sequence $\mathbf{Y}^u \in \mathcal{D}_{\text{text}}$. Noted that $\mathcal{D}_{\text{text}}$ is a text-only corpus, which is unpaired with $\mathcal{D}_{\text{speech}}$. Apart from the adversarial loss, wav2vec-U 2.0 also utilizes several other types of losses (i.e., gradient penalty $\mathcal{L}_{\text{gp}}$, segment smoothness $\mathcal{L}_{\text{sp}}$, phoneme diversity $\mathcal{L}_{\text{pd}}$, and auxiliary K-means clusters $\mathcal{L}_{\text{ss}}$ from Mel frequency cepstral coefficients (MFCC)) to stabilize the training. The final optimization target is formulated as:

$$\begin{split}\min_{\mathcal{G}} \max_{\mathcal{C}} \mathbb{E}_{\mathbf{Y}^u}[\log \mathcal{C}(\mathbf{Y}^u)] - \mathbb{E}_{\mathbf{H}}[\log(1 - \mathcal{C}(\mathcal{G}(\mathbf{H})))] \\ + \lambda \mathcal{L}_{\text{gp}} + \gamma \mathcal{L}_{\text{sp}} + \eta \mathcal{L}_{\text{pd}} + \delta \mathcal{L}_{\text{ss}},\end{split} \tag{25}$$

where $\lambda, \gamma, \eta, \delta$ are the weights for losses.

UASR with SSL provides the foundation of our following investigation. In later sections, we primarily focus on a simplified version of wav2vec-U 2.0 that is without auxiliary cluster prediction (i.e., remove $\mathcal{L}_{\text{ss}}$), as we empirically find the training is harder to converge with the proposed version in [85].

3) Problem Formulation: As speech SSLs train on a large number of speech signals from $\mathcal{D}_{\text{speech}}$, it has shown impressive performances in several speech tasks [1], [2]. However, recent works also reveal that speech SSLs cannot handle some high-level semantic tasks (e.g., spoken question answering) [252]. To reach reasonable performances, the framework needs the following textual pre-trained model f_{text} that trains on text-only corpus $\mathcal{D}_{\text{text}}$. To match the embeddings in textual pre-trained models, previous work applied K-means clustering to convert the speech SSL features into cluster IDs, which are further used as a token ID for the textual model [252]. The sequential application of speech and textual pre-trained models results in an issue of modalities mismatch between speech and text token IDs, so a fine-tuning stage is necessary to connect these modalities. On the other hand, our method can connect both SSL models in the pre-training stage without fine-tuning thanks to UASR, and build an **unsupervised speech-to-semantic pre-trained model** straightforwardly.

In the following subsections, we first add UASR as an augmentation module to the speech SSL model as a base framework, resulting in a speech-to-text pre-trained model.

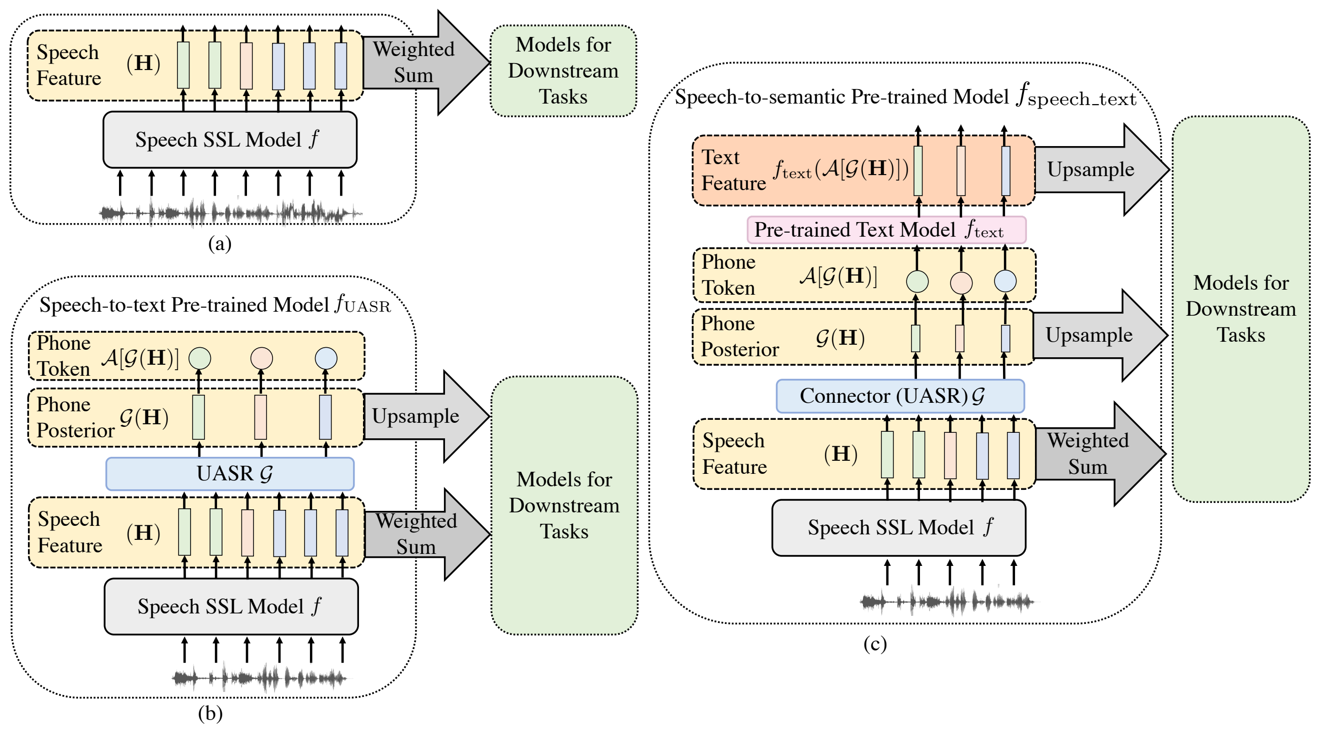

Speech Feature
(H)
Weighted Sum
Models for Downstream Tasks
Speech SSL Model f
(a)
Speech-to-text Pre-trained Model f_{UASR}
Phone Token $\mathcal{A}[\mathcal{G}(\mathbf{H})]$
Phone Posterior $\mathcal{G}(\mathbf{H})$
Upsample
UASR $\mathcal{G}$
(b)
Speech-to-semantic Pre-trained Model $f_{\text{speech\_text}}$
Text Feature $f_{\text{text}}(\mathcal{A}[\mathcal{G}(\mathbf{H})])$
Pre-trained Text Model f_{text}
Connector (UASR) $\mathcal{G}$
(c)

Fig. 27: The comparison between different frameworks: (a) the application of speech SSL models as a feature extractor to downstream tasks, discussed in Sec X-B2. (b) the framework of using UASR as an augmentation module to the speech SSL, introduced in Sec X-B4. (c) the framework of using UASR as a connector to utilize a pre-trained text model for downstream tasks, proposed in Sec X-B5.

Then, we further extend it by combining a textual pre-trained model, where we utilize UASR as a connector and contract a speech-to-semantic pre-trained model.

4) UASR as an Augmentation Module: In this subsection, we propose using UASR as an augmentation module for the speech SSL model for the following reasons.

Unsupervised property: As discussed in Sec. X-B2, the UASR are jointly trained with inputs that include both $\mathbf{X} \in \mathcal{D}_{\text{speech}}$ and $\mathbf{Y}^u \in \mathcal{D}_{\text{text}}$. As discussed in Sec. X-B2, adding an unsupervised ASR network maintains the unsupervised property of the whole model, which keeps the benefits of learning from a vast amount of unsupervised data.

Textual benefits: The incorporation of text-only corpus $\mathcal{D}_{\text{text}}$ is the major difference between speech SSL and the proposed method. According to (25), the textual information Y^u is explicitly applied with the adversarial objective.

Performance stability: To keep the performance stable for various tasks, we can always apply the original speech SSL with the UASR representation.

The application of UASR follows the illustration of Figure 27 (b). In short, the augmented model is defined as follows,

$$f_{\text{UASR}}(\mathbf{H}) \triangleq (\mathbf{H}, \text{UP}(\mathcal{G}(\mathbf{H}))), \tag{26}$$

where UP$(\cdot)$ is an upsampling function to match the resolution of $\mathbf{H}$ and $\mathcal{G}(\mathbf{H})$.[26] The $f_{\text{UASR}}(\mathbf{H})$ is used as the input of downstream models for various tasks.

[26]In our experiments, we simply use the repeat upsampling.

5) UASR as a Connector: As discussed in Sec. X-B1 and Sec. X-B3, the modalities mismatch issue blocks constructing pre-trained speech-to-semantic models in an unsupervised manner. Since UASR targets conducting recognition in an unsupervised fashion, it could naturally be a glue to connect the acoustic and textual pre-trained models. We argue that using UASR as a connector would have three more benefits, apart from the mitigation of the modalities mismatch issue.

Unsupervised property: Similar to f_{UASR} in Sec. X-B4, adding UASR as a connector can still maintain the unsupervised property of the whole model.

Enhanced textual benefits: As explained in Sec. X-B4, the UASR module can introduce implicit textual knowledge from the adversarial objective. On the other hand, when applying UASR as a connector, it can adopt explicit textual resources that hold semantic information from the pre-trained text model f_{text}.

Fine-tuning free: As previous methods usually need fine-tuning to mitigate the modalities mismatch between speech and text modalities, using UASR as a connector can relax the requirement for fine-tuning since the output of UASR is a text (phoneme)-level distribution.

The application of the UASR connector follows Figure 27 (c). The augmented model is defined in a similar manner to f_{UASR}, as follows,

$$f_{\text{speech\_text}}(\mathbf{H}) \triangleq (\mathbf{H}, \text{UP}'(\mathcal{G}(\mathbf{H})), \text{UP}'(f_{\text{text}}(\mathcal{A}[\mathcal{G}(\mathbf{H})]))), \tag{27}$$

TABLE XXVI: Freeze setting experiments with UASR and PT5: pre-trained models are frozen during the training. The "+UASR" option utilizes the method explained in Sec. X-B4, while the "+UASR +PT5" option utilizes the method showed in Sec. X-B5.

SSL	PR(↓)	ASR(↓)	IC(↑)	ST(↑)	SID(↑)
wav2vec2	5.51	3.79	94.38	13.01	**83.69**
+UASR	**4.53**	**3.76**	94.33	13.00	82.85
+UASR +PT5	4.68	3.97	**94.88**	**13.53**	82.74

where $\mathrm{UP}'(\cdot)$ is another upsampling function to match the resolution of $\mathbf{H}$ and $\mathcal{A}$ is an assignment procedure that assigns phone tokens to each frame, given a phonetic posteriorgram. $\mathcal{A}$ can be a $\mathrm{GumbleSoftmax}$ module to keep the differentiable property, but it can also be an argmax operation for simplicity. Similar to Sec. X-B4, the $f_{\text{speech\_text}}(\mathbf{H})$ is used as the input of downstream models for tasks.

6) Experiments: Downstream Tasks and Datasets: To fully investigate the proposed methods, we conduct experiments in two folds: the first freezes the parameters of the pre-trained model; the second fine-tunes the textual pre-trained model.

Freeze setting: In the freeze setting, we follow the exact setting of SUPERB benchmark [1]. As discussed in the previous section, we primarily focus on understanding tasks, including intent classification (IC), emotion recognition (ER), and speech translation (ST). However, as UASR are designed for recognition tasks, we also conduct experiments in ASR and phone recognition (PR). Meanwhile, we carry out the speaker identification (SID) task to verify if the UASR and the pre-trained text model are effective for semantic-related tasks.

Fine-tuning setting: In the fine-tuning setting, we only conduct experiments that fine-tunes the textual model to match the configuration in [252] and to avoid joint training of a large model. We adopt ESPnet [247], [256] that can support stronger downstream models. We intentionally select some different datasets that are larger than the ones in the SUPERB benchmark, evaluating the proposed method apart from the potential over-fitting issue in low-resource scenarios. Similar to the freeze setting, we focus on understanding tasks, including IC, ER, ST, and slot-filling (SF). We adopt SLURP [241] for IC and ST, IEMOCAP for ER [257], and CoVOST2 for ST [258]. In addition, We also tested on a publicly available spoken question answering (SQA) NMSQA dataset [252].

Three test sets included are derived from SQuAD [259], NEWSQA [260] and QuAC [261], whereas the latter two are reported together as out-of-domain (OOD) samples.

7) Experimental Settings: **Speech SSL**: Given that the UASR model is based on wav2vec2 [127], we adopt pre-trained wav2vec2 that trained on LibriLight [230].

UASR: To keep the same as [85], we utilize Librispeech speech data for the speech-only corpus $\mathcal{D}_{\text{speech}}$ and Librispeech language modeling data for the text-only corpus $\mathcal{D}_{\text{text}}$. As noted in Sec X-B3, we do not use $\mathcal{L}_{\text{ss}}$. For other settings, we use the hyper-parameters released on Fairseq.[27]

[27]https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec/unsupervised/config/gan/w2vu2.yaml

Pre-trained text model: As discussed in Sec. X-B1 and Sec. X-B5, we select PhonemeT5 (PT5)[28] as our main textual pre-trained model, because it has the same token space as our UASR model. PT5 is a variant of ByT5 [262], which uses phonemicized text[29] as input with the span reconstruction objective.

Model candidates: For the freeze setting, aligned with Figure 27, we compare the models in three settings, including speech pre-trained model (i.e., wav2vec2), speech pre-trained model with UASR, and speech pre-trained model with UASR + textual pre-trained model (i.e., PT5). For the fine-tuning setting, we compare four settings: **A** utilizes only the speech pre-trained model; **B** applies UASR as an augmented module to the speech pre-trained model; **C** employs K-means as a connector between speech and textual pre-trained models [252];[30] **D** uses UASR instead of K-means as a connector.

To further investigate the effects of different pre-trained text models, we provide two other models with the same network architecture, namely randomized T5 (RT5) and textual T5 (TT5). RT5 is a model with the same architecture as PT5 but with randomized parameters. TT5 is the original version of ByT5 that is pre-trained on text data [262]. Because of the token mismatch (i.e., phone tokens from UASR and byte-level tokens from ByT5), we utilize a randomly initialized dictionary to map UASR's phone tokens into bytes, similar to the K-means approach [252]. Given the same framework as **D**, we name **E** as the one with RT5 and **F** as the one with TT5. Except for SQA which utilizes whole pre-trained models (i.e., both encoder and decoder), other tasks only employ the encoders of the T5 variants (i.e., **D**, **E**, and **F**).

Dowstream models: For the freeze setting, we follow exact settings as the SUPERB public benchmark, which adopts simple downstream models (e.g., stacked linear or recurrent layers). For the fine-tuning setting, we utilize ESPnet [247], [256] for the downstream models except for SQA. For IC, SF, and ER, we adopt a conformer-based encoder-decoder network with the hybrid connectionist temporal classification (CTC)/attention style of training [208], [225]. While for ST, we adopt a transformer-based encoder-decoder with auxiliary ASR loss computed from CTC. Detailed network settings follow the corresponding configuration.[31] For classification tasks like IC and ER, we also include ASR transcriptions as a joint learning objective. For all models in ESPnet, we apply specaugment to the hidden states from pre-trained models [1]. For SQA, it is difficult to hold the whole speech SSL features for the spoken context of answers. Therefore, we employ a two-step approach. Specifically, questions and contexts are converted into time-coded phoneme sequences by UASR. Given questions and contexts, the PT5 is fine-tuned to predict answer time spans in the context. The model is trained for 3

[28]https://github.com/voidful/t5lephone

[29]Each phoneme token is mapped to a single character with a predefined mapping dictionary to reduce the sequence length.

[30]We adopt the official K-means checkpoint at https://dl.fbaipublicfiles.com/textless_nlp/gslm/w2v2/km50/km.bin. 50-cluster version is selected because UASR has a similar vocabulary size (i.e., 45).

[31]https://github.com/espnet/espnet/blob/master/egs2/{slurp_entity,slurp,iemocap,covost1}/{asr1,st1}

TABLE XXVII: Fine-tuning setting experiments with UASR and PT5: textual pre-trained models are jointly fine-tuned with downstream models during training. **B** utilizes the method explained in Sec. X-B4, while **D** utilize the method showed in Sec. X-B5. Other models (i.e., **A** and **C**) and more detailed configurations are introduced in Sec. X-B7.

	Connector	Text Model	IC(↑)	SF(↑)	ER(↑)	ST(↑)
A	/	/	82.82	65.82	66.35	22.1
B	UASR	/	82.93	64.93	64.11	22.2
C	K-means	PT5	86.41	71.70	72.61	22.5
D	UASR	PT5	**87.10**	**74.03**	**73.57**	**24.3**

TABLE XXVIII: Effect of different textual pre-trained models when using UASR as a connector: **E** employs a T5 architecture with randomized parameters; **F** uses the original ByteT5; **D** applies the phoneme T5 introduced in Sec X-B7.

	Text Model	IC(↑)	SF(↑)	ER(↑)	ST(↑)
E	RT5	86.61	71.83	72.19	23.9
F	TT5	85.93	72.31	72.19	**24.6**
D	PT5	**87.10**	**74.03**	**73.57**	24.3

TABLE XXIX: Effect of different textual pre-trained models for SQA when using UASR as a connector. The Longformer model is the state-of-the-art model in [252].

	Text Model	SQuAD		OOD	
		AOS(↑)	FF1(↑)	AOS(↑)	FF1(↑)
[252]	Longformer	49.1	55.9	/	/
F	TT5	56.0	61.5	38.4	42.4
D	PT5	**65.8**	**69.7**	**42.1**	**46.2**

epochs with a 3e-4 learning rate and the best model is selected based on the performances on the development set.

Evaluation metric: For PR, we adopt phone error rate (PER), while for ASR, we utilize word error rate (WER). For IC and ER, we employ the accuracy rate. For slot-filling, we apply the slot-type F1 score. For ST, we use the BLEU score. For SQA, we adopt the same evaluation metrics as [252], including the area overlapping score (AOS) and frame F1 (FF1). Note that when the answers cannot be extracted from the given context, we set AOS and FF1 to 0.

8) Experiments: Results and Discussion: We first discuss the results of the freeze setting as shown in Table XXVI. As UASR is optimized for PR, it is reasonable to have better performances in PR. However, ASR does not have significant gain with UASR, which shows that the PR advantage from UASR may not directly enhance the speech SSL. When it comes to understanding tasks (i.e., IC and ST), the one with UASR and PT5 achieves the best performance, demonstrating the semantic benefits introduced by PT5. Given the wav2vec2, UASR, and PT5 do not update their parameters, we could infer that the UASR as a connector could mitigate the modalities mismatch issue discussed in Sec. X-B3. Last, it is reasonable that UASR and PT5 cannot improve the performances of speaker identification, as their features barely contain speaker-related information at the design level.

Table XXVII shows the results in the fine-tuning setting. For all the selected tasks, **D** achieves the best performance by using UASR as a connector. When applying K-means as a connector (i.e., **C**), we find similar performance improvements in all four tasks as [252]. However, with UASR as a connector, the improvements are even larger.

As in the ablation study shown in Table XXVIII, the one with PT5 (i.e., **D**) reaches the best performance for IC, SF, and ER, while the one with TT5 (i.e.,**F**) has the best performance for ST. Based on the experimental results, we would argue that there is a balance between modalities matching and textual semantics. As for text generation tasks, **F** has better results. Table XXIX shows our proposed method on SQA. We only conduct the model with textual pre-trained models, as we find the text information is essential to train the model. For example, model **E** can hardly converge for the task, resulting in 0.0 scores for the test sets. On the contrary, the model **D** with matched modalities has shown outstanding performances, significantly better than the model **F** and the best model in [252].

XI. Concluding Remarks

Here we summarize what we have achieved:

- We study several standard compression techniques for Transformer-based speech SSL models, including weight pruning, head pruning, low-rank approximation, and knowledge distillation. We comprehensively analyze the compression-performance trade-off, charting the landscape of model compression for speech SSL models in Section II. In addition, we study fixed-length subsampling along the time axis in self-supervised learning and propose a variable-length subsampling approach in Section III. We found that ASR with larger word pieces is more resistant to more aggressive subsampling, while ASR with characters can only sustain subsampling up to approximately 25 Hz. Subsampling while training SSL models not only improves the overall performance on downstream tasks under certain frame rates but also brings significant speed-up in inference. Variable-length subsampling performs particularly well under low frame rates. Moreover, if we have access to phonetic boundaries, we find no degradation in performance for an average frame rate as low as 10 Hz. For future work, the effect of subsampling on large SSL models, particularly, by training them with subsampling from random initialization, is largely unexplored. And we believe it is even possible to encode an utterance into a single vector [263], independent of the input length. Much research is needed in this direction.
- We find that though having similar performance as original SSL models, distilled SSL models suffer from

performance degradation even more than their original versions in distorted environments. We propose cross-distortion mapping (CDM) during distillation to improve the generalizability of distilled SSL models. Results show consistent improvements under both in- and out-of-domain distorted setups for different downstream tasks while keeping efficient model size. Please refer to Section IV for the complete results.

- We introduces SpeechCLIP , a novel framework integrating CLIP into visually grounded speech models, in Section V. We demonstrate significant improvements in image-speech retrieval with CLIP's supervision. Moreover, the proposed methods can perform zero-shot speech-text retrieval and capture semantically related keywords in speech signals. Results indicate that bridging speech and text domains with CLIP's supervision is possible and promising. Overall, SpeechCLIP opens a new research direction of indirectly supervising speech models with text via other modalities. We suggest some topics in SpeechCLIP are worth investigating in the future, including integrating parallel and cascaded in the same model and cascaded structure with variable length prediction aiming for unsupervised ASR. Furthermore, extending SpeechCLIP to a multilingual model is possible using spoken captions from other languages or Multilingual-CLIP models. Finally, we wish to inspect how CLIP can enhance speech SSL models' performance on downstream problems like speech recognition and intent classification.
- We explore the effectiveness of efficient tuning methods for SSL speech representation transfer. Extensive experiments are conducted to investigate the various adapter types on different SSL speech models on a wide range of speech processing tasks in Section VI. Other than finding adapters capable of achieving comparable performance to the fully fine-tuned models, we further examine the stability of adapters compared with fine-tuning. We then discussed comparing efficient methods in NLP and Speech. To our best knowledge, this is the most comprehensive work exploring adapter methods on a wide range of downstream speech tasks so far. In addition, we propose a prompt tuning framework based on Generative Spoken Language Model (GSLM) for speech processing tasks in Section VII. The experiment results show that the language model can be guided to directly generate the answers by tuning a limited number of task-specific vectors. In classification tasks, the framework achieves competitive performance compared to fine-tuning the entire downstream models. We also investigate the limitation of the framework on challenging sequence generation tasks. This work is the first exploration of prompt tuning paradigm for speech processing tasks. We believe this study can motivate the speech research communities to explore the prompting paradigm more.
- We explore the utility of SSL models for prosody-conveyed pragmatic functions. Using the newly-proposed SUPERB-prosody evaluation framework, we find that SSL models provide significant value for prosody-intensive tasks, and that they are good at extracting prosodic information in pseudo tasks. Furthermore, we analyze the layer contribution and discover that most SSL models tend to store prosodic information in the first few layers. However, the field still lacks a good understanding of why different SSL models are better for different tasks [264], and this is an important topic for future work.
- We explored using unlabelled speech data to improve the accuracy of the ASR of South African languages, using a small amount of manually transcribed speech data along with a weak language model. We confirmed the previous findings [190] that semi-supervised training with a weak language model does not work very well. The best approach to improve the initial acoustic models (WER of 54.7%) is to continue self-supervised pre-training of the multilingual model XLSR-53 (WER of 50.9%), which does not require any language model. This approach is beneficial in South African code-switching, where we can only train a weak language model due to the lack of sufficient amount of in-domain text. When we subsequently used this model as a seed model for semi-supervised training, we obtained a relative improvement of 17% (WER of 45.2%) compared to the supervised baseline model trained with MFCC features. However, these improvements were much smaller than we would expect with a strong language model. For this reason, in the future, we would like to explore ways of training better language models in low-resource and especially code-switched settings to improve the performance during semi-supervised training and decoding. We would also like to use the large XLS-R pre-trained multilingual model [193] because it achieved better performance than the multilingual model XLSR-53 [184] in our preliminary experiments. Finally, we would also like to follow [196] and use better regularization methods during continued self-supervised pre-training. Both these methods should allow for a more efficient continued pre-training. We believe our findings will apply to other low-resource languages with limited amounts of text corpora available.
- We introduce a new toolkit for unsupervised ASR, namely EURO. The toolkit is developed as an open platform for the research field of unsupervised ASR. The current architecture of EURO is based on the Wav2vec-U framework but greatly improves the reproducibility with flexible frontends of almost 30 SSL models and a faster preparation/inference compared to its original implementation in FAIRSEQ. By integrating with k2, EURO provides a WFST decoder for word recognition. Our experiments on TIMIT and LibriSpeech show that we could get comparable performances with wav2vec-U in FAIRSEQ but even better results with Hubert and WavLM as new frontends.
- Though pre-trained models in speech and languages have shown to be effective, it is difficult to sequentially apply speech and language pre-trained models to scenarios of spoken language understanding. In Section X-B, we propose using unsupervised ASR to mitigate the mismatch

between modalities. We first show that adding unsupervised ASR could enhance the original speech pre-trained model, then unsupervised ASR is applied as a connector between speech and textual pre-trained models. Extensive experiments, in various tasks and settings, demonstrate the effectiveness of our proposed method, which also shows possible solutions to other scenarios other than speech and language.

The six-week workshop was a resounding success, with our international team making significant advancements in speech SSL technology. Our efforts paid off, as five cutting-edge papers were accepted and presented at SLT 2022 [9], [12], [18], [22], [25]. And the cherry on top, one of these papers was awarded the Best Paper Award at the conference [25]. Our team pushed the boundaries of speech SSL technology in several areas, including computation, robustness, visual enhancement, and efficient usage of these models. We have made great progress in improving speech SSL models in different aspects individually, but there was simply not enough time to fully integrate all of our findings into a single, cohesive model. However, the spirit of collaboration and innovation lives on, as some members of the team continue their work to build upon and integrate the technology we developed during the workshop. The team made remarkable achievements in just six short weeks, and we are eager to see where our continued work will take us. The future of speech SSL technology is bright, and we are proud to have played a role in shaping it.

References

[1] S. wen 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, T.-H. Huang, W.-C. Tseng, K. tik Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H. yi Lee, "SUPERB: Speech Processing Universal PERformance Benchmark," in *INTERSPEECH*, 2021.

[2] H.-S. Tsai, H.-J. Chang, W.-C. Huang, Z. Huang, K. Lakhotia, S.-w. Yang, S. Dong, A. Liu, C.-I. Lai, J. Shi *et al.*, "SUPERB-SG: Enhanced speech processing universal performance benchmark for semantic and generative capabilities," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 8479–8492.

[3] anonymous, "SUPERB @ SLT 2022: Challenge on generalization and efficiency of self-supervised speech representation learning," in *SLT*, 2022.

[4] S. Evain, H. Nguyen, H. Le, M. Z. Boito, S. Mdhaffar, S. Alisamir, Z. Tong, N. Tomashenko, M. Dinarelli, T. Parcollet, A. Allauzen, Y. Estève, B. Lecouteux, F. Portet, S. Rossato, F. Ringeval, D. Schwab, and L. Besacier, "LeBenchmark: A Reproducible Framework for Assessing Self-Supervised Representation Learning from Speech," in *INTERSPEECH*, 2021.

[5] J. Shor, A. Jansen, R. Maor, O. Lang, O. Tuval, F. de Chaumont Quitry, M. Tagliasacchi, I. Shavitt, D. Emanuel, and Y. Haviv, "Towards Learning a Universal Non-Semantic Representation of Speech," in *INTERSPEECH*, 2020.

[6] J. Pu, Y. Yang, R. Li, O. Elibol, and J. Droppo, "Scaling effect of self-supervised models," in *INTERSPEECH*, 2021.

[7] P. Ganesh, Y. Chen, X. Lou, M. A. Khan, Y. Yang, H. Sajjad, P. Nakov, D. Chen, and M. Winslett, "Compressing large-scale transformer-based models: A case study on BERT," *Transactions of the Association for Computational Linguistics*, 2021.

[8] T.-Q. Lin, T.-H. Yang, C.-Y. Chang, K.-M. Chen, T. hsun Feng, H. yi Lee, and H. Tang, "Compressing transformer-based self-supervised models for speech processing," in *ICASSP*, 2023.

[9] Y. Meng, H.-J. Chen, J. Shi, S. Watanabe, P. Garcia, H.-y. Lee, and H. Tang, "On Compressing Sequences for Self-Supervised Speech Models," in *SLT*, 2022.

[10] W.-N. Hsu, A. Sriram, A. Baevski, T. Likhomanenko, Q. Xu, V. Pratap, J. Kahn, A. Lee, R. Collobert, G. Synnaeve, and M. Auli, "Robust wav2vec 2.0: Analyzing domain shift in self-supervised pre-training," in *INTERSPEECH*, 2021.

[11] K. P. Huang, Y.-K. Fu, Y. Zhang, and H.-y. Lee, "Improving distortion robustness of self-supervised speech processing tasks with domain adaptation," in *INTERSPEECH*, 2022.

[12] K.-P. Huang, Y.-K. Fu, T.-Y. Hsu, F. R. Gutierrez, F.-L. Wang, L.-H. Tseng, Y. Zhang, and H. yi Lee, "Improving generalizability of distilled self-supervised speech processing models under distorted settings," in *SLT*, 2022.

[13] P. Peng and D. Harwath, "Fast-slow transformer for visually grounding speech," in *ICASSP*, 2022.

[14] G. Chrupała, "Visually grounded models of spoken language: A survey of datasets, architectures and evaluation techniques," *arXiv preprint arXiv:2104.13225*, 2021.

[15] P. Peng and D. Harwath, "Self-supervised representation learning for speech using visual grounding and masked language modeling," in *AAAI SAS workshop*, 2022.

[16] ——, "Word discovery in visually grounded, self-supervised speech models," *arXiv preprint arXiv:2203.15081*, 2022.

[17] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," in *ASRU*, 2021.

[18] Y.-J. Shih, H.-F. Wang, H.-J. Chang, L. Berry, H. yi Lee, and D. Harwath, "SpeechCLIP: Integrating speech with pre-trained vision and language model," in *SLT*, 2022.

[19] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "BitFit: Simple parameter-efficient fine-tuning for Transformer-based masked language-models," 2021.

[20] D. Guo, A. Rush, and Y. Kim, "Parameter-efficient transfer learning with diff pruning," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, August 2021, pp. 4884–4896. [Online]. Available: https://aclanthology.org/2021.acl-long.378

[21] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *CoRR*, vol. abs/2107.13586, 2021.

[22] Z.-C. Chen, C.-L. Fu, C.-Y. Liu, S.-W. D. Li, and H. yi Lee, "Exploring efficient-tuning methods in self-supervised speech models," in *SLT*, 2022.

[23] K.-W. Chang, W.-C. Tseng, S.-W. Li, and H. yi Lee, "SpeechPrompt: An exploration of prompt tuning on generative spoken language model for speech processing tasks," *INTERSPEECH*, 2022.

[24] K.-W. Chang, Y.-K. Wang, H. Shen, I. thing Kang, W.-C. Tseng, S.-W. Li, and H. yi Lee, "SpeechPrompt v2: Prompt tuning for speech classification tasks," *submitted to ICASSP*, 2023.

[25] O. the Utility of Self-supervised Models for Prosody-related Tasks, "Guan-ting lin and chi-luen feng and wei-ping huang and yuan tseng and tzu-han lin and chen-an li and hung-yi lee and nigel g. ward," in *SLT*, 2022.

[26] E. van der Westhuizen and T. Niesler, "A first South African corpus of multilingual code-switched soap opera speech." in *LREC*, 2018.

[27] L.-M. Lam-Yee-Mui, L. Ondel, and O. Klejch, "Self-supervised pre-training and semi-supervised training for speech recognition for low-resource south african language," in *submitted to ICASSP*, 2023.

[28] A. Baevski, W.-N. Hsu, A. Conneau, and M. Auli, "Unsupervised speech recognition," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 826–27 839, 2021.

[29] D. Gao, J. Shi, S.-P. Chuang, L. P. Garcia, H.-y. Lee, S. Watanabe, and S. Khudanpur, "EURO: ESPnet unsupervised ASR open-source toolkit," *arXiv*, 2023.

[30] S. Shon, A. Pasad, F. Wu *et al.*, "Slue: New benchmark tasks for spoken language understanding evaluation on natural speech," in *ICASSP*, 2022.

[31] G.-T. Lin, Y.-S. Chuang, H.-L. Chung, S.-w. Yang, H.-J. Chen, S. Dong, S.-W. Li, A. Mohamed, H.-y. Lee, and L.-s. Lee, "DUAL: Discrete spoken unit adaptive learning for textless spoken question answering," *CoRR*, 2022.

[32] J. Shi, C.-J. Hsu, H. Chung, D. Gao, P. Garcia, S. Watanabe, A. Lee, and H.-y. Lee, "Bridging speech and textual pre-trained models with unsupervised asr," *arXiv*, 2022.

[33] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[34] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2021.

[35] M. Behnke and K. Heafield, "Losing heads in the lottery: Pruning transformer attention in neural machine translation," in *EMNLP*, 2020, pp. 2664–2674.

[36] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *ICASSP*, 2013.

[37] J. Max, V. Andrea, and Z. Andrew, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[38] D. Emily, Z. Wojciech, B. Joan, n. L. Yan, and F. Rob, "Exploiting linear structure within convolutional networks for efficient evaluation," *arXiv preprint arXiv:1404.0736*, 2014.

[39] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[40] R. Prabhavalkar, O. Alsharif, A. Bruguier, and L. McGraw, "On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition," in *ICASSP*, 2016.

[41] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," *CoNLL*, 2016.

[42] C.-I. J. Lai, Y. Zhang, A. H. Liu, S. Chang, Y.-L. Liao, Y.-S. Chuang, K. Qian, S. Khurana, D. Cox, and J. Glass, "PARP: Prune, adjust and re-prune for self-supervised speech recognition," in *NIPS*, 2021.

[43] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, M. Carbin, and Z. Wang, "The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models," in *CVPR*, 2021, pp. 16 306–16 316.

[44] H.-J. Chang, S. wen Yang, and H. yi Lee, "DistilHuBERT: Speech representation learning by layer-wise distillation of hidden-unit BERT," 2021.

[45] R. Wang, Q. Bai, J. Ao, L. Zhou, Z. Xiong, Z. Wei, Y. Zhang, T. Ko, and H. Li, "Lighthubert: Lightweight and configurable speech representation learning with once-for-all hidden-unit bert," *arxiv:2203.15610*, 2022.

[46] Y. Lee, K. Jang, J. Goo, Y. Jung, and H. Kim, "FitHuBERT: Going thinner and deeper for knowledge distillation of speech self-supervised learning," *Interspeech*, 2022.

[47] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *NIPS*, 2015.

[48] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *NIPS*, 1990.

[49] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *arXiv preprint arXiv:1905.09418*, 2019.

[50] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?" *NeurIPS*, 2019.

[51] Y.-C. Hsu, T. Hua, S. Chang, Q. Lou, Y. Shen, and H. Jin, "Language model compression with weighted low-rank factorization," *ICLR*, 2022.

[52] J. Ba and R. Caruana, "Do deep nets really need to be deep?" *Advances in neural information processing systems*, 2014.

[53] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[54] G. Aguilar, Y. Ling, Y. Zhang, B. Yao, X. Fan, and C. Guo, "Knowledge distillation from internal representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[55] T. Kim, J. Oh, N. Kim, S. Cho, and S.-Y. Yun, "Comparing Kullback-Leibler divergence and mean squared error loss in knowledge distillation," *arXiv preprint arXiv:2105.08919*, 2021.

[56] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *arXiv preprint arXiv:1904.03240*, 2019.

[57] M. N. Rabe and C. Staats, "Self-attention does not need $O(n^2)$ memory," *arXiv preprint arXiv:2112.05682*, 2021.

[58] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A time-restricted self-attention layer for ASR," in *ICASSP*. IEEE, 2018, pp. 5874–5878.

[59] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[60] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[61] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *CSUR*, 2022.

[62] V. Vanhoucke, M. Devin, and G. Heigold, "Multiframe deep neural networks for acoustic modeling," in *ICASSP*, 2013.

[63] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *ASRU*, 2015.

[64] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.

[65] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *ICASSP*. IEEE, 2016, pp. 4945–4949.

[66] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *ICASSP*. IEEE, 2017, pp. 4835–4839.

[67] S. Bengio and G. Heigold, "Word embeddings for speech recognition," in *INTERSPEECH*, 2014.

[68] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *ICASSP*, 2016.

[69] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *ICASSP*. IEEE, 2017, pp. 4845–4849.

[70] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in Joint CTC-Attention Based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM," in *Interspeech*, 2017, pp. 949–953.

[71] F. Wu, K. Kim, J. Pan, K. J. Han, K. Q. Weinberger, and Y. Artzi, "Performance-efficiency trade-offs in unsupervised pre-training for speech recognition," in *ICASSP*, 2022.

[72] A. Vyas, W.-N. Hsu, M. Auli, and A. Baevski, "On-demand compute reduction with stochastic wav2vec 2.0," in *Interspeech*, 2022, pp. 3048–3052.

[73] Y. Lee, K. Jang, J. Goo, Y. Jung, and H. R. Kim, "FitHuBERT: Going Thinner and Deeper for Knowledge Distillation of Speech Self-Supervised Models," in *Interspeech*, 2022, pp. 3588–3592.

[74] J. Chorowski, G. Ciesielski, J. Dzikowski, A. Lancucki, R. Marxer, M. Opala, P. Pusz, P. Rychlikowski, and M. Stypulkowski, "Aligned contrastive predictive coding," in *Interspeech*, 2021.

[75] S. Bhati, J. Villalba, P. Żelasko, L. Moro-Velázquez, and N. Dehak, "Segmental contrastive predictive coding for unsupervised word segmentation," in *Interspeech*, 2021.

[76] S. Dieleman, C. Nash, J. Engel, and K. Simonyan, "Variable-rate discrete representation learning," *arXiv preprint arXiv:2103.06089*, 2021.

[77] L. Dong and B. Xu, "CIF: Continuous integrate-and-fire for end-to-end speech recognition," in *ICASSP*, 2020.

[78] 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.

[79] M. Ott *et al.*, "FAIRSEQ: A fast, extensible toolkit for sequence modeling," in *NAACL*, 2019.

[80] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books," in *ICASSP*, 2015.

[81] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *EMNLP*, 2018.

[82] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *ACL*, 2016.

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

[84] H. Kamper and B. van Niekerk, "Towards unsupervised phone and word segmentation using self-supervised vector-quantized neural networks," *arXiv preprint arXiv:2012.07551*, 2020.

[85] A. H. Liu, W.-N. Hsu, M. Auli, and A. Baevski, "Towards end-to-end unsupervised speech recognition," *arXiv preprint arXiv:2204.02492*, 2022.

[86] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using Kaldi," in *Interspeech*, 2017.

[87] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[88] J. Huh, H. S. Heo, J. Kang, S. Watanabe, and J. S. Chung, "Augmentation adversarial training for self-supervised speaker recognition," *arXiv preprint arXiv:2007.12085*, 2020.

[89] H. Hu, X. Yang, Z. Raeesy, J. Guo, G. Keskin, H. Arsikere, A. Rastrow, A. Stolcke, and R. Maas, "Redat: Accent-invariant representation for end-to-end asr by domain adversarial training with relabeling," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6408–6412.

[90] N. Das, S. Bodapati, M. Sunkara, S. Srinivasan, and D. H. Chau, "Best of both worlds: Robust accented speech recognition with adversarial transfer learning," *arXiv preprint arXiv:2103.05834*, 2021.

[91] C.-F. Liao, Y. Tsao, H.-Y. Lee, and H.-M. Wang, "Noise adaptive speech enhancement using domain adversarial training," *arXiv preprint arXiv:1807.07501*, 2018.

[92] H. Wang, Y. Qian, X. Wang, Y. Wang, C. Wang, S. Liu, T. Yoshioka, J. Li, and D. Wang, "Improving noise robustness of contrastive speech representation learning with speech reconstruction," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6062–6066.

[93] Q.-S. Zhu, J. Zhang, Z.-Q. Zhang, M.-H. Wu, X. Fang, and L.-R. Dai, "A noise-robust self-supervised pre-training model based speech representation learning for automatic speech recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3174–3178.

[94] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[95] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8342–8360. [Online]. Available: https://aclanthology.org/2020.acl-main.740

[96] Y. Wang, J. Li, H. Wang, Y. Qian, C. Wang, and Y. Wu, "Wav2vec-switch: Contrastive learning from original-noisy speech pairs for robust speech recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7097–7101.

[97] Q.-S. Zhu, J. Zhang, Z.-Q. Zhang, and L.-R. Dai, "Joint training of speech enhancement and self-supervised model for noise-robust asr," *arXiv preprint arXiv:2205.13293*, 2022.

[98] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[99] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 750–15 758.

[100] D. Snyder, G. Chen, and D. Povey, "MUSAN: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[101] G. Wichern, J. Antognini, M. Flynn, L. R. Zhu, E. McQuinn, D. Crow, E. Manilow, and J. L. Roux, "Wham!: Extending speech separation to noisy environments," *arXiv preprint arXiv:1907.01160*, 2019.

[102] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "FSD50K: an open dataset of human-labeled sound events," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 829–852, 2022.

[103] H. Dubey, V. Gopal, R. Cutler, A. Aazami, S. Matusevych, S. Braun, S. E. Eskimez, M. Thakker, T. Yoshioka, H. Gamper *et al.*, "Icassp 2022 deep noise suppression challenge," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 9271–9275.

[104] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[105] E. Kharitonov, M. Rivière, G. Synnaeve, L. Wolf, P.-E. Mazaré, M. Douze, and E. Dupoux, "Data augmenting contrastive learning of speech representations in the time domain," *arXiv preprint arXiv:2007.00991*, 2020.

[106] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'chime'speech separation and recognition challenge: Dataset, task and baselines," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 504–511.

[107] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[108] A. R. *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021.

[109] H.-H. Wu, P. Seetharaman, K. Kumar, and J. P. Bello, "Wav2clip: Learning robust audio representations from clip," in *ICASSP*, 2022.

[110] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "Audioclip: Extending clip to image, text and audio," in *ICASSP*, 2022.

[111] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[112] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[113] D. Harwath and J. R. Glass, "Deep multimodal semantic embeddings for speech and images," in *ASRU*, 2015.

[114] W.-N. Hsu, D. Harwath, C. Song, and J. Glass, "Text-free image-to-speech synthesis using learned segmental units," *arxiv:2012.15454*, 2020.

[115] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *CVPR*, 2015.

[116] R. Sanabria, A. Waters, and J. Baldridge, "Talk, don't write: A study of direct speech-based image retrieval," in *INTERSPEECH*, 2021.

[117] W.-N. Hsu, D. Harwath, and J. Glass, "Transfer learning from audio-visual grounding to speech recognition," *Interspeech*, 2019.

[118] P.-A. Duquenne, H. Gong, and H. Schwenk, "Multimodal and multilingual embeddings for large-scale speech mining," in *NeurIPS*, 2021.

[119] S. Khurana, A. Laurent, and J. Glass, "SAMU-XLSR: Semantically-aligned multimodal utterance-level cross-lingual speech representation," *arxiv preprint*, 2022.

[120] H. Kamper, G. Shakhnarovich, and K. Livescu, "Semantic speech retrieval with a visually grounded model of untranscribed speech," *IEEE Trans. Audio, Speech, Language Process.*, vol. 27, no. 1, pp. 89–98, 2019.

[121] A. Pasad, B. Shi, H. Kamper, and K. Livescu, "On the contributions of visual and textual supervision in low-resource semantic speech retrieval," *INTERSPEECH*, 2019.

[122] K. Olaleye, D. Oneata, and H. Kamper, "Keyword localisation in untranscribed speech using visually grounded speech models," *JSTSP*, 2022.

[123] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[124] Y. Zhu, J. Feng, C. Zhao, M. Wang, and L. Li, "Counter-interference adapter for multilingual machine translation," *arXiv preprint arXiv:2104.08154*, 2021.

[125] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "Mad-x: An adapter-based framework for multi-task cross-lingual transfer," *arXiv preprint arXiv:2005.00052*, 2020.

[126] B. Thomas, S. Kessler, and S. Karout, "Efficient adapter transfer of self-supervised speech models for automatic speech recognition," in *ICASSP*, 2022.

[127] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *NIPS*, vol. 33, 2020.

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

[129] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[130] C.-L. Fu, Z.-C. Chen, Y.-R. Lee, and H.-y. Lee, "AdapterBias: Parameter-efficient token-dependent representation shift for adapters in NLP tasks," in *Findings of the Association for Computational Linguistics: NAACL 2022*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 2608–2621. [Online]. Available: https://aclanthology.org/2022.findings-naacl.199

[131] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=0RDcd5Axok

[132] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[133] R. He, L. Liu, H. Ye, Q. Tan, B. Ding, L. Cheng, J.-W. Low, L. Bing, and L. Si, "On the effectiveness of adapter-based tuning for pretrained language model adaptation," *arXiv preprint arXiv:2106.03164*, 2021.

[134] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT understands, too," *CoRR*, vol. abs/2103.10385, 2021.

[135] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *CoRR*, vol. abs/2110.07602, 2021.

[136] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *EMNLP (1)*. Association for Computational Linguistics, 2021, pp. 3045–3059.

[137] K. Lakhotia, E. Kharitonov, W. Hsu, Y. Adi, A. Polyak, B. Bolte, T. A. Nguyen, J. Copet, A. Baevski, A. Mohamed, and E. Dupoux, "Generative spoken language modeling from raw audio," *CoRR*, vol. abs/2102.01192, 2021.

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

[139] G. I. Winata, A. Madotto, Z. Lin, R. Liu, J. Yosinski, and P. Fung, "Language models are few-shot multilingual learners," *CoRR*, vol. abs/2109.07684, 2021.

[140] T. Schick and H. Schütze, "It's not just size that matters: Small language models are also few-shot learners," in *NAACL-HLT*. Association for Computational Linguistics, 2021, pp. 2339–2352.

[141] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," in *EMNLP (1)*. Association for Computational Linguistics, 2020, pp. 4222–4235.

[142] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020.

[143] G. F. Elsayed, I. J. Goodfellow, and J. Sohl-Dickstein, "Adversarial reprogramming of neural networks," in *ICLR (Poster)*. OpenReview.net, 2019.

[144] C. H. Yang, Y. Tsai, and P. Chen, "Voice2series: Reprogramming acoustic models for time series classification," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 11 808–11 819.

[145] H. Yen, P. Ku, C. H. Yang, H. Hu, S. M. Siniscalchi, P. Chen, and Y. Tsao, "A study of low-resource speech commands recognition based on adversarial reprogramming," *CoRR*, vol. abs/2110.03894, 2021.

[146] T. Schick and H. Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference," in *EACL*. Association for Computational Linguistics, 2021, pp. 255–269.

[147] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *CoRR*, vol. abs/1804.03209, 2018.

[148] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio, "Speech Model Pre-Training for End-to-End Spoken Language Understanding," in *INTERSPEECH*, 2019.

[149] C. Lai, Y. Chuang, H. Lee, S. Li, and J. R. Glass, "Semi-supervised spoken language understanding via self-supervised speech and language model pretraining," in *ICASSP*. IEEE, 2021, pp. 7468–7472.

[150] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *ACL/IJCNLP (1)*. Association for Computational Linguistics, 2021, pp. 3816–3830.

[151] P. Yang, X. Sun, W. Li, S. Ma, W. Wu, and H. Wang, "SGM: sequence generation model for multi-label classification," in *COLING*. Association for Computational Linguistics, 2018, pp. 3915–3926.

[152] Z. Du, Y. Qian, X. Liu, M. Ding, J. Qiu, Z. Yang, and J. Tang, "All NLP tasks are generation tasks: A general pretraining framework," *CoRR*, vol. abs/2103.10360, 2021.

[153] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "Ptr: Prompt tuning with rules for text classification," *arXiv preprint arXiv:2105.11259*, 2021.

[154] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H. Hon, "Unified language model pre-training for natural language understanding and generation," in *NeurIPS*, 2019, pp. 13 042–13 054.

[155] H. Bao, L. Dong, F. Wei, W. Wang, N. Yang, X. Liu, Y. Wang, J. Gao, S. Piao, M. Zhou *et al.*, "Unilmv2: Pseudo-masked language models for unified language model pre-training," in *International Conference on Machine Learning*. PMLR, 2020, pp. 642–652.

[156] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *ACL*. Association for Computational Linguistics, 2020, pp. 7871–7880.

[157] C. Qin and S. Joty, "LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5," in *International Conference on Learning Representations*, 2022.

[158] A. Mohamed, H.-y. Lee, L. Borgholt, J. D. Havtorn, J. Edin, C. Igel, K. Kirchhoff, S.-W. Li, K. Livescu, L. Maaløe *et al.*, "Self-supervised speech representation learning: A review," *arXiv preprint arXiv:2205.10643*, 2022.

[159] M. Marge, C. Espy-Wilson, N. G. Ward *et al.*, "Spoken language interaction with robots: Recommendations for future research," *Computer Speech & Language*, vol. 71, p. 101255, 2022.

[160] N. G. Ward, *Prosodic Patterns in English Conversation*. Cambridge University Press, 2019.

[161] O. Niebuhr, "Stepped intonation contours: A new field of complexity," in *Tackling the Complexity of Speech*, R. Skarnitzl and O. Niebuhr, Eds. Charles University Press, 2015, pp. 39–74.

[162] J. Weston, R. Lenain, U. Meepegama, and E. Fristed, "Learning de-identified representations of prosody from raw audio," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 11 134–11 145. [Online]. Available: https://proceedings.mlr.press/v139/weston21a.html

[163] D. Aguirre, N. Ward, J. E. Avila, and H. Lehnert-LeHouillier, "Comparison of models for detecting off-putting speaking styles," *Proc. Interspeech*, pp. 2303–2307, 2022.

[164] J. Shah, Y. K. Singla, C. Chen, and R. R. Shah, "What all do audio transformer models hear? probing acoustic representations for language delivery and its structure," *arXiv preprint arXiv:2101.00387*, 2021.

[165] Y.-A. Chung, Y. Belinkov, and J. Glass, "Similarity analysis of self-supervised speech representations," in *ICASSP*, 2021.

[166] Y.-A. Chung, H. Tang, and J. Glass, "Vector-Quantized Autoregressive Predictive Coding," in *INTERSPEECH*, 2020.

[167] A. H. Liu, Y.-A. Chung, and J. Glass, "Non-Autoregressive Predictive Coding for Learning Speech Representations from Local Dependencies," in *INTERSPEECH*, 2021.

[168] 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.

[169] 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.

[170] M. Riviere, A. Joulin, P.-E. Mazaré, and E. Dupoux, "Unsupervised pretraining transfers well across languages," in *ICASSP*, 2020.

[171] S. Schneider, A. Baevski, R. Collobert *et al.*, "wav2vec: Unsupervised pre-training for speech recognition." in *Interspeech*, 2019.

[172] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *ICLR*, 2020.

[173] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, J. Wu, L. Zhou, S. Ren, Y. Qian, Y. Qian, J. Wu, M. Zeng, and F. Wei, "WavLM: Large-scale self-supervised pre-training for full stack speech processing," 2021.

[174] A. B. Zadeh, P. P. Liang, S. Poria, E. Cambria, and L.-P. Morency, "Multimodal language analysis in the wild: CMU-MOSEI dataset and interpretable dynamic fusion graph," in *ACL*, 2018.

[175] S. Castro, D. Hazarika, V. Pérez-Rosas, R. Zimmermann, R. Mihalcea, and S. Poria, "Towards multimodal sarcasm detection (an _Obviously_ perfect paper)," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4619–4629. [Online]. Available: https://aclanthology.org/P19-1455

[176] S. Park, H. S. Shim, M. Chatterjee, K. Sagae, and L.-P. Morency, "Computational analysis of persuasiveness in social multimedia: A novel dataset and multimodal prediction approach," in *Proceedings of the 16th International Conference on Multimodal Interaction*, 2014, pp. 50–57.

[177] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A corpus derived from librispeech for text-to-speech," *CoRR*, vol. abs/1904.02882, 2019. [Online]. Available: http://arxiv.org/abs/1904.02882

[178] N. Ding, S.-w. Tian, and L. Yu, "A multimodal fusion method for sarcasm detection based on late fusion," *Multimedia Tools and Applications*, vol. 81, no. 6, pp. 8597–8616, 2022.

[179] B. Nojavanasghari, D. Gopinath, J. Koushik, T. Baltrušaitis, and L.-P. Morency, "Deep multimodal fusion for persuasiveness prediction," in *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, 2016, pp. 284–288.

[180] D. Talkin, "Reaper: Robust epoch and pitch estimator," 2015, https://github.com/google/REAPER.

[181] 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.

[182] Y. Shi, H. Bu, X. Xu, S. Zhang, and M. Li, "Aishell-3: A multi-speaker Mandarin TTS corpus and the baselines," 2015. [Online]. Available: https://arxiv.org/abs/2010.11567

[183] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "MLS: A large-scale multilingual dataset for speech research," in *INTERSPEECH*, 2020.

[184] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," *CoRR, abs/2006.13979*, 2020.

[185] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *ICASSP*, 2013.

[186] L. Lamel, J.-L. Gauvain, and G. Adda, "Lightly supervised and unsupervised acoustic model training," *Computer Speech & Language*, 2002.

[187] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Interspeech*, 2016.

[188] Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, and M. Auli, "Self-training and pre-training are complementary for speech recognition," *arXiv preprint arXiv:2010.11430*, 2020.

[189] A. Carmantini, P. Bell, and S. Renals, "Untranscribed web audio for low resource speech recognition." in *Interspeech*, 2019.

[190] E. Wallington, B. Kershenbaum, P. Bell, and O. Klejch, "On the learning dynamics of semi-supervised training for ASR," in *Interspeech*, 2021.

[191] I. Caswell, T. Breiner, D. van Esch, and A. Bapna, "Language id in the wild: Unexpected challenges on the path to a thousand-language web text corpus," in *ACL*, 2020.

[192] T. Reitmaier, E. Wallington, D. Kalarikalayil Raju, O. Klejch, J. Pearson, M. Jones, P. Bell, and S. Robinson, "Opportunities and challenges of automatic speech recognition systems for low-resource language speakers," in *CHI*, 2022.

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

[194] K. D. N, P. Wang, and B. Bozza, "Using Large Self-Supervised Models for Low-Resource Speech Recognition," in *Interspeech*, 2021.

[195] S. Kessler, B. Thomas, and S. Karout, "An Adapter Based Pre-Training for Efficient and Scalable Self-Supervised Speech Representation Learning," in *ICASSP*, 2022.

[196] J.-H. Lee, C.-W. Lee, J.-S. Choi, J.-H. Chang, W. K. Seong, and J. Lee, "CTRL: Continual Representation Learning to Transfer Information of Pre-trained for WAV2VEC 2.0," *Interspeech*, 2022.

[197] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, no. C, May 2019. [Online]. Available: https://doi.org/10.1016/j.neunet.2019.01.012

[198] H. Y. Chan and P. Woodland, "Improving broadcast news transcription by lightly supervised discriminative training," in *ICASSP*, 2004.

[199] V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Semi-supervised training of acoustic models using lattice-free MMI," in *ICASSP*, 2018.

[200] O. Klejch, E. Wallington, and P. Bell, "The CSTR System for Multilingual and Code-Switching ASR Challenges for Low Resource Indian Languages," in *Interspeech*, 2021.

[201] E. Yılmaz, A. Biswas, E. van der Westhuizen, F. de Wet, and T. Niesler, "Building a Unified Code-Switching ASR System for South African Languages," in *Interspeech*, 2018.

[202] A. Biswas, E. Yilmaz, F. De Wet, E. Van der westhuizen, and T. Niesler, "Semi-supervised Development of ASR Systems for Multilingual Code-switched Speech in Under-resourced Languages," in *LREC*, 2020.

[203] N. Wilkinson, A. Biswas, E. Yilmaz, F. De Wet, E. Van der westhuizen, and T. Niesler, "Semi-supervised Acoustic Modelling for Five-lingual Code-switched ASR using Automatically-segmented Soap Opera Speech," in *SLTU & CCURL*, 2020.

[204] E. Barnard, M. H. Davel, C. van Heerden, F. De Wet, and J. Badenhorst, "The nchlt speech corpus of the south african languages," in *SLTU*, 2014.

[205] P. Bell, M. J. Gales, T. Hain, J. Kilgour, P. Lanchantin, X. Liu, A. McParland, S. Renals, O. Saz, M. Wester *et al.*, "The MGB challenge: Evaluating multi-genre broadcast media recognition," in *ASRU*, 2015.

[206] W. Han, Z. Zhang, Y. Zhang *et al.*, "Contextnet: Improving convolutional neural networks for automatic speech recognition with global context," *Interspeech*, 2020.

[207] 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.

[208] P. Guo, F. Boyer, X. Chang *et al.*, "Recent developments on ESPnet toolkit boosted by conformer," in *ICASSP*, 2021.

[209] L. Grenoble, P. Austin, and J. Sallabank, "Handbook of endangered languages," 2011.

[210] A. Michaud, E. Castelli *et al.*, "Towards the automatic processing of Yongning Na (Sino-Tibetan): developing a light acoustic model of the target language and testing heavyweight models from five national languages," in *SLTU*, 2014.

[211] J. Shi, J. Amith, R. Castillo Garcia *et al.*, "Leveraging end-to-end asr for endangered language documentation: An empirical study on yoloxochitl mixtec," in *EACL*, 2020.

[212] J. S. Garofolo, "TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1," *Linguistic Data Consortium*, 1993. [Online]. Available: https://catalog.ldc.upenn.edu/LDC93S1

[213] H. Gelas, L. Besacier, and F. Pellegrino, "Developments of Swahili resources for an automatic speech recognition system," in *Spoken Language Technologies for Under-Resourced Languages*, 2012.

[214] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[215] Q. Xu, T. Likhomanenko, J. Kahn, A. Hannun, G. Synnaeve, and R. Collobert, "Iterative Pseudo-Labeling for Speech Recognition," in *INTERSPEECH*, 2020.

[216] D. Povey, A. Ghoshal, G. Boulianne *et al.*, "The Kaldi speech recognition toolkit," in *ASRU*, 2011.

[217] S. Watanabe, H. Hori, S. Karita *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Interspeech*, 2018.

[218] F. Kuang, M. Song, H. Qiu, and D. Povey, "k2," *https://github.com/k2-fsa/k2*, 2020.

[219] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *NIPS*, 2019.

[220] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[221] J. D. Kahn, V. Pratap, T. Likhomanenko, Q. Xu, A. Hannun, J. Cai, P. Tomasello, A. Lee, E. Grave, G. Avidov *et al.*, "Flashlight: Enabling innovation in tools for machine learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 10 557–10 574.

[222] "Can, dogan and martinez, victor r and papadopoulos, pavlos and narayanan, shrikanth s," in *ICASSP*, 2018.

[223] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5769–5779, 2017.

[224] A. Graves, "Supervised sequence labelling with recurrent neural networks," Ph.D. dissertation, Technical University of Munich, 2008.

[225] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[226] D. Povey, M. Hannemann, G. Boulianne *et al.*, "Generating exact lattices in the wfst framework," in *ICASSP*, 2012.

[227] H. Xu, T. Chen, D. Gao *et al.*, "A pruned RNNLM lattice-rescoring algorithm for automatic speech recognition," in *ICASSP*, 2018.

[228] K. Li, D. Povey, and S. Khudanpur, "A parallelizable lattice rescoring strategy with neural language models," in *ICASSP*, 2021.

[229] C.-K. Yeh, J. Chen, C. Yu, and D. Yu, "Unsupervised speech recognition via segmental empirical output distribution matching," *ICLR*, 2018.

[230] J. Kahn *et al.*, "Libri-light: A benchmark for ASR with limited or no supervision," in *ICASSP*, 2020.

[231] G. Chen, S. Chai, G.-B. Wang, J. Du, W.-Q. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang, M. Jin, S. Khudanpur, S. Watanabe, S. Zhao, W. Zou, X. Li, X. Yao, Y. Wang, Z. You, and Z. Yan, "GigaSpeech: An Evolving, Multi-Domain ASR Corpus with 10,000 Hours of Transcribed Audio," in *INTERSPEECH*, 2021.

[232] C. Wang, M. Riviere, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," in *ACL*, 2021.

[233] Y. Lee, S. Shon, and T. Kim, "Learning pronunciation from a foreign language in speech synthesis networks," *arXiv preprint arXiv:1811.09364*, 2018.

[234] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional Transformers for language understanding," in *NAACL*, 2019.

[235] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 2015, pp. 2048–2057.

[236] D. Yu and L. Deng, *Automatic speech recognition*. Springer, 2016, vol. 1.

[237] T. Qian, J. Shi, S. Guo, P. Wu, and Q. Jin, "Training strategies for automatic song writing: A unified framework perspective," in *ICASSP*, 2022.

[238] L. Gomez, Y. Patel, M. Rusinol, D. Karatzas, and C. Jawahar, "Self-supervised learning of visual features through embedding images into text topic spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4230–4239.

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

[240] X. Chang, T. Maekaku, P. Guo, J. Shi, Y.-J. Lu, A. S. Subramanian, T. Wang, S.-w. Yang, Y. Tsao, H.-y. Lee *et al.*, "An exploration of self-supervised pretrained representations for end-to-end speech recognition," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 228–235.

[241] E. Bastianelli, A. Vanzo, P. Swietojanski, and V. Rieser, "Slurp: A spoken language understanding resource package," in *EMNLP*, 2020.

[242] A. Bapna, Y. an Chung, N. Wu, A. Gulati, Y. Jia, J. H. Clark, M. Johnson, J. Riesa, A. Conneau, and Y. Zhang, "SLAM: A unified encoder for speech and language modeling via speech-text joint pre-training," 2021.

[243] A. Bapna, C. Cherry, Y. Zhang, Y. Jia, M. Johnson, Y. Cheng, S. Khanuja, J. Riesa, and A. Conneau, "mslam: Massively multilingual joint pre-training for speech and text," *arXiv preprint arXiv:2202.01374*, 2022.

[244] M. Dinarelli, M. Naguib, and F. Portet, "Toward low-cost end-to-end spoken language understanding," *arXiv preprint arXiv:2207.00352*, 2022.

[245] L. Borgholt, J. D. Havtorn, M. Abdou, J. Edin, L. Maaløe, A. Søgaard, and C. Igel, "Do we still need automatic speech recognition for spoken language understanding?" *arXiv preprint arXiv:2111.14842*, 2021.

[246] A. Wu, C. Wang, J. Pino, and J. Gu, "Self-supervised representations improve end-to-end speech translation," *INTERSPEECH*, 2020.

[247] S. Arora, S. Dalmia, P. Denisov, X. Chang, Y. Ueda, Y. Peng, Y. Zhang, S. Kumar, K. Ganesan, B. Yan *et al.*, "ESPnet-SLU: Advancing spoken language understanding through espnet," in *ICASSP*, 2022.

[248] S. Arora, S. Dalmia, X. Chang, B. Yan, A. Black, and S. Watanabe, "Two-pass low latency end-to-end spoken language understanding," *INTERSPEECH*, 2022.

[249] Y.-A. Chung, C. Zhu, and M. Zeng, "SPLAT: Speech-language joint pre-training for spoken language understanding," in *NAACL*, 2021.

[250] Y. Huang, H.-K. Kuo, S. Thomas, Z. Kons, K. Audhkhasi, B. Kingsbury, R. Hoory, and M. Picheny, "Leveraging unpaired text data for training end-to-end speech-to-intent systems," in *ICASSP*, 2020.

[251] 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.

[252] G.-T. Lin, Y.-S. Chuang, H.-L. Chung, S.-w. Yang, H.-J. Chen, S.-W. Li, A. Mohamed, H.-y. Lee, and L.-s. Lee, "DUAL: Textless spoken question answering with speech discrete unit adaptive learning," *INTERSPEECH*, 2022.

[253] C.-J. Hsu, H.-L. Chung, H.-y. Lee, and Y. Tsao, "T5lephone: Bridging speech and text self-supervised models for spoken language understanding via phoneme level t5," *arXiv preprint arXiv:2211.00586*, 2022.

[254] K.-Y. Chen, C.-P. Tsai, D.-R. Liu, H.-Y. Lee, and L.-s. Lee, "Completely unsupervised speech recognition by a generative adversarial network harmonized with iteratively refined hidden Markov models," *INTERSPEECH*, pp. 1856–1860, 2019.

[255] A. H. Liu, T. Tu, H.-y. Lee, and L.-s. Lee, "Towards unsupervised speech recognition and synthesis with quantized speech representation learning," in *ICASSP*, 2020.

[256] H. Inaguma, S. Kiyono, K. Duh, S. Karita, N. Yalta, T. Hayashi, and S. Watanabe, "Espnet-st: All-in-one speech translation toolkit," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020, pp. 302–311.

[257] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "IEMOCAP: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, pp. 335–359, 2008.

[258] C. Wang, A. Wu, and J. Pino, "CoVoST 2 and massively multilingual speech-to-text translation," *INTERSPEECH*, 2021.

[259] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *EMNLP*, 2016, pp. 2383–2392.

[260] A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman, "Newsqa: A machine comprehension dataset," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017, pp. 191–200.

[261] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer, "Quac: Question answering in context," in *EMNLP*, 2018, pp. 2174–2184.

[262] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 291–306, 2022.

[263] A. Haque, M. Guo, P. Verma, and L. Fei-Fei, "Audio-linguistic embeddings for spoken sentences," in *ICASSP*, 2019.

[264] S. H. Dumpala, C. S. Sastry, R. Uher, and S. Oore, "On Combining Global and Localized Self-Supervised Models of Speech," in *Proc. Interspeech 2022*, 2022, pp. 3593–3597.