Unsupervised Speech Recognition

Alexei Baevski[△], **Wei-Ning Hsu**[△], **Alexis Conneau**[□*], **Michael Auli**[△]
[△] Facebook AI [□] Google AI

Abstract

Despite rapid progress in the recent past, current speech recognition systems still require labeled training data which limits this technology to a small fraction of the languages spoken around the globe. This paper describes wav2vec-U, short for wav2vec Unsupervised, a method to train speech recognition models without any labeled data. We leverage self-supervised speech representations to segment unlabeled audio and learn a mapping from these representations to phonemes via adversarial training. The right representations are key to the success of our method. Compared to the best previous unsupervised work, wav2vec-U reduces the phoneme error rate on the TIMIT benchmark from 26.1 to 11.3. On the larger English Librispeech benchmark, wav2vec-U achieves a word error rate of 5.9 on test-other, rivaling some of the best published systems trained on 960 hours of labeled data from only two years ago. We also experiment on nine other languages, including low-resource languages such as Kyrgyz, Swahili and Tatar. The code is available at `https://github.com/pytorch/fairseq/tree/master/examples/wav2vec/unsupervised`

1. Introduction

Speech recognition performance on the much studied English Librispeech benchmark (Panayotov et al., 2015) has seen rapid improvement over the last few years due to advances in model architectures (Dong et al., 2018; Synnaeve et al., 2020; Gulati et al., 2020), semi-supervised learning (Xu et al., 2020b; Park et al., 2020) and self-supervised learning (van den Oord et al., 2018; Chung and Glass, 2018; Chung et al., 2019b; Baevski et al., 2020c). However, all of these techniques require transcribed speech data which is not available for the vast majority of the nearly 7,000 languages of the world (Lewis et al., 2016). As a result, speech recognition technology is only available for about 125 different languages (Google, 2021). On the other hand, humans learn a lot about speech simply by listening to others around them and without explicit supervision (Werker and Tees, 1984; Hirsh-Pasek et al., 1987; Polka and Werker, 1994; Jusczyk et al., 1999; Johnson and Jusczyk, 2001).

Unsupervised learning has been very successful in machine translation resulting in systems that obtain remarkable accuracy given no labeled training data at all (Conneau et al., 2018; Lample et al., 2018; Artetxe et al., 2018). Inspired by this, there has been some work on unsupervised speech recognition based on learning to align unlabeled text and audio (Yeh et al., 2019) or adversarial learning (Liu et al., 2018; Chen et al., 2019). These approaches showed promising initial results but their error rates are still high, with evaluation being limited to the small-scale TIMIT benchmark.

In this work, we introduce a framework for unsupervised learning of speech recognition models. Wav2vec-U, or wav2vec Unsupervised, leverages self-supervised representations from wav2vec 2.0 (Baevski et al., 2020c) to embed the speech audio and to segment the

∗. Work done while at Facebook AI.

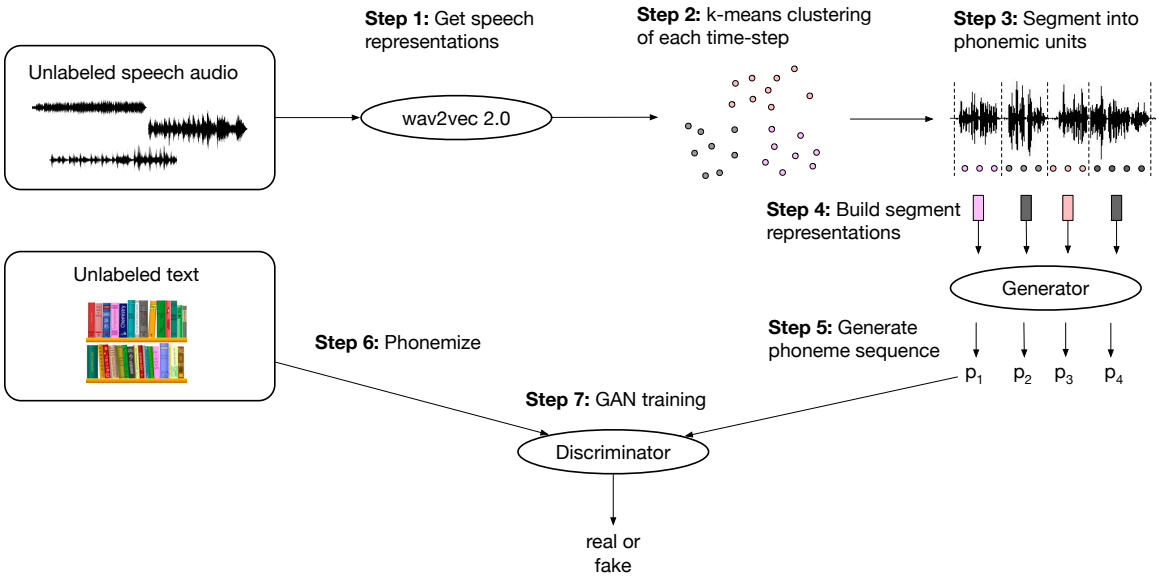

Figure 1: Illustration of wav2vec Unsupervised: we learn self-supervised representations with wav2vec 2.0 on unlabeled speech audio (Step 1), then identify clusters in the representations with k-means (Step 2) to segment the audio data (Step 3). Next, we build segment representations by mean pooling the wav2vec 2.0 representations, performing PCA and a second mean pooling step between adjacent segments (Step 4). This is input to the generator which outputs a phoneme sequence (Step 5) that is fed to the discriminator, similar to phonemized unlabeled text (Step 6) to perform adversarial training (Step 7).

audio into units with a simple k-means clustering method (see Figure 1 for an illustration of our approach). We find that the quality of the audio representations is key to the success of unsupervised speech recognition. Similar to Liu et al. (2018) and Chen et al. (2019), we learn a mapping between segments and phonemes using adversarial training but different to their work, we also enable the algorithm to ignore segments, as they may just correspond to silences or noise. We also introduce an unsupervised cross-validation metric to enable model development without labeled development data. Our unsupervised speech recognition model, the generator, is very lightweight: it consists of a single temporal convolution comprising only about 90k parameters to which we input frozen wav2vec 2.0 representations.

Experimental results demonstrate the viability of the framework for a variety of settings and languages. wav2vec-U improves the phoneme error rate (PER) on the small-scale TIMIT benchmark from 26.1 to 11.3 compared to the next best known unsupervised approach. To get a better sense of the performance compared to the best supervised methods, we measure performance on the larger Librispeech benchmark where our method achieves word error rate (WER) 5.9 on test-other. We also evaluate on six other European languages of the multilingual Librispsech benchmark (Pratap et al., 2020) and on three non-European low-resource languages.

2. Background

2.1 Supervised ASR

Automatic speech recognition (ASR) is the task of transcribing a speech waveform into a text transcript. In a supervised setup, a dataset $\mathcal{D}_l = \{(X_i, Y_i)\}_{i=1}^N$ of N speech X and text Y pairs are provided. Specifically, speech is usually represented as a sequence of feature vectors $X = [x_1, \cdots x_T] \in (\mathbb{R}^m)^*$, where each feature frame x_t is an m-dimensional continuous vector such as Mel-frequency cepstral coefficients (MFCC). Text is usually represented as a sequence of discrete units $Y = [y_1, \cdots, y_L] \in B^*$, where B denotes the vocabulary of text sequences (e.g., letters or words). The goal is to build a model, typically a probabilistic one such as $p_\theta(Y \mid X)$, to predict the transcription given speech audio. Such models are often classified into two categories: hybrid (Juang et al., 1986; Young, 1996; Povey, 2005; Hinton et al., 2012; Abdel-Hamid et al., 2012; Bourlard and Morgan, 2012) or end-to-end (Graves et al., 2006; Graves, 2012; Chorowski et al., 2015; Rao et al., 2017).

2.2 Hybrid System for Supervised ASR

Hybrid systems parameterize the *joint distribution* of speech and text $p_\theta(X, Y)$ with three components: an acoustic model (AM), a pronunciation model (PM), and a language model (LM) (Young, 1996). A language model describes the distribution over text sequences: $p_{LM}(Y) = \prod_{i=1}^L p(y_i \mid y_1, \cdots, y_{i-1})$. A pronunciation model is a lexicon mapping each word to its pronunciation represented as a sequence of phonemes, often created by linguists. For simplicity, we consider a deterministic lexicon here where each word only has one pronunciation, and therefore each word sequence Y can be transcribed into one phoneme sequence $P = [p_1, \cdots, p_M] = f_{PM}(Y) \in O^*$, where O is the phoneme inventory.

The last component, the acoustic model $p_{AM}(X \mid P)$, parameterizes the distribution over speech X given a phoneme sequence P. Since each phoneme $p \in O$ can generate a speech feature sequence of variable length, this component is often formulated as a linear hidden Markov model (HMM), where each phoneme has its corresponding HMM model and each state can transition to itself or the next state. The probability of a phoneme sequence is described by concatenating the corresponding HMM models (linking the forward transition from the last state of a phoneme to the first state of the next phoneme in that sequence). Let an alignment $A = [a_1, \cdots, a_T]$ denote a sequence of states from the concatenated HMM each speech frame x_t corresponds to, the joint probability of speech and alignment can be written as

$$\begin{aligned} p_{AM}(X, A \mid P) &= p_{emit}(X \mid A)\ p_{tran}(A \mid P) \\ &= \prod_t p_{emit}(x_t \mid a_t)\ p_{tran}(a_t \mid a_{t-1};\ P), \end{aligned}$$

where p_{tran} denotes the transition probability derived from the concatenated HMM model, and p_{emit} denotes the emission probability, which is often parameterized by a Gaussian mixture model (GMM).

To compute the acoustic model probability, one can marginalize over all alignments: $p_{AM}(X \mid P) = \sum_A p_{emit}(X \mid A)\ p_{tran}(A \mid P)$ efficiently with the forward-backward algorithm. Oftentimes, the probability is approximated with $p_{AM}(X \mid P) \approx \max_A p_{emit}(X \mid$

A) $p_{tran}(A \mid P)$, which can also be computed efficiently with the Viterbi algorithm. The resulting alignment, called the *forced alignment*, can be used to segment the speech by phone identities (Chen et al., 2019) or to create frame-level targets for training neural network acoustic model that parameterizes $p_{emit}(a_t \mid x_t)$ (Bourlard and Morgan, 2012).

2.2.1 Training a Hybrid System

Combining AM, PM and LM, a hybrid system computes the joint probability of speech and text as

$$\begin{aligned} p_\theta(X, Y) &= p_{AM}(X \mid f_{PM}(Y))\, p_{LM}(Y) \\ &= \sum_A p_{emit}(X \mid A)\, p_{tran}(A \mid f_{PM}(Y))\, p_{LM}(Y) \\ &\approx \max_A p_{emit}(X \mid A)\, p_{tran}(A \mid f_{PM}(Y))\, p_{LM}(Y). \end{aligned}$$

The acoustic and the language model are often learned separately with a maximal likelihood objective (Bahl et al., 1983; Juang et al., 1986):

$$p^*_{LM} = \arg\max_{p_{LM}} \sum_{i=1}^{N} \log p_{LM}(Y_i) \qquad p^*_{AM} = \arg\max_{p_{AM}} \sum_{i=1}^{N} \log p_{AM}(X_i \mid f_{PM}(Y_i)),$$

but alternative discriminative objectives for AMs taking LMs into account have also been explored (Bahl et al., 1986; Povey, 2005). Note that since training of language models requires only text data, one can leverage not only text from the paired data in $\mathcal{D}_l$, but also additional unpaired text data for training.

2.2.2 Decoding a Hybrid System with Weighted Finite State Transducers

Once the hybrid system is trained, decoding amounts to finding the most probable text sequence for a given speech sequence $\arg\max_Y p_\theta(Y \mid X)$, which is equivalent to finding the sequence with the highest joint probability $\arg\max_Y p_\theta(X, Y)$. In practice, decoding is often carried out by searching the minimal distortion path on a weighted finite state transducer (WFST; Mohri 1997; Mohri et al. 2002), obtained by composing WFSTs representing HMM emission, HMM transition, pronunciation model, and language model, respectively. Moreover, it is common to introduce a weighting factor α to balance the importance of acoustic and language model by re-writing the joint probability as

$$p_{\theta,\alpha}(X, Y) \propto p_{AM}(X \mid f_{PM}(Y))\, p_{LM}(Y)^\alpha, \tag{1}$$

where α is determined by minimizing the edit distance ED on a development set $\{(\tilde{X}_i, \tilde{Y}_i)\}_{i=1}^{\tilde{N}}$:

$$\alpha = \arg\min_\alpha \sum_{i=1}^{\tilde{N}} ED\left(\tilde{Y}_i,\ \arg\max_Y \left(\log p_{AM}(\tilde{X} \mid f_{PM}(Y)) + \alpha * \log p_{LM}(Y)\right)\right). \tag{2}$$

In our setting, we use either phoneme error rate (PER) or word error rate (WER) but other choices are possible. After α is determined, we decode an utterance X by finding

$$Y^* = \arg\max_Y p_{\theta,\alpha}(X, Y). \tag{3}$$

2.3 End-to-End Systems for Supervised ASR

More recently, end-to-end approaches that directly parametrize $p_\theta(Y \mid X)$ have gained increasing interest. This line of approaches includes but is not limited to connectionist temporal classification (CTC; Graves et al. 2006), recurrent neural network transducer (RNN-T; Graves 2012), sequence-to-sequence models (seq2seq; Chorowski et al. 2015). We focus our discussion on CTC, which is most relevant to this work.

2.3.1 Training a CTC Model

As mentioned above, the emission model in the hybrid system can be replaced with a neural network predicting the posterior probability over HMM states for each frame, but it requires a seed HMM-GMM model to derive the forced-alignment. Instead of maximizing the probability of the derived forced alignment to HMM states, CTC parameterizes the distribution over alignment to text sequences directly, and marginalizes over all possible alignments for the target text sequence during training to compute the posterior directly. Formally speaking, for an input speech sequence X of T frames, CTC predicts a distribution over $B' = B \cup \{\epsilon\}$ for each input step, where B is the text alphabet and ϵ is a special blank symbol, representing empty output. The probability of an alignment $A = [a_1, \cdots, a_T]$ is defined as $\prod_{t=1}^{T} p_\theta(a_t \mid X)$. Each alignment is mapped to a text sequence with a function g, which first removes consecutive repeating units and then removes all ϵ. For example, an alignment "cϵaaϵabb" is mapped to a text sequence "caab". The posterior probability of a text sequence Y given speech X of length T is therefore defined as:

$$p_\theta(Y \mid X) = \sum_{A: g(A)=Y, A \in (B')^T} \prod_{t=1}^{T} p_\theta(a_t \mid X), \tag{4}$$

and the marginalization on the right hand side can be computed efficiently with dynamic programming. Training of CTC optimizes the likelihood of the posterior distribution:

$$p_\theta^* = \arg\max_{p_\theta} \sum_{i=1}^{N} \log p_\theta(Y_i \mid X_i), \tag{5}$$

and decoding is approximated with finding the most probable alignment and mapping that alignment to a text sequence:

$$Y^* = g\left(\arg\max_{A} \prod_{t=1}^{T} p_\theta(a_t \mid X)\right) = g\left(\prod_{t=1}^{T} \arg\max_{a_t} p_\theta(a_t \mid X)\right).$$

2.3.2 Decoding CTC with a Language Model Using WFST

While CTC is motivated by end-to-end modeling of speech recognition, it can also fit nicely into the hybrid framework. The HMM associated with CTC has one state for each letter and a special state for the blank symbol (Zeyer et al., 2017). Each state can transit to any state including itself with the same transition probability. See Hannun (2017) and Hannun et al. (2020) Figure 3 for an illustration. As a result, one can also compose a CTC acoustic model with a letter-PM (mapping words to letters instead of phonemes) and an LM into a

single WFST for decoding, or train a CTC model that predicts phonemes to compose with regular PM and LM for decoding.

Likewise, a weighting factor α can be introduced to balance the CTC acoustic model and the language model. By re-arranging Equation 3 as

$$\begin{aligned} Y^* &= \arg\max_Y p_{AM}(X \mid f_{PM}(Y))\; p_{LM}(Y)^{\alpha} = \arg\max_Y p_\theta(X, Y)\; p_{LM}(Y)^{\alpha-1} \\ &= \arg\max_Y p_\theta(Y \mid X)\; p_{LM}(Y)^{\alpha-1} \\ &= \arg\max_Y \log p_\theta(Y \mid X) + (\alpha - 1) \log p_{LM}(Y), \end{aligned}$$

we can observe that this is equivalent to the formulation of language model fusion for end-to-end systems that combines posterior and prior probabilities of text for decoding (Chorowski and Jaitly, 2017).

2.4 Self-Training for Semi-Supervised ASR

In the semi-supervised setup, one is provided with unpaired speech $\mathcal{D}_u^s = \{X_j\}_{j=1}^{N_s}$ and text $\mathcal{D}_u^t = \{Y_j\}_{k=1}^{N_t}$ in addition to a labeled dataset $\mathcal{D}_l$. Most of the semi-supervised ASR studies focus on how to utilize unpaired speech, because it is straightforward to utilize additional text by training a better language model with both paired and unpaired text for decoding. Self-training is one of the simplest but most effective approaches to leveraging unpaired speech (Kahn et al., 2020a). It first trains a seed ASR model using labeled data $\mathcal{D}_l$, and then transcribes unpaired speech $\mathcal{D}_u^s$ into text $\{\hat{Y}_j\}_{j=1}^{N_s}$ using the trained model to generate a pseudo-labeled dataset $\hat{\mathcal{D}}_l = \{(X_j, \hat{Y}_j\}_{j=1}^{N_s}$. The transcribed text is often called the *pseudo labels*. Combining the real and the generated paired data, an ASR model can be trained with any supervised objective (Kahn et al., 2020a), or with separate objectives and weights (Veselỳ et al., 2017; Manohar et al., 2018; Hsu et al., 2020). This process can also can be repeated for multiple iterations to further improve performance (Xu et al., 2020b; Park et al., 2020).

3. Speech and Text Representations

In the following, we describe how we build suitable speech and text representations for unsupervised learning. Good representations are essential to learning a mapping from unlabeled speech audio to unlabeled text in a completely unsupervised fashion.

3.1 Self-supervised Learning of Speech Audio Representations

In a first step, we learn representations of the speech audio signal using self-supervised learning. There has been a lot of recent work in this direction which has shown strong performance in extremely low-labeled data setups across a range of languages (Conneau et al., 2020) and tasks (Fan et al., 2021; Pepino et al., 2021; Wang et al., 2021).

Wav2vec 2.0 consists of a convolutional feature encoder $f : \mathcal{X} \mapsto \mathcal{Z}$ that maps a raw audio sequence X to latent speech representations $z_1, \ldots, z_T$, which a Transformer $g : \mathcal{Z} \mapsto \mathcal{C}$ then turns into context representations $c_1, \ldots, c_T$ (Baevski et al., 2020b,a). Each z_t represents about 25ms of audio strided by 20ms and the Transformer architecture follows

BERT (Vaswani et al., 2017; Devlin et al., 2019). During training, latent representations are discretized to $q_1, \ldots, q_T$ with a quantization module $\mathcal{Z} \mapsto \mathcal{Q}$ to represent the targets in the objective. The quantization module uses a Gumbel softmax to choose entries from $E = 2$ codebooks with $V = 320$ entries each and the chosen entries are concatenated to obtain q (Jegou et al., 2011; Jang et al., 2016; Baevski et al., 2020b).

The model is trained to identify the true quantized latent q_t for each masked time-step within a set of distractors Q_t sampled from other masked time steps:

$$-\log \frac{\exp(sim(c_t, q_t))}{\sum_{\tilde{q} \sim Q_t} \exp(sim(c_t, \tilde{q}))}$$

where c_t is the output of the Transformer, and $sim(a, b)$ denotes cosine similarity.

This is augmented by a codebook diversity penalty to encourage the model to use all codebook entries (Dieleman et al., 2018). The codebook diversity penalty maximizes the entropy of the averaged softmax distribution over the codebook entries for each group $\bar{p}_g$ across a batch of utterances:

$$\frac{1}{EV} \sum_{g=1}^{E} -H(\bar{p}_g) = \frac{1}{EV} \sum_{g=1}^{E} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v}.$$

In our experiments, we use the publicly available English model pre-trained on 53k hours of Libri-Light (Kahn et al., 2020b) as well as XLSR-53 which was pre-trained on nearly 60k hours of speech audio in 53 languages (Conneau et al., 2020). Both use Large transformers with 24 blocks, model dimension 1,024, inner-dimension 4,096 and 16 attention heads (Baevski et al., 2020c).

3.2 Choosing Audio Representations

Next, we embed the speech audio using self-supervised representations from wav2vec 2.0. Before that, we also remove silences from the speech audio using an off-the-shelf unsupervised method. One exception is the TIMIT benchmark, where silences are part of the transcription. Silence removal is important to learning a better mapping between speech audio representations and transcriptions.

Removing Silences. Most datasets we use for our experiments have audio data with silences. However, these parts of the audio do not correspond to any transcription and we therefore remove silences as much as possible. To remove silences, we apply rVAD, an unsupervised voice activity detection (VAD) model which determines the segments in the audio data corresponding to silences, and we remove these sections (Tan et al., 2020). We ablate this choice in § 3.4.

Speech Audio Representations. After silence removal, we embed the unlabeled speech audio with wav2vec 2.0 to obtain speech representations. Specifically, we use the representations of the context Transformer network $c_1, \ldots, c_T$ (§ 3.1). Each c_t represents features computed from the entire utterance using self-attention and centered over the current time-step t; there is a 20ms gap between subsequent context representations (§ 3.1). The context network contains 24 Transformer blocks and we denote the output of block l at time-step t as c_t^l.

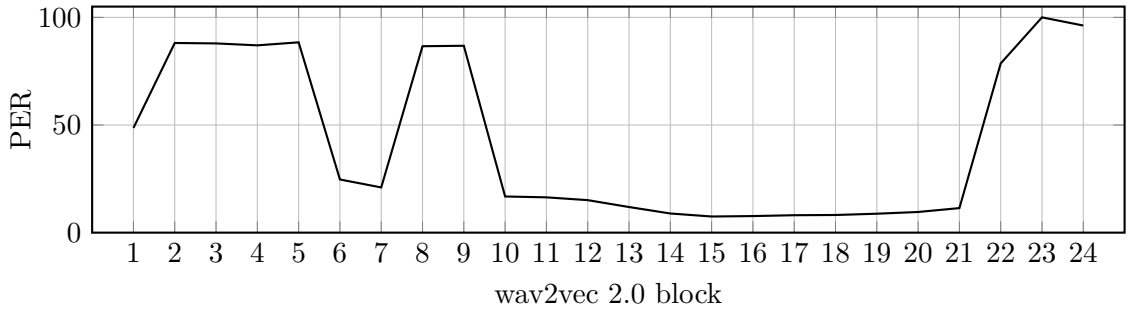

Figure 2: Supervised phoneme classification using representations from different wav2vec 2.0 blocks on dev-other of English Librispeech. Low and high blocks do not provide good features, while as blocks 14-19 do. Block 15 performs best.

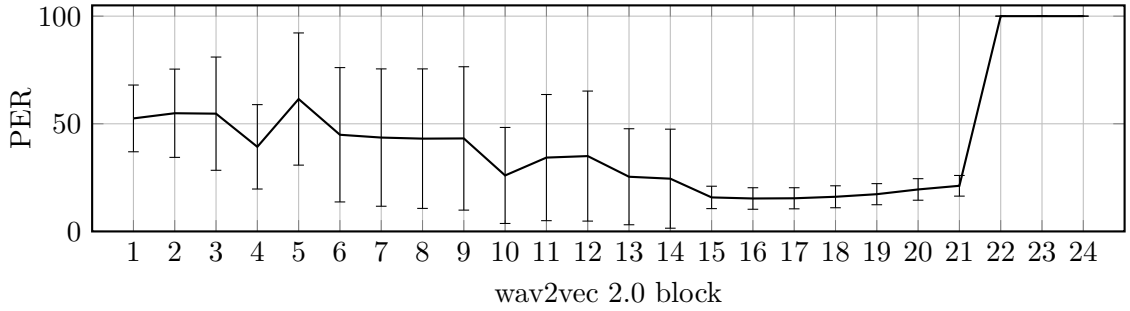

Figure 3: Supervised phoneme classification on eight languages of the MLS dataset in terms of mean PER and standard deviation for different wav2vec 2.0 blocks to represent the raw audio (cf. Figure 2). We consider English, German, Spanish, French, Italian, Dutch, Polish and Portuguese.

Our goal is to learn a model which can map from audio representations c_t^l to phonemes using no supervision. However, the representations of the uppermost block of wav2vec 2.0 may not be well suited for this task. These features are trained to directly predict masked latent representations spanning 25ms of speech audio which is much shorter than the typical duration of a phoneme. The uppermost block is therefore likely not well suited to learning a good mapping from the current time–step to phonemes. Similarly in natural language understanding, probing Transformer layers of the BERT models showed that blocks behave differently, with earlier blocks encoding syntactic information, while high-level semantic information appears at higher blocks (Tenney et al., 2019; Jawahar et al., 2019).

To get a better sense of this, we train supervised phoneme classifiers on top of the frozen representations of each of the 24 blocks of the English wav2vec 2.0 LARGE model pre-trained on Libri-Light. We then evaluate phoneme error rate (PER) with respect to the phonemized transcriptions of Librispeech dev-other. The classifier takes as input c_t^l and contains a single softmax-normalized linear layer mapping to the phoneme inventory.

Figure 2 shows that most of the first ten blocks as well as the final blocks provide very poor performance, while blocks 15-19 provide error rates below 9 PER. Block 15 achieves the best error rate of 7.5 PER. A similar insight has been used in the concurrent work

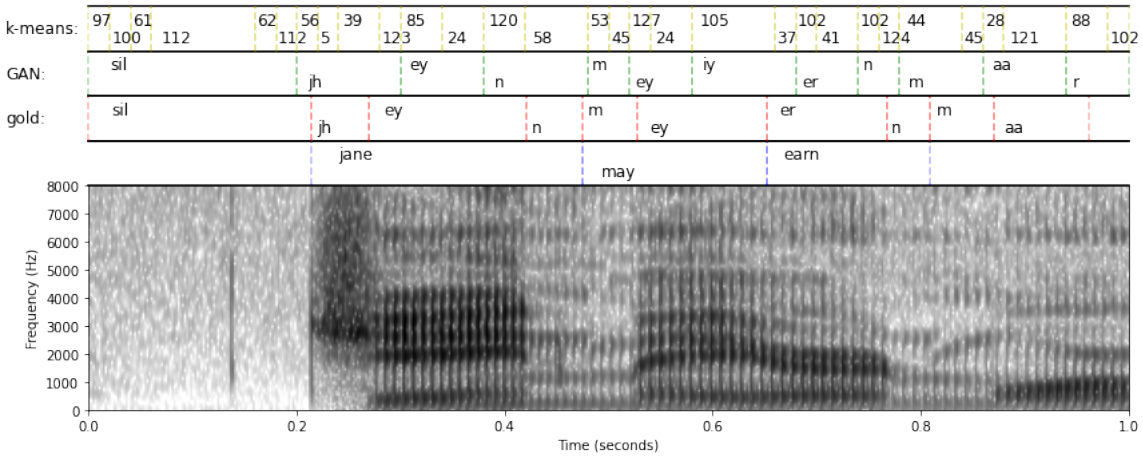

Figure 4: Example of segmenting the audio signal of the utterance *jane may earn more*. The top line shows the segmentation by the k-means segmentation method, the second line is the segmentation of Viterbi decoding with the GAN, and the third line shows the gold segmentation of human annotators from the TIMIT dataset. At the bottom, we show the corresponding spectogram, although, the input to our method is raw audio. The utterance has TIMIT ID MGLB0_SX4.

of Hsu et al. (2021b). Does this choice of representation generalize across languages? To get a sense of this, we measure PER for eight different languages of the MLS dataset (§ 5.1) when inputting representations of the multilingual wav2vec 2.0 XLSR-53 model. Figure 3 confirms that block 15 provides good performance across a range of languages and we will use block 15 as the representation for speech audio in all subsequent experiments. For brevity we drop the superscript l and refer to block 15 representations simply as $c_1, \ldots, c_T$.

3.3 Segmenting the Audio Signal

Once the speech signal is embedded, we identify segments corresponding to meaningful units that can be mapped to phonemes. Segmentation has been shown to be crucial in prior work (Chung et al., 2018) since the right boundaries in the input representations make it easier to predict the output sequence.

Identifying Speech Audio Segments. There has been a lot of prior work in unsupervised speech segmentation (Kamper et al., 2017a,b; Rasanen et al., 2015; Kreuk et al., 2020) but here we simply use a method based on clustering the wav2vec 2.0 speech representations $c_1, \ldots, c_T$. In a first step, we collect all the speech representations for the unlabeled speech data and perform k-means clustering to identify $K = 128$ clusters. We use the FAISS library to do fast clustering on GPUs (Johnson et al., 2019). Next, each c_t is labeled with the corresponding cluster ID $i_t \in \{1, \ldots, K\}$ and we introduce speech segment boundaries whenever the cluster ID changes.

Building Segment Representations. Once the speech audio representations are segmented, we compute a 512-dimensional PCA over all speech representations output by

Method	Precision	Recall	F1
DAVEnet + peak detection (Harwath and Glass, 2019)	.893	.712	.792
CPC + peak detection (Kreuk et al., 2020)	.839	.836	.837
k-means on wav2vec 2.0 features	.935	.379	.539
wav2vec-U Viterbi prediction	.598	.662	.629

Table 1: Quantitative evaluation of segment boundaries with respect to human labeled segment boundaries. We report precision, recall and f-measure using a 20ms tolerance.

wav2vec 2.0 for the training set. Next, we mean-pool the PCA representations for a particular segment to obtain an average representation of the segment. The PCA retains only the most important features and we found this to be effective. Segment boundaries are noisy due to the lack of supervision and we therefore found it useful to also mean-pool pairs of adjacent segment representations to increase robustness. This results in sequences of speech segment representation $S = s_1, \ldots, s_T, S \in \mathcal{S}$ for a given utterance.

Figure 4 illustrates how the k-means segmentation strategy results in very granular units compared to the gold segments corresponding to phonemes. Based on the k-means units, the unsupervised model can then recover segments that correspond very closely to phonemic units identified by humans.

Table 1 shows that k-means clustering results in very high precision but low recall when recovering gold phoneme boundaries on TIMIT. The Viterbi outputs of our model (wav2vec-U) result in more balanced, and better, accuracy because neighboring segments with the same predicted label are combined into a larger segment.

3.4 Pre-processing the Text Data

Similar to how we segment the unlabeled speech audio data into suitable units for unsupervised learning, we do the same for the unlabeled text data. We apply two pre-processing steps to the text data: phonemization and silence token insertion.

Phonemization. Phonemes characterize the different sounds which distinguish words from each other, e.g., for the word *cat* there are three phonemes corresponding to the three distinct sounds in the pronunciation of the word: /K/, /AE/, /T/. We phonemize the text data because we found it easier to learn a mapping between speech audio and the different sounds of a word rather than between audio and words or letters. Phonemization converts a sequence of words Y into a sequence of phonemes $P = [p_1, \cdots, p_M] \in O^*$, where O is the phoneme inventory. We use off-the-shelf tools for this step which we detail in § 5.2.

Silence token insertion. The unlabeled speech audio data is pre-processed by applying unsupervised silence removal. However, this process is not always accurate and many silences in the speech audio remain. To deal with this, we enable the unsupervised model to label some segments with a phonemic silence token (SIL; § 4.1). However, the phonemized unlabeled text data does not contain any silence tokens and this may pose difficulties for adversarial learning (§ 4). We remedy this by inserting silence markers into the phonemized unlabeled text data.

	PER
Baseline	21.4 ± 1.2
- begin/end SIL tokens	25.8 ± 0.7
- audio silence removal	29.3 ± 2.0

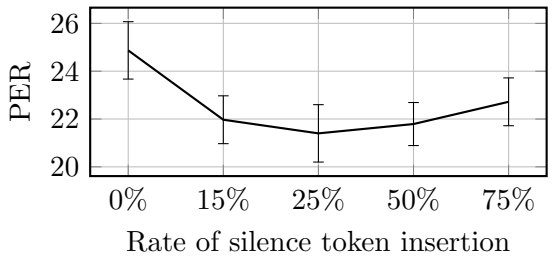

Figure 5: Unsupervised performance when augmenting the unlabeled text data with silence tokens. We add silence tokens to the unlabeled text to better resemble the speech audio which does contain silences. Silence tokens surrounding sentences and not removing silences from the audio results in better performance (left), and we show different rates of silence token insertion in the unlabeled text data (right). We report mean PER and standard deviation over 20 random seeds of unsupervised training on Librispeech dev-other.

First, we add a SIL token to the beginning and the end of all phonemized unlabeled text sentences. Figure 5 (left) shows that this improves accuracy. The same table shows the detrimental effect of not removing silences from the audio data (§ 3.2). Second, we randomly insert SIL between words, or groups of phonemes corresponding to words. Figure 5 (right) shows that inserting the silence token at a rate of 0.25 yields the best end accuracy.

4. Unsupervised Learning

We use adversarial training to train an unsupervised speech recognition model using the representations of the unlabeled speech audio data and the unlabeled phonemized text data (Liu et al., 2018; Chen et al., 2019). In the following, we detail the model architecture, the training objective as well as the unsupervised cross-validation metric we developed.

4.1 Model Architecture

Generative adversarial networks (GAN; Goodfellow et al. 2014) train a generator network $\mathcal{G}$ and a discriminator/critic network $\mathcal{C}$ where the generator produces samples which are then judged by the discriminator. The discriminator is trained to classify whether samples are from the generator or from the real data distribution. The objective of the generator is to produce samples that are indistinguishable by the discriminator.

Concretely, $\mathcal{G}$ takes as input a sequence of T segment representations $S = [s_1, \ldots, s_T]$ (§ 3.3) which are then mapped to a sequence of M phonemes $P = [p_1, \ldots, p_M]$. The generator predicts a distribution over the phoneme set O for each segment and outputs the phoneme with the highest probability. If the argmax prediction of consecutive segments result in the same phoneme, then we average the scores across all possible phoneme predictions for these segments, therefore $M \leq T$.

The phoneme set O includes a silence label SIL to enable labeling silences in the speech audio as such. Without a silence label, we noticed that the model was repurposing a particular phoneme to label silences which resulted in much lower performance since it interfered with subsequent language model decoding. In the backward pass, we back-propagate

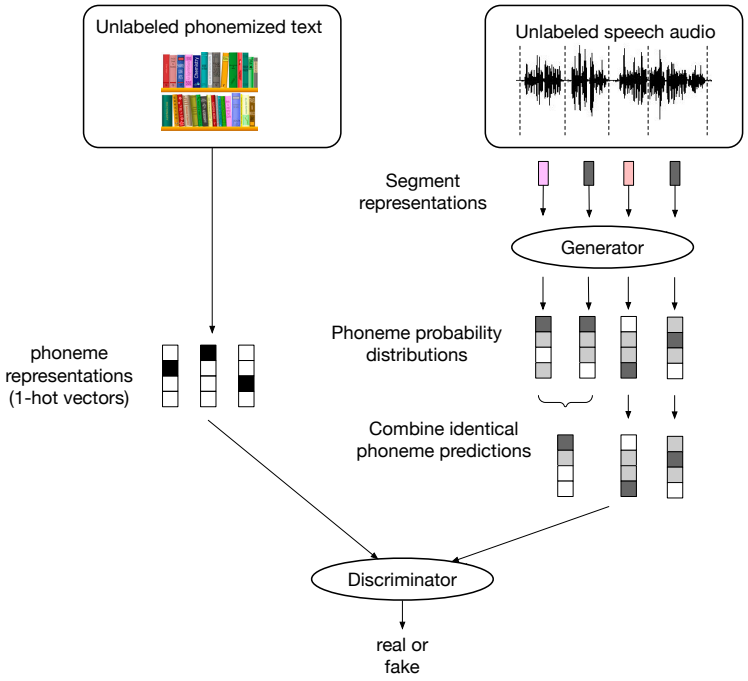

Figure 6: Illustration of how generator outputs and real phonemized text are turned into inputs to the discriminator. Real text is represented as a sequence of 1-hot vectors and generator outputs for different segment representations are collapsed if consecutive segments result in the same argmax prediction.

through one randomly chosen segment. We do not modify the segment representations during unsupervised training. The generator is parameterized as a single layer convolutional neural network (CNN; § 5.3).

The discriminator takes as input either a sequence $P^r \in \mathcal{P}^r$ of one-hot vectors denoting phonemized text from the real data distribution or a sequence of outputs from the generator P. Each input vector has $|O|$ dimensions to represent the distribution over phonemes for each segment (see Figure 6 for an illustration). The discriminator is also a CNN which outputs a probability indicating how likely the sample is to be from the data distribution (§ 5.3).

4.2 Objective

In our setup we use the original GAN objective with a gradient penalty (Goodfellow et al., 2014; Arjovsky et al., 2017), a segment smoothness penalty and a phoneme diversity penalty:

$$\min_{\mathcal{G}} \max_{\mathcal{C}} \mathop{\mathbb{E}}_{P^r \sim \mathcal{P}^r} [\log \mathcal{C}(P^r)] - \mathop{\mathbb{E}}_{S \sim \mathcal{S}} [\log (1 - \mathcal{C}(\mathcal{G}(S)))] - \lambda \mathcal{L}_{gp} + \gamma \mathcal{L}_{sp} + \eta \mathcal{L}_{pd}$$

where $P^r \in \mathcal{P}^r$ is phonemized unlabeled text, $\mathcal{G}(S)$ is the transcription output by the generator when input segment representations S for some unlabeled speech audio. The first term trains the discriminator to assign high probability to real transcriptions, the second term encourages the discriminator to assign low probability to generator outputs, $\mathcal{L}_{gp}$ is a

gradient penalty, $\mathcal{L}_{sp}$ is a smoothness penalty and $\mathcal{L}_{pd}$ is a phoneme diversity loss which we detail next. During training we alternate updates for the discriminator and the generator. We also alternate batches of predicted transcriptions from the generator and phonemized unlabeled text.

Gradient penalty. To stabilize training, we penalize the gradient norm of the discriminator with respect to the input (Gulrajani et al., 2017). The penalty is computed for random samples $\tilde{P} \in \tilde{\mathcal{P}}$ which are a linear combination of the activations of pairs of real and fake samples.[1]

$$\mathcal{L}_{gp} = \mathop{\mathbb{E}}_{\tilde{P} \sim \tilde{\mathcal{P}}} \left[\left(\|\nabla \mathcal{C}(\tilde{P})\| - 1 \right)^2 \right]$$

Segment smoothness penalty. The k-means segmentation of the speech audio is more granular than a typical phonemized transcription and neighboring representations are highly correlated. We therefore found it useful to add a penalty which encourages the generator to produce similar outputs for adjacent segments:

$$\mathcal{L}_{sp} = \sum_{(p_t, p_{t+1}) \in \mathcal{G}(S)} \|p_t - p_{t+1}\|^2$$

where $p_t \in \mathbb{R}^{|O|}$.

Phoneme diversity loss. We also found it helpful to penalize low usage of the phoneme vocabulary by the generator network on the batch level. In particular, we maximize the entropy of the averaged softmax distribution $H_{\mathcal{G}}(\mathcal{G}(S))$ of the generator over the phoneme vocabulary across a batch B of utterances:

$$\mathcal{L}_{pd} = \frac{1}{|B|} \sum_{S \in B} -H_{\mathcal{G}}(\mathcal{G}(S))$$

4.3 Unsupervised Cross-Validation Metric

Our goal is to build speech recognition models without any supervision. To this end, we developed the following cross-validation metric which does not require labeled data. We use the metric for early stopping and to select training hyper-parameters (λ, γ, η).

We consider two quantities in our metric: language model entropy and vocabulary usage. Entropy serves as an indicator of fluency for a given transcription and it is measured with a language model p_{LM} trained on phonemized text data (§ 3.4). Vocabulary usage is the proportion of the phoneme vocabulary being output by the model via Viterbi decoding. Measuring vocabulary usage identifies degenerate models which output fluent but trivial transcriptions. We denote Viterbi phoneme transcriptions for a given generator configuration $\mathcal{G}$ and unlabeled speech audio $\{X_j\}_{j=1}^{N_s}$ as $\mathcal{P} = \{P_j\}_{j=1}^{N_s}$. Entropy is measured in the standard way over the phonemized transcriptions: $H_{LM}(\mathcal{P}) = \frac{1}{N_s} \sum_{j=1}^{N_s} H_{LM}(P_j)$ where $H_{LM}(P) = -\frac{1}{M} \sum_{t=1}^{M} p_{LM}(p_t) \log p_{LM}(p_t)$ using $p_{LM}(p_t)$ as shorthand for $p_{LM}(p_t | p_{t-1}, \ldots, p_1)$.

In a first step, we generate phoneme transcriptions for different training checkpoints or hyper-parameter settings and denote the transcriptions of the configuration with the lowest

1. We simply shorten longer sequence if the lengths differ.

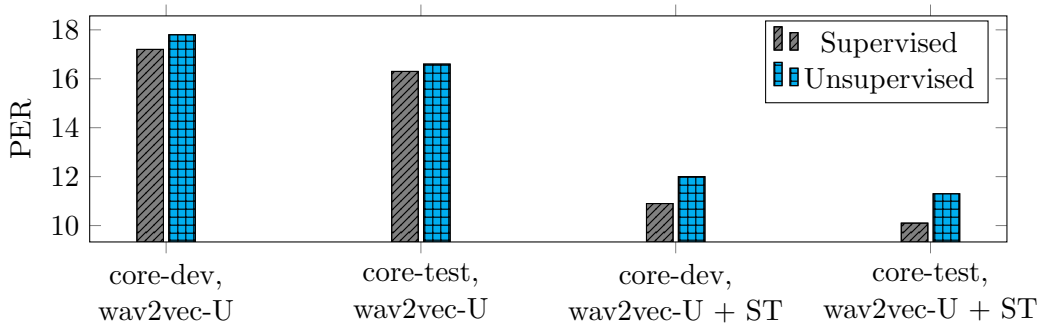

Figure 7: Effectiveness of the unsupervised cross-validation metric for model development compared to using a labeled development set (Supervised). We report PER on TIMIT core-dev/test (§ 5.1) for the GAN (wav2vec-U) and with self-training (wav2vec-U + ST).

vocabulary-usage adjusted entropy as $\hat{\mathcal{P}} = \arg\min_{\mathcal{P}} H(\mathcal{P}) - \log U(\mathcal{P})$ where $U(\mathcal{P}) \in [0, 1]$ is the vocabulary usage of $\mathcal{P}$.[2]

Next, we discard any model configurations which do not satisfy the following:

$$H(\mathcal{P}) < H(\hat{\mathcal{P}}) + \log\left(1.2 \times \frac{U(\mathcal{P})}{U(\hat{\mathcal{P}})}\right)$$

The last term introduces a margin over the lowest entropy transcription $H(\hat{\mathcal{P}})$ based on the vocabulary usage of $\mathcal{P}$ and $\hat{\mathcal{P}}$: If $U(\hat{\mathcal{P}})$ is much lower compared to $U(\mathcal{P})$, then we allow model configurations which produce transcriptions with higher entropy compared to $\hat{\mathcal{P}}$. However, if $U(\hat{\mathcal{P}})$ is a lot higher than $U(\mathcal{P})$, then the model configuration will not satisfy the constraint. The 1.2 factor permits slightly lower vocabulary usage for $\mathcal{P}$ compared to $\hat{\mathcal{P}}$.

In a final step, we take into account the length of the transcriptions: out of the configurations which satisfy the above constraint, we select the one which has the lowest sum of phoneme entropy scores per utterance:

$$\arg\min_{\mathcal{P}} H_{LM}(\mathcal{P}) = \frac{1}{N_s} \sum_{j=1}^{N_s} \sum_{t=1}^{M} p_{LM}(p_t^j) \log p_{LM}(p_t^j), M = |P^j|, P^j = [p_1^j, \ldots, p_M^j]$$

This selects model configurations which produce phoneme sequences that score high under the language model but are not too long.

How effective is this metric? To get a sense of this, we compare the performance of cross-validation with a labeled development to cross-validation with our unsupervised metric on the TIMIT benchmark. We cross-validate GAN hyper-parameters, different checkpoints for early stopping, language model decoding hyper-parameters (§ 5.4) and HMM decoding hyper-parameters for self-training (§ 5.5). Figure 7 shows that the unsupervised metric has only between 0.3-1.2 higher PER compared to using a labeled development set. This enables model development of unsupervised speech recognition systems without labeled data at only a small drop in accuracy compared to the ideal setting where labeled development data is available.

2. In practice, we used language model perplexity which is equivalent to entropy after taking the log.

5. Experimental Setup

5.1 Datasets

We consider several corpora and a variety of languages to evaluate our approach. TIMIT is a small English dataset on which previous unsupervised speech recognitition work was conducted. Librispeech is a standard benchmark with nearly 1,000 hours of labeled English speech audio and MLS is a multilingual benchmark with eight European languages. In addition, we also consider non-European languages from the ALFFA and CommonVoice corpora.

TIMIT. This dataset contains about five hours of audio recordings with time-aligned phonetic transcripts (Garofolo et al., 1993). To compare to prior work, we consider two setups: the *matched* setting uses text and speech from the same set of utterances to train the model while the *unmatched* setting ensures that the unlabeled text data does not contain the transcriptions of the audio data. For the matched setup, we follow the standard train/dev/test split of TIMIT as done in Yeh et al. (2019). This is 3,696/400/192 train/dev/test utterances which contains only SX (compact) and SI (diverse) sentences. For the unmatched setting, we follow Chen et al. (2019) by training on 3,000 speech utterances and 1,000 transcriptions from the training portion of the complete dataset split. We use the remaining 620 training utterances for validation, and test on 1,680 sentences for testing. The complete dataset split contains 4,620 training and 1,680 testing utterances, with additional SA (dialect) sentences.

Librispeech and Libri-Light. The Librispeech corpus contains 960 hours of transcribed speech audio (LS-960) for training. The data is based on read English audio books. For unsupervised training, we only use the speech audio data but not the transcriptions. We use the official Librispeech language modeling data as unlabeled text data with the Libri-Light data removed (Synnaeve et al., 2020).[3] This is a large text corpus of 635m words but we show that much smaller amounts of text and speech data still result in the same performance (§ 6.6). We evaluate on the standard dev-other/clean and test-clean/other sets. For development, we compute the unsupervised metric (§ 4.3) on dev-other. We also experiment with the audio data from Libri-Light (LL-60k) for which we follow the pre-processing of Kahn et al. (2020b) resulting in 53.2k hours of speech audio.

Multilingual LibriSpeech (MLS). The Multilingual Librispeech dataset (Pratap et al., 2020) is a large corpus of read audiobooks from Librivox in eight languages and we experiment with the following six languages: *Dutch (du), French (fr), German (de), Italian (it), Portuguese (pt), Spanish (es)*. The latest version of this corpus contains around 50k hours including 44k hours in English. However, for unsupervised learning we only use 100 hours of speech audio for each language. As unlabeled text data, we use the LM data provided by MLS.

ALFFA. We experiment with the Swahili data from the ALFFA project (Gelas et al., 2012; Abate et al., 2005; Tachbelie et al., 2014) which is read speech. There are 9.2 hours

3. https://github.com/flashlight/wav2letter/tree/master/recipes/sota/2019#non-overlap-lm-corpus-librispeech-official-lm-corpus-excluded-the-data-from-librivox

of speech audio training data and we use the language modeling data provided by ALFFA as unlabeled text data as well as newscrawl 2008-2014, 2018-2020.[4]

CommonVoice. This is a multilingual corpus of read speech for 38 languages (Ardila et al., 2020). We focus on two low-resource languages Kyrgyz (ky) and Tatar (tt) and use 1.8 hours and 4.6 hours of speech audio, respectively. As unlabeled text data for Kyrgyz we use the Kyrgyz community corpus 2017,[5] and newscrawl 2008-2014 and 2018-2020.[6] For Tatar we use the Tatar community corpus 2017,[7] and newscrawl 2005-2011.[8] We evaluate on the dev and test split of Rivière et al. (2020).

5.2 Phonemization

TIMIT provides time-aligned phonetic transcriptions annotated with an inventory of 60 phones adapted from the ARPAbet system, which treats silence as a phoneme. In addition, it includes a mapping from the 60 phoneme inventory to 48- and 39-phoneme inventories. Phoneme error rates are typically computed on the 39-phoneme inventory (Povey et al., 2011), which we map the phonetic transcripts to for training.

For Librispeech, we use the G2P phonemizer (Park and Kim, 2019) which uses the CMU dictionary to look up English word pronunciations, falling back to a neural network trained to output a phoneme sequence given a word. We phonemize the Librispeech LM corpus, with Librispeech and Librivox text data removed (Synnaeve et al., 2020). We convert the full phoneme set to a reduced set containing 39 phonemes by removing the numerical stress markers from the vowels.

For other corpora, including English in the MLS dataset, we use Phonemizer which supports a large number of various languages, but is less accurate than G2P for English.[9] We disable the language-switching labels and prune phonemes that appear fewer than 1000 times in the text corpus.

5.3 Unsupervised Training Details

Models are implemented in fairseq (Ott et al., 2019). The generator and discriminator are optimized with Adam (Kingma and Ba, 2015) using $\beta_1 = 0.5$ and $\beta_2 = 0.98$. The discriminator has a weight decay of $1e-4$ while the generator does not use weight decay. The discriminator is trained with a learning rate of $1e-5$ and the generator with $1e-4$, which are held constant throughout the training. We train for a total of 150k steps, during which we alternate optimizing the discriminator and the generator (so each are updated 75k times in total). Each training step is performed using a batch of 160 randomly chosen samples from the unlabeled audio data and 160 randomly chosen text samples from the unlabeled text data. Training takes about 12 hours on a single V100 GPU.

The discriminator is composed of three causal convolution blocks with a hidden size of 384 and a kernel size of 6, resulting in a receptive field size of 16 segments. The input into

4. http://data.statmt.org/news-crawl/sw
5. https://corpora.uni-leipzig.de?corpusId=kir_community_2017
6. http://data.statmt.org/news-crawl/ky
7. https://corpora.uni-leipzig.de?corpusId=tat_community_2017
8. https://corpora.uni-leipzig.de/en?corpusId=tat_news_2005-2011
9. https://github.com/bootphon/phonemizer

the discriminator is an $|O|$ dimensional vector representing the probability distribution over the phoneme vocabulary, and the output is a single logit for each time-step, indicating how likely the sample is to be from the data distribution. The first layer serves as embedding for the $|O|$ phonemes.

The generator is a single non-causal convolution with kernel size 4. The input to the generator are the segment representations S of dimension 512 and the output is an $|O|$ dimensional vector. The generator contains about 90k parameters and we do not backpropagate to the segment representations. We combine subsequent generator predictions prior to feeding them into the discriminator as described in § 4.1 and apply a softmax normalization. During training, we use dropout with $p = 0.1$ to the input of the generator (Srivastava et al., 2014).

For each language, we tune the following hyper-parameters using the unsupervised cross-validation metric (§ 4.3): the gradient penalty weight λ is selected from the range $[1.5, 2.0]$, the smoothness penalty weight γ from $[0.5, 0.75]$, the phoneme diversity loss weight η from $[2, 4]$, and we train 5 seeds for each configuration for a total of 40 models.

5.4 Decoding

We wish to decode the output of either phoneme-based models, resulting from unsupervised training, or letter-based models, resulting from subsequent self-training (§ 5.5) using a language model.

To do so we build WFSTs (§ 2.2.2) using PyKaldi (Can et al., 2018), a Python port of Kaldi (Povey et al., 2011). The WFST takes as input the model emissions and if we decode to words, then we use the same phonemizer with which we pre-processed the unlabeled text data to build a mapping between phonemes to words. The WFST is composed with a 4-gram language model (Heafield, 2011) pruned to keep only 4-grams occurring more than 3 times. We add self-loops that mimic CTC behavior (Zhang et al., 2020a) where blank symbols (silence for the GAN or actual blank symbol for letter models) are mapped to epsilons and consecutive predictions of the same symbol are collapsed.

During decoding, we provide acoustic scale as a parameter to the Kaldi decoder and we also add a scalar ν to the blank token emission (silence for GAN models and blank for others). We tune the optimal weights for these two parameters by minimizing a quantity that measures fluency of the output as well as faithfulness to the model output. In particular we minimize the following quantity on an unlabeled development set - assuming a phoneme-based model which we wish to decode to words:

$$\sum_{j=1}^{N_s} H_{LM}(\bar{P}_j) \times \max\left(ED(\bar{P}_j, P_j), \mu\right)$$

where $\{P_j\}_{j=1}^{N_s}$ are the Viterbi model outputs, $\{\bar{P}_j\}_{j=1}^{N_s}$ are the word-based outputs of the WFST converted to phonemes (or simply phoneme-based outputs if decoded to phonemes), $H_{LM}(P_j)$ is the entropy of a language model, ED is an edit distance such as PER for phonemes or character error rate for letter-based model models, and $\mu = 0.03$. In practice, trivial solutions may achieve very low entropy. We counteract this by replacing $H_{LM}(\bar{P}_j)$ by the average entropy of the language model training data if $H_{LM}(\bar{P}_j)$ is lower than the entropy of the training data. We tune acoustic scale in the interval $[0, 8]$, and ν in $[-3, 8]$.

For Librispeech experiments, we also decode with a Transformer language model (Baevski and Auli, 2018) trained on the Librispeech LM corpus using the beam search decoder of Pratap et al. (2019). The Transformer LM is identical to Synnaeve et al. (2020) and contains 20 blocks, model dimension 1,280, inner dimension 6,144 and 16 attention heads. We tune acoustic scale with a beam of 50 and test performance is measured with beam 500.

5.5 Self-Training

For self-training, we perform two iterations: first, we pseudo-label the training data with the unsupervised GAN model and train an HMM on the pseudo-labels (§ 2.2). Second, we relabel the training data with the HMM and then fine-tune the original wav2vec 2.0 model using the HMM pseudo-labels with a CTC loss (§ 2.3). HMM models use phonemes as output, while as wav2vec 2.0 models use letters. Both are decoded using WFST decoders into words. wav2vec 2.0 self-training for Librispeech uses the same fine-tuning parameters as the original wav2vec 2.0 model fine-tuned on 100 hours of Librispeech data, but we reduce masking probability to 0.25, reduce the batch size to 800k frames and train on 8 GPUs for 80k updates.[10] We use the last checkpoint instead of early stopping. For TIMIT self-training, we use the one hour fine-tuning parameters of the original wav2vec 2.0 model.[11] This performs 13k updates on 4 GPUs.

6. Results

We first evaluate our approach on the Librispeech benchmark to compare to competitive systems trained on 960 hours of labeled data (§ 6.1). This is followed by a comparison to prior unsupervised work on the small-scale TIMIT benchmark (§ 6.2). To get a sense of how well our approach works on non-English languages we evaluate on several European languages of the MLS benchmark (§ 6.3) as well as non-European low-resource languages (§ 6.4). We evaluate the effectiveness of different self-training strategies (§ 6.5) and show that good performance can be achieved with very little unlabeled data (§ 6.6).

6.1 Comparison to Supervised Speech Recognition on Librispeech

We first test our approach on Librispeech to get a sense of how viable unsupervised speech recognition can be compared to the best supervised systems trained on a large amount of labeled data. Librispeech is a standard benchmark in the speech recognition community which provides about 960 hours of transcribed read audiobooks. We use the language modeling data of Librispeech as unlabeled text data for unsupervised training.[12] We experiment with the frozen representations of a wav2vec 2.0 LARGE model trained on the 53.2k hours of Libri-Light (LL-60k) which we denote as wav2vec-U LARGE and we also consider self-training (§ 5.5).

10. https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/config/finetuning/vox_100h.yaml
11. https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/config/finetuning/vox_1h.yaml
12. Below we show that much less unlabeled text and speech audio are sufficient to reach a similar level of performance (§ 6.6).

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
960h - Supervised learning						
DeepSpeech 2 (Amodei et al., 2016)	-	5-gram	-	-	5.33	13.25
Fully Conv (Zeghidour et al., 2018)	-	ConvLM	3.08	9.94	3.26	10.47
TDNN+Kaldi (Xu et al., 2018)	-	4-gram	2.71	7.37	3.12	7.63
SpecAugment (Park et al., 2019)	-	-	-	-	2.8	6.8
SpecAugment (Park et al., 2019)	-	RNN	-	-	2.5	5.8
ContextNet (Han et al., 2020)	-	LSTM	1.9	3.9	1.9	4.1
Conformer (Gulati et al., 2020)	-	LSTM	2.1	4.3	1.9	3.9
960h - Self and semi-supervised learning						
Transf. + PL (Synnaeve et al., 2020)	LL-60k	CLM+Transf.	2.00	3.65	2.09	4.11
IPL (Xu et al., 2020b)	LL-60k	4-gram+Transf.	1.85	3.26	2.10	4.01
NST (Park et al., 2020)	LL-60k	LSTM	1.6	3.4	1.7	3.4
wav2vec 2.0 (Baevski et al., 2020c)	LL-60k	Transf.	1.6	3.0	1.8	3.3
wav2vec 2.0 + NST (Zhang et al., 2020b)	LL-60k	LSTM	1.3	2.6	1.4	2.6
Unsupervised learning						
wav2vec-U LARGE	LL-60k	4-gram	13.3	15.1	13.8	18.0
wav2vec-U LARGE + ST	LL-60k	4-gram	3.4	6.0	3.8	6.5
	LL-60k	Transf.	3.2	5.5	3.4	5.9

Table 2: WER on the Librispeech dev/test sets when using 960 hours of unlabeled audio data from Librispeech (LS-960) or 53.2k hours from Libri-Light (LL-60k) using representations from wav2vec 2.0 LARGE. Librispeech provides clean dev/test sets which are less challenging than the other sets. We report results for GAN training only (wav2vec-U) and with subsequent self-training (wav2vec-U + ST).

wav2vec-U LARGE with self-training (wav2vec-U + ST) and a Transformer language model achieves WER 5.9 on test-other, the noisy test set. This shows that unsupervised speech recognition can perform remarkably well compared to the best supervised systems of the recent past on this much studied benchmark. Also, self-training is effective even when the teacher model is unsupervised as per the improvement over GAN training (wav2vec-U). Interestingly, self-training on just Librispeech, or 960 hours of unlabeled speech audio, achieves already very good performance of WER 6.4 on dev-other compared to self-training on all of Libri-Light (53.2k hours) which compares at 6.0 WER. We note that the number of parameters trained during adversarial training is very small: the generator contains only about 90k parameters for a single temporal convolution mapping to the phoneme set from frozen wav2vec 2.0 representations.

6.2 Comparison to Prior Unsupervised Work

Prior work on unsupervised speech recognition focused on the TIMIT benchmark. In order to perform a direct comparison to these approaches, we report results on this benchmark as well. We consider two setups to compare to previous work: in the matched setting, the unlabeled text data is simply the transcriptions of the unlabeled audio data but unpaired. In the unmatched setup, the unlabeled text data does not contain the transcriptions for the audio data which is a more realistic setting.

Model	LM	core-dev	core-test	all-test
Supervised learning				
LiGRU (Ravanelli et al., 2018)	-	-	14.9	-
LiGRU (Ravanelli et al., 2019)	-	-	14.2	-
Self and semi-supervised learning				
vq-wav2vec (Baevski et al., 2020b)	-	9.6	11.6	-
wav2vec 2.0 (Baevski et al., 2020c)	-	7.4	8.3	-
Unsupervised learning - matched setup				
EODM (Yeh et al., 2019)	5-gram	-	36.5	-
GAN* (Chen et al., 2019)	9-gram	-	-	48.6
GAN + HMM* (Chen et al., 2019)	9-gram	-	-	26.1
wav2vec-U	4-gram	17.0	17.8	16.6
wav2vec-U + ST	4-gram	11.3	12.0	11.3
Unsupervised learning - unmatched setup				
EODM (Yeh et al., 2019)	5-gram	-	41.6	-
GAN* (Chen et al., 2019)	9-gram	-	-	50.0
GAN + HMM* (Chen et al., 2019)	9-gram	-	-	33.1
wav2vec-U*	4-gram	21.3	22.3	24.4
wav2vec-U + ST*	4-gram	13.8	15.0	18.6

Table 3: TIMIT Phoneme Error Rate (PER) in comparison to previous work for the matched and unmatched training data setups (§ 5.1). PER is measured on the standard Kaldi dev and test sets (core-dev/core-test) as well as a slightly larger version of the test set (all-test) as used by some of the prior work. (*) indicates experiments that do not use the standard split excluding SA utterances.

We measure performance on the standard Kaldi dev and test sets (core-dev/core-test) as well as a slightly larger version of the test set (all-test) to be able to compare to Liu et al. (2018) and Chen et al. (2019). Further details of the two setups can be found in § 5.1. We report performance for wav2vec-U with a 4-gram language model trained on the language modeling data of TIMIT and we also consider self-training (wav2vec-U + ST).

Table 3 shows that wav2vec-U outperforms prior unsupervised work in both the matched and unmatched settings, reducing PER on all-test in the matched setup by 57% relative compared to Chen et al. (2019). Our method has lower performance than the best supervised methods but it performs still very well at PER 12 on core-test in the matched setup compared to PER 8.3 for the state of the art (Baevski et al., 2020c).

6.3 Performance on non-English languages

To get a sense of how well the method works on non-English data, we experiment on six languages of the multilingual Librispeech corpus (MLS; Pratap et al. 2020). As baseline we consider the supervised systems of Pratap et al. (2020) trained on between 2k and 161 hours of labeled data, depending on the language. For adversarial learning we use 100 hours of unlabeled audio data from MLS for every language as well as the MLS language modeling

Model	Labeled data used	LM	de	nl	fr	es	it	pt	Avg
Labeled training hours (full)			2k	1.6k	1.1k	918	247	161	
Supervised learning									
Pratap et al. (2020)	full	5-gram	6.49	12.02	5.58	6.07	10.54	19.49	10.0
Unsupervised learning									
wav2vec-U	0h	4-gram	32.5	40.2	39.8	33.3	58.1	59.8	43.9
wav2vec-U + ST	0h	4-gram	11.8	21.4	14.7	11.3	26.3	26.3	18.6

Table 4: WER on the Multilingual Librispeech (MLS) dataset using representations from the wav2vec 2.0 XLSR-53 model. We consider German (de), Dutch (nl), French (fr), Spanish (es), Italian (it), Portuguese (pt).

Model	tt	ky
Supervised learning		
Fer et al. (2017)	42.5	38.7
m-CPC (Rivière et al., 2020)	42.0	41.2
XLSR-53 (Conneau et al., 2020)	5.1	6.1
Unsupervised learning		
wav2vec-U	25.7	24.1
wav2vec-U + HMM	13.7	14.9

Table 5: PER for low-resource languages, Tatar (tt) and Kyrgyz (ky).

Model	sw
Supervised learning	
Besacier et al. (2015)	27.36
Unsupervised learning	
wav2vec-U	52.6
wav2vec-U + ST	32.2

Table 6: WER for Swahili from the ALFFA corpus. We compare to the supervised baseline of the ALFFA project.

data. As input to wav2vec-U we use the representations from XLSR-53 (Conneau et al., 2020), a wav2vec 2.0 model pre-trained on 53 languages. Table 4 shows that wav2vec-U generalizes across a range of languages. Performance is lower than supervised systems but it shows the viability for other languages.

6.4 Application to Low-resource Languages

Experiments so far focused on European languages, for most of which relatively large amounts of labeled data exist. Next, we turn to three low-resource languages, Swahili, Kyrgyz, and Tatar. Swahili is an African language, Kyrgyz and Tatar are Turkic languages with only about 4.3m and 5.2m speakers, respectively.[13] We use between 1.8 hours (Kyrgyz) and 9.2 hours of unlabeled audio (Swahili), see § 5.1. To compare to prior work, we measure WER for Swahili and PER for Kyrgyz and Tatar. For Tatar and Kyrgyz we opted to use a reduced self-training regime for faster experimental turn-around where we only perform HMM self-training and we expect better performance with the full self-training setup (§ 6.5). Table 5 and Table 6 show that wav2vec-U achieves good performance on

13. `https://en.wikipedia.org/wiki/{Kyrgyz,Tatar}_language`

Model	LM	core-dev	core-test	all-test
wav2vec-U	4-gram	17.0	17.8	16.6
+ HMM	4-gram	13.7	14.6	13.5
+ HMM + HMM	4-gram	13.3	14.1	13.4
+ HMM resegment + GAN	4-gram	13.6	14.4	13.8
+ fine-tune	4-gram	12.0	12.7	12.1
+ fine-tune	-	12.1	12.8	12.0
+ fine-tune + fine-tune	-	12.0	12.7	12.0
+ HMM + fine-tune	-	11.3	11.9	11.3
+ HMM + fine-tune	4-gram	11.3	12.0	11.3

Table 7: PER on TIMIT for various self-training strategies. We compare the performance of just the GAN output (wav2vec-U) to one or two iterations of subsequent self-training with an HMM. We contrast this to using the HMM for re-segmenting the audio data as done in prior work (Chen et al., 2019). We also consider self-training based on fine-tuning the original wav2vec 2.0 model (fine-tune) in or two self-training iterations (Xu et al., 2020a) as well as a combination of HMM and fine-tuning-based self-training.

these low-resource languages compared to previous work that utilized labeled data. We note that for Tatar and Kyrgyz we use a much smaller amount of speech audio than prior work: compared to XLSR-53 we use 1.8h unlabeled data vs 17h of labeled data for Kyrgyz and 4.6h vs. 17h for Tatar.

6.5 Self-training Strategies

The self-training strategy we use is as follows: once the GAN is trained, we use it together with a language model to pseudo-label the unlabeled audio, then we train an HMM on the labels and repeat pseudo-labeling with the HMM in order to fine-tune the wav2vec 2.0 model whose representation were originally fed to the GAN. Finally, we use the fine-tuned wav2vec 2.0 model to decode the test set.

Table 7 shows that fine-tuning with an HMM (wav2vec-U + HMM) leads to substantial improvements, however, a second iteration of HMM self-training leads to much smaller additional gains (wav2vec-U + HMM + HMM). Using the HMM to re-segment the speech audio followed by repeated GAN training (wav2vec-U + HMM resegment + GAN), similar to Chen et al. (2019), does not improve performance over just HMM self-training.

Another option is to directly fine-tune wav2vec 2.0 on the labels assigned by the GAN model (wav2vec-U + fine-tune) and this performs very well. However, another round of self-training based on a fine-tuned wav2vec 2.0 model does not improve performance (wav2vec 2.0 + fine-tune + fine-tune). We believe that this is due to overfitting since the fine-tuned model has over 300m parameters. This is in line with recent observations about overfitting in self-training for speech recognition (Likhomanenko et al., 2021).

The HMM is less likely to overfit in the way the LARGE wav2vec 2.0 model does. We therefore found it effective to perform a single round of HMM self-training followed by wav2vec 2.0 fine-tuning (wav2vec-U + HMM + fine-tune). After two rounds of self-training,

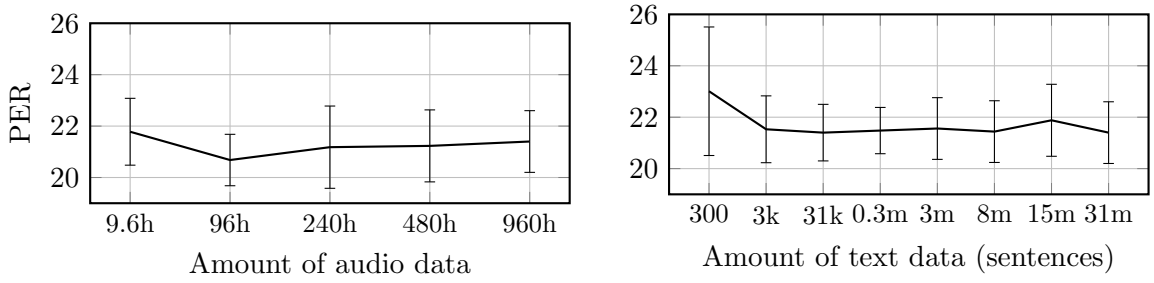

Figure 8: Effect of the amount of unlabeled audio data (left) and text data (right) on unsupervised training in terms of PER on Librispeech dev-other.

we do not require a language model anymore for this benchmark which is likely because the language model has been distilled into the model to a large degree.

6.6 Amount of Unlabeled Data Needed

For the experiments on Librispeech we used large amounts of unlabeled speech audio and text data for adversarial learning (960 hours of unlabeled speech audio and nearly 31m sentences of text data). For TIMIT we used much less data, only about 3.15h of speech audio and 140k phonemes of text data for the matched setup. Next, we perform controlled experiments on Librispeech to get a sense of how much data is sufficient to achieve good performance.

Figure 8 (left) shows that 9.6h of speech audio data still achieves excellent performance. Similarly, Figure 8 (right) shows that only about 3,000 sentences of text data are sufficient to achieve a similar level of accuracy as using all of the text data. We show further ablations in Appendix A.

7. Related Work

This paper builds on a large body of prior work which can be sub-divided into self-supervised learning, semi-supervised learning, unsupervised learning for speech recognition and unsupervised learning for machine translation.

Semi-supervised Speech Recognition. This is a long-standing research area in speech recognition with the most recent work focusing on self-training (Kahn et al., 2020a) and iterative self-training (Xu et al., 2020b) by carefully filtering the unlabeled data to match the target domain (Park et al., 2020). There has also been recent work on distilling knowledge from a strong prior such as a language model to provide a learning signal to a discriminative model trained on unlabeled speech (Hsu et al., 2020). Other work explored data augmentation via a technique similar to back-translation (Hayashi et al., 2018) as used in machine translation (Bojar and Tamchyna, 2011; Sennrich et al., 2015; Edunov et al., 2018). Another line of work composes text-to-speech and speech-to-text models to enforce cycle-consistency (Hori et al., 2019), similar to dual learning in machine translation (Xia et al., 2016).

Self-supervised Learning for Speech. Contrastive predictive coding (CPC; van den Oord et al. 2018) explored unsupervised or self-supervised representation learning for phoneme recognition. Their model architecture and loss function was simplified in Schneider et al. (2019) who also applied it to full speech recognition. Chung et al. (2019a) explores language model-style pre-training on speech audio data with recurrent neural networks and Chung and Glass (2018) learns fixed size representations of audio segments. Another line of work explored quantization of the continuous speech data to identify speech units in order to learn representations (Baevski et al., 2020b,a; Liu et al., 2019; van Niekerk et al., 2020; Baevski et al., 2020c). Hsu et al. (2021b) leverages automatically discovered speech units to learn representations with a BERT-like masked prediction objective and achieves comparable results with models based on contrastive learning. More recently, Transformer architectures have been adopted for self-supervised pre-training (Jiang et al., 2019; Baevski et al., 2020c). There has also been work on robustness of self-supervised learning to domain shift (Hsu et al., 2021a), multilingual self-supervised speech representation learning (Kawakami et al., 2020; Conneau et al., 2020) as well as work on combining modalities such as speech and vision (Harwath et al., 2020).

Unsupervised Speech Recognition. Learning to map speech to phonemes without supervision using adversarial learning has been explored by Liu et al. (2018) who learn a mapping matrix between segment identifiers and phonemes. However, their work still relied on data segmented into phonemes by human annotators. This has been later extended to use an automatic segmentation (Chen et al., 2019) which is iteratively refined with HMMs. However, cross validation is still performed using labeled data (personal communication with authors). We also explored HMMs to refine segmentation boundaries but did not find it helpful in our setting. Instead, we used HMMs for self-training.

Speech data has been segmented and clustered in a large body of prior work (Varadarajan et al., 2008; Zhang and Glass, 2009; Gish et al., 2009; Lee and Glass, 2012; Lee et al., 2015; Ondel et al., 2016), including Bayesian methods (Kamper et al., 2017a) and more efficient approximations thereof (Kamper et al., 2017b). More recent work includes self-supervised approaches to detect phoneme boundaries based on detecting spectral changes in the signal (Kreuk et al., 2020).

Unsupervised Machine Translation. Our work is in part inspired by unsupervised machine translation that showed the possibility of aligning language embedding spaces in an unsupervised way. Mikolov et al. (2013a) aligned word2vec embedding spaces (Mikolov et al., 2013b) by building a linear mapping from a source language to a target language using supervision from a seed dictionary of word translation pairs. Artetxe et al. (2017) greatly reduced the number of word translation pairs needed to learn the linear mapping, by iteratively refining an initial model through semi-supervised learning. Conneau et al. (2018) showed that the supervision used to build the initial alignment could be completely removed by introducing an adversarial game (Goodfellow et al., 2014; Ganin et al., 2016) in which a generator (the linear mapping from source to target) and a discriminator compete (a classifier trained to identify the languages of randomly sampled word vectors from the projected source space and the target space). The generator is trained to fool the discriminator. By exploiting the similarties of word2vec spaces across languages, the algorithm converges to a suboptimal mapping that leads to strong unsupervised word translation accuracy.

Going from unsupervised word translation to unsupervised sentence translation, Lample et al. (2018); Artetxe et al. (2018) exploited the similarity of neural representations across languages to build fully unsupervised machine translation systems. They used sequence to sequence architectures (Sutskever et al., 2014; Bahdanau et al., 2014) with shared encoder/decoders across languages, and the word embedding layer is initialized with the unsupervised bilingual word embeddings learned in Conneau et al. (2018). With this parameter sharing and initialization, a mere bilingual denoising auto-encoding loss provided a first unsupervised machine translation model. Similar to word embeddings, semi-supervised learning in the form of back-translation (Sennrich et al., 2015) further refines this initial model to provide better BLEU scores. It was later shown in Conneau and Lample (2019) that a better self-supervised initialization of the encoder and decoder using cross-lingual masked language modeling (Devlin et al., 2019) led to significantly better performance. Our work follows similar steps than in unsupervised machine translation: pretraining, then unsupervised alignment and finally semi-supervised learning.

8. Discussion

Phonemization. Our approach requires tools to phonemize text for the language of interest. Even for existing tools, we found that the quality of phonemization differs, e.g., G2P vs Phonemizer (§ 5.2) on the same English data results in a performance difference. Moreover, phonemizers are not available for all languages and this presents a bottleneck. To address this, future work may develop phonemizers for more languages, explore phonemization approaches that generalize across languages, or unsupervised training with graphemic text units such as letters.

Segmentation. In our work, we explored a simple segmentation technique based on self-supervised representations, however, there is a large body of work on segmentation and some of these techniques may lead to improvements over our simple approach. Also, wav2vec 2.0 learns representations for fixed size units with a fixed stride, however, phonemic units are of variable size. Another direction is to learn variable sized representations during pre-training.

9. Conclusion

wav2vec-U is a framework which enables building speech recognition models without labeled data. It embeds and segments the speech audio with self-supervised representations from wav2vec 2.0, learns a mapping to phonemes with adversarial learning, and cross-validates hyper-parameter choices as well as early stopping with an unsupervised metric. Experiments on the standard Librispeech benchmark show performance close to the state of the art models from only a few years ago, even though these models relied on nearly 1,000 hours of labeled data. Compared to the previous best unsupervised speech recognition approach, wav2vec-U reduces TIMIT phoneme error rate from 26.1 to 11.3. We also demonstrate the viability of our approach on several languages other than English, some of which are low-resource. The ability to build speech recognition models solely from unlabeled speech audio and unlabeled text drastically lowers the effort to build speech technology for many more languages of the world.

Acknowledgments

We thank Zhouhan Lin for helping with initial explorations in this project, Tatiana Likhomanenko for helpful discussions about self-training, Da-Rong Liu for sharing details to reproduce the setup of Chen et al. (2019), Marc'Aurelio Ranzato for general helpful discussions, and Ruth Kipng'eno, Ruth Ndila Ndeto as well as Mark Mutitu for error analysis of our Swahili model.

Appendix A. Hyperparameter Ablations

Ablation	mean PER ± std	%-converged (PER < 40)
Baseline	21.4 ± 1.2	100%
9.6h audio, 3k text	21.2 ± 1.1	100%
96h audio, 3k text	21.1 ± 1.3	95%
w/o clustering, pca, mean pool	-	0%
w/o clustering	-	0%
w/o 2nd stage mean pool	-	0%
w/o PCA	-	0%
64 clusters	23.1 ± 0.7	100%
256 clusters	22.3 ± 1.1	100%
256 PCA	21.6 ± 1.1	100%
768 PCA	28.0 ± 1.5	90%
use full phone set	23.51 ± 1.3	100%

Table 8: Ablation of various data settings, pre-processing steps, cluster sizes, PCA sizes and using the full phoneme set.

References

S. T. Abate, W. Menzel, and B. Tafila. An amharic speech corpus for large vocabulary continuous speech recognition. In *Proc. of Interspeech*, 2005.

O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Proc. of ICASSP*, 2012.

D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proc. of ICML*, 2016.

R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber. Common voice: A massively-multilingual speech corpus. *Proc. of LREC*, 2020.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *Proc. of ICML*, 2017.

M. Artetxe, G. Labaka, and E. Agirre. Learning bilingual word embeddings with (almost) no bilingual data. In *Proc. of ACL*, 2017.

M. Artetxe, G. Labaka, E. Agirre, and K. Cho. Unsupervised neural machine translation. *Proc. of ICLR*, 2018.

A. Baevski and M. Auli. Adaptive input representations for neural language modeling. In *Proc. of ICLR*, 2018.

A. Baevski, M. Auli, and A. Mohamed. Effectiveness of self-supervised pre-training for speech recognition. *Proc. of ICASSP*, 2020a.

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

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

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *Proc. of ICLR*, 2014.

L. Bahl, P. Brown, P. De Souza, and R. Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proc. of ICASSP*, volume 11, pages 49–52. IEEE, 1986.

L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, 1983.

L. Besacier, E. Gauthier, M. Mangeot, P. Bretier, P. Bagshaw, O. Rosec, T. Moudenc, F. Pellegrino, S. Voisin, E. Marsico, and P. Nocera. Speech technologies for african languages: example of a multilingual calculator for education. In *Proc. of Interspeech*, 2015.

O. Bojar and A. Tamchyna. Improving translation model by monolingual data. In *Proc. of WMT*, 2011.

H. A. Bourlard and N. Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 2012.

D. Can, V. R. Martinez, P. Papadopoulos, and S. S. Narayanan. Pykaldi: A python wrapper for kaldi. In *Proc. of ICASSP*, 2018.

K.-Y. Chen, C.-P. Tsai, D.-R. Liu, H.-Y. Lee, and L. shan Lee. Completely unsupervised speech recognition by a generative adversarial network harmonized with iteratively refined hidden markov models. In *Proc. of Interspeech*, 2019.

J. Chorowski and N. Jaitly. Towards better decoding and language model integration in sequence to sequence models. *Proc. of Interspeech*, 2017.

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

Y. Chung, W. Weng, S. Tong, and J. R. Glass. Unsupervised cross-modal alignment of speech and text embedding spaces. *Proc. of NIPS*, 2018.

Y. Chung, W. Hsu, H. Tang, and J. R. Glass. An unsupervised autoregressive model for speech representation learning. *Proc. of Interspeech*, 2019a.

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

Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass. An unsupervised autoregressive model for speech representation learning. *Proc. of Interspeech*, 2019b.

A. Conneau and G. Lample. Cross-lingual language model pretraining. *Proc. of NeurIPS*, 2019.

A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data. *Proc. of ICLR*, 2018.

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

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proc. of NAACL*, 2019.

S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. *Proc of NIPS*, 2018.

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

S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding back-translation at scale. In *Proc. of EMNLP*, 2018.

Z. Fan, M. Li, S. Zhou, and B. Xu. Exploring wav2vec 2.0 on speaker verification and language identification. *arXiv*, 2021.

R. Fer, P. Matějka, F. Grézl, O. Plchot, K. Veselỳ, and J. H. Černockỳ. Multilingually trained bottleneck features in spoken language recognition. *Computer Speech & Language*, 46, 2017.

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*, 17(1):2096–2030, 2016.

J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM. *Linguistic Data Consortium*, 1993.

H. Gelas, L. Besacier, and F. Pellegrino. Developments of Swahili resources for an automatic speech recognition system. In *Proc. of SLTU*, 2012.

H. Gish, M. Siu, A. Chan, and W. Belfield. Unsupervised training of an hmm-based speech recognizer for topic classification. In *Proc. of Interspeech*, 2009.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Proc. of NIPS*, 2014.

Google. Google cloud: Speech-to-text. `https://cloud.google.com/speech-to-text`, 2021. Accessed: 2021-05-13.

A. Graves. Sequence transduction with recurrent neural networks. *Proc. of ICML workshop on Representation Learning*, 2012.

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

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

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *Proc. of NIPS*, 2017.

W. Han, Z. Zhang, Y. Zhang, J. Yu, C.-C. Chiu, J. Qin, A. Gulati, R. Pang, and Y. Wu. Contextnet: Improving convolutional neural networks for automatic speech recognition with global context. *Proc. of Interspeech*, 2020.

A. Hannun. Sequence modeling with ctc. *Distill*, 2017. doi: 10.23915/distill.00008. https://distill.pub/2017/ctc.

A. Hannun, V. Pratap, J. Kahn, and W.-N. Hsu. Differentiable weighted finite-state transducers. *arXiv preprint arXiv:2010.01003*, 2020.

D. Harwath and J. Glass. Towards visually grounded sub-word speech unit discovery. In *Proc. of ICASSP*, pages 3017–3021. IEEE, 2019.

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

T. Hayashi, S. Watanabe, Y. Zhang, T. Toda, T. Hori, R. F. Astudillo, and K. Takeda. Back-translation-style data augmentation for end-to-end ASR. In *Proc. of SLT*, 2018.

K. Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

K. Hirsh-Pasek, D. G. Kemler Nelson, P. W. Jusczyk, K. W. Cassidy, B. Druss, and L. Kennedy. Clauses are perceptual units for young infants. *Cognition*, 26(3):269–286, 1987.

T. Hori, R. Astudillo, T. Hayashi, Y. Zhang, S. Watanabe, and J. Le Roux. Cycle-consistency training for end-to-end speech recognition. In *Proc. of ICASSP*, 2019.

W.-N. Hsu, A. Lee, G. Synnaeve, and A. Hannun. Semi-supervised speech recognition via local prior matching. *arXiv preprint arXiv:2002.10336*, 2020.

W.-N. Hsu, A. Sriram, A. Baevski, T. Likhomanenko, Q. Xu, V. Pratap, J. Kahn, A. Lee, R. Collobert, G. Synnaeve, et al. Robust wav2vec 2.0: Analyzing domain shift in self-supervised pre-training. *arXiv preprint arXiv:2104.01027*, 2021a.

W.-N. Hsu, Y.-H. H. Tsai, B. Bolte, R. Salakhutdinov, and A. Mohamed. Hubert: How much can a bad teacher benefit ASR pre-training? In *Proc. of ICASSP*, 2021b.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *Proc. of ICLR*, 2016.

G. Jawahar, B. Sagot, and D. Seddah. What does bert learn about the structure of language? In *Proc. of ACL*, 2019.

H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, Jan. 2011.

D. Jiang, X. Lei, W. Li, N. Luo, Y. Hu, W. Zou, and X. Li. Improving transformer-based speech recognition using unsupervised pre-training. *Proc. of Interspeech*, 2019.

E. K. Johnson and P. W. Jusczyk. Word segmentation by 8-month-olds: When speech cues count more than statistics. *Journal of Memory and Language*, 44(4):548–567, 2001.

J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

B.-H. Juang, S. Levinson, and M. Sondhi. Maximum likelihood estimation for multivariate mixture observations of markov chains (corresp.). *IEEE Transactions on Information Theory*, 32(2):307–309, 1986.

P. W. Jusczyk, D. M. Houston, and M. Newsome. The beginnings of word segmentation in english-learning infants. *Cognitive Psychology*, 39(3):159–207, 1999.

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

J. Kahn et al. Libri-light: A benchmark for asr with limited or no supervision. In *Proc. of ICASSP*, 2020b.

H. Kamper, A. Jansen, and S. Goldwater. A segmental framework for fully-unsupervised large-vocabulary speech recognition. *Comput. Speech Lang.*, 46(C), Nov. 2017a.

H. Kamper, K. Livescu, and S. Goldwater. An embedded segmental k-means model for unsupervised segmentation and clustering of speech. *Proc. of ASRU*, 2017b.

K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. van den Oord. Learning robust and multilingual speech representations. *Proc. of EMNLP*, 2020.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*, 2015.

F. Kreuk, J. Keshet, and Y. Adi. Self-supervised contrastive learning for unsupervised phoneme segmentation. *Proc. of Interspeech*, 2020.

G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. In *Proc. of ICLR*, 2018.

C. Lee and J. R. Glass. A nonparametric bayesian approach to acoustic model discovery. In *Proc. of ACL*, 2012.

C. Lee, T. J. O'Donnell, and J. R. Glass. Unsupervised lexicon discovery from acoustic input. *TACL*, 2015.

M. P. Lewis, G. F. Simon, and C. D. Fennig. Ethnologue: Languages of the world, nineteenth edition. Online version: `http://www.ethnologue.com`, 2016.

T. Likhomanenko, Q. Xu, J. Kahn, G. Synnaeve, and R. Collobert. slimipl: Language-model-free iterative pseudo-labeling. *arXiv*, 2021.

A. H. Liu, T. Tu, H. yi Lee, and L. shan Lee. Towards unsupervised speech recognition and synthesis with quantized speech representation learning. *Proc. of ICASSP*, 2019.

D.-R. Liu, K.-Y. Chen, H.-Y. Lee, and L. shan Lee. Completely unsupervised phoneme recognition by adversarially learning mapping relationships from audio embeddings. *Proc. of Interspeech*, 2018.

V. Manohar, H. Hadian, D. Povey, and S. Khudanpur. Semi-supervised training of acoustic models using lattice-free mmi. In *Proc. of ICASSP*, 2018.

T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, 2013b.

M. Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997.

M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

L. Ondel, L. Burget, and J. Cernocký. Variational inference for acoustic unit discovery. In *Proc. of SLTU*, 2016.

M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL System Demonstrations*, 2019.

V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proc. of ICASSP*, pages 5206–5210. IEEE, 2015.

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

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

K. Park and J. Kim. g2pe. `https://github.com/Kyubyong/g2p`, 2019.

L. Pepino, P. Riera, and L. Ferrer. Emotion recognition from speech using wav2vec 2.0 embeddings. *arXiv*, 2021.

L. Polka and J. F. Werker. Developmental changes in perception of nonnative vowel contrasts. *Journal of Experimental Psychology: Human perception and performance*, 20(2): 421, 1994.

D. Povey. *Discriminative training for large vocabulary speech recognition.* PhD thesis, University of Cambridge, 2005.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. In *Proc. of ASRU*, 2011.

V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky, and R. Collobert. Wav2letter++: A fast open-source speech recognition system. In *Proc. of ICASSP*, 2019.

V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert. Mls: A large-scale multilingual dataset for speech research. In *Proc. of Interspeech*, 2020.

K. Rao, H. Sak, and R. Prabhavalkar. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 193–199. IEEE, 2017.

O. Rasanen, G. Doyle, and M. C. Frank. Unsupervised word discovery from speech using automatic segmentation into syllable-like units. In *Proc. of Interspeech*, 2015.

M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio. Light gated recurrent units for speech recognition. *IEEE Trans. on Emerging Topics in Comp. Intel.*, 2, 2018.

M. Ravanelli, T. Parcollet, and Y. Bengio. The pytorch-kaldi speech recognition toolkit. *Proc. of ICASSP*, 2019.

M. Rivière, A. Joulin, P.-E. Mazaré, and E. Dupoux. Unsupervised pretraining transfers well across languages. In *Proc. of ICASSP*, 2020.

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

R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *Proc. of ACL*, 2015.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Proc. of NIPS*, 2014.

G. Synnaeve, Q. Xu, J. Kahn, T. Likhomanenko, E. Grave, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert. End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures. *Proc. of ICML workshop on Self-supervision in Audio and Speech (SAS)*, 2020.

M. Tachbelie, S. T. Abate, and L. Besacier. Using different acoustic, lexical and language modeling units for asr of an under-resourced language - amharic. *Speech Communication*, 56, 2014.

Z. Tan, A. K. Sarkar, and N. Dehak. rvad: An unsupervised segment-based robust voice activity detection method. *Computer speech & language*, 59:1–21, 2020.

I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. *Proc. of ACL*, 2019.

A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *Proc. of NIPS*, 2018.

B. van Niekerk, L. Nortje, and H. Kamper. Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge. *Proc. of Interspeech*, 2020.

B. Varadarajan, S. Khudanpur, and E. Dupoux. Unsupervised learning of acoustic sub-word units. In *Proc. of ACL*, 2008.

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

K. Veselỳ, L. Burget, and J. Cernockỳ. Semi-supervised dnn training with word selection for asr. In *Proc. of Interspeech*, 2017.

C. Wang, A. Wu, J. Pino, A. Baevski, M. Auli, and A. Conneau. Large-scale self- and semi-supervised learning for speech translation. *arXiv*, 2021.

J. F. Werker and R. C. Tees. Cross-language speech perception: Evidence for perceptual reorganization during the first year of life. *Infant behavior and development*, 7(1):49–63, 1984.

Y. Xia, D. He, T. Qin, L. Wang, N. Yu, T. Liu, and W. Ma. Dual learning for machine translation. In *Proc. of NeurIPS*, 2016.

H. Xu, K. Li, Y. Wang, J. Wang, S. Kang, X. Chen, D. Povey, and S. Khudanpur. Neural network language modeling with letter-based features and importance sampling. In *Proc. of ICASSP*, 2018.

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. In *Proc. of ICASSP*, 2020a.

Q. Xu, T. Likhomanenko, J. Kahn, A. Hannun, G. Synnaeve, and R. Collobert. Iterative pseudo-labeling for speech recognition. *Proc. of Interspeech*, 2020b.

C.-K. Yeh, J. Chen, C. Yu, and D. Yu. Unsupervised speech recognition via segmental empirical output distribution matching. In *Proc. of ICLR*, 2019.

S. Young. Large vocabulary continuous speech recognition: A review. *IEEE Signal Processing Magazine*, 13(5):45–57, 1996.

N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert. Fully convolutional speech recognition. *arXiv*, abs/1812.06864, 2018.

A. Zeyer, E. Beck, R. Schlüter, and H. Ney. Ctc in the context of generalized full-sum hmm training. In *Proc. of Interspeech*, 2017.

F. Zhang, Y. Wang, X. Zhang, C. Liu, Y. Saraf, and G. Zweig. Faster, simpler and more accurate hybrid asr systems using wordpieces. *Proc. of Interspeech*, 2020a.

Y. Zhang and J. R. Glass. Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams. *IEEE Workshop on Automatic Speech Recognition & Understanding*, 2009.

Y. Zhang, J. Qin, D. S. Park, W. Han, C.-C. Chiu, R. Pang, Q. V. Le, and Y. Wu. Pushing the limits of semi-supervised learning for automatic speech recognition. *Proc. of NeurIPS SAS Workshop*, 2020b.