Wav2vec-C: A Self-supervised Model for Speech Representation Learning

Samik Sadhu\textsuperscript{1}, Di He\textsuperscript{2}, Che-Wei Huang\textsuperscript{2}, Sri Harish Mallidi\textsuperscript{2}, Minhua Wu\textsuperscript{2}, Ariya Rastrow\textsuperscript{2}, Andreas Stolcke\textsuperscript{2}, Jasha Droppo\textsuperscript{2}, Roland Maas\textsuperscript{2}

\textsuperscript{1}Johns Hopkins University, USA  
\textsuperscript{2}Amazon Alexa, USA

\{deehe, cheweh, mallidih, wuminhua, arastrow, stolcke, drojasha, rmaas\}@amazon.com

Abstract

Wav2vec-C introduces a novel representation learning technique combining elements from wav2vec 2.0 and VQ-VAE. Our model learns to reproduce quantized representations from partially masked speech encoding using a contrastive loss in a way similar to wav2vec 2.0. However, the quantization process is regularized by an additional consistency network that learns to reconstruct the input features to the wav2vec 2.0 network from the quantized representations in a way similar to a VQ-VAE model. The proposed self-supervised model is trained on 10k hours of unlabeled data and subsequently used as the speech encoder in an RNN-T ASR model and fine-tuned with 1k hours of labeled data. This work is one of the very few studies of self-supervised learning on speech tasks with a large volume of real far-field labeled data. The proposed self-supervised model is trained on 10k hours of unlabeled data and subsequently used as the speech encoder in an RNN-T ASR model and fine-tuned with 1k hours of labeled data.

\section{Introduction}

Self-supervision [1, 2, 3, 4] is a paradigm of machine learning (ML) that deals with unsupervised learning of structural patterns in data by exploiting contextual information. Self-supervision has been of significant interest in the automatic speech recognition (ASR) literature primarily as a pre-training step before a fully supervised task. In particular, it is widely used for problems with some amount of labeled data (for supervised training) and a significantly larger volume of unlabeled data (for self-supervised training). The recently proposed wav2vec 2.0 [5] is one such self-supervised learning model that learns to predict masked discrete speech encodings using a contextualized representation from a transformer model [6].

In this paper, we introduce the wav2vec-C model that solves a more rigorously defined self-supervised learning problem compared to the wav2vec 2.0. In the latter, a contrastive loss defined on discretized codes drives the self-supervised learning - including the codebook in the built-in differentiable Vector Quantization module. In contrast, wav2vec-C facilitates codebook learning through an additional regularization on the discrete speech representations by reconstructing the discrete codes to the input features. Thus, wav2vec-C maintains a consistency between the learnt representations and the input features to the network.

Our main contributions in this paper are:

\begin{itemize}
  \item The wav2vec-C model (Section 2)
  \item We use real world far-field voice query speech with varied degrees of SNR ranging between -40 to 50 dB, whereas most studies on self-supervised learning in the literature use clean read speech [7, 8] and some use simulated noisy speech [9].
\end{itemize}

\begin{itemize}
  \item Self-supervised learning has been shown to be useful for settings with little labeled data [1, 9]. It has been observed that the effectiveness of self-supervision decreases as the amount of labeled data increases[10, 7]. In this work, we explore the applicability of self-supervision with a relatively large amount of labeled data (1k hours).
  \item We also limit our model size to facilitate low-latency production level ASR models, which goes against the general trend of exceedingly large self-supervised models proposed in the literature [2].
  \item We explore and compare different variants of our framework in the choice of the vector quantization framework and the effect it has on robustness and codebook utilization.
\end{itemize}

\section{The Wav2vec-C Model}

\subsection{Summary of wav2vec 2.0}

Our model is similar to wav2vec 2.0 [5], but differs in the way we use log short-term Fourier transform (log-STFT) features as input to our model. An encoder network \( f : X \rightarrow Z \) maps the input features \( X = [x_1, x_2, \ldots, x_T] \) to a latent embedding space. These embeddings are quantized by a vector quantization module \( q : Z \rightarrow \hat{Z} \). The embedded vectors \( Z = [z_1, z_2, \ldots, z_T] \in Z \) are passed through a SpecAugment [11] module that randomly masks a portion of these embeddings to generate \( Z_{\text{masked}} \). These masked embeddings are fed into a context network \( g : Z \rightarrow C \) that generates a set of context representations \( C = [c_1, c_2, \ldots, c_T] \). A contrastive score between the context representations and the vector quantized embeddings \( \hat{Z} = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_T] \) is maximized during network training.

\subsection{Wav2vec-C}

The wav2vec 2.0 model relies on a diverse set of codes correlating to the underlying speech units learned by \( q \) to enable \( g \) to learn good contextual representations via the contrastive loss. However, the wav2vec 2.0 problem formulation can result in several locally optimal codebooks. A few highly probable optima observed in our experiments were:

\begin{itemize}
  \item Voice activity detection (VAD) codebooks - where two codes are assigned; one for speech and the other for non-speech.
\end{itemize}
• Temporally invariant codebooks - where the model assigns specific codes to fixed temporal locations to enable a good contrastive loss.

Our training data consists of many similar query terms occurring at fixed temporal locations which also contributed to the model assigning fixed codes at specific temporal instances via the recurrent encoder (Section 2.3) irrespective of the underlying speech sounds. Hence, the codebook learning methodology adopted for wav2vec 2.0 might not generalize well to other datasets and different model architectures, as in our case.

In wav2vec-C (Figure 1) we enforce the codes to explicitly carry information about the input features X to help mitigate the described codebook learning issues. We define an additional consistency network \( r: \hat{Z} \rightarrow S \) that reconstructs the quantized encodings \( \hat{Z} = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_T] \) to consistency vectors \( S = [s_1, s_1, \ldots, s_T] \) and minimize the normed distance between the inputs X and S during network training. This network allows a flow of information from the input log-STFT features back to the feature domain and enforces the latent space to preserve meaningful information that enable a low reconstruction error. Hence, in a way, wav2vec-C can be seen as an integration of the ideas behind wav2vec 2.0 and VQ-VAE [12].

2.3. Encoder network (f)

Our encoder network \( f \) consists of three layers of long short-term memory network (LSTM) with a hidden dimension of 768. The encoder gradients are scaled by a factor \( \gamma = 0.1 \) as in wav2vec 2.0 to help stabilize the codebook during training.

2.4. Vector quantization (q)

We use a product quantization module [8, 13] with \( G \) codebooks \( Q = [Q^{(1)}, Q^{(2)}, \ldots, Q^{(G)}] \). Each codebook \( Q^{(i)} \in \mathbb{R}^{V \times K} \) is represented by a set of V codes, each of dimension K. The LSTM encoded representations \( z \in \mathbb{R}^{768} \) are split into a set of \( G \) representations \( z_{\text{split}} = \{z^{(1)}, z^{(2)}, \ldots, z^{(G)}\} \) with \( z^{(i)} \in \mathbb{R}^{768}, i \in \{1, 2, \ldots, G\} \). Every \( z^{(i)} \) is used to select one code \( e \in \mathbb{R}^K \) from \( Q^{(i)} \) to obtain a quantized representation \( \hat{z}^{(i)} \). The representations \( \hat{z}^{(i)} \) are fed to the context network to enable code selection process completely differentiable.

2.4.1. Gumbel-softmax [14]

Each split \( \hat{z}^{(i)} \) is passed through a trainable linear transformation to generate logits \( f^{(i)} \in \mathbb{R}^V \) which are passed through a Gumbel-softmax to generate a hard distribution over V codes that can be used as a code selector during the forward pass. During back propagation, we use the true gradient of the softmax distribution, thereby making the code selection process completely differentiable.

2.4.2. K-means [12]

During forward pass, a k-means codebook selects the code \( e \) from \( Q^{(i)} \) which has the closest squared distance to \( \hat{z}^{(i)} \) as

\[
\hat{z}^{(i)} = \arg \min_{e \in Q^{(i)}} ||z^{(i)} - e||_2^2
\]

However, during back-propagation, a straight-through estimator [15] bypasses gradient computation w.r.t the quantized embedding and copies the gradients [12] to the continuous embedding \( z \). Since this process puts the codebook out of the training graph, there are two loss terms incorporated into training as

\[
\mathcal{L}_k = ||sg(\hat{z}^{(i)} - \hat{z}^{(i)})||_2 + \beta||z^{(i)} - sg(\hat{z}^{(i)})||_2
\]

On minimization of \( \mathcal{L}_k \), the first term pushes the quantized representations close to the continuous encoded representation and the second term (also called commitment loss) enforces encodings \( z^{(i)} \) to commit to quantized embeddings \( \hat{z}^{(i)} \) during training. In eq. 2, \( sg(.) \) is the stop gradient operator [8] and \( \beta = 0.25 \) as is the optimal value reported in [12].

2.5. Masking

We use a SpecAugment [11] module to mask out portions of the continuous encodings \( Z = [z_1, z_2, \ldots, z_T] \) before feeding them to the context network. We use five masks for every utterance. Each mask has maximum width of 16% of the utterance length. On average 40% of the encoded frames are masked.

2.6. Context network (r)

The context network consists of five transformer layers, with model dimension 1024 and inner feed-forward dimension of 4096 with 16 attention heads. We use sinusoidal positional embedding for the transformer layers. A contrastive score between the context representations \( C = [c_1, c_2, \ldots, c_T] \) and the quantized encodings \( \hat{Z} = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_T] \) is computed as

\[
\mathcal{L}_m = -\log \frac{\exp(d(c_t, \hat{z}_t))/\kappa}{\sum_{z \in \Theta} \exp(d(c_t, z))/\kappa}
\]

where \( t \in \{1, 2, \ldots, T\} \), \( \Theta \) is a set consisting of \( \hat{z}_t \) and a selection of \( N \) negative samples, \( \kappa \) is the temperature variable and \( d \) calculates the cosine similarity \( d(x, y) = \frac{\langle x, y \rangle}{||x|| ||y||} \). In our experiments, we uniformly sample \( N = 50 \) negative samples from the encodings \( \hat{Z} \) of the utterance and \( \kappa \) is updated as proposed in [5].

2.7. Consistency network (r)

The consistency network \( r \) consists of a 3-layer LSTM that maps the quantized embedding \( \hat{Z} = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_T] \) to the consistency vectors \( S = [s_1, s_1, \ldots, s_T] \). We minimize the \( L_2 \) normed distance between \( S \) and \( X \) as
During training, we minimize the primary contrastive loss together with a codebook loss component and the consistency loss as

$$\mathcal{L} = \mathcal{L}_m + \mathcal{L}_{cb} + \gamma \mathcal{L}_c$$

(5)

The codebook loss \(\mathcal{L}_{cb}\) (section 2.4) takes a different form according to the type of VQ used. Wav2vec 2.0 and wav2vec-C are generalized by the parameter \(\gamma\), where a value \(\gamma = 0\) results in the wav2vec 2.0 model as the consistency loss is ignored for model training, while \(\gamma = 1\) leads to our wav2vec-C model in full effect.

For a Gumbel-softmax VQ module, the codebook loss is given by \(\mathcal{L}_{cb} = \alpha \mathcal{L}_d\), where \(\mathcal{L}_d\) is a diversity loss on the Gumbel-softmax distribution given by

$$\mathcal{L}_d = \frac{GV - \sum_{g=1}^{G} \exp\left(\frac{-\sum_{V=1}^{V} p_{g,v} \log p_{g,v}}{GV}\right)}{GV}$$

(6)

where \(p_{g,v}\) is the probability assignment by the \(g^{th}\) codebook on the \(v^{th}\) code. The weight \(\alpha\) on the diversity loss determines the relative importance of the component and is instrumental in avoiding the codebook collapse that is commonly observed in VQ problems [16, 12]. In our experiments, we found \(\alpha = 1.5\) to be suitable to avoid catastrophic codebook collapse issues. For k-means VQ, the codebook loss is simply equal to the k-means loss, i.e., \(\mathcal{L}_{cb} = \mathcal{L}_k\).

3. Experimental Setup

3.1. Data sets

The goal of this study is to evaluate the effectiveness of self-supervised pre-training for real-world applications. Hence, instead of using publicly available clean read speech we use in-house training and evaluation data consisting of real-world far-field English voice command and voice query speech collected from home environments similar to [17] with varying degrees of SNR in the range -40 to 50 dB.

3.1.1. Training data

We use 10k hours of unlabeled and 1k hours of transcribed de-identified English language training data collected from native and non-native English speakers. To our knowledge, this work is one of the first few instances where a large proportion of labeled data is used alongside self-supervised pre-training for ASR tasks, especially realistic speech queries instead of clean read speech data.

3.1.2. Test data

We test our ASR models on four different test sets summarized in Table 1.

<table>
<thead>
<tr>
<th>Test set</th>
<th>Details</th>
<th># Utterances</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean(snr20)</td>
<td>Average SNR (\approx) 20 dB</td>
<td>118.0k</td>
</tr>
<tr>
<td>clean(snr16)</td>
<td>Average SNR (\approx) 16 dB</td>
<td>43.2k</td>
</tr>
<tr>
<td>noisy(n1)</td>
<td>background multimedia speech</td>
<td>31.0k</td>
</tr>
<tr>
<td>noisy(n2)</td>
<td>background speech, multiple speakers</td>
<td>5.8k</td>
</tr>
</tbody>
</table>

3.3. Training details

We train 4 different self-supervised models

1. wav2vec 2.0 (GS): \(\gamma = 0\), Gumbel-softmax codebook
2. wav2vec 2.0 (KM): \(\gamma = 0\), k-means codebook
3. wav2vec-C (GS): \(\gamma = 1\), Gumbel-softmax codebook
4. wav2vec-C (KM): \(\gamma = 1\), k-means codebook

Subsequently, we train RNN-T models with the speech encoder replaced by the self-supervised models.

Our models are trained using Tensorflow 2.0. The self-supervised models are trained for 100k steps with 30 minutes of speech per step. We use an Adam optimizer [21], where the learning rate is warmed up from \(1 \times 10^{-7}\) and held at \(5 \times 10^{-6}\) after 3k steps. The RNN-T models are trained for 60k steps, with an average of 1 hour of speech per step. The learning rate is warmed up from \(1 \times 10^{-7}\) and held at \(5 \times 10^{-8}\) after 3k steps.

4. Results

We compare the word error rate reduction relative to the baseline model (rWERR) for the different pre-trained RNN-T models evaluated on the four test sets in Table 2. The baseline ASR model has < 10% absolute word error rate. To smooth out error fluctuations, we report the mean rWERR computed after 50k, 55k and 60k RNN-T training steps. The average rWERR in the last column is the rWERR for each test set weighted by the number of utterances in that test set.

Our implementation of the wav2vec 2.0 pre-trained RNN-T model does not show noticeable performance improvement over baseline for the clean test sets. Whereas, for the noisy test sets, some gains can be observed - with wav2vec 2.0 (KM)
performing better, on average, compared to wav2vec 2.0 (GS). This trend is comparable to the results reported in [5], where pre-training is shown to be most beneficial for the challenging test other test set of Librispeech [22]. However, while drawing this comparison we should keep in mind the major differences between the best performing wav2vec 2.0 models in [5] and our implementation, namely

1. We use a much smaller context network (5 layers) compared to the original (24 layers).
2. We use a 3-layer LSTM as encoder with log-STFT input features.

Table 2: Comparison of rWERR of different pre-trained RNN-T models.
(*The errors on each test set weighted by the number of utterances in each test set)

<table>
<thead>
<tr>
<th>RNN-T encoder</th>
<th>rWERR on different test sets</th>
<th>SNR 20</th>
<th>SNR 16</th>
<th>N1</th>
<th>N2</th>
<th>Average*</th>
</tr>
</thead>
<tbody>
<tr>
<td>wav2vec 2.0 (KM)</td>
<td></td>
<td>0</td>
<td>0.6</td>
<td>0.7</td>
<td>3.2</td>
<td>0.7</td>
</tr>
<tr>
<td>wav2vec 2.0 (GS)</td>
<td></td>
<td>0</td>
<td>0.6</td>
<td>0.7</td>
<td>2.7</td>
<td>0.3</td>
</tr>
<tr>
<td>wav2vec-C (KM)</td>
<td></td>
<td>0.8</td>
<td>0.6</td>
<td>0.7</td>
<td>2.7</td>
<td>0.8</td>
</tr>
<tr>
<td>wav2vec-C (GS)</td>
<td></td>
<td>1.6</td>
<td>1.2</td>
<td>0.7</td>
<td>1.6</td>
<td>1.4</td>
</tr>
</tbody>
</table>

The wav2vec-C encoded RNN-T models, on the other hand, show a positive rWERR for both SNR 20 as well as SNR 16 clean test sets. In particular, wav2vec-C (GS) gains 1.6% rWERR on SNR 20 and 1.2% rWERR on SNR 16. However, there is a reduction in performance (in comparison to wav2vec 2.0) for the noisy test sets. This suggests that the reconstruction idea adopted for wav2vec-C leads to an overall better performance of the pre-trained RNN-T model, however with a slight loss in robustness.

4.1. Discussions on codebook utilization

Our codebooks have a maximum capacity of $320 \times 320 = 102.4k$ codes with wav2vec-C (GS) utilizing the full 100% of the codebook (see Table 3). Hence, the consistency loss together with the weight $\alpha$ on the diversity loss enforces the model to pick a variety of codes to minimize the reconstruction loss.

Table 3: Codebook utilization of the self-supervised models

<table>
<thead>
<tr>
<th>self-supervised model</th>
<th>codebook utilization(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>wav2vec 2.0 (KM)</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>wav2vec 2.0 (GS)</td>
<td>15</td>
</tr>
<tr>
<td>wav2vec-C (KM)</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>wav2vec-C (GS)</td>
<td>100</td>
</tr>
</tbody>
</table>

A t-SNE plot of the 102.4k codes in the 100% utilized codebook of the wav2vec-C (GS) model can be seen in Figure 2b showing the clusters formed by the codes over the course of training. On the other hand, the 102.4k codes learnt by wav2vec 2.0 (GS), as shown in Figure 2a, form a smaller number of clusters with significant inter-cluster overlap possibly due to the under-utilized codebook.

![t-SNE clustering of the wav2vec 2.0 (GS) and wav2vec-C (GS) codebooks](image)

The k-means codebook uses only a small fraction of the codes but is more robust compared to Gumbel-softmax models for noisy test sets, in particular the N2 noisy test set. For example, a comparison of the ASR performances of wav2vec-C (GS) and wav2vec-C (KM) would show that wav2vec-C (GS) gives a better rWERR for clean test sets in comparison to noisy test sets, whereas wav2vec-C (KM) shows the opposite characteristics. This observation highlights the importance of codebook diversity for different application domains. For example, a small codebook diversity is not necessarily a bad design choice if robustness is of importance during model evaluation.

5. Conclusions

In this paper we propose wav2vec-C, a new self-supervised learning model which is based on an amalgamation of the ideas from wav2vec 2.0 and VQ-VAE with the goal of solving the codebook utilization difficulties observed for wav2vec 2.0. We used real-world far-field noisy data for self-supervised learning and 1k hours of data for supervised ASR training. The proposed self-supervised model after RNN-T fine-tuning achieved, on average, a 1.4% relative WER reduction over baseline compared to a 0.7% reduction from wav2vec 2.0. Furthermore, we also observed that ASR robustness is correlated with codebook diversity, validating our motivation for the wav2vec-C architecture.
6. References


