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

Samik Sadhu[1]*, Di He*[2]*, Che-Wei Huang*[2]*, Sri Harish Mallidi*[2]*, Minhua Wu*[2]*, Ariya Rastrow*[2]*, Andreas Stolcke*[2]*, Jasha Droppo*[2]*, Roland Maas*[2]

[1]Johns Hopkins University, USA
[2]Amazon Alexa, USA

samiksadhu@jhu.edu,{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 a 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 wav2vec-C encoded representations achieve, on average, twice the error reduction over baseline and a higher codebook utilization in comparison to wav2vec 2.0.
Index Terms: Self-supervised learning, representation learning, end-to-end automatic speech recognition

1. 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 out 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

- The wav2vec-C model (Section 2)
- 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].
- 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).
- 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].
- 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.

2. The Wav2vec-C Model

2.1. 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* $f : \mathcal{X} \rightarrow \mathcal{Z}$ maps the input features $X = [x_1, x_2, \dots x_T]$ to a latent embedding space. These embeddings are quantized by a *vector quantization* module $q : \mathcal{Z} \rightarrow \hat{\mathcal{Z}}$. The embedded vectors $Z = [z_1, z_2, \dots z_T] \in \mathcal{Z}$ are passed through a SpecAugment [11] module that randomly masks a portion of these embeddings to generate Z_{masked}. These masked embeddings are fed into a *context network* $g : \mathcal{Z} \rightarrow \mathcal{C}$ that generates a set of context representation $C = [c_1, c_2, \dots c_T]$. A contrastive score between the context representations and the vector quantized embeddings $\hat{Z} = [\hat{z}_1, \hat{z}_1, \dots \hat{z}_T]$ is maximized during network training.

2.2. 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

- *Voice activity detection (VAD) codebooks* - where two codes are assigned; one for speech and the other for non-speech

- *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.

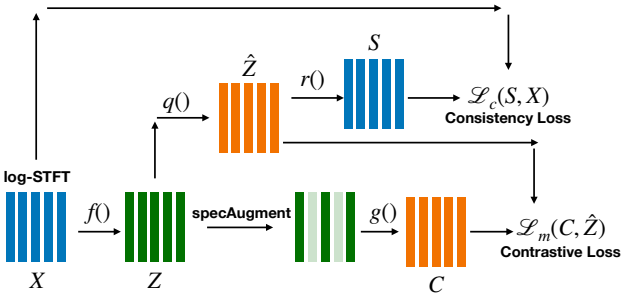


Figure 1: *Overview of the wav2vec-C self-supervised learning model*

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* $r : \hat{\mathcal{Z}} \rightarrow \mathcal{S}$ that reconstructs the quantized encodings $\hat{Z} = [\hat{z}_1, \hat{z}_1, \dots \hat{z}_T]$ to consistency vectors $S = [s_1, s_1, \dots s_T]$ 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 $Q = [Q^{(1)}, Q^{(2)}, \dots Q^{(G)}]$. 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 $z_{split} = \{z^{(1)}, z^{(2)}, \dots z^{(G)}\}$ with $z^{(i)} \in \mathbb{R}^{768/G}, i \in \{1, 2, \dots G\}$. 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 $\hat{z}^{(i)}, i = \{1, 2, \dots G\}$ from all the codebooks are concatenated to form the final quantized encoding $\hat{z}$. In our experiments we use $G = 2$ codebooks, each with $V = 320$ codes and dimension $K = 384$ which is consistent with the original wav2vec 2.0 model. We use two different VQ techniques

2.4.1. Gumbel-softmax [14]

Each split $z^{(i)} \in z_{split}$ is passed through a trainable linear transformation to generate logits $l^{(i)} \in \mathbb{R}^V$ 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 $z^{(i)}$ as

$$\hat{z}^{(i)} = \arg\min_{e \in Q^{(i)}} ||z^{(i)} - e||_2 \tag{1}$$

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

$$\mathcal{L}_k = ||sg(z^{(i)}) - \hat{z}^{(i)}||_2 + \beta||z^{(i)} - sg(\hat{z}^{(i)})||_2 \tag{2}$$

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 $Z = [z_1, z_2, \dots, z_T]$ 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(g)

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 $C = [c_1, c_2, \dots c_T]$ and the quantized encodings $\hat{Z} = [\hat{z}_1, \hat{z}_2, \dots \hat{z}_T]$ is computed as

$$\mathcal{L}_m = -\log \frac{\exp(d(c_t, \hat{z}_t))/\kappa}{\sum_{z \in \Theta} \exp(d(c_t, z))/\kappa} \tag{3}$$

where $t \in \{1, 2, \dots T\}$, Θ is a set consisting of $\hat{z}_t$ and a selection of N negative samples, κ is the temperature variable and d calculates the cosine similarity $d(x, y) = \frac{x^T y}{||x||||y||}$. In our experiments, we uniformly sample $N = 50$ negative samples from the encodings $\hat{Z}$ of the utterance and κ 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 $\hat{Z} = [\hat{z}_1, \hat{z}_2, \dots \hat{z}_T]$ to the consistency vectors $S = [s_1, s_1, \dots s_T]$. We minimize the L_2 normed distance between S and X as

$$\mathcal{L}_c = ||x_t - s_t||_2 \quad (4)$$

2.8. Loss

During training, we minimize the primary contrastive loss together with a codebook loss component and the consistency loss as

$$\mathcal{L} = \mathcal{L}_m + \mathcal{L}_{cb} + \gamma\mathcal{L}_c \quad (5)$$

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 γ, 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

$$\mathcal{L}_d = \frac{GV - \sum_{g=1}^{G} \exp(-\sum_{v=1}^{V} p_{g,v} \log p_{g,v})}{GV} \quad (6)$$

where $p_{g,v}$ is the probability assignment by the g^{th} codebook on the v^{th} code. The weight α 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 1: *Different test sets for RNN-T model evaluation*

Test set	Details	# Utterances
clean($_{SNR_{20}}$)	Average SNR $\approx$ 20 dB	118.0k
clean($_{SNR_{16}}$)	Average SNR $\approx$ 16 dB	43.2k
noisy($_{N_1}$)	background multimedia speech	31.0k
noisy($_{N_2}$)	background speech, multiple speakers	5.8k

3.2. Recurrent Neural Network Transducer (RNN-T) Model

RNN-T [18, 19, 20] ASR models are widely used for deployable end-to-end speech recognition systems because of their fast online streaming capability. We use the pre-trained wav2vec-C and wav2vec 2.0 models to initialize the speech encoder for a RNN-T ASR model.

3.2.1. Pre-trained RNN-T

After training the self-supervised model on unlabeled data, we use the output of the context network g as speech representations. Thus, the RNN-T *speech encoder* consists of three LSTM layers followed by five layers of transformer extracted from the self-supervised model with the masking module eliminated. We use two LSTM layers with 1024 hidden units as the RNN-T *prediction network* and a simple single layer feed-forward *joint network*. The pre-trained speech encoder is also fine-tuned during RNN-T training. We use a total of 4000 sub-word tokens together with a blank token to generate the targets for RNN-T training. The RNN-T network is also regularized with SpecAugment on the input features with 10% of the temporal frames and 30% of the frequency bins randomly masked with noise. 25% dropout is applied on the transformer weights.

3.2.2. Baseline RNN-T

Our baseline model consists of an RNN-T with the same architecture as the pre-trained model but without pre-training the speech encoder.

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 1×10^{-7} and held at 5×10^{-6} after 3k steps. The RNN-T models are trained for 60k steps, with an average of 1 hours of speech per step. The learning rate is warmed up from 1×10^{-7} and held at 5×10^{-4} 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)*

RNN-T encoder	rWERR on different test sets				
	SNR_{20}	SNR_{16}	N_1	N_2	Average*
wav2vec 2.0 (KM)	0	0.6	0.7	3.2	0.7
wav2vec 2.0 (GS)	0	0.6	0.7	2.7	0.3
wav2vec-C (KM)	0.8	0.6	0.7	2.7	0.8
wav2vec-C (GS)	1.6	1.2	0.7	1.6	1.4

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.

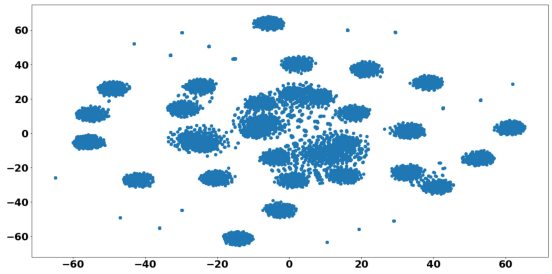
4.1. Discussions on codebook utilization

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

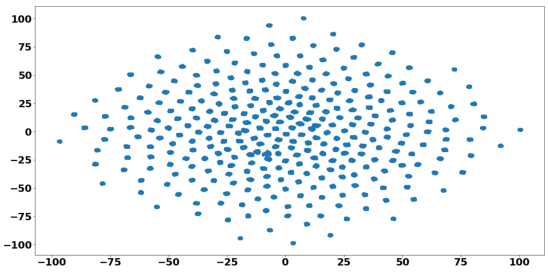
Table 3: *Codebook utilization of the self-supervised models*

self-supervised model	codebook utilization(%)
wav2vec 2.0 (KM)	< 1
wav2vec 2.0 (GS)	15
wav2vec-C (KM)	< 1
wav2vec-C (GS)	100

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.



(a) *wav2vec 2.0 (GS)*



(b) *wav2vec-C (GS)*

Figure 2: *t-SNE clustering of the wav2vec 2.0 (GS) and wav2vec-C (GS) codebooks*

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 N_2 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

[1] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *Proc. Interspeech 2019*, pp. 3465–3469, 2019.

[2] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

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

[4] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2019.

[5] A. Baevski, Y. Zhou, A.-r. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

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

[8] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," in *International Conference on Learning Representations*, 2019.

[9] M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, J. Monteiro, J. Trmal, and Y. Bengio, "Multi-task self-supervised learning for robust speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6989–6993.

[10] S. Ling, Y. Liu, J. Salazar, and K. Kirchhoff, "Deep contextualized acoustic representations for semi-supervised speech recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6429–6433.

[11] 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," *Proc. Interspeech 2019*, pp. 2613–2617, 2019.

[12] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6306–6315.

[13] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[14] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

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

[16] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 12, pp. 2041–2053, 2019.

[17] M. Van Segbroeck, A. Zaid, K. Kutsenko, C. Huerta, T. Nguyen, X. Luo, B. Hoffmeister, J. Trmal, M. Omologo, and R. Maas, "DiPCo–dinner party corpus," *arXiv preprint arXiv:1909.13447*, 2019.

[18] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[19] J. Guo, G. Tiwari, J. Droppo, M. Van Segbroeck, C.-W. Huang, A. Stolcke, and R. Maas, "Efficient minimum word error rate training of rnn-transducer for end-to-end speech recognition," *Proc. Interspeech 2020*, pp. 2807–2811, 2020.

[20] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 114–121.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.