*INTERSPEECH 2020*
October 25–29, 2020, Shanghai, China

# Vector-Quantized Autoregressive Predictive Coding

*Yu-An Chung, Hao Tang, James Glass*

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{andyyuan,haotang,glass}@mit.edu

## Abstract

Autoregressive Predictive Coding (APC), as a self-supervised objective, has enjoyed success in learning representations from large amounts of unlabeled data, and the learned representations are rich for many downstream tasks. However, the connection between low self-supervised loss and strong performance in downstream tasks remains unclear. In this work, we propose Vector-Quantized Autoregressive Predictive Coding (VQ-APC), a novel model that produces quantized representations, allowing us to explicitly control the amount of information encoded in the representations. By studying a sequence of increasingly limited models, we reveal the constituents of the learned representations. In particular, we confirm the presence of information with probing tasks, while showing the *absence* of information with mutual information, uncovering the model's preference in preserving speech information as its capacity becomes constrained. We find that there exists a point where phonetic and speaker information are amplified to maximize a self-supervised objective. As a byproduct, the learned codes for a particular model capacity correspond well to English phones.

**Index Terms**: self-supervised learning, unsupervised learning, representation learning, vector quantization, transfer learning

## 1. Introduction

Many high-level properties of speech, e.g., phonetic content and speaker characteristics, are not easily accessible[1] without sufficiently powerful transformations from the surface features such as audio waveforms and spectrograms. Speech representation learning aims to search for a transformation from the surface features that better reveals these properties to downstream tasks.

Recently, self-supervised learning—a paradigm that treats the input itself or modifications of the input as learning targets—has obtained promising results for learning such transformations [1, 2, 4, 5, 6, 7, 8, 9, 3, 10]. These methods, mostly inspired by the techniques for pre-training NLP models [11, 12, 13], learn speech representations by either inferring future information conditioned on historical audio [1, 2], or predicting masked parts of input audio [9, 3]. The resulting representations, obtained without requiring any additional labeled data, are able to outperform the surface features across downstream tasks such as speech recognition, speech translation, speaker identification, and emotion recognition [14, 15].

Autoregressive Predictive Coding (APC) [2] is one of the recent self-supervised speech representation models. APC defines a prediction task that trains an autoregressive neural model (e.g., a unidirectional RNN) to predict a future frame considering the past context. Although the learned representations contain highly accessible phonetic and speaker information, the reason why this seemingly unrelated self-supervised objective produces such a representation remains unclear. In this work, we aim to provide an explanation, investigating the constituents that lead to low objective values, and connect them with the properties of the learned representations.

Our approach is to study the properties of the learned representation as we limit the model capacity. The models with limited capacity are forced to retain information to achieve maximal prediction, thereby allowing us to study the constituents of the task and the learned representations. Several options are available to obtain a spectrum of models with different capacity, including reducing the number of layers, reducing the hidden layer size, or enforcing a bottleneck along the feed-forward process. The impact of different numbers of hidden layers has been studied in prior work [2]. Regardless, it is difficult to quantify the amount of information by changing the number of layers, changing the hidden layer size, or using low-rank matrices to enforce bottlenecks. In this work, we study the use of vector quantization (VQ), where the amount of information (i.e., bits required to transmit the codebook and the sequence of codes) can be exactly quantified, to control the capacity of the models.

Recent studies on VQ for representation learning, mostly motivated by the discrete nature of phonetic units, attempt to show that enforcing the quantization leads to a better representation for acoustic unit discovery [16, 17] and automatic speech recognition [7, 18]. In contrast, our goal is not to discover the discrete units in speech. We treat VQ as a general approach to limit the model capacity, and study its impact on the information encoded in the learned representations.

## 2. Proposed Models

### 2.1. A review on APC

Given a sequence of acoustic feature vectors $(x_1, x_2, ..., x_t)$ as context, APC incorporates an autoregressive neural model $g_{AR}$, e.g., a unidirectional RNN or a Transformer decoder [19, 12], to summarize the sequence for predicting a future frame $x_{t+n}$ that is $n$ steps ahead of $x_t$. Let $y_t$ denote the prediction of $g_{AR}$ at time $t$. In practice, for a speech utterance $\mathbf{x} = (x_1, x_2, ..., x_T)$, where $T$ is the sequence length, $g_{AR}$ is trained by minimizing the $L1$ frame-wise loss between the predicted sequence $(y_1, y_2, ..., y_{T-n})$ and the target sequence $(x_{1+n}, x_{2+n}, ..., x_T)$:

$$\sum_{t=1}^{T-n} |x_{t+n} - y_t|. \quad (1)$$

Code is available at https://github.com/iamyuanchung/VQ-APC.

[1]Following other recent works [1, 2, 3], in this study we use linear separability (or separability with a shallow network) to define the accessibility of information for a downstream task.



http://dx.doi.org/10.21437/Interspeech.2020-1228

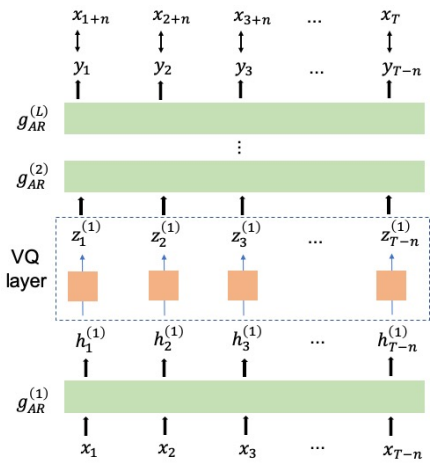


Figure 1: *A diagram of VQ-APC. $g_{AR}$ is an autoregressive model with L layers with $g_{AR}^{(\ell)}$ denoting the $\ell$-th layer and $h_t^{(\ell)}$ denoting the output vector of $g_{AR}^{(\ell)}$ at time t. The figure is an example of inserting a VQ layer (area inside the dashed box) between the first and second layers $g_{AR}^{(1)}$ and $g_{AR}^{(2)}$. The orange block represents the code variable lookup process that replaces $h_t^{(\ell)}$ by $z_t^{(\ell)}$, where $z_t^{(\ell)}$ is one of the elements in a codebook. The quantized hidden vectors are fed to the next layer and the feed-forward process continues. The training objective is the same as APC: to minimize the $L1$ loss between the predicted frame $y_t$ and the target future frame $x_{t+n}$.*

Once $g_{AR}$ is trained, one can extract features by taking its hidden representations, e.g., the last layer output, to replace the surface features as the new input to downstream models.

### 2.2. VQ-APC

Figure 1 illustrates the VQ-APC architecture, which is based upon APC with additional quantization layer(s). Assume $g_{AR}$ has $L$ layers. Let $g_{AR}^{(\ell)}$ denote the $\ell$-th layer of $g_{AR}$. After feeding $\mathbf{x} = (x_1, x_2, ..., x_T)$ to $g_{AR}$, each $g_{AR}^{(\ell)}$ will produce a sequence of hidden vectors $\mathbf{h}^{(\ell)} = (h_1^{(\ell)}, h_2^{(\ell)}, ..., h_T^{(\ell)})$. Next, we add a vector quantization (VQ) layer [16] that replaces $h_t^{(\ell)}$ by $z_t^{(\ell)}$, where $z_t^{(\ell)}$ is one of the $V$ elements in a codebook $\{c_1, \ldots, c_V\}$. We then pass the resulting hidden vectors $\mathbf{z}^{(\ell)} = (z_1^{(\ell)}, z_2^{(\ell)}, \ldots, z_T^{(\ell)})$ to the next layer $g_{AR}^{(\ell+1)}$ and continue the feed-forward process.

We use the Gumbel-Softmax with the straight-through estimator [20] for selecting discrete codebook variables in a fully differentiable way. Specifically, we apply a linear layer to map $h_t^{(\ell)}$ to a vector $r \in \mathbb{R}^V$. At test time, we simply choose the largest index in $r$. At training time, the probability $p_i$ of selecting the $i$-th code variable $c_i$ is computed as follows:

$$p_i = \frac{e^{(r_i + v_i)/\tau}}{\sum_{j=1}^{V} e^{(r_j + v_j)/\tau}}, \tag{2}$$

where $v = -\ln(-\ln(u))) \in \mathbb{R}^V$ and $u$ is uniformly sampled from $\mu(0, 1)$. $\tau$ controls how close the approximation is to argmax. During the forward pass, the argmax code $c_k$ where $k = \text{argmax}_i p_i$ is chosen; during the backward pass, the true gradients of the Gumbel-Softmax outputs are used. The training objective is the same as APC (Equation 1).

The codebook size and the code dimension of a VQ layer control the amount of information from the previous layer flowing to the next, and changing either of these two factors allows us to explicitly control the capacity of the models. As we limit the model capacity, the model is forced to retain information to achieve maximal prediction. By studying a sequence of increasingly limited models, we are able to reveal the constituents of the prediction task and the learned representations.

## 3. Experiments

We conduct experiments to study the properties of the learned representations and their connection to the self-supervised objective. Since VQ layers are known to significantly disrupt model training, we first examine where VQ layer(s) should be inserted. Next, by using phonetic and speaker classification as probing tasks, we show the model's preference in preserving speech information as its capacity becomes constrained. We then visualize the learned VQ codes to show the presence and absence of phonetic information and the correspondence between codes and phones. Finally, we compare VQ-APC with other self-supervised speech representation models.

### 3.1. Setup

**Training of self-supervised models.** All self-supervised models, including the VQ-APC variants and other models to be compared, are trained on the clean 360-hour subset of LibriSpeech [21]. We use 80-dimensional log Mel spectrograms (normalized to zero mean and unit variance per speaker) as input features, that is, $x_t \in \mathbb{R}^{80}$, and train all models for 100 epochs using Adam [22] with a batch size of 32 and an initial learning rate of $10^{-3}$.

**Probing tasks.** We consider phonetic and speaker classification for measuring the accessibility of the phonetic and speaker information contained in the representations, respectively. Both experiments are carried out on the Wall Street Journal corpus (WSJ) [23]. For phonetic classification, the goal is to correctly classify each frame in an utterance into one of the 42 phones. The phone alignments are generated with a speaker adapted GMM-HMM model. We follow the standard split of WSJ, using 90% of `si284` for training, the rest 10% for validation, and reporting phone error rate on `dev93`. For speaker classification, the goal is to correctly predict the speaker identity of an utterance. We follow [14] and consider a 259-class classification task where each class corresponds to an unique speaker, using 80% of `si284` for training, the other 10% for validation, and reporting classification error rate on the rest 10%. We note that speaker classification is not a typical task for WSJ, and only serves as a sanity check for the presence of speaker information (and its potentially correlated channel information) [1, 3]. The classifier for both tasks is a linear logistic regression that takes the features extracted from the self-supervised models as input. For speaker classification, the features from the same utterance are averaged before being fed to the classifier. All self-supervised models are kept frozen when training the linear classifier. All reported numbers are an average of 5 runs, of which variances are negligibly small and not included.

### 3.2. Preliminary VQ experiments

We first explore several potential places to insert VQ layers. For all VQ-APC variants in our experiments, the autoregressive model $g_{AR}$ is a 3-layer unidirectional GRU with 512 hidden

Table 1: *Phonetic classification results of VQ-APC with different VQ configurations. The layers where VQ is inserted are denoted as a set, and $\emptyset$ is equivalent to the regular APC. We compare both the hidden vectors $\mathbf{h}^{(\ell)}$ and their corresponding VQ codes $\mathbf{z}^{(\ell)}$ (when applicable) for $\ell = 1, 2, 3$ as extracted features. Training loss on LibriSpeech is also reported. The lowest phone error rate is highlighted in bold.*

| VQ config. | Train loss | Phone error rate | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\mathbf{h}^{(1)}$ | $\mathbf{h}^{(2}$ | $\mathbf{h}^{(3)}$ | $\mathbf{z}^{(1)}$ | $\mathbf{z}^{(2)}$ | $\mathbf{z}^{(3)}$ |
| $\emptyset$ | 0.68 | 46.5 | 38.6 | 33.3 | – | – | – |
| $\{1\}$ | 0.70 | 43.0 | 37.5 | 32.3 | 43.4 | – | – |
| $\{2\}$ | 0.72 | 45.8 | 35.4 | 31.5 | – | 36.0 | – |
| $\{3\}$ | 0.73 | 46.4 | 35.7 | **30.5** | – | – | 30.8 |
| $\{1, 2\}$ | 1.22 | 75.0 | 72.3 | 70.7 | 74.8 | 72.6 | – |
| $\{1, 3\}$ | 0.83 | 59.9 | 54.1 | 51.0 | 61.2 | – | 51.4 |
| $\{2, 3\}$ | 0.87 | 63.1 | 58.7 | 54.7 | – | 59.9 | 55.2 |
| $\{1, 2, 3\}$ | 1.21 | 75.3 | 74.1 | 68.5 | 75.4 | 75.2 | 67.8 |

units, and the target frame in the future, $n$, is set to 5 when training (Equation 1) on LibriSpeech. Whenever a VQ layer is added, the embedding dimension of each code is 512, and $\tau$ for the Gumbel-Softmax straight-through estimator (Equation 2) is a fixed value of 0.1 throughout training.

Table 1 presents the phonetic classification results of adding VQ layers to different layers in $g_{AR}$. In the "VQ config." column, the numbers inside the parenthesis denote the layers we insert a VQ layer. For example, $\{1\}$ means that we only add VQ layer after $g_{AR}^{(1)}$. $\emptyset$ denotes the case where no VQ layer is applied, equivalent to the regular APC. The codebook size here is 128. We try using both the hidden vectors $\mathbf{h}^{(\ell)}$ and their quantized codes $\mathbf{z}^{(\ell)}$ (when applicable) for $\ell = 1, 2, 3$ as extracted features. We also include the final VQ-APC training loss on the LibriSpeech 360-hour subset after 100 epochs (not the downstream linear classifier's training loss on WSJ).

**Quantizing one layer.** As indicated by the training loss, we see that the bottleneck imposed by the VQ layer indeed handicaps the models' ability to predict the future, as $\{1\}$, $\{2\}$, and $\{3\}$ all have higher training loss than $\emptyset$. In terms of phone error rate, regardless of where VQ is inserted, we see improvement over the APC representations. Inserting VQ at the third layer leads to the most improvement, from 33.3 to 30.5. The quantized codes $\mathbf{z}^{(\ell)}$, when applicable, could also be used as extracted features, which perform similarly to their corresponding pre-quantized representations. For example, in $\{3\}$, $\mathbf{z}^{(3)}$ (30.8) is close to $\mathbf{h}^{(3)}$ (30.5).

**Quantizing multiple layers.** We find that our VQ-APC models with multiple VQ layers have trouble fitting the training set. Their representations are also much worse than the regular APC on phonetic classification. One potential remedy is to enable VQ with a schedule [17], but is beyond the scope of the paper.

### 3.3. The constituents of the learned representations

Experiments so far suggest that the phonetic information is still present (if not better) after using VQ. For the rest of the paper, we will focus on the case where VQ is inserted at the third layer, i.e., the case of $\{3\}$. To study the constituents of the learned representations, we train a series of increasingly limited VQ-APC models by decreasing the codebook size from 2048 to 64 while fixing the code dimension to 512. As the codebook size becomes smaller, the model is forced to choose what information to encode and what to discard, thus revealing the constituents of the learned representations. We show the training losses of these models at convergence and the respective phone and speaker error rates in Figure 2. The dashed lines are the training loss, phone error rate, and speaker error rate of a regular APC model.

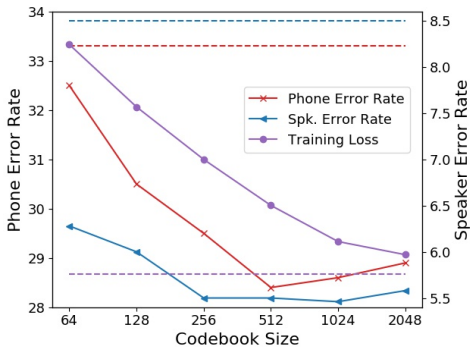


Figure 2: *Training loss (purple), phonetic classification (red), and speaker classification (blue) results of VQ-APC with VQ configuration $\{3\}$ using varying codebook sizes. The x-axis is the codebook size, decreasing from 2048 to 64, and the y-axis on the left is the corresponding phone error rate and on the right the speaker (spk.) error rate. For clarity, the vertical axis for training loss is not displayed. The three dash horizontal lines show the corresponding results of a regular APC, i.e., $\emptyset$.*

First, the training loss (purple curve) increases as expected, showing worse fit on the training set as we limit the codebook size. Note that in theory, when the codebook size goes to infinity, we recover the regular APC. The phone error rate (red curve) obtains a minimum at codebook size 512, and starts to worsen with smaller codebook size. The sharp degradation in phone error rate suggests that the model discards certain phonetic information to achieve maximal self-supervised objective.

The speaker error rate (blue curve), on the contrary, does not change by much as we limit the codebook size. This shows that the speaker information (and its potentially correlated channel information) is mostly retained. Given the sharp degradation in phone error rate, we can conclude that the model prefers to retain speaker information over phonetic information to achieve maximal future prediction. The preference of information can potentially stem from the use of GRUs, the VQ configuration, and the self-supervised, future prediction objective. More analyses are needed to disentangle the among these causes.

On the other hand, when the codebook size becomes large, the model falls back to regular APC and might suffer from overfitting [24], paying unnecessary attention to the spectral details that does not generalize for predicting future frames. Finally, even with a codebook size of 64, we still see gains over regular APC, showing the strong performance of VQ-APC in representation learning.

### 3.4. Visualizations of learned codes

To better measure the correspondence between the learned VQ codes and English phones, we compute co-occurrence statistics (at the frame level) across the 360-hour subset of LibriSpeech, the dataset we use to train the VQ-APC models. We compare three settings, $\{1\}$, $\{2\}$, and $\{3\}$ with a codebook size of 128. The conditional probability $P(\text{phone}|\text{code})$, as shown

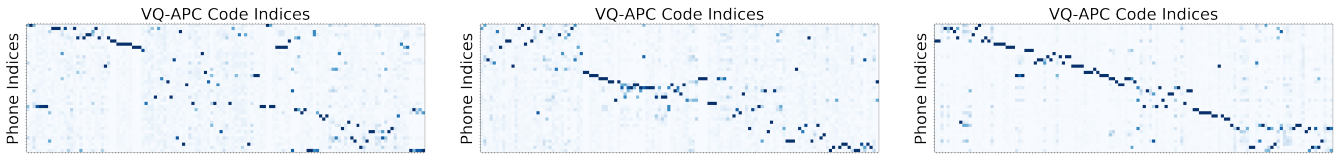


Figure 3: *From left to right, visualizations of the conditional probability matrix* $P(\text{phone}|\text{code})$ *for configurations* $\{1\}$, $\{2\}$, *and* $\{3\}$ *with 128 codes, respectively. Each sub-figure is a* $42 \times 128$ *conditional probability matrix, where each row and column correspond to a phone and code index, respectively. Color scaling is saturated at probability 0.5 for better visualization.*

in Figure 3, are estimated based on the co-occurrence statistics, i.e., via maximum likelihood. In each sub-figure, the rows and columns are ordered via spectral co-clustering with 15 clusters to group together phones that share similar sets of codes, and a diagonal segment would imply a high correspondence between phones and codes. Note that the phone labels of LibriSpeech are only used for analysis and never seen during training.

From Figure 3, we see that the correspondence between phones and VQ codes is stronger when quantized at higher layers, and is especially strong for $\{3\}$. Recall that probing tasks are useful for showing the presence of certain information, but have difficulty showing the absence of it. In contrast, given the co-occurrence statistics, mutual information can be estimated to support the absence of information. The normalized mutual information are 0.167, 0.285, and 0.406 for $\{1\}$, $\{2\}$, and $\{3\}$, respectively. In other words, not only can we conclude that the learned representations in $\{3\}$ contain phonetic information, we can also readily conclude that $\{1\}$ and $\{2\}$ contain much less information for certain phones.

### 3.5. A comparison with other self-supervised models

Finally, we compare VQ-APC with other self-supervised speech representation models, including Contrastive Predictive Coding (CPC) [1], Bidirectional Masked Reconstruction [9], Mockingjay [3], and Multi-Target APC [24]. We briefly review these methods below, and show their results on phonetic and speaker classification in Table 2. To stay as close to the original implementation as possible, we do not separate the discussion of models, such as the use RNNs or Transformers, and the self-supervised objectives.

CPC and APC share a similar methodology as both use an autoregressive model to learn representations through conditioning on past frames to predict information about a future frame. Their difference is that while APC tries to directly predict the future frame via regression, CPC aims to learn representations containing information that most discriminates the future frame from a set of negative samples. We mainly follow the original paper [1] for implementing CPC with some modifications described in [2]. These modifications are meant to minimize the architectural differences between APC and CPC while maintaining their training objectives.

Multi-Target APC (MT-APC) is an extension of APC. It incorporates an auxiliary objective serving as a regularizer to improve the generalization of the main future prediction task. The exact same setup described in [24] is used in our experiments.

Different from CPC and APC that are based on the idea of future prediction, Bidirectional Masked Reconstruction (BMR) and Mockingjay are under the category of *masked prediction*. Inspired by the masked language modeling technique from BERT [13], BMR and Mockingjay mask parts of the input signals, and predict them through conditioning on both past and future contexts with a bidirectional RNN and Transformer encoder [25], respectively. For experiments, we mainly follow the implementations in the original papers [9, 3], except that the number of layers are reduced to match ours to minimize the architectural differences.

Table 2: *Phonetic and speaker classification results of different self-supervised speech representation models. All features are fed to a linear logistic repression. For log Mel, we also include the results of using a 1- and 3-layer multi-layer perceptron, denoted as MLP-1 and MLP-3, respectively. We also note the neural architectures used by each model.*

| Model | Network | Phone error rate | Speaker error rate |
|---|---|---|---|
| log Mel | – | 50.3 | 17.6 |
| log Mel + MLP-1 | – | 43.1 | 12.3 |
| log Mel + MLP-3 | – | 41.2 | 11.9 |
| CPC [1] | 3-layer uni-GRU | 34.1 | 9.7 |
| APC [2] | 3-layer uni-GRU | 33.3 | 8.5 |
| MT-APC [24] | 3-layer uni-GRU | 30.5 | 7.3 |
| VQ-APC (ours) | 3-layer uni-GRU | 28.4 | 5.5 |
| BMR [9] | 3-layer bi-GRU | 32.4 | 6.2 |
| Mockingjay [3] | 3-layer Transformer | 30.8 | 5.1 |

On phonetic classification, we see that VQ-APC (28.4) improves over APC (33.3) and MT-APC (30.5), demonstrating the effectiveness of VQ layers. It also outperforms other self-supervised models despite using the same (vs. CPC) or smaller (vs. BMR and Mockingjay) network. On speaker classification, VQ-APC (5.5) again improves over the other two APC models (8.5 and 7.3), and is on par with the best model (Mockingjay, 5.1). On both tasks, all self-supervised models outperform log Mel regardless of the type of classifier it uses.

## 4. Conclusions

We have demonstrated that incorporating vector quantization (VQ) layers into an Autoregressive Predictive Coding model imposes a bottleneck, forcing the model to learn better representations. Extensive experiments have been conducted to compare different VQ configurations, to study the effect of varying codebook sizes, and to compare with other self-supervised speech representation models. We show evidence for the presence and absence of phonetic and speaker information in the learned representations, and also show the model's preference in retaining information when the model capacity is limited, in the hope to bridge the connection between the self-supervised objective and the properties of the learned representations. When the phonetic information is present, the learned VQ codes also correspond well with English phones.

# 5. References


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

[2] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," in *Interspeech*, 2019.

[3] A. Liu, S.-W. Yang, P.-H. Chi, P.-C. Hsu, and H.-Y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional Transformer encoders," in *ICASSP*, 2020.

[4] J. Chorowski, R. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *TASLP*, vol. 27, no. 12, pp. 2041–2053, 2019.

[5] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Interspeech*, 2019.

[6] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," in *Interspeech*, 2019.

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

[8] A. Baevski and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," in *ICASSP*, 2020.

[9] W. Wang, Q. Tang, and K. Livescu, "Unsupervised pre-training of bidirectional speech encoders via masked reconstruction," in *ICASSP*, 2020.

[10] S. Ling, Y. Liu, J. Salazar, and K. Kirchhoff, "Deep contextualized acoustic representations for semi-supervised speech recognition," in *ICASSP*, 2020.

[11] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018.

[12] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, Tech. Rep., 2018.

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

[14] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP*, 2020.

[15] 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.

[16] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *NIPS*, 2017.

[17] D. Harwath, W.-N. Hsu, and J. Glass, "Learning hierarchical discrete linguistic units from visually-grounded speech," in *ICLR*, 2020.

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

[19] P. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," in *ICLR*, 2018.

[20] E. Jang, S. Gu, and B. Poole, "Categorical reparametrization with gumble-softmax," in *ICLR*, 2017.

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

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[23] D. Paul and J. Baker, "The design for the wall street journal-based CSR corpus," in *Speech and Natural Language Workshop*, 1992.

[24] Y.-A. Chung and J. Glass, "Improved speech representations with multi-target autoregressive predictive coding," in *ACL*, 2020.

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