VQ-WAV2VEC: SELF-SUPERVISED LEARNING OF DISCRETE SPEECH REPRESENTATIONS

Anonymous authors
Paper under double-blind review

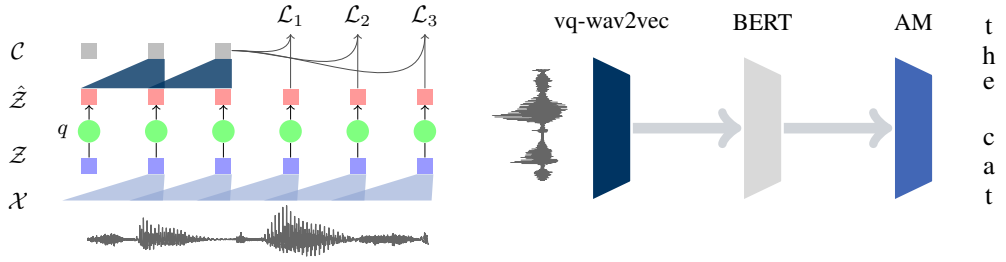
ABSTRACT

We propose vq-wav2vec to learn discrete representations of audio segments through a wav2vec-style self-supervised context prediction task. The algorithm uses either a gumbel softmax or online k-means clustering to quantize the dense representations. Discretization enables the direct application of algorithms from the NLP community which require discrete inputs. Experiments show that BERT pre-training achieves a new state of the art on TIMIT phoneme classification and WSJ speech recognition.[1]

1 INTRODUCTION

Self-supervised representation learning has led to large improvements in natural language processing (Peters et al., 2018; Radford et al., 2018; Baevski et al., 2019; Devlin et al., 2018; Edunov et al., 2019; Lample & Conneau, 2019; Yang et al., 2019; Liu et al., 2019). The same topic has also been an active area of research in computer vision and speech processing (van den Oord et al., 2018; Hénaff et al., 2019; Bachman et al., 2019; Schneider et al., 2019). However, the algorithms used to perform self-supervision differ because computer vision and speech processing data is continuous, whereas in NLP both inputs and outputs are discrete units.

In this paper, we aim to make well performing NLP algorithms more widely applicable. We propose a new algorithm to discretize the speech signal and then apply NLP algorithms to speech data (Figure 1a). Different to Chorowski et al. (2019) who learn discrete latent representations based on autoencoding the audio signal, we learn latent variables that are useful to predict context information (van den Oord et al., 2018; Schneider et al., 2019).



(a) vq-wav2vec (b) Discretized speech training pipeline

Figure 1: (a) The vq-wav2vec encoder maps raw audio ($\mathcal{X}$) to a dense representation ($\mathcal{Z}$) which is quantized (q) to $\hat{\mathcal{Z}}$ and aggregated into context representations ($\mathcal{C}$); training requires future time step prediction. (b) Acoustic models are trained by quantizing the raw audio with vq-wav2vec, then applying BERT to the discretized sequence and feeding the resulting representations into the acoustic model to output transcriptions.

Our new discretization algorithm learns discrete representations of fixed length segments of audio signal by utilizing the wav2vec loss and architecture (Schneider et al, 2019; §2). To choose the

[1]We will open source the code for our models.

discrete variables, we consider a gumbel softmax approach (Jang et al., 2016) as well as online k-means clustering, similar to VQ-VAE (Oord et al., 2017; §3).

We then train a Deep Bidirectional Transformer (BERT; Devlin et al., 2018; Liu et al., 2019) on the discretized unlabeled speech data and input these representations to a standard acoustic model (Figure 1b; §4). Our experiments show that BERT representations perform significantly better than log-mel filterbank inputs as well as dense wav2vec representations on both TIMIT and WSJ benchmarks. Discretization of audio data enables the direct application of a whole host of algorithms from the NLP literature to speech data. For example, we show initial results of training a standard sequence to sequence model from the NLP literature as a speech recognition model over discrete audio tokens (§5, §6).

2 BACKGROUND

2.1 WAV2VEC

wav2vec (Schneider et al., 2019) learns representations of audio data by solving a self-supervised context-prediction task with the same loss function as word2vec (Mikolov et al., 2013; van den Oord et al., 2018). The model is based on two convolutional neural networks where the the *encoder* produces a representation $\mathbf{z}_i$ for each time step *i* at a rate of 100 Hz and the *aggregator* combines multiple encoder time steps into a new representation $\mathbf{c}_i$ for each time step *i*. Given an aggregated representation $\mathbf{c}_i$, the model is trained to distinguish a sample $\mathbf{z}_{i+k}$ that is k steps in the future from distractor samples $\tilde{\mathbf{z}}$ drawn from a distribution p_n, by minimizing the contrastive loss for steps $k = 1, \dots, K$:

$$\mathcal{L}_k^{\text{wav2vec}} = -\sum_{i=1}^{T-k} \Big(\log \sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i)) + \lambda \mathop{\mathbb{E}}_{\tilde{\mathbf{z}} \sim p_n} [\log \sigma(-\tilde{\mathbf{z}}^\top h_k(\mathbf{c}_i))] \Big) \tag{1}$$

where T is the sequence length, $\sigma(x) = 1/(1 + \exp(-x))$, and where $\sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i))$ is the probability of $\mathbf{z}_{i+k}$ being the true sample. We consider a step-specific affine transformation $h_k(\mathbf{c}_i) = W_k \mathbf{c}_i + \mathbf{b}_k$ that is applied to $\mathbf{c}_i$ (van den Oord et al., 2018). We optimize the loss $\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k$, summing (1) over different step sizes. After training, the representations produced by the context network $\mathbf{c}_i$ are input to the acoustic model instead of log-mel filterbank features.

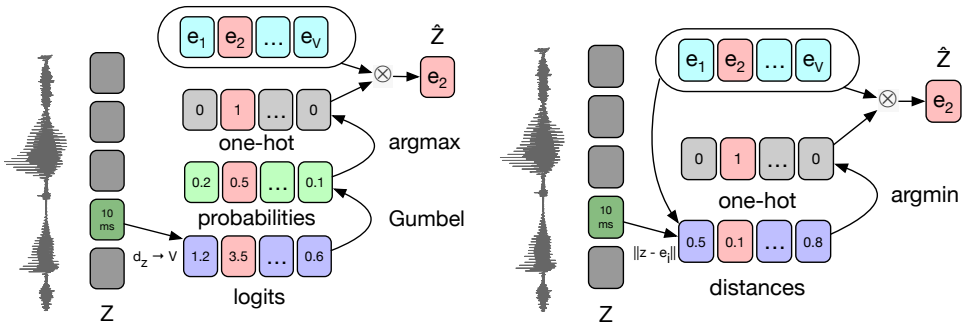
2.2 BERT

BERT (Devlin et al., 2018) is a pre-training approach for NLP tasks, which uses a transformer encoder model to build a representation of text. Transformers uses self-attention to encode the input sequence as well as an optional source sequence (Vaswani et al., 2017). The original BERT model combined two tasks for training: first, masked language modeling randomly removes some of the input tokens and the model has to predict those missing tokens. Second, next sentence prediction splices two different text passages together into a single example and the model needs to predict whether the passages are from the same document.

3 VQ-WAV2VEC

Our approach, vq-wav2vec learns vector quantized (VQ) representations of audio data using a future time-step prediction task. We follow the same architectual choices as wav2vec (§2.1) with two convolutional networks $f : \mathcal{X} \mapsto \mathcal{Z}$ and $g : \hat{\mathcal{Z}} \mapsto \mathcal{C}$ for feature extraction and aggregation, as well as a new *quantization* module $q : \mathcal{Z} \mapsto \hat{\mathcal{Z}}$ to build discrete representations (Figure 1a). We first map 30ms segments of raw speech to a dense feature representation $\mathbf{z}$ at a stride of 10ms using the encoder network f. Next, the quantizer (q) turns these dense representations into discrete indices which are mapped to a reconstruction $\hat{\mathbf{z}}$ of the original representation $\mathbf{z}$. We feed $\hat{\mathbf{z}}$ into the aggregator g and optimize the same context prediction task as wav2vec outlined in §2.1.

The quantization module replaces the original representation $\mathbf{z}$ by $\hat{\mathbf{z}} = \mathbf{e}_i$ from a fixed size codebook $\mathbf{e} \in \mathbb{R}^{V \times d}$ which contains V representations of size d. We consider the Gumbel Softmax which is a differentiable approximation of the argmax for computing one-hot representations (§3.1; Figure 2a)



(a) Gumbel Softmax

(b) K-means clustering.

Figure 2: (a) The gumbel softmax quantization computes logits representing the codebook vectors ($\mathbf{e}$). In the forward pass the argmax codeword ($\mathbf{e}_2$) is chosen and for backward (not shown) the exact probabilities are used. (b) K-means vector quantization computes the distance to all codeword vector and chooses the closest (argmin).

as well as online k-means clustering, similar to the vector quantized variational autoencoder (VQ-VAE; Oord et al., 2017; §3.2; Figure 2b). Finally, we perform multiple vector quantizations over different parts of $\mathbf{z}$ to mitigate mode collapse (§3.3).

3.1 Gumbel Softmax

The Gumbel-Softmax (Gumbel, 1954; Jang et al., 2016; Maddison et al., 2014) enables selecting discrete codebook variables in a fully differentiable way and we use the straight-through estimator of Jang et al. (2016). Given the dense representation $\mathbf{z}$, we apply a linear layer, followed by a ReLU and another linear which outputs $\mathbf{l} \in \mathbb{R}^V$ logits for the Gumbel softmax. At inference, we simply pick the largest index in l. At training, the output probabilities for choosing the j-th variable are

$$p_j = \frac{\exp(l_j + v_j)/\tau}{\sum_{k=1}^{V} \exp(l_k + v_k)/\tau}, \tag{2}$$

where $v = -\log(-\log(u))$ and u are uniform samples from $\mathcal{U}(0, 1)$. During the forward pass, $i = \text{argmax}_j p_j$ and in the backward pass, the true gradient of the Gumbel softmax outputs is used.

3.2 K-Means

The vector quantization approach of van den Oord et al. (2017) is an alternative to making the index selection procedure fully differentiable. Different to their setup, we optimize a future time step prediction loss instead of the reconstruction loss of an autoencoder.

We choose the codebook variable representation by finding the closest variable to the input features $\mathbf{z}$ in terms of the Euclidean distance, yielding $i = \text{argmin}_j \|\mathbf{z} - \mathbf{e}_j\|_2^2$. During the forward pass, we select $\hat{\mathbf{z}} = \mathbf{e}_i$ by choosing the corresponding variable from the codebook. We obtain gradients for the encoder network by back-propagating $\mathrm{d}\mathcal{L}^{\text{wav2vec}}/\mathrm{d}\hat{\mathbf{z}}$ (van den Oord et al., 2017). The final loss has two additional terms:

$$\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k^{\text{wav2vec}} + \Big(\|\text{sg}(\mathbf{z}) - \hat{\mathbf{z}}\|^2 + \gamma \|\mathbf{z} - \text{sg}(\hat{\mathbf{z}})\|^2 \Big), \tag{3}$$

where $\text{sg}(x) \equiv x$, $\frac{\mathrm{d}}{\mathrm{d}x}\text{sg}(x) \equiv 0$ is the stop gradient operator and γ is a hyper-parameter. The first term is the future prediction task and gradients do not change the codebook because of the straight-through gradient estimation of mapping $\mathbf{z}$ to $\hat{\mathbf{z}}$. The second term $\|\text{sg}(\mathbf{z}) - \hat{\mathbf{z}}\|^2$ moves the codebook vectors closer to the encoder output, and the third term $\|\mathbf{z} - \text{sg}(\hat{\mathbf{z}})\|^2$ makes sure that the encoder outputs are close to a centroid (codeword).

3.3 VECTOR QUANTIZATION WITH MULTIPLE VARIABLE GROUPS

So far, we considered replacing the encoder feature vector $\mathbf{z}$ by a single entry $\mathbf{e}_i$ in the codebook. This approach is prone to mode collapse, which can be mitigated my workarounds such as re-initializing codewords or applying additional regularizers to the loss function. In the following, we describe another work around where we independently quantize partitions of $\mathbf{z}$.

The dense feature vector $\mathbf{z} \in \mathbb{R}^d$ is first organized into multiple *groups* G into the matrix form $\mathbf{z}' \in \mathbb{R}^{G \times (d/G)}$. We then represent each row by an integer index, and hence can represent the full feature vector by the indices $\mathbf{i} \in [V]^G$, where V again denotes the possible number of *variables* for this particular group and each element $\mathbf{i}_j$ corresponds to a fixed codebook vector. For each of the G groups, we apply either one of the two VQ approaches (§3.1 and §3.2).

The codebook itself can be initialized in two possible ways: Codebook variables can be shared across groups, i.e., a particular index in group j would reference the same vector as the same index in group j'. This yields a codebook $\mathbf{e} \in \mathbb{R}^{V \times (G/d)}$. In contrast, not sharing the codebook variables yields a codebook of size $\mathbf{e} \in \mathbb{R}^{V \times G \times (G/d)}$. In practise, we observe that sharing the codebook variables generally yields competitive results to a non-shared representation.

4 BERT PRE-TRAINING ON QUANTIZED SPEECH

Once we trained a vq-wav2vec model we can discretize audio data and make it applicable to algorithms that require discrete inputs. One possibility is to use the discretized training data and apply BERT pre-training where the task is to predict masked input tokens based on an encoding of the surrounding context (Devlin et al., 2018). Once the BERT model is trained, we can use it to build representations and feed them into an acoustic model to improve speech recognition. We follow recent advances in BERT training which only use the masked input token prediction Liu et al. (2019).

Since each of the discretized tokens represents around 10 ms of audio it is likely too easy to predict a single masked input token. We therefore change BERT training by masking *spans* of consecutive discretized speech tokens, similar to Joshi et al. (2019). To mask the input sequence, we randomly sample $p = 0.05$ of all tokens to be a starting index, without replacement, and mask $M = 10$ consecutive tokens from every sampled index; spans may overlap. This makes the masked token prediction harder and we show later that it improves accuracy over masking individual tokens (§6.5).

5 EXPERIMENTAL SETUP

5.1 DATASETS

We generally pre-train vq-wav2vec and BERT on the full 960h of Librispeech (Panayotov et al., 2015) and after vq-wav2vec training it is discretized to 345m tokens. Where indicated we perform ablations on a clean 100h subset which is discretized to 36M tokens. We evaluate models on two benchmarks: TIMIT (Garofolo et al., 1993b) is a 5h dataset with phoneme labels and Wall Street Journal (WSJ; Garofolo et al. 1993a) is a 81h dataset for speech recognition. For TIMIT, we apply the standard evaluation protocol and consider 39 different phonemes. For WSJ, we train acoustic models directly on 31 graphemes, including the English alphabet, the apostrophe, the silence token and tokens for repeating characters.

5.2 VQ-WAV2VEC

We adapt the fairseq implmentation of wav2vec (Schneider et al., 2019; Ott et al., 2019) and use vq-wav2vec/wav2vec models with 34×10^6 parameters. The encoder has 8 layers with 512 channels each, kernel sizes (10,8,4,4,4,1,1,1) and strides (5,4,2,2,2,1,1,1), yielding a total stride of 160. Each layer contains a convolution, followed by dropout, group normalization with a single group (Wu & He, 2018) and a ReLU non-linearity. The aggregator is composed of 12 layers, with 512 channels, stride 1, and kernel sizes starting at 2 and increasing by 1 for every subsequent layer. The block structure is the same as for the encoder network, except we introduce skip connections between each subsequent block.

We train with the wav2vec context prediction loss (Equation 1) for 400k updates, predicting $K = 8$ steps into the future and sample 10 negatives from the same audio example. Training is warmed up for 500 steps where the learning rate is increased from 1×10^{-7} to 5×10^{-3}, and then annealed to 1e-06 using a cosine schedule (Loshchilov & Hutter, 2016). The batch size is 10, and we crop a random section of 150,000 frames for each example (approximately 9.3 seconds for 16kHz sampling rate). All models are trained on 8 GPUs.

For ablations and experiments on the 100h Librispeech subset, we use a smaller model with kernels (10,8,4,4,4) and strides (5,4,2,2,2) in the encoder and seven convolutional layers with stride one and kernel size three in the aggregator. This model is trained for 40k updates.

Gumbel Softmax Models. We use $G = 2$ groups and $V = 320$ latents per group, the linear layer projects the features produced by the encoder to 1024 dimensions and we compute $G \cdot V$ logits. The Gumbel softmax produces a one-hot vector for each group G. The temperature τ is linearly annealed from 2 to 0.5 over the first 70% of updates and then kept constant at 0.5. This enables the model to learn which latents work best for each input before committing to a single latent. After training this model on 960h of Librispeech and quantizing the training dataset, we are left with 13.5k unique codewords combinations (out of V^G = 102k possible codewords).

k-means Models. We use $G = 2$ groups and $V = 320$ variables per group. vq-wav2vec on full Librispeech yields 23k unique codewords. Following van den Oord et al. (2017), we found $\gamma = 0.25$ to be a robust choice for balancing the VQ auxiliary loss.

5.3 BERT

BERT base models have 12 layers, model dimension 768, inner dimension (FFN) 3072 and 12 attention heads (Devlin et al., 2018). The learning rate is warmed up over the first 10,000 updates to a peak value of 1×10^{-5}, and then linearly decayed over a total of 250k updates. We train on 128 GPUs with a batch size of 3072 tokens per GPU giving a total batch size of 393k tokens (Ott et al., 2018). Each token represents 10ms of audio data.

BERT small. For our ablations we use a smaller setup with model dimension 512, FFN size 2048, 8 attention heads and dropout 0.05. These models are trained for 250k updates with a batch size of 2 examples per GPU.

5.4 Acoustic Model

We use wav2letter as accoustic model (Collobert et al., 2016; 2019) and train for 1k epochs on 8 GPUs for both TIMIT and WSJ using the auto segmentation criterion. For decoding the emissions from the acoustic model on WSJ we use a lexicon as well as a separate language model trained on the WSJ language modeling data only. We consider a 4-gram KenLM language model (Heafield et al., 2013) and a character based convolutional language model (Likhomanenko et al., 2019) and tune the models with the same protocol as Schneider et al. (2019).

6 Results

6.1 WSJ Speech Recognition

We first evaluate on the WSJ speech recognition benchmark. We train a vq-wav2vec model on the unlabeled version of Librispeech, then discretize the same data with the resulting model to estimate a BERT model. Finally, we train a wav2letter acoustic model on WSJ by inputting either the BERT or vq-wav2vec representations instead of log-mel filterbanks.

We compare to various results from the literature, including wav2vec (Schneider et al., 2019) and we consider three setups: performance without any language model (No LM), with an n-gram LM (4-gram LM) and with a character convolutional LM (Char ConvLM). We report the accuracy of wav2letter with log-mel filterbanks as input (Baseline) and wav2vec. For vq-wav2vec we first experiment with the gumbel softmax, with and without a BERT base model (§5.3).

	nov93dev		nov92	
	LER	WER	LER	WER
Deep Speech 2 (12K h labeled speech; Amodei et al., 2016)	-	4.42	-	3.1
Trainable frontend (Zeghidour et al., 2018)	-	6.8	-	3.5
Lattice-free MMI (Hadian et al., 2018)	-	5.66†	-	2.8†
Supervised transfer-learning (Ghahremani et al., 2017)	-	4.99†	-	2.53†
NO LM				
Baseline (log-mel)	6.28	19.46	4.14	13.93
wav2vec (Schneider et al., 2019)	5.07	16.24	3.26	11.20
vq-wav2vec gumbel	7.04	20.44	4.51	14.67
+ BERT base	**4.13**	**13.40**	**2.62**	**9.39**
4-GRAM LM (Heafield et al., 2013)				
Baseline (log-mel)	3.32	8.57	2.19	5.64
wav2vec (Schneider et al., 2019)	2.73	6.96	1.57	4.32
vq-wav2vec gumbel	3.93	9.55	2.40	6.10
+ BERT base	**2.41**	**6.28**	**1.26**	**3.62**
CHAR CONVLM (Likhomanenko et al., 2019)				
Baseline (log-mel)	2.77	6.67	1.53	3.46
wav2vec (Schneider et al., 2019)	2.11	5.10	0.99	2.43
vq-wav2vec gumbel + BERT base	**1.79**	**4.46**	**0.93**	**2.34**

Table 1: WSJ accuracy of vq-wav2vec on the development (nov93dev) and test set (nov92) in terms of letter error rate (LER) and word error rate (WER) without s language modeling (No LM), a 4-gram LM and a character convolutional LM. vq-wav2vec with BERT pre-training improves over the best wav2vec model (Schneider et al., 2019).

	nov93dev		nov92	
	LER	WER	LER	WER
NO LM				
wav2vec (Schneider et al., 2019)	5.07	16.24	3.26	11.20
vq-wav2vec gumbel	7.04	20.44	4.51	14.67
+ BERT small	4.52	14.14	2.81	9.69
vq-wav2vec k-means (39m codewords)	5.41	17.11	3.63	12.17
vq-wav2vec k-means	7.33	21.64	4.72	15.17
+ BERT small	4.31	13.87	2.70	9.62
4-GRAM LM (Heafield et al., 2013)				
wav2vec (Schneider et al., 2019)	2.73	6.96	1.57	4.32
vq-wav2vec gumbel	3.93	9.55	2.40	6.10
+ BERT small	2.67	6.67	1.46	4.09
vq-wav2vec k-means (39m codewords)	3.05	7.74	1.71	4.82
vq-wav2vec k-means	4.37	10.26	2.28	5.71
+ BERT small	2.60	6.62	1.45	4.08

Table 2: Comparison of gumbel softmax and k-means vector quantization on WSJ (cf. Table 1).

Table 1 shows that vq-wav2vec together with BERT training can achieve a new state of the art of 2.34 WER on nov92. Gains are largest when no language model is used which is the fastest setting. vq-wav2vec with gumbel uses only 13.5k distinct codewords to represent the audio signal and this limited set of codewords is not sufficient to outperform the baseline. However, it does enable training BERT models which require a relatively small vocabulary.

Next, we compare gumbel softmax to k-means for vector quantization. For this experiment we use the faster to train BERT small configuration (§5.3). We also train a vq-wav2vec k-means model with a very large number of codewords (39m) to test whether a more expressive model can close

	dev PER	test PER
CNN + TD-filterbanks (Zeghidour et al., 2018)	15.6	18.0
Li-GRU + fMLLR (Ravanelli et al., 2018)	–	14.9
wav2vec (Schneider et al., 2019)	12.9	14.7
Baseline (log-mel)	16.9	17.6
vq-wav2vec, gumbel	15.34	17.78
+ BERT small	9.64	**11.64**
vq-wav2vec, k-means	15.65	18.73
+ BERT small	9.80	11.40

Table 3: TIMIT phoneme recognition in terms of phoneme error rate (PER). All our models use the CNN-8L-PReLU-do0.7 architecture (Zeghidour et al., 2018).

	dev clean	dev other	test clean	test other
Cnv Cxt Tsf (Mohamed et al., 2019)	4.8	12.7	4.7	12.9
vq-wav2vec gumbel + Transformer Big	6.7	18.8	6.9	21.4

Table 4: Librispeech results for a standard sequence to sequence model trained over discretized transcribed data. All results without a supplemental language model.

the gap to wav2vec. Table 2 shows that gumbel softmax and k-means clustering perform relatively comparably: in the no language model setup without BERT, gumbel softmax is more accurate than k-means but these differences disappear with BERT. For 4-gram LM setup, k-means is better but those differences disappear again after BERT training. Finally, the large codeword model can substantially reduce the gap to the original wav2vec model.

6.2 TIMIT Phoneme Recognition

Next, we experiment on the much smaller TIMIT phoneme recognition task where we also pre-train vq-wav2vec on the full Librispeech corpus. Table 3 shows that vq-wav2vec and BERT achieve a new state of the art of 11.67 PER which corresponds to a 21% reduction in error over the previous best result of wav2vec.

6.3 Sequence to Sequence Modeling

So far we used vq-wav2vec to train BERT models on discretized speech data. However, once the audio data is discretized we can also train a standard sequence to sequence model to perform speech recognition. In preliminary experiments, we trained an off-the-shelf Big Transformer with minimal tuning (Vaswani et al., 2017; Ott et al., 2019) on the vq-wav2vec gumbel discretized Librispeech corpus and evaluated on the Librispeech dev/test sets using the standard data splits. As output vocabulary we use a 4k BPE types (Sennrich et al., 2016) estimated on the Librispeech transcriptions of the training set. We compare against Mohamed et al. (2019) which is the current best result without an additional language model, similar to our setup. Table 4 shows that our first results are promising, even though they are not as good as the state of the art.

6.4 Accuracy vs. Bitrate

Next, we investigate how well vq-wav2vec can compress the audio data. Specifically, we train models with different numbers of groups G and variables V to vary the size of the possible codebook size V^G and measure accuracy on TIMIT phoneme recognition without BERT training. We measure compression with the bitrate $r \cdot G \log_2 V$ at sampling rate $r = 100$Hz and report the tradeoff between bitrate and accuracy on our phoneme recognition task.

We experiment with vq-wav2vec k-means and train models with 1,2,4,8,16 and 32 groups, using 40,80,160,...,1280 variables, spanning a bitrate range from 0.53 kbit/s (G = 1, V = 40) to 33.03

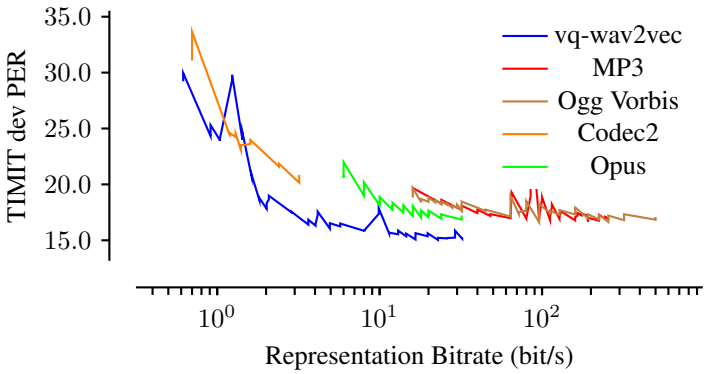


Figure 3: Comparison of PER on the TIMIT dev set for various audio codecs and vq-wav2vec k-means trained on Librispeech 100h.

kbit/s (G = 32, V = 1280). We place the quantization module after the aggregator module and train all models in the small vq-wav2vec setup (§5.2) on the 100h clean Librispeech subset.

As baselines, we consider various lossy compression algorithms applied to the TIMIT audio data and train wav2vec models on the resulting audio: Codec2[2] as a low bitrate codec, Opus (Terriberry & Vos, 2012) as a medium bitrate codec and MP3 and Ogg Vorbis (Montgomery, 2004) as high bitrate codecs. We use the whole spectrum of both variable and constant bitrate settings of the codecs; we encode and decode with `ffmpeg` (ffmpeg developers, 2016).

Figure 3 shows the trade-off between the bitrate and TIMIT accuracy. Acoustic models on vq-wav2vec achieve the best results across most bitrate settings.

6.5 Ablations

Table 5a shows that masking entire spans of tokens performs significantly better than individual tokens ($M = 1$). Furthermore, BERT training on discretized audio data is fairly robust to masking large parts of the input (Table 5b).

M	dev	test
1	14.94	17.38
5	13.62	15.78
10	12.65	15.28
20	13.04	15.56
30	13.18	15.64

(a) Mask length.

p	dev	test
0.015	12.65	15.28
0.020	12.51	14.43
0.025	12.16	13.96
0.030	11.68	14.48
0.050	11.45	13.62

(b) Mask probabilities.

Table 5: TIMIT PER for (a) different mask sizes M with $pM = 0.15$ in BERT training and (b) mask probabilities p for a fixed mask length $M = 10$.

7 Conclusion

vq-wav2vec is a self-supervised training algorithm that vector quantizes unlabeled audio data which makes it amenable to algorithms requiring discrete data. This approach improves the state of the art on the WSJ and TIMIT benchmarks by leveraging BERT pre-training. In future work, we are planning to apply other algorithms requiring discrete inputs to audio data.

[2]https://github.com/drowe67/codec2

REFERENCES

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proc. of ICML*, 2016.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv*, abs/1906.00910, 2019.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. Cloze-driven pretraining of self-attention networks. *arXiv*, abs/1903.07785, 2019.

Jan Chorowski, Ron J. Weiss, Samy Bengio, and Aäron van den Oord. Unsupervised speech representation learning using wavenet autoencoders. *arXiv*, abs/1901.08810, 2019.

Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv*, abs/1609.03193, 2016.

Ronan Collobert, Awni Hannun, and Gabriel Synnaeve. A fully differentiable beam search decoder. *arXiv*, abs/1902.06022, 2019.

FFmpeg Developers. ffmpeg tool software, 2016. URL `http://ffmpeg.org/`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, abs/1810.04805, 2018.

Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. In *Proc. of NAACL*, 2019.

John S. Garofolo, David Graff, Doug Paul, and David S. Pallett. CSR-I (WSJ0) Complete LDC93S6A. Web Download. *Linguistic Data Consortium*, 1993a.

John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathon G. Fiscus, David S. Pallett, and Nancy L. Dahlgren. The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM. *Linguistic Data Consortium*, 1993b.

Pegah Ghahremani, Vimal Manohar, Hossein Hadian, Daniel Povey, and Sanjeev Khudanpur. Investigation of transfer learning for asr using lf-mmi trained neural networks. In *Proc. of ASRU*, 2017.

Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954.

Hossein Hadian, Hossein Sameti1, Daniel Povey, and Sanjeev Khudanpur. End-to-end speech recognition using lattice-free mmi. In *Proc. of Interspeech*, 2018.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified Kneser-Ney language model estimation. In *Proc. of ACL*, 2013.

Olivier J. Hénaff, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aäron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv*, abs/1905.09272, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv*, abs/1611.01144, 2016.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *arXiv*, abs/1907.10529, 2019.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv*, abs/1901.07291, 2019.

Tatiana Likhomanenko, Gabriel Synnaeve, and Ronan Collobert. Who needs words? lexicon-free speech recognition. In *Proc. of Interspeech*, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *arXiv*, abs/1608.03983, 2016.

Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. In *Advances in Neural Information Processing Systems*, pp. 3086–3094, 2014.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, 2013.

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. Transformers with convolutional context for ASR. *CoRR*, abs/1904.11660, 2019.

C Montgomery. Vorbis i specification, 2004.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proc. of WMT*, 2018.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL System Demonstrations*, 2019.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proc. of ICASSP*, pp. 5206–5210. IEEE, 2015.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of ACL*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf`, 2018.

Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, and Yoshua Bengio. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102, 2018.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *CoRR*, abs/1904.05862, 2019. URL `http://arxiv.org/abs/1904.05862`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proc. of ACL*, 2016.

Tim Terriberry and Koen Vos. Definition of the opus audio codec. 2012.

Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv*, abs/1807.03748, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NIPS*, 2017.

Yuxin Wu and Kaiming He. Group normalization. *arXiv*, abs/1803.08494, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv*, abs/1906.08237, 2019.

Neil Zeghidour, Nicolas Usunier, Iasonas Kokkinos, Thomas Schaiz, Gabriel Synnaeve, and Emmanuel Dupoux. Learning filterbanks from raw speech for phone recognition. In *Proc. of (ICASSP)*, 2018.

APPENDIX A NUMBER OF VARIABLES VS. GROUPS

We investigate the relationship between number of variables V and groups G. Table 6 shows that multiple groups are beneficial compared to a single group with a large number of variables. Table 7 shows that with a single group and many variables, only a small number of codewords survive.

V	1 group	2 groups	4 groups	8 groups	16 groups	32 groups
40	33.44 ± 0.24	23.52 ± 0.53	18.76 ± 0.20	17.43 ± 0.14	15.97 ± 0.21	15.44 ± 0.32
80	29.14 ± 0.70	25.36 ± 4.62	17.32 ± 0.28	16.36 ± 0.27	17.55 ± 0.27	15.49 ± 0.14
160		24.27 ± 0.35	17.55 ± 0.03	16.36 ± 0.13	15.64 ± 0.03	15.11 ± 0.10
320	27.22 ± 0.25	20.86 ± 0.09	16.49 ± 0.07	15.88 ± 0.10	15.74 ± 0.18	15.18 ± 0.02
640	26.53 ± 2.02	18.64 ± 0.12	16.60 ± 0.22	15.62 ± 0.16	15.45 ± 0.13	15.54 ± 0.31
1280	32.63 ± 5.73	18.04 ± 0.26	16.37 ± 0.07	15.85 ± 0.05	15.13 ± 0.29	15.18 ± 0.05

Table 6: PER on TIMIT dev set for vq-wav2vec models trained on Libri100. Results are based on three random seeds.

V	1 group	2 groups	4 groups	8 groups	16 groups	32 groups
40	100 % (40)	95.3 % (1.6k)	27.4 % (2.56m)	74.8 % (39.9m)	99.6 % (39.9m)	99.9 % (39.9m)
80	92.5 % (80)	78.5 % (6.4k)	11.8 % (39.9m)	91.5 % (39.9m)	99.3 % (39.9m)	100 % (39.9m)
160	95 % (160)	57.2 % (25.6k)	35.2 % (39.9m)	97.6 % (39.9m)	99.8 % (39.9m)	100 % (39.9m)
320	33.8 % (320)	24.6 % (102.4k)	57.3 % (39.9m)	98.7 % (39.9m)	99.9 % (39.9m)	100 % (39.9m)
640	24.6 % (640)	10 % (409.6k)	60.2 % (39.9m)	99.3 % (39.9m)	99.9 % (39.9m)	100 % (39.9m)
1280	7.2 % (1.28k)	4.9 % (1.63m)	67.9 % (39.9m)	99.5 % (39.9m)	99.9 % (39.9m)	100 % (39.9m)

Table 7: Fraction of used codewords vs. number of theoretically possible codewords V^G in brackets; 39.9m is the number of tokens in Librispeech 100h .