INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Exploration of Efficient End-to-End ASR using Discretized Input from Self-Supervised Learning

Xuankai Chang[1], *Brian Yan*[1], *Yuya Fujita*[2], *Takashi Maekaku*[2], *Shinji Watanabe*[1]

[1]Carnegie Mellon University, PA, USA
[2]Yahoo Japan Corporation, Japan

xuankaic,byan,swatanab@andrew.cmu.edu, yuyfujit,tmaekaku@yahoo-corp.jp

Abstract

Self-supervised learning (SSL) of speech has shown impressive results in speech-related tasks, particularly in automatic speech recognition (ASR). While most methods employ the output of intermediate layers of the SSL model as real-valued features for downstream tasks, there is potential in exploring alternative approaches that use discretized token sequences. This approach offers benefits such as lower storage requirements and the ability to apply techniques from natural language processing. In this paper, we propose a new protocol that utilizes discretized token sequences in ASR tasks, which includes de-duplication and subword modeling to enhance the input sequence. It reduces computational cost by decreasing the length of the sequence. Our experiments on the LibriSpeech dataset demonstrate that our proposed protocol performs competitively with conventional ASR systems using continuous input features, while reducing computational and storage costs.

Index Terms: self-supervised learning, discrete tokens, discretized input, speech recognition.

1. Introduction

Over the past decade, remarkable advancements have been made in automatic speech recognition (ASR), largely due to the rapid development of deep neural networks [1–8]. These networks have significantly expanded the capabilities of speech recognition models. Additionally, the increasing availability of computing resources has enabled the training of ASR models using vast amounts of transcribed data, resulting in further improved performance [9, 10]. However, since deep neural networks require substantial amounts of data, some researchers have sought to increase their capacity by incorporating more transcribed data [11]. Nevertheless, this approach has limitations, as a significant portion of available data remains untranscribed. To address this, researchers have proposed leveraging untranscribed data through unsupervised and semi-supervised learning techniques [12–14]. Among these methods, self-supervised learning (SSL) [15–19] has achieved impressive results in speech related downstream tasks [20]. There are several methods to make models more suitable for various downstream tasks, including fine-tuning pre-trained models [15, 16], extracting robust speech features [21], and inserting adapters [22].

Advances in speech processing technology have led to the development of various applications that improve the convenience of human life, such as voice enabled robots and smart speaker. These advancements have greatly enhanced the ability of machines to interact with humans. However, the collection of speech data through such systems raises concerns about privacy [23]. Users may worry about the potential for their personal information to be leaked during data transmission or due to security issues in the storage system of such systems. One solution to these concerns is to transform speech signals into a different form of encoding that does not contain speaker-specific information while retaining the essential linguistic information.

In a study by Van et al. [24], it was observed that training a vector quantization-variational autoencoder (VQ-VAE) model with general speech representation learning can extract speaker-independent features from speech. Using VQ-VAE, the speech is discretized and encoded in discrete tokens, where each token represents the speech information in a short time interval. Later, several other SSL-based methods have been developed for learning general speech representations [15–17, 25]. These SSL models can also be used to discretize speech either through the vector quantization module in the model [15, 25], or by applying k-means clustering on hidden embeddings from these models [16, 17]. Note that it was shown that the higher layers of SSL models retain less speaker information than the lower layers, and the discretization step can further cleanse speaker-specific information. Using discretized tokens from an SSL model as speech representations has several other advantages:

1. Small storage and transmission size.
2. Preservation of original speech duration information.
3. Intermediate representation with both acoustic and linguistic information but with less speaker specific information.

Previous studies have investigated the use of discrete-token input for ASR models [25, 26], where 13.5 thousand unique discrete tokens are used. However, this approach did not outperform conventional log-mel-filterbank features in terms of ASR performance, unless a small BERT [27] model trained with discretized token sequences is additionally used in front of ASR model. As a result, the wide application of discrete token input in speech-processing tasks may be limited.

Given the advantages of using discretized token and the result of previous study, we are motivated to investigate using discrete tokens extracted from state-of-the-art (SOTA) SSL models to replace conventional speech processing inputs such as raw wave or acoustic features, such as mel filter bank [28]. In our study, we use WavLM [17] to extract speech representations and a k-means model to obtain discrete tokens. Specifically, maximum $2,000$ unique tokens are used to represent the speech features, less than that in previous study [26]. Once the speech data is discretized, it can be used in training and inference. The discrete tokens offer a significant reduction in data size. For example, in certain conditions, 1,000 hours of speech data can be compressed from around 100 GB to less than 1 GB, which can be conveniently loaded to RAM at once. To prove the feasibility of this approach, we conduct experiments in end-to-end (E2E) speech recognition (ASR) task, using sequence-

10.21437/Interspeech.2023-2051

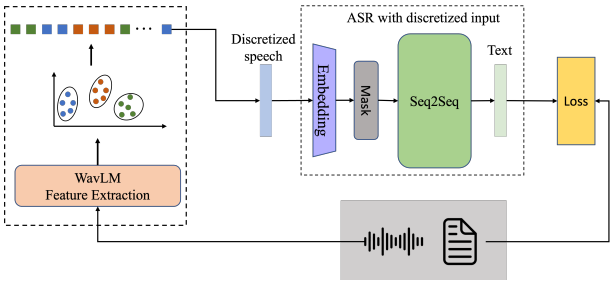

Figure 1: *Illustration of proposed ASR with discretized input. The speech discretization process is shown on the left. On the right side is Seq2Seq model from discrete tokens to text (upper).*

to-sequence (Seq2Seq) models. So far, we only reduce the size of data in terms of storage. However, it doesn't affect the computation efficiency, which is dominated by the model size and sequence length. To address the latter factor, we propose two methods. First, we can combine consecutively repeated tokens into a single one [29]. Second, we apply subword modeling on discrete tokens [30] to divide the sequence into subsequences, each of which is represented as a single meta-token. We evaluated our method on LibriSpeehch 100 hr subset for fast evaluation and LibriSpeech 960 hr. All pre-trained models and source code required for conducting experiments will be made publicly available with a license that allows free usage for research purposes.

2. Speech Recognition Using Discretized Representation

In this section, we present the details of the speech recognition model using discrete tokens in details, shown in Fig. 1, including data processing, speech recognition models and data augmentation methods.

2.1. Discretized Speech and Length Reduction

This section provides a detailed description of the speech discretization process. Instead of using the original continuous speech or acoustic features, discrete tokens are employed as input in our proposed ASR model. This approach is similar to the one used in pre-training HuBERT [16]. k-means clustering is applied to the hidden embeddings from intermediate SSL models to generate discretized speech encoding. It is worth noting that different features can be used for k-means clustering, depending on the task, including various SSL models or pre-trained supervised-learning models. Additionally, other discretization techniques, such as those proposed in [24, 31–33], can also be used.

In our experiments, we use a pre-trained WavLM model[1] [17] to extract hidden embeddings. It was trained primarily on learning the masked language models by predicting the pseudo-labels while performing speech denoising simultaneously. It achieved the best results on many downstream tasks in the SUPERB [20]. Note that the WavLM large model used comprises 24 hidden layers. The last Transformer-Encoder layer from the model is chosen to extract hidden embeddings because it has been reported that the last layer contributes the most to ASR downstream tasks [17, 34], according to a learned weight vector used to combine the hidden embeddings from all layers during training the ASR model. A k-means is trained to cluster the hidden embeddings and obtain the cluster indices as discrete tokens used as input to the ASR model. The resulting discretized input sequence has the same length as the sequence of hidden embeddings from the WavLM model, which produces speech features at a rate of 50 frames per second.

In Tab.1, we compare the data sizes of different data formats including raw waveform, conventional acoustic features, SSL-based features, and discrete tokens. Let us take a single-channel speech utterance of T seconds as an example. The size of raw waveform data depends on the sampling rate and the audio sample encoding. Here we take the common settings used in speech recognition, i.e. 16 kHz wav in 16-bit signed integer form. For acoustic features, we use a D-dimensional float (4 Bytes or 32 bits) vector with a frame shift of 10ms, which corresponds to a rate of 100 frames per second. An example value of D is 80 used in [35]. We illustrate the SSL-based features using a single-layer hidden embedding from WavLM model stored as float vectors. While for the discrete tokens, we take the WavLM features clustered with a k-means model of maximum 4096 clusters (i.e. 12-bit).

Table 1: *Data size (bit) comparison of a T-second utterance among: 1) 16 kHz raw waveform in 16-bit wav format; 2) acoustic features in D-dimensional float vector with 100 frames/sec.; 3) SSL-based features using WavLM; 4) discrete tokens in 12-bit with 50 frames/sec.*

Data format	Data size (bits)
Raw waveform	$16 \times 16000 \times T$
Acoustic features	$32 \times D \times 100 \times T$
SSL-based features	$32 \times 1024 \times 50 \times T$
Discrete tokens	$12 \times 50 \times T$

While this rate is efficient in most cases, we can further improve computational efficiency by **removing repeated tokens** and applying **subword modeling**. In [29], consecutively repeated discrete tokens are combined into a single token. Subword modeling, originally proposed in [36, 37] to address the open vocabulary problem in text, was applied to discrete speech tokens from vector-quantization [30] to reduce time resolution by identifying and grouping frequently occurring patterns as a single meta-token. To achieve this, we use Sentencepiece[2] with unigram model [37].

2.2. ASR with Discretized Input

This section provides an overview of the model utilized in our study. First, every D-dimensional continuous SSL feature vector of speech data is mapped to an N-bit discrete token, which serves as an intermediate representation between the acoustic features and the linguistic units. For example, D is 1024 and N is 12 in our experiments. The model acts as a "translator" to convert discrete tokens into text transcriptions. The input sequence comprises of discretized speech, while the output sequence is the transcription text. Typically, the input sequence is longer than the output sequence. Monotonic alignment is maintained between the input and output, allowing us to incorporate the connectionist temporal classification (CTC) [1] loss. To accomplish ASR with discretized input, we utilize

[1]We use WavLM Large model from `https://github.com/microsoft/unilm/tree/master/wavlm`

[2]`https://github.com/google/sentencepiece`

the joint CTC/attention-based encoder-decoder models that are used in acoustic feature-based end-to-end speech recognition systems [38]. Note that a randomly initialized linear embedding layer is used before the encoder to extract learnable features for input discrete tokens.

2.3. Data Augmentation for ASR with Discretized Input

In this section, we present the data augmentation method employed in our experiments.

Time-masking Masking is a fundamental technique that has been widely adopted in various machine learning tasks, such as speech recognition [39], computer vision [40], and natural language processing [27]. This technique involves concealing a portion of the input data during the training phase, which forces the model to learn to make predictions based on incomplete information. Time masking is a specific type of masking where a continuous segment in the input sequence is masked, and the model is expected to predict the correct output based on the remaining information. By exposing the model to partially masked inputs, time-masking helps to improve the model's generalization ability and makes it more robust to different types of noise and signal perturbations. In our work, we apply multiple time masks on the embedding sequence of discretized tokens.

3. Experimental Results

3.1. Setup

In this study, we examined the effectiveness of the proposed protocol on the LibriSpeech corpus [9], which is a widely used benchmark dataset for ASR. To expedite the process of tuning hyper-parameters, we employed the subset comprised of 100 hour of clean speech as the primary dataset for conducting our experiments. All evaluations are performed on dev-{clean,other} and test-{clean,other} sets. When doing experiments with all 960 hours LibriSpeech data, speed perturbation with factors 0.9 and 1.1 are used to increase training data by two folds. However, no speed perturbation is applied for experiments with train_clean_100. All k-means models are trained using around 100 hours of speech from training sets. To this end, a 100-hour subset is randomly selected when using all LibriSpeech as the training set.

Our implementation is based on ESPnet [41], an open-source toolkit for end-to-end speech processing. Our ASR models use the joint CTC/attention-based encoder-decoder architecture based on the E-Branchformer [42]. The encoder consists of 12 blocks, each with 4 self-attention heads, a convolutional gating multi-layer perceptron (cgMLP), and a feed-forward network (FFN) with an intermediate hidden dimension of 1024. The cgMLP convolution kernel size is 31. For the decoder, we used a 6-layer Transformer with a FFN dimension of 2048. We set the dropout rate to 0.1 and used 5,000 BPE subword units for output tokens. We set the CTC weight to 0.3 and did not employ language models in our experiments. For ASR decoding, we set the CTC weight to 0.3 and the beam size to 20.

3.2. LibriSpeech100 Baselines

The performance of the baseline systems is presented in Table 2. We employ three baseline systems to evaluate the performance of our proposed method. The first system uses log-MelFilterbank (FBank) features with a frame shift of 10ms, while the other two systems adopt self-supervised learning features previously used in [21]. The difference between the latter two is whether the acoustic model uses hidden representations from the last layer of the WavLM large model or a weighted-sum of all layers as input features. A convolutional subsampling layer is applied between the features and acoustic models to reduce the time resolution as the default setting, resulting in a frame shift of 40ms. SpecAugment [39] is applied to both FBank and WavLM speech features. The results show that using features from the WavLM-Large model can achieve significantly better performance, especially on the dev-other and test-other sets compared to FBank. Note that using only the embedding of the last-layer is slightly inferior to using a weighted-sum of all layers.

Table 2: *LibriSpeech-100 baseline WERs (%) of continuous feature-based ASR model using log-Mel-Filterbank (FBank), WavLM-Large with weighted-sum of all layers' and WavLM-Large only last layer's acoustic features.*

Feature	dev		test	
	clean	other	clean	other
FBank	8.1	21.6	8.3	22.2
WavLM-Large weighted-sum	3.5	**6.0**	3.5	**6.2**
WavLM-Large last-layer	**3.4**	6.4	**3.4**	6.5

3.3. Number of Discrete Tokens

Table 3: *LibriSpeech-100 WERs (%) of discretized input-based ASR with various number of discrete tokens: {100, 500, 1000, 2000}. Phoneme purity (phn_pur), discrete token purity (dsc_pur) and phone-normalized mutual information (PNMI) are listed for each type of discrete tokens.*

# of tokens	k-means quality			dev		test	
	phn_pur	dsc_pur	PNMI	clean	other	clean	other
100	0.5750	0.3183	0.5591	8.1	18.2	8.4	19.0
500	0.6732	0.1096	0.6740	6.8	14.5	6.8	14.9
1000	0.7075	0.0721	0.7075	5.8	13.0	6.2	13.2
2000	0.7357	0.0391	**0.7394**	**5.6**	**12.5**	**5.9**	**12.8**

We report the ASR performance of models trained on train_clean_100 dataset using discretized inputs with different discrete tokens numbers, as shown in Table 3. Our experiments involve setting varying numbers of clusters for the k-means model to obtain discrete tokens. These tokens are fed into the seq2seq model as discretized inputs, without reducing the length. To evaluate the quality of k-means labels, we follow the approach of [16] to compute the phoneme purity, discrete token purity, and phone-normalized mutual information (PNMI). These metrics are computed by comparing the discrete tokens with the phoneme alignment[3] [43] using the dev sets. In [16], PNMI was used to assess the quality of discrete tokens. Our ASR results show that more discrete tokens lead to better PNMI, as well as better ASR performance. Using 2000 clusters yields the best word error rates (WERs) of 5.9% and 12.8% on test-clean and test-other, respectively. These two numbers are about 29% and 42% better than those of the FBank-based ASR system, however they are worse than the WavLM feature-based systems. Based on the results, we use 2000 as the number of tokens in the subsequent experiments.

[3]It is based on a Montreal Forced Aligner, available at `https://zenodo.org/record/2619474#.ZAblHuyZNqs`

3.4. Length reduction method

Table 4: *LibriSpeech-100 WERs (%) of discretized input based ASR with different length reduction methods on 2000-token representation, including de-duplication (dedup), apply subword modeling (SW) and combined together. In [26], a VQ-Wav2Vec is used to discretize speech (13.5K tokens), together with fine-tuning a pre-trained discrete token-BERT and using a 4-gram LM in decoding.*

Length Reduction	Avg. Input Length (train / dev)	dev		test	
		clean	other	clean	other
vq-wav2vec [26] + token-BERT + 4-gram LM	- / -	4.0	10.9	4.5	12.1
WavLM-token	633.8 / 358.1	5.6	12.5	5.9	12.8
+SW_6000	349.9 / 200.1	5.5	11.8	5.8	12.0
WavLM-token dedup	468.7 / 270.9	6.4	12.7	6.4	13.2
+SW_6000	**249.4 / 144.5**	5.8	12.3	6.0	12.5

We evaluated the ASR performance of models trained on the LibriSpeech-100 dataset using different methods to reduce the length of the input sequence. First, we used subword modeling alone and set the total number of subwords to 6,000 based on the original WavLM discrete tokens. This resulted in an average input sequence length reduction of approximately 44%, and improved the ASR performance by 6% on test-other. Next, we applied de-duplication by combining consecutively repeated tokens from the original WavLM token sequence, which reduced the average input sequence length by about 24%. The ASR performance degrades by 3% on test-other set. Finally, we applied subword modeling on the discrete token sequence after de-duplication, resulting in a total sequence length reduction of 60%. Using this shorter sequence, we achieved a final performance of 12.5% on test-other. Among the above 4 different types of sequences, subword modeling on original discrete tokens achieves the best performance. Comparing our result against the results in the previous study [26], it is worse. Note that in [26], a VQ-Wav2Vec [25] with 13.5 thousand discrete tokens is used, together with a token-BERT fine-tuned on the 100-hour training set. A 4-gram language model (LM) was used in decoding. Given this, our method is much simpler while achieving close performance. Since applying both subword modeling and de-duplication achieves the best computation efficiency due to the shortest sequence length, we use them together in our subsequent experiments on the large-scale data for fast training speed.

3.5. LibriSpeech960 Results

Table 5: *WERs(%) of continuous features and discretized input w/. and w/o. 1-D Convolutional downsampling at a rate of 2 (1D-Conv) on LibriSpeech960.*

Feature	dev		test	
	clean	other	clean	other
ASR-FBank	2.5	6.3	2.6	6.2
ASR-WavLM last-layer	1.9	3.9	2.0	4.0
Discrete tokens (dedup)	2.9	6.8	3.0	7.0
Discrete tokens (dedup+1D_Conv)	2.9	6.8	3.1	6.9

In Table. 5, we present the performance on LibriSpeech 960 hours data with speed perturbation. As baselines, we listed the conventional FBank-based ASR system and the WavLM-Large feature-based ASR models. We can see that using discrete token input with 2000 unique tokens achieves slightly worse performance than the FBank. However, the gap is relatively small. More hyper-parameter searching efforts are required on LibriSpeech 960 hours, including the length reduction method, time masking ratio, number of k-means clusters and the vocabulary size of subword modeling, etc. We found that training ASR models using the discrete units on large-scale data can be quite efficient. We further reduce the input sequence length by employing a 1-D convolution layer with a downsampling rate of 2. It is observed that applying a downsampling layer didn't hurt the performance while reducing the length by 50%. About computation efficiency, it took us 23 minutes/epoch to train a discrete token-based ASR model with 1-D convolution layer on 4 Nvidia V100 GPUs, which is about half of that using FBank features. We attributed this to three reasons. First, input sequence length was reduced significantly, resulting in small computation and memory footprint. Second, large batch sizes can be achieved given that the input sequences are short. Last, the I/O overhead can be reduced because all the training data can be loaded to RAM at the beginning. In theory, the total size of 960 hours of training data can be as small as 0.3 GB ($\approx 960 * 3600 * 50 * 12$ bits) [4].

4. Conclusions

In this paper, we proposed a new protocol for E2E-ASR that uses discrete tokens as input to replace conventional raw waveform or acoustic feature input. These tokens are computed as the k-means cluster indices of hidden embeddings derived from state-of-the-art semi-supervised learning (SSL) models, specifically WavLM. Using discrete speech data can considerably reduce the size of data required for transmission and storage. By implementing de-duplication and subword modeling, the sequence length can be reduced, resulting in better computation efficiency. We experimentally compared different combinations of length-reduction method and provide the results. In addition, the discrete tokens computed from hidden embeddings of SSL models trained in general representation learning may preserve less speaker information, thus providing the benefit of preserving speaker privacy. Compared with previous studies on ASR with discrete tokens, our proposed methods is more straightforward. The experiments on the LibriSpeech dataset show that the proposed methods can achieve competitive results compared to conventional acoustic features. Future research could involve investigating other discretization techniques and ensembling discrete tokens to further enhance speech recognition performance.

5. Acknowledgements

Some experiments of this work used the Bridges2 system at PSC and Delta system at NCSA through allocation CIS210014 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the RTX 8000 GPUs used for this research.

[4] To clarify, we didn't implement the data storing part in bit in our implementation. Instead, we used the normal int32 for convenience and compatibility with neural network toolkits. However, in that case, the data is still less than 1 GB for 960 hours of speech.

6. References

[1] A. Graves *et al.*, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[2] O. Abdel-Hamid *et al.*, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Proc. ICASSP*, 2012, pp. 4277–4280.

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

[4] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[5] W. Chan *et al.*, "Listen, attend and spell," in *Proc. ICASSP*, 2016, pp. 4960–4964.

[6] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.

[7] L. Dong, S. Xu, and B. Xu, "Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018, pp. 5884–5888.

[8] A. Gulati *et al.*, "Conformer: Convolution-augmented Transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[9] V. Panayotov *et al.*, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[10] G. Chen *et al.*, "GigaSpeech: An evolving, multi-domain ASR corpus with 10,000 hours of transcribed audio," in *Proc. Interspeech*, 2021.

[11] W. Chan *et al.*, "SpeechStew: Simply mix all available speech recognition data to train one large neural network," in *Proc. Interspeech*, 2021.

[12] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. ICML*, 2013, p. 896.

[13] G. Synnaeve *et al.*, "End-to-end ASR: From supervised to semi-supervised learning with modern architectures," in *Proc. ICML*, 2020.

[14] J. Kahn, A. Lee, and A. Hannun, "Self-training for end-to-end speech recognition," in *Proc. ICASSP*, 2020, pp. 7084–7088.

[15] A. Baevski *et al.*, "Wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[16] W.-N. Hsu *et al.*, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[17] S. Chen *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022.

[18] A. Baevski *et al.*, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *Proc. ICML*, 2022, pp. 1298–1312.

[19] A. Mohamed *et al.*, "Self-supervised speech representation learning: A review," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[20] S.-w. Yang *et al.*, "Superb: Speech processing universal performance benchmark," *arXiv preprint arXiv:2105.01051*, 2021.

[21] X. Chang *et al.*, "An exploration of self-supervised pretrained representations for end-to-end speech recognition," in *Proc. ASRU*, 2021, pp. 228–235.

[22] B. Thomas, S. Kessler, and S. Karout, "Efficient adapter transfer of self-supervised speech models for automatic speech recognition," in *Proc. ICASSP*, 2022, pp. 7102–7106.

[23] A. Nautsch *et al.*, "Preserving privacy in speaker and speech characterisation," *Computer Speech & Language*, vol. 58, pp. 441–480, 2019.

[24] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

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

[26] A. Baevski and A. Mohamed, "Effectiveness of self-supervised pre-training for asr," in *Proc. ICASSP*, 2020, pp. 7694–7698.

[27] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[28] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[29] A. Lee *et al.*, "Direct speech-to-speech translation with discrete units," in *Proc. ACL*, 2022, pp. 3327–3339.

[30] T. Hayashi and S. Watanabe, "Discretalk: Text-to-speech as a machine translation problem," *arXiv preprint arXiv:2005.05525*, 2020.

[31] N. Zeghidour *et al.*, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.

[32] C. Wang *et al.*, "Neural codec language models are zero-shot text to speech synthesizers," *arXiv preprint arXiv:2301.02111*, 2023.

[33] A. Défossez *et al.*, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[34] Y. Masuyama *et al.*, "End-to-end integration of speech recognition, dereverberation, beamforming, and self-supervised learning representation," *arXiv preprint arXiv:2210.10742*, 2022.

[35] P. Guo *et al.*, "Recent developments on espnet toolkit boosted by conformer," in *Proc. ICASSP*, 2021, pp. 5874–5878.

[36] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. ACL*, 2016, pp. 1715–1725.

[37] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proc. ACL*, 2018, pp. 66–75.

[38] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017, pp. 4835–4839.

[39] D. S. Park *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," *Proc. Interspeech 2019*, pp. 2613–2617, 2019.

[40] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[41] S. Watanabe *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

[42] K. Kim *et al.*, "E-branchformer: Branchformer with enhanced merging for speech recognition," *arXiv preprint arXiv:2210.00077*, 2022.

[43] L. Lugosch *et al.*, "Speech model pre-training for end-to-end spoken language understanding," *Proc. Interspeech 2019*, pp. 814–818, 2019.