离散token用于多语言的识别的方法

1.论文

- Towards Universal Speech Discrete Tokens: A Case Study for ASR and TTS

> (Shanghai Jiao Tong University & Xiaomi; Yifan Yang, Feiyu Shen, Chenpeng Du, Ziyang Ma, Kai Yu, Daniel Povey, Xie Chen; ICASSP 2024)

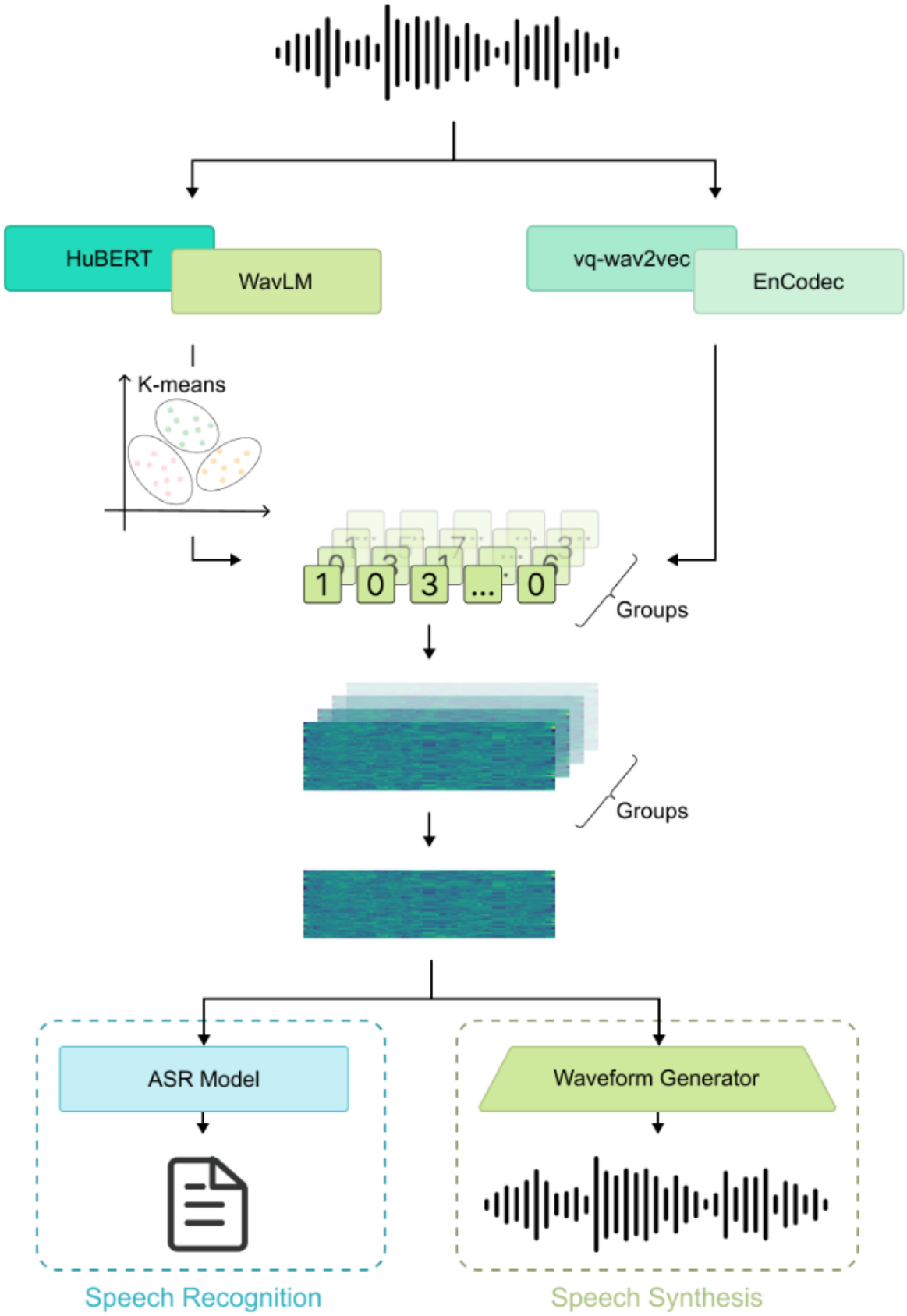


Fig. 1: Illustration of the pipeline for speech discrete tokens.

1. 任务： ASR & TTS
2. audio tokenizers：

    - WavLM-Large
    - HuBERT-Large
    - EnCodec
    - vq-wav2vec
- DASB - Discrete Audio and Speech Benchmark （speechbrain Benchmark）

> (康考迪亚大学,加拿大; Pooneh Mousavi;arXiv:2406.14294)

> Github: DASB - Discrete Audio and Speech Benchmark
>
> This repository provides a benchmark for evaluating discrete audio representations using popular audio tokenizers like **EnCodec**, **DAC**, and many more, integrated with SpeechBrain.
>
> The package helps integrate and evaluate new audio tokenizers in speech tasks of great interest such as *speech recognition*, *speaker identification*, *emotion recognition*, *keyword spotting*, *intent classification*, *speech enhancement*, *separation*, and *text-to-speech*. It offers an interface for easy model integration and testing and a protocol for comparing different audio tokenizers.
>
> This repository can be used to benchmark new audio tokenizers reliably. It includes a benchmark on 9 audio and speech datasets using 6 popular discrete audio encoders: **semantic** (*Discrete HuBERT*, *Discrete WavLM*, *Discrete Wav2Vec2*), **compression** (*EnCodec*, *DAC*), and **hybrid** (*SpeechTokenizer*). We consider different downstream architectures for each task and report the best-performing architecture.

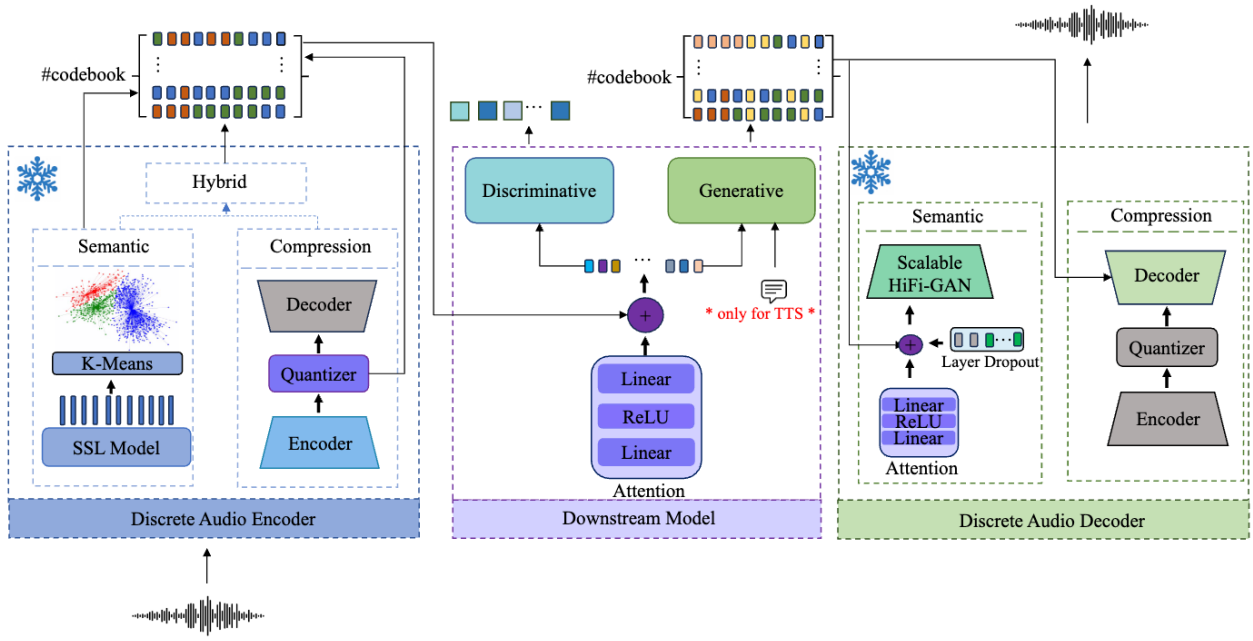


Figure 1: The workflow of DASB consists of three steps. First, a discrete audio encoder converts the audio signal into discrete tokens (*left*). Then, the tokens are combined using attention and fed to a neural model for the final prediction (*middle*). For generative tasks, the predicted tokens are passed to a discrete decoder, which converts them back into an audio waveform (*right*). Both the encoder and decoder are pretrained and frozen during downstream model training.

    -

Table 1: Key Features of the Discrete Audio Encoders. #Params is computed for medium bitrate.

Model	#Params	Sampling Rate	Bitrate (kbps)			#Codebooks		
			low	medium	high	low	medium	high
Discrete HuBERT	309.0M	16KHz	0.98	2.9	-	2	6	-
Discrete WavLM	309.0M	16KHz	0.98	2.9	-	2	6	-
Discrete Wav2Vec2	309.0M	16KHz	0.98	2.9	-	2	6	-
EnCodec [29]	17.9M	24KHz	1.5	6.0	24.0	2	8	32
DAC [30]	22.4M	24KHz	1.5	6.0	24.0	2	8	32
SpeechTokenizer [8]	85.3M	16KHz	1.0	4.0	-	2	8	-

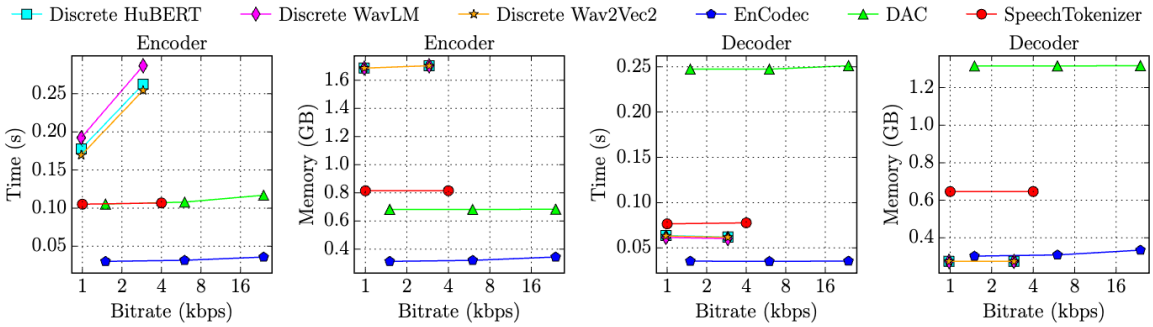


Figure 2: Time and memory required to process an utterance of 16 seconds for encoders and decoders of the considered audio tokenizers on an NVIDIA GeForce RTX 3070 GPU @ 8 GB.

Table 2: Benchmarking results for discriminative tasks.

Models/Tasks	ASR-En		ASR-multiling		ER	IC	KS	SI	SV
	WER ↓		WER ↓		ACC ↑	ACC ↑	ACC ↑	ACC ↑	EER ↓
	Clean	Other	Welsh	Basque					
Low Bitrate									
Discrete Hubert	**8.99**	**21.14**	**58.50**	**26.83**	57.20	68.70	90.54	0.90	24.99
Discrete WavLM	11.72	27.56	60.37	28.63	**59.80**	73.40	**97.94**	0.70	26.02
Discrete Wav2Vec2	12.14	28.65	66.30	32.25	57.80	**74.10**	96.16	0.40	33.53
EnCodec	52.37	77.04	92.01	58.20	44.70	31.50	86.00	**58.30**	**17.40**
DAC	63.96	83.61	94.86	66.29	49.20	22.10	81.00	45.10	20.62
SpeechTokenizer	19.77	43.12	76.67	47.92	49.10	57.90	95.09	47.40	20.41
Medium Bitrate									
Discrete Hubert	**7.91**	**18.95**	54.77	23.63	**62.10**	70.50	94.69	67.40	15.71
Discrete WavLM	8.52	20.35	**54.22**	**22.06**	57.60	**78.00**	**98.09**	80.80	8.00
Discrete Wav2Vec2	8.76	21.32	60.39	26.64	59.10	75.10	96.64	65.47	17.64
EnCodec	46.80	74.24	91.23	47.95	51.30	31.40	88.70	**91.90**	**7.81**
DAC	59.54	81.48	97.43	56.16	45.80	18.90	76.60	83.80	11.78
SpeechTokenizer	18.32	41.21	75.17	38.94	52.10	57.80	94.86	91.40	7.88
High Bitrate									
EnCodec	**45.18**	**72.56**	**93.40**	**87.65**	46.40	**19.60**	**83.60**	**92.81**	**7.18**
DAC	99.53	99.38	99.40	99.68	**46.00**	15.70	75.20	85.61	10.89
Continuous Baseline									
SSL	3.370	7.04	41.77	14.32	63.10	86.10	99.00	99.70	2.10

- dMel: Speech Tokenization made Simple

  > (Apple; He Bai,Tatiana Likhomanenko;arXiv:2407.15835)

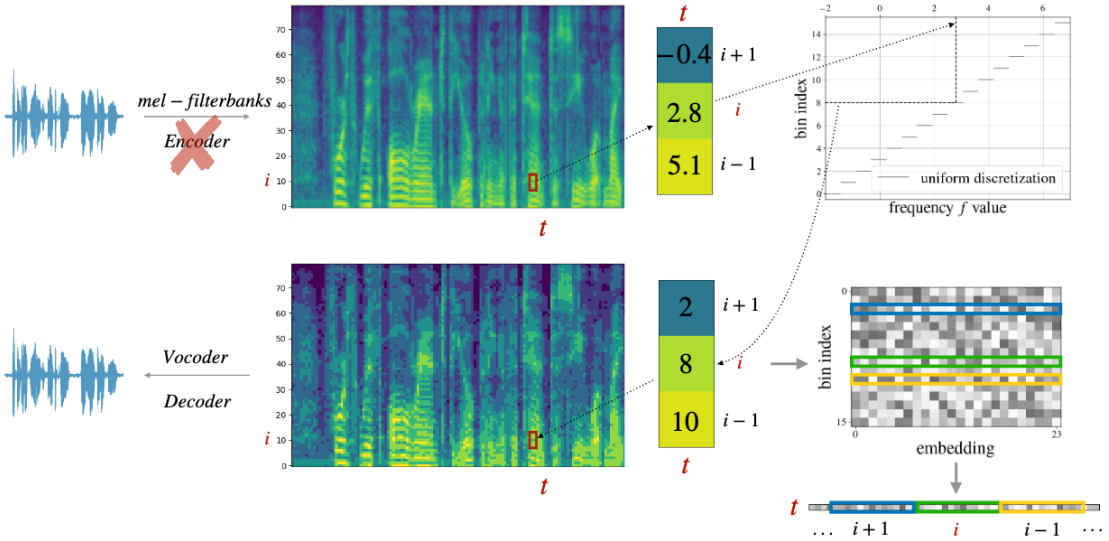


Figure 1: Prior works on speech tokenization use either heavy self-supervised pretrained encoders [2, 14] to extract semantic tokens (and train a separate decoder for it) [19] or learn compression encoder-decoder models with residual vector quantizations [37, 11] to obtain acoustic tokens. By contrast we eliminate the encoder and simply discretize mel-filerbanks (`dMel`) to encode audio, and use a simple mel-filterbank vocoder [36] to reconstruct speech signals.

We denote tensors as $\mathbf{X}$ while $\mathbf{X}_{i,\dots}$ denote the $(i, \dots)$-th component of tensor $\mathbf{X}$. First, the speech tokenizer takes the input speech signal $\mathbf{x}$ and computes the mel-filterbanks representation $\mathbf{M}$:

$$\mathbf{M} = \text{Mel}(\mathbf{x}), \tag{1}$$

where $\text{Mel}(\cdot)$ represents the function that computes the mel-filterbanks, $\mathbf{M} \in \mathbb{R}^{T \times N}$, N is the number of filterbanks and T is the number of frames in the spectrogram.

Tokenization To discretize the mel-filterbanks representation $\mathbf{M}$ into speech tokens, we adopt a codebook $\mathbf{C}$. In this paper, we apply a simple linear discretization, so that the codebook $\mathbf{C} \in \mathbb{R}^{2^K}$ and its values are evenly spaced in the range of the mel-filterbanks values:

$$m = \min_{t,i}(\mathbf{M}_{t,i}), \qquad M = \max_{t,i}(\mathbf{M}_{t,i}) \qquad \delta = \frac{M - m}{2^K}, \tag{2}$$

$$\mathbf{C} = \left[m,\ m + \delta,\ m + 2\delta,\ \dots,\ m + (2^K - 1)\delta\right]. \tag{3}$$

In practice, we compute the minimum m and maximum M values of mel-filterbanks across the entire dataset to define the codebook $\mathbf{C}$. Then we map a magnitude $\mathbf{M}_{t,i}$ of every frequency channel $i = 1 \dots N$ for the time frame $t = 1 \dots T$ into a bin index of the codebook $\mathbf{C}$ in the following way:

$$\mathbf{S}_{t,i} = \text{Discretize}(\mathbf{M}_{t,i}) = \text{argmin}_j |\mathbf{M}_{t,i} - \mathbf{C}_j| \tag{4}$$

where $\mathbf{S} \in \mathbf{B}^{T \times N}$ represents the discretized mel-filterbanks (`dMel`) with $\mathbf{B} = \{j | j = 1, 2, 3, \dots 2^K\}$ and $\mathbf{S}_t \in \mathbf{B}^N$ being the t-th speech token. As the codebook $\mathbf{C}$ has 2^K distinct values and thus number of bins $|\mathbf{B}| = 2^K$, each speech token is represented by $N \cdot K$ bits where every K bits are used to represent one of N frequency channels.

- SpeechGPT: Empowering Large Language Models with Intrinsic Cross-Modal Conversational Abilities

> Fudan University; Dong Zhang, Xin Zhang, Jun Zhan, Shimin Li, Yaqian Zhou, Xipeng Qiu; EMNLP2023 Findings

- SpeechGPT-Gen: Scaling Chain-of-Information Speech Generation

> Fudan University; Dong Zhang, Xin Zhang, Jun Zhan, Shimin Li, Yaqian Zhou, Xipeng Qiu; arXiv: 2401.13527

- SpeechTokenizer: Unified Speech Tokenizer for Speech Language Models

> Fudan University; Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, Xipeng Qiu; ICLR 2024

- WavTokenizer: an Efficient Acoustic Discrete Codec Tokenizer for Audio Language Modeling

> Zhejiang University & Alibaba Group & Fundamental AI Research (FAIR), Meta; Shengpeng Ji, Ziyue Jiang, Xize Cheng, Yifu Chen, Minghui Fang, Jialong Zuo, Qian Yang, Ruiqi Li, Ziang Zhang, Xiaoda Yang, Rongjie Huang, Yidi Jiang, Qian Chen, Siqi Zheng, Wen Wang, Zhou Zhao; arXiv:2408.16532

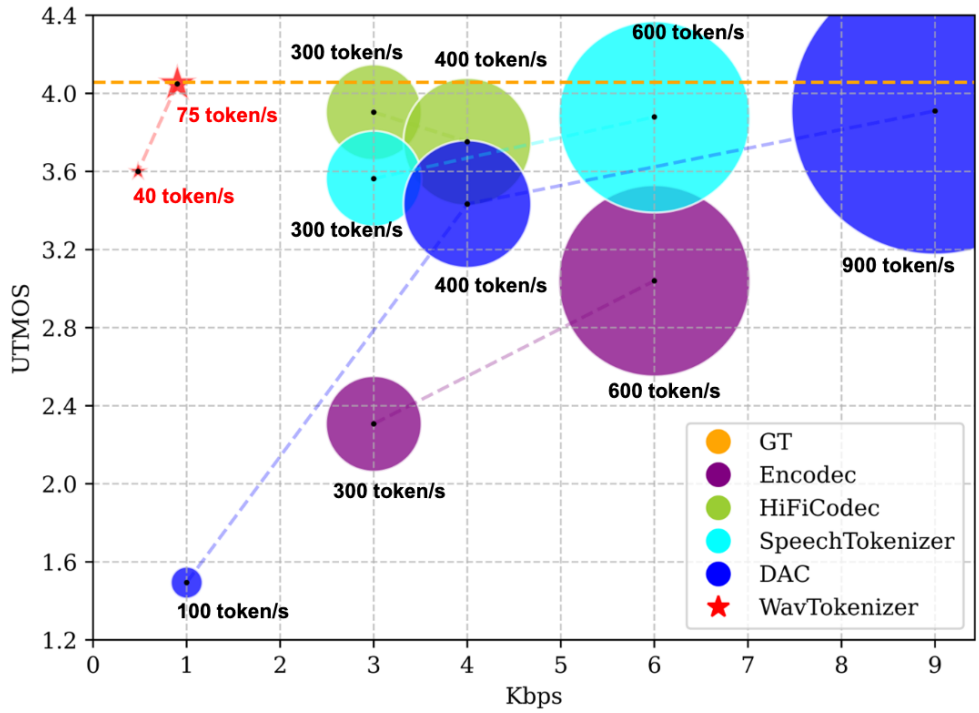


Figure 1: Comparison between WavTokenizer and state-of-the-art acoustic codec model. The vertical axis UTMOS represents reconstructed quality closer to human auditory perception, the horizontal axis kbps represents audio compression levels. The size of circles represents the number of discrete tokens per second.

> 所以相当于1个token 10bit ，然后对应的就是每个码本数量是1024，10位～，那100token/s代表的就是每一秒有100个采样点。
>
> 如果使用两个码本的话，**100 token/s** 仍然表示每秒有 100 个 token，但每个 token 可以通过组合两个码本中的条目来表示，从而增加了表达能力。
>
> **一个码本**：
>
> - 每个 token 对应一个码本，假设这个码本有 1024 个条目（10 bit 表示）。
> - 每秒 100 token，表示每秒有 100 个采样点，每个采样点用一个 10 bit 的条目表示。
>
> **两个码本**：

> - 当使用两个码本时，每个 token 实际上是通过两个码本的组合来表示。
> - 假设每个码本也有 1024 个条目（10 bit），那么两个码本的组合可以表示 1024×1024=2201024 \times 1024 = 2^{20}1024×1024=220 个不同的条目，这相当于每个 token 用 20 bit 来表示（因为 210×210=2202^{10} \times 2^{10} = 2^{20}210×210=220）。

2. Metric

> compression, reconstruction quality, and semantic modeling

- Compression: the number of quantizers and the temporal dimension of the codec. 【low bitrates】

> - **多量化器下的下游模型设计**：当量化器数量多于一个时，下游模型如 Valle、SoundStorm、MusicGen 和 UniAudio 等分别采用不同的特殊结构设计，包括 AR 和 NAR 结构、并行生成、倾斜自回归结构以及全局和局部注意力结构等。
> - **单量化器的优势**：单个量化器可使语音模态直接自回归地嵌入到大型多模态模型中，相对更简洁高效。
> - **编解码器时间维度的影响**：编解码器的时间维度，以 DAC 为例每秒需 900 个标记，会对语言模型的生成质量和资源消耗产生影响。

Feature	# Units	# Frame Rate	$\text{bitrate} = \log_2 V \cdot C \cdot R$	Bandwidth (kbps)
FBank	-	100 帧 (帧移10ms)	32 bit * 80 * 100 = 256 000	256.00
WavLM-Large	2000	50 Hz		0.55
HuBERT-Large	2000	50 Hz		0.55
EnCodec	1024^8	75 Hz embeddings	10位(bit) * 8组 * 75Hz = 6000	6.00
vq-wav2vec	320^2	100 Hz embeddings		1.66

- Pursuing Better Reconstruction Quality

> AudioDec, PromptCodec, DAC, HiFi-Codec, APCodec, Single-Codec, Language-Codec

3. Speech discrete tokens

3.1 semantic tokens

> vq-wav2vec, wav2vec 2.0, HuBERT, and WavLM (Discrete HuBERT, Discrete WavLM, Discrete Wav2Vec2)

- vq-wav2vec employs autoregressive Convolutional Neural Networks(CNNs) to extract feature representations from raw audio, which are then organized into multiple groups. Subsequently, vector quantization is applied in each group, facilitated by either the Gumbel-Softmax or online k-means clustering, yielding multiple sets of discrete token sequences. In practice, a pre-trained vq-wav2vec model is employed to process 16 kHz audio and outputs embeddings at a 100 Hz rate with 2 groups.
- For wav2vec 2.0, HuBERT, and WavLM, the discretization of speech can be realized by applying k-means clustering to the hidden embeddings from some specified Transformer-Encoder layer.

3.1.1 CPC: Representation Learning with Contrastive Predictive Coding

> 基于判别学习的语音预训练模型（2-1）---CPC from DeepMind - 知乎 (zhihu.com)

> 作者通过训练一个线性分类器来验证CPC提取的特征中包含的音素。

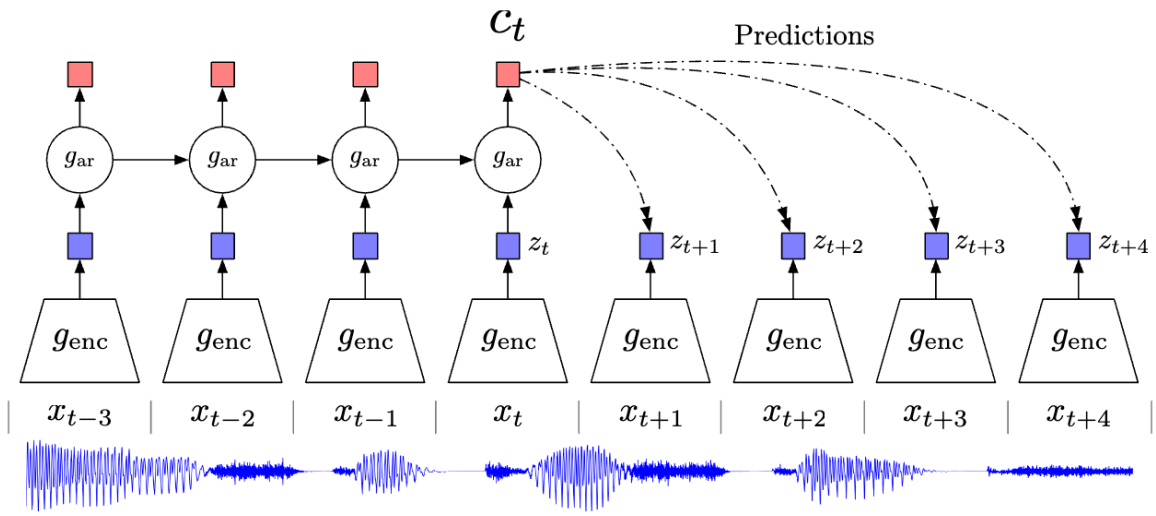


Figure 1: Overview of Contrastive Predictive Coding, the proposed representation learning approach. Although this figure shows audio as input, we use the same setup for images, text and reinforcement learning.

Method	ACC
Phone classification	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
Speaker classification	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

Table 1: LibriSpeech phone and speaker classification results. For phone classification there are 41 possible classes and for speaker classification 251. All models used the same architecture and the same audio input sizes.

3.1.2 wav2vec: Unsupervised Pre-training for Speech Recognition

> 不同于CPC文章中作者只训练一个线性层来评估预训练特征向量中包含的语义信息，wav2vec的作者们**将预训练的wav2vec模型作为特征提取器，代替人工定义的声学特征输入给ASR模型**，并且直接在多个语音识别数据集上直接与baseline ASR模型以及SOTA结果进行比较。

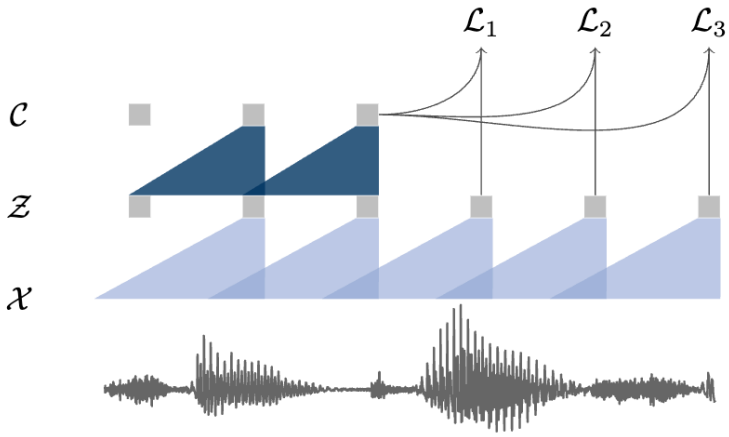
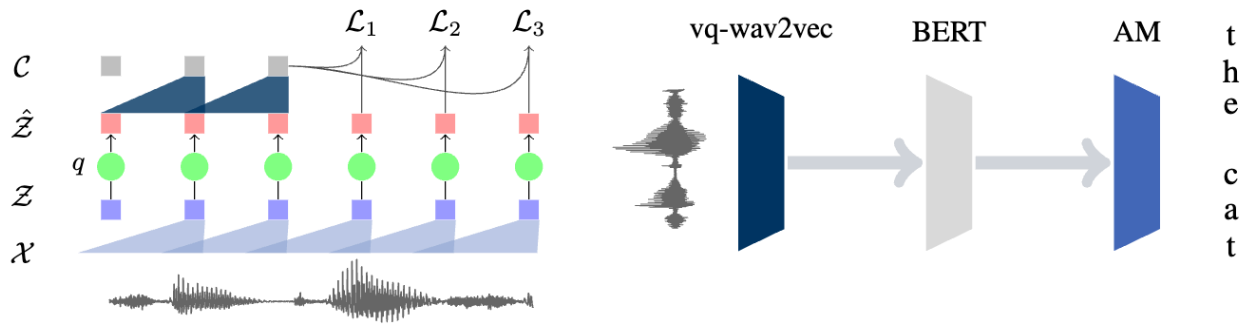


Figure 1: Illustration of pre-training from audio data $\mathcal{X}$ which is encoded with two convolutional neural networks that are stacked on top of each other. The model is optimized to solve a next time step prediction task.

3.1.3 vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations

> 随着NLP领域中的基于自监督学习的预训练模型BERT被提出并受到广泛的关注，语音领域的研究者们开始思考能不能也使用语音数据训练一个BERT，再使用BERT提供的语义特征向量来提升下游语音识别任务的性能呢？然而不同于NLP领域中的文本具有天然的离散特性（每个token对应着一个id，每个id对应着一个embedding向量），语音信号本是连续的信号，并且由wav2vec模型学习得到的特征也是连续的向量表征（从waveform经过下采样得到的每一帧直接对应着一个向量）。于是，这项工作的作者提出在wav2vec模型中加入量化模块，这样我们就可以按照以下流程（如下图b），**利用BERT预训练的方法提升语音识别任务**的效果。
>
> 1. 首先经过自监督的预训练得到的vq-wav2vec模型，从原始音频中抽取出离散化的特征表示（每一帧对应着一个id，每个id对应着一个特征向量）。
> 2. 有了离散化的特征表示，我们便可以将作为embedding，并使用大规模无标签的语音数据训练一个BERT。
> 3. 我们再将预训练BERT得到的语义表征作为输入，有监督地训练下游语音识别模型（AM），希望能够提升在各种语音识别任务上的性能。

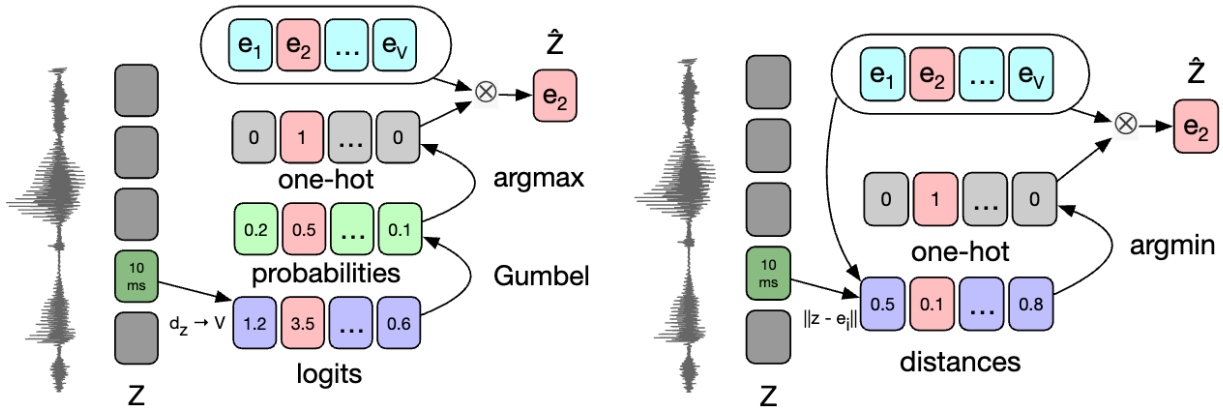


(a) vq-wav2vec (b) Discretized speech training pipeline

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.

vq-wav2vec的整体结构如上图a所示，给定输入语音信号 X，我们首先使用encoder网络（与wav2vec相同）进行编码得到隐变量 Z，再通过引入量化模块（wav2vec中没有）将隐变量 Z 映射为离散化隐变量 Z^，最后使用context网络（与wav2vec相同）编码历史时刻的离散化隐变量得到上下文特征向量 C 。

文章中介绍了**两种量化模块**可供选择：Straight Through Gumbel-Softmax与Online K-means clustering（后者类似于VQ-VAE中的量化模块），如下图所示：



(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).

给定包含 V 个 d 维向量的codebook $\mathbf{e} \in \mathbb{R}^{V \times d}$，**Straight Through Gumbel-Softmax量化方法**首先通过linear+ReLU+linear层将隐变量 $\mathbf{z}$ 映射为logits $\mathbf{l} \in \mathbb{R}^{V}$，再使用Gumbel-Softmax得到选择codebook中每个向量的概率分布 $\mathbf{p} \in \mathbb{R}^{V}$。在模型*前馈传播*时，我们使用argmax直接挑选出codebook中概率值最高的向量作为量化的隐变量 $\hat{\mathbf{z}}$。在模型*反向传播*时，我们假装在前传时没有argmax的步骤，而是直接将概率分布 $\mathbf{p}$ 乘上codebook中的向量 $\mathbf{e}$ 得到的 $\hat{\mathbf{z}}$，来计算codebook中的向量与encoder网络中的梯度值。

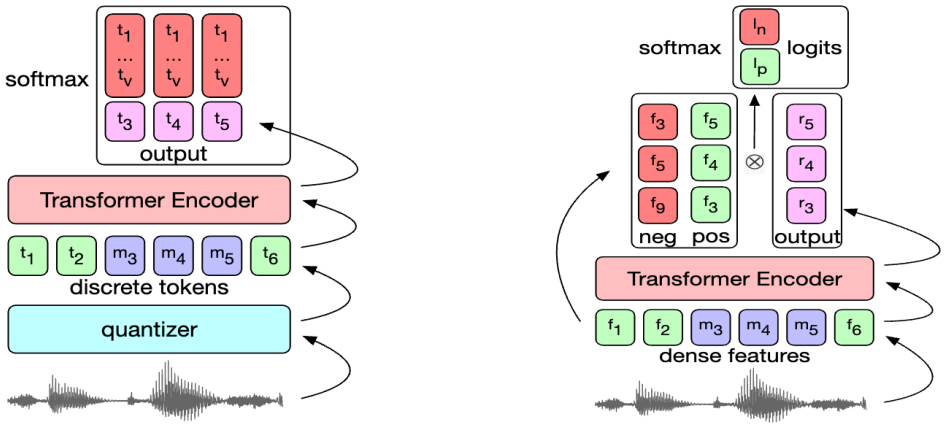
Gumbel-Softmax具体公式为：$p_j = \frac{\exp(l_j + v_j)/\tau}{\sum_{k=1}^{V} \exp(l_k + v_k)/\tau}$，其中 v 为标准Gumbel分布中的采样值，τ 为temperature超参。

Online K-means clustering量化方法首先将隐变量 $\mathbf{z}$ 与codebook中的每个向量 $\mathbf{e}_j$ 计算欧氏距离得到 $||\mathbf{z} - \mathbf{e}_j||_2^2$。在模型*前馈传播*时，我们使用argmin直接挑选出codebook中与隐变量 $\mathbf{z}$ 最接近的向量作为量化的隐变量 $\hat{\mathbf{z}} = \mathbf{e}_i, \text{where } i = \text{argmin}_j ||\mathbf{z} - \mathbf{e}_j||_2^2$。在模型*反向传播*时，我们通过在wav2vec的损失函数 $\mathcal{L}^{\text{wav2vec}}$ 上加上两个正则项得到vq-wav2vec的损失函数 $\mathcal{L}_{\text{k-means}}^{\text{vq-wav2vec}}$ 来计算网络中各模块的梯度值。

基于Online K-means clustering量化方法的vq-wav2vec模型的损失函数的具体公式为：$\mathcal{L}_{\text{k-means}}^{\text{vq-wav2vec}} = \mathcal{L}^{\text{wav2vec}} + \left(||\text{sg}(\mathbf{z}) - \hat{\mathbf{z}}||_2^2 + \lambda ||\text{sg}(\hat{\mathbf{z}}) - \mathbf{z}||_2^2 \right)$，其中 $\text{sg}(\cdot)$ 表示停止梯度运算，两个正则项分别将codebook中的向量和encoder输出的隐变量彼此拉近距离。

3.1.4 Effectiveness of self-supervised pre-training for speech recognition

> 当使用自监督学习方法逐步得到预训练的vq-wav2vec模型和BERT模型之后，不同于**vq-wav2vec将BERT作为特征提取器**，再使用BERT提供的语义特征有监督地训练下游ASR模型，**Discrete BERT直接将BERT作为ASR模型**，通过加上随机初始化的linear层并且使用CTC loss有监督地fine-tune BERT模型来进行语音识别任务。



(a) Quantized Inputs (b) Continuous Inputs

Fig. 1: Illustration of BERT pre-training. m_i refers to masked time-steps chosen at random during each forward pass. (a) Inputs are quantized with a vq-wav2vec quantizer or, for MFCC and FBANK variants, by finding the closest centroids and are then used for training a BERT model with a masked language model objective. (b) wav2vec, MFCC or FBANK inputs are masked, encoded by a transformer, and then fed into a classifier to predict which features were masked at each time-step.

Discrete BERT的训练框架如上左图所示。我们首先使用预训练的vq-wav2vec模型为输入音频得到**量化**的特征向量（也可以是MFCC或FBANK声学特征的k-means聚类中心，只要是每个特征向量都是codebook中的某个id对应的向量即可）。然后，我们便可以将量化的特征向量作为输入，按照BERT的masked prediction的自监督训练方式（正例是mask之前的输入中的id，负例是codebook中的其他id）训练出BERT模型。

为了体现将输入特征进行量化的好处，作者还提出了**Continuous BERT**来进行对比，如上右图所示。我们首先使用预训练的wav2vec模型为输入音频得到**连续**的特征向量（也可以是MFCC或FBANK声学特征向量）。然后，我们可以将连续的特征向量直接作为输入，随机mask输入序列中的一些位置，再使用CPC中提出的InfoNCE loss使用对比学习的方式在mask的位置上计算损失函数（正例是mask之前的输入特征向量，负例是从当前batch的其他被mask的位置上随机sample得到的特征向量），训练出BERT模型。

3.1.5 [wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations]

> **与wav2vec，vq-wav2vec，Discrete BERT模型的区别**
>
> 不同于wav2vec和vq-wav2vec模型需要将特征向量输入给下游ASR模型进行训练，wav2vec 2.0本身即为ASR模型，fine-tune时只需加上随机初始化的linear层将特征向量映射到预测文本，使用CTC loss训练即可。
>
> 不同于Discrete BERT需要先训练一个vq-wav2vec作为特征提取器，再预训练+微调BERT用于语音识别任务，wav2vec 2.0提出了一个端到端的架构，把vq-wav2vec中的Gumbel softmax量化模块和BERT结合到一起，只需要一步预训练+微调即可。

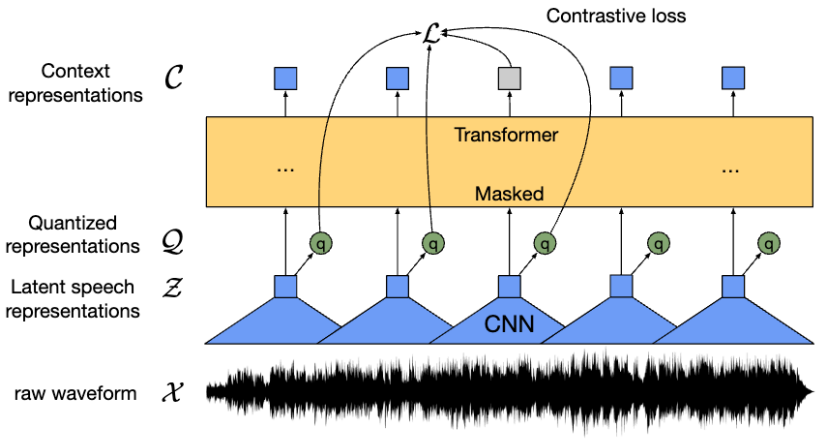


Figure 1: Illustration of our framework which jointly learns contextualized speech representations and an inventory of discretized speech units.

3.1.6 wav2vec-U: [Unsupervised Speech Recognition]

> **完全不使用任何的标注数据，通过无监督的方法也能够训练出很好的ASR模型**

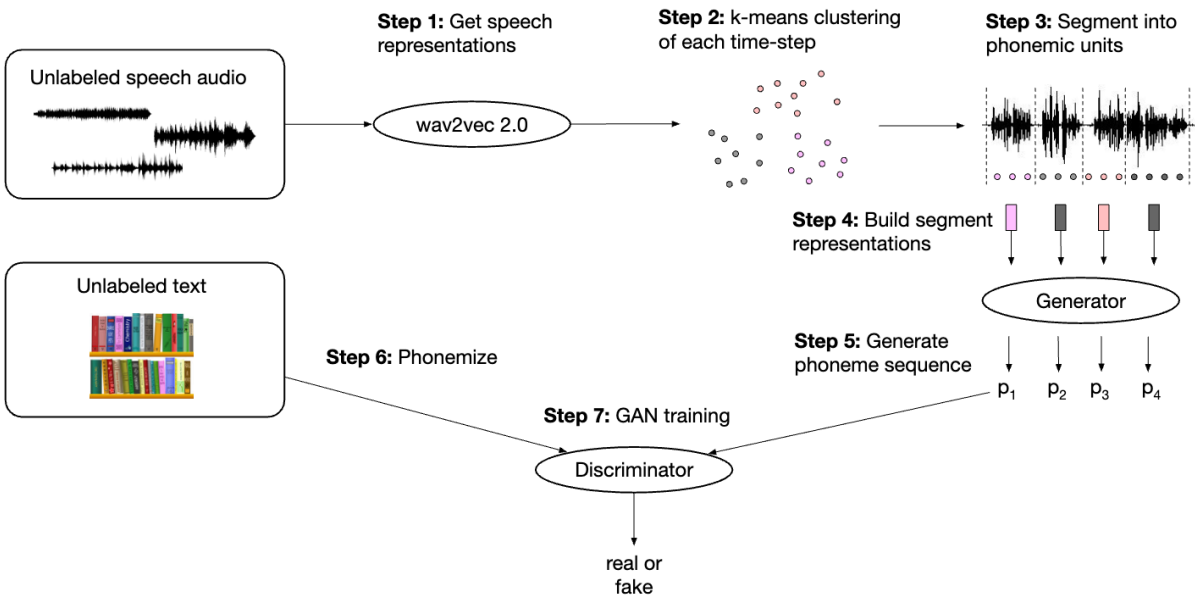


Figure 1: Illustration of wav2vec Unsupervised: we learn self-supervised representations with wav2vec 2.0 on unlabeled speech audio (Step 1), then identify clusters in the representations with k-means (Step 2) to segment the audio data (Step 3). Next, we build segment representations by mean pooling the wav2vec 2.0 representations, performing PCA and a second mean pooling step between adjacent segments (Step 4). This is input to the generator which outputs a phoneme sequence (Step 5) that is fed to the discriminator, similar to phonemized unlabeled text (Step 6) to perform adversarial training (Step 7).

由于作者发现相比于word或者letter，模型更容易学习从audio到phoneme（音素）之间的映射，wav2vec-U的无监督学习方法首先利用预训练wav2vec 2.0 large模型从**audio**中提取得到**语义表征序列**，然后通过量化再单元化分割得到更短的**单元表征序列**，接着通过使用无监督对抗训练的生成器由单元表征序列生成**音素序列**，于是我们就可以得到对于的**预测文本**。

3.1.7* wav2vec 2.0 +ST

> Self-training and Pre-training are Complementary for Speech Recognition

wav2vec-U中，我们看到**自我习方法**（Self-training，谷歌称之为Noisy Student Training）能够非常有效地提升无监督语音识别模型的性能。那么，如果我们在使用有标注数据fine-tune wav2vec 2.0预训练模型时引入自我学习方法，能不能得到更好的性能呢?

Step 1: 首先，我们使用无标注的数据预训练一个wav2vec 2.0模型，再使用有标注的数据集通过CTC loss fine-tune此wav2vec 2.0预训练模型，得到一个wav2vec 2.0 ASR模型。

Step 2: 接着，对于每一条无标注的数据，我们使用wav2vec 2.0 ASR模型加上4-gram语言模型使用beam-search方法进行解码得到50条转录文本作为候选伪标注。

Step 3: 然后，我们使用Transformer语言模型对于50条候选伪标注重新进行打分，并挑选出分数最高的转录文本作为最终的伪标注。

Step 4: 最后，我们使用带有伪标注的数据训练第二个ASR模型（可能是Transformer-based seq2seq模型，也可能是wav2vec 2.0预训练模型）。

3.1.8 Hubert

> [HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units]

与wav2vec 2.0的区别

不同于wav2vec 2.0在模型中引入了**量化模块**，在自监督预训练时采用**端到端的方式进行联合训练**，HuBERT模型**事先**通过无监督方法训练得到聚类模型，为所有无标注语音信号生成**离散化的目标序列**，再直接使用**MLM自监督预训练方法**预测掩码位置的目标值。通过使用更为简洁的自监督训练方法，HuBERT的预训练进程往往比wav2vec 2.0更加**稳定**，尤其当我们为了提升ASR性能的绝对数值而需要训练很大的预训练模型时。此外，不同于wav2vec 2.0模型在预训练时**实时**地从量化模块中的codebook通过概率分布选择**正例**，从相同句子的其他位置随机选择**负例**，用于计算**对比学习**的损失函数，HuBERT模型在离线聚类的步骤中**提前确定**了每段语音信号所对应的**聚类中心序列**（也就是离散化的目标序列），以及聚类中心的总数量，再使用交叉熵损失函数进行模型的预训练。由于无监督聚类模型提供的离散化的目标序列中不仅仅包含与ASR相关的词汇信息，还包含着说话者、情感等其他方面的信息，预训练得到的HuBERT模型能够学习到**更丰富的多层面的语音信息**，从而在多种语音处理下游任务中展现出**更强的泛化性能**。

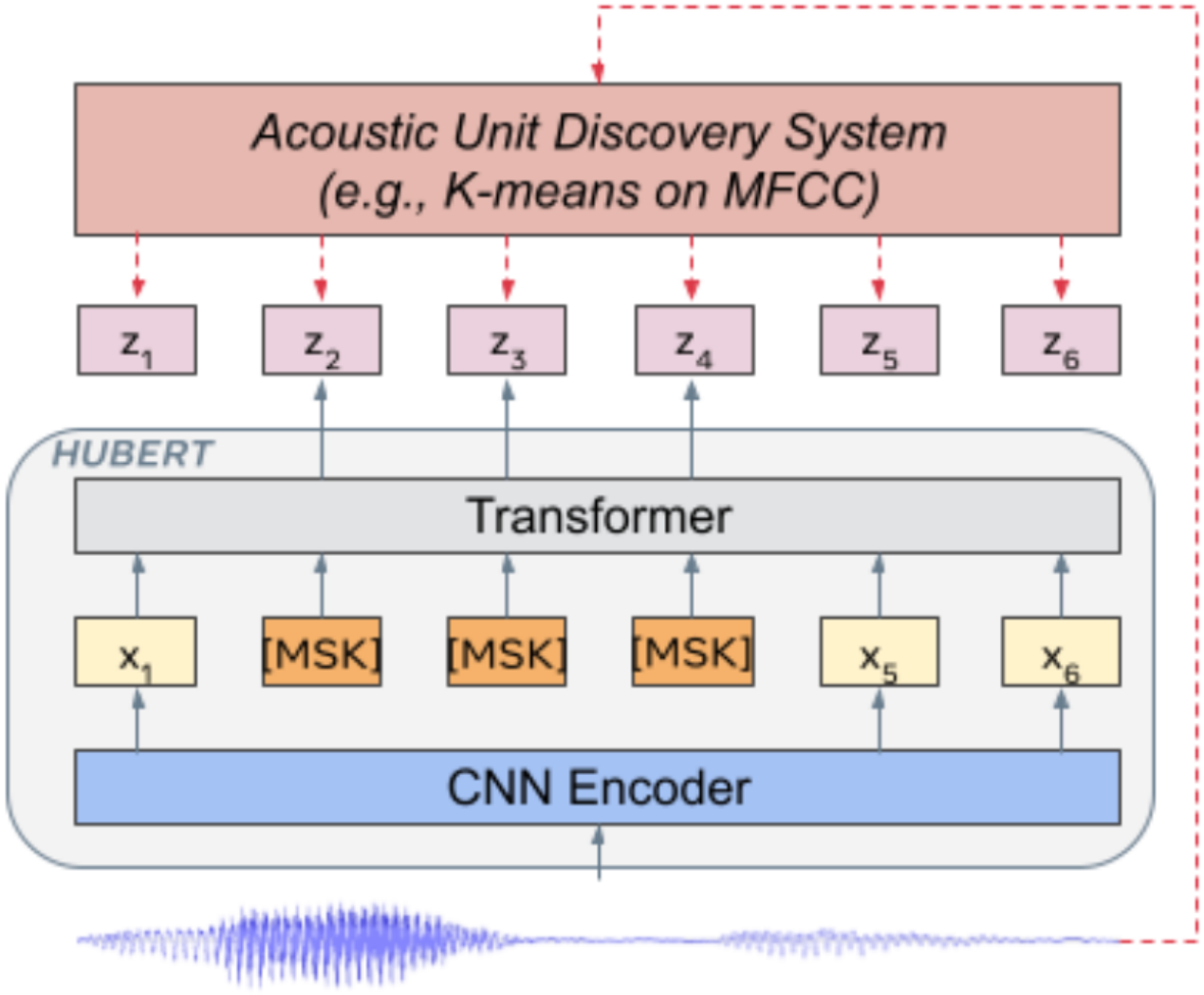


Fig. 1: The HuBERT approach predicts hidden cluster assignments of the masked frames (y_2, y_3, y_4 in the figure) generated by one or more iterations of k-means clustering.

3.1.9 wav2vec-U 2.0

> Towards End-to-end Unsupervised Speech Recognition
>
> 新一代的改进版本wav2vec-U 2.0，通过去除了wav2vec-U中的语音预处理步骤，大大**简化了整体的模型架构**，并使得模型可以**端到端地进行无监督的训练**；通过在无监督训练中引入自监督学习目标，最终实现了无监督语音识别的**进一步性能提升**。

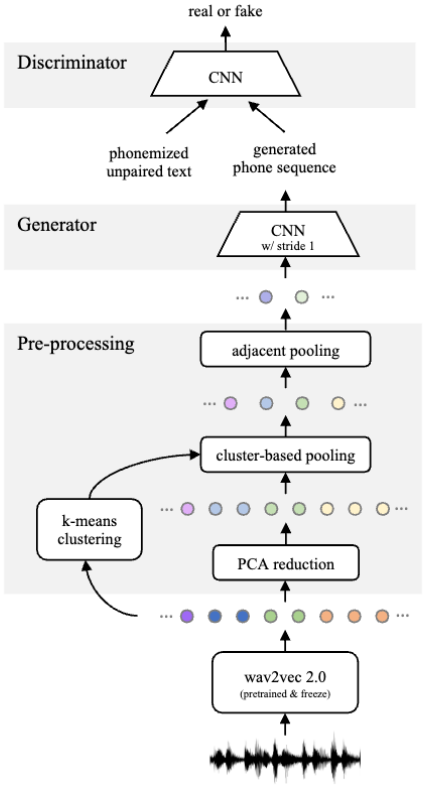


Figure 1: Wav2vec-U [6]. The input wav2vec2.0 feature is pre-processed before feeding into the generator as described in Section 2.2. The generator is optimized through adversarial training against the discriminator as described in Section 3.1.

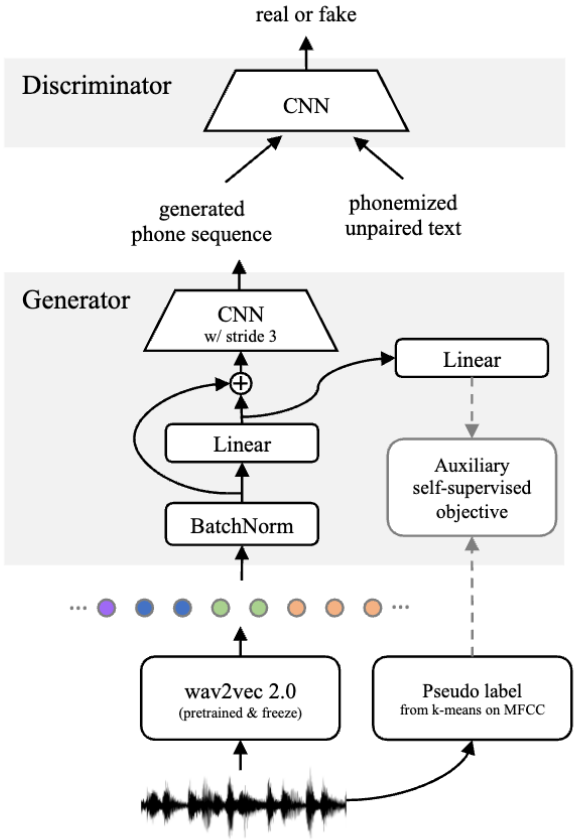


Figure 2: Proposed wav2vec-U 2.0. The generator takes raw wav2vec 2.0 feature as input without pre-processing step as described in Section 3.2. In addition to adversarial training, an auxiliary self-supervised objective is introduced with pseudo label derived from the raw waveform as described in Section 3.3.

3.2 acoustic tokens

> Soundstream, EnCodec, AudioDec, PromptCodec, DAC, HiFi-Codec, APCodec, Single-Codec, Language-Codec

Compared to semantic tokens, acoustic tokens offer the advantage of a **reconstruction paradigm** that can uniformly model speech, music, and audio. Current acoustic codec models have achieved human-level **reconstruction quality**, validated in downstream generative models.

3.2.1 Soundstream (SoundStream: An End-to-End Neural Audio Codec)

> [2107.03312] SoundStream: An End-to-End Neural Audio Codec (arxiv.org)

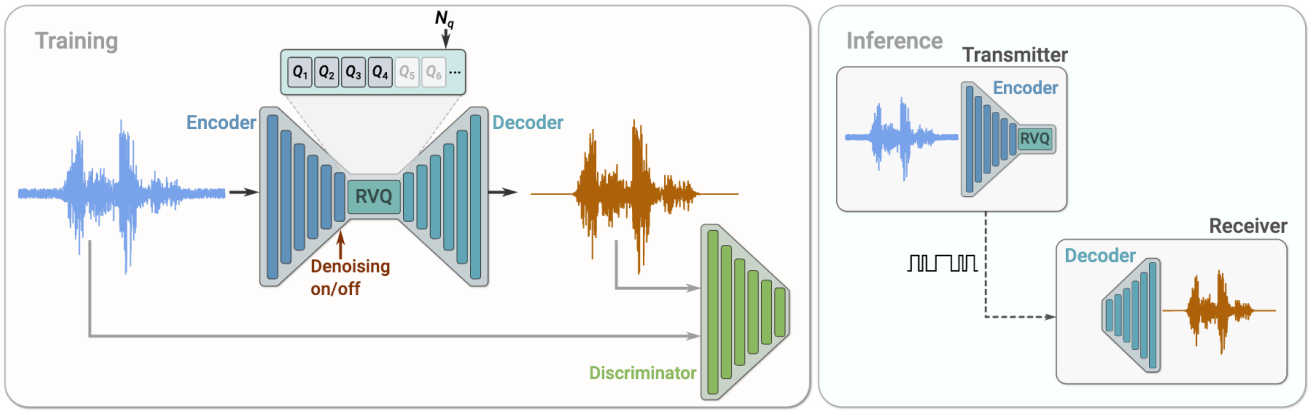


Fig. 2: *SoundStream* model architecture. A convolutional encoder produces a latent representation of the input audio samples, which is quantized using a variable number n_q of residual vector quantizers (RVQ). During training, the model parameters are optimized using a combination of reconstruction and adversarial losses. An optional conditioning input can be used to indicate whether background noise has to be removed from the audio. When deploying the model, the encoder and quantizer on a transmitter client send the compressed bitstream to a receiver client that can then decode the audio signal.

1. Task: text-to-speech and speech enhancement
2. RVQ

> The goal of the quantizer is to compress the output of the encoder enc(x) to a target bitrate R, expressed in bits/second (bps).The vector
> quantizer (VQ) in the context of VQ VAEs meets this requirement. This vector quantizer learns a codebook of N vectors to encode each D-dimensional frame of enc(x). The encoded audio $\mathrm{enc}(x) \in \mathrm{R}^{S \times D}$ is then mapped to a sequence of one-hot vectors of shape $S \times N$, which can be represented using $S \times \log_2 N$ bits.

3.2.2 EnCodec (EnCodec: High Fidelity Neural Audio Compression)

> arxiv.org/pdf/2210.13438
>
> 从框架结构来看，SoundStream和EnCodec大体是一样的，包括[损失函数，区别是其中的实现细节和功能偏向。和SoundStream一样，也是实时的端到端音频编解码器，采用encoder-decoder结构，但是EnCodec模型还包含了序列建模部分（LSTM），这一点与SoundStream是不同的，SoundStream只有卷积结构。encoder-decoder的结构在很多语音相关的任务上表现得非常好，包括分离和加强，神经网络vocoders，和编解码器。

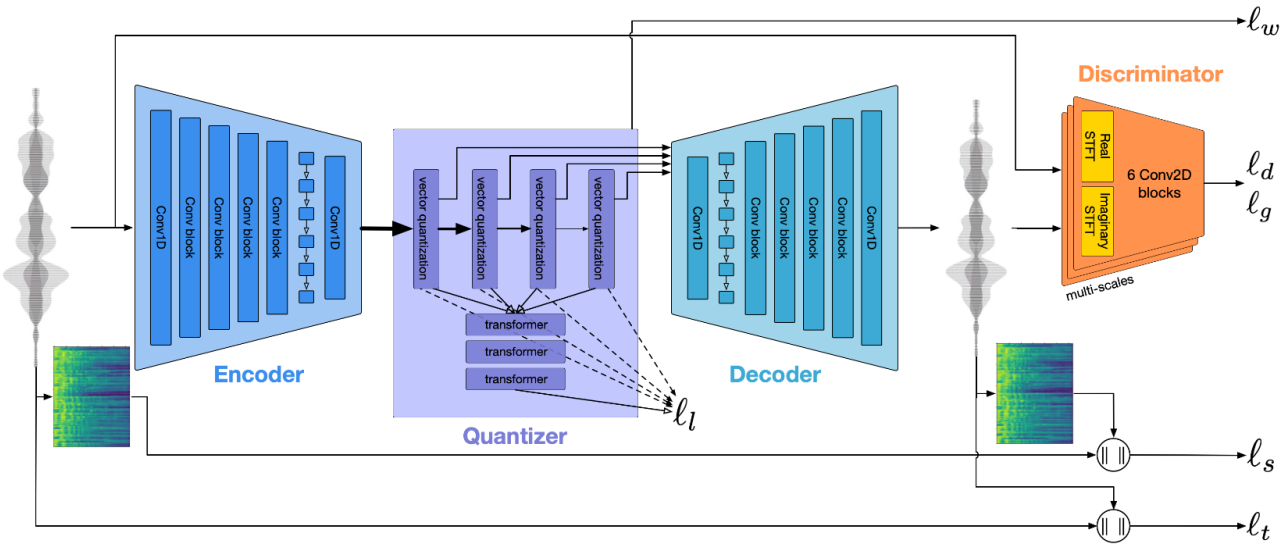


Figure 1: EnCodec : an encoder decoder codec architecture which is trained with reconstruction (ℓ_f and ℓ_t) as well as adversarial losses (ℓ_g for the generator and ℓ_d for the discriminator). The residual vector quantization commitment loss (ℓ_w) applies only to the encoder. Optionally, we train a small Transformer language model for entropy coding over the quantized units with ℓ_l, which reduces bandwidth even further.

- EnCodec applies Residual Vector Quantization(RVQ) on the output of the convolutional-based encoder. A pre-trained EnCodec processes 24 kHz audio at varying bitrates, which generates 75 Hz embeddings from 24 kHz inputs. When doing variable bandwidth training, we select randomly a number of codebooks as a multiple of 4, i.e. corresponding to a bandwidth 1.5, 3, 6, 12 or 24 kbps at 24 kHz.

> Vector quantization consists in projecting an input vector onto the closest entry in a codebookof a given size.
>
> (向量量化是将输入向量投射到固定大小码本中最接近的对应码本元素上。)
>
> RVQ refines this process by computing the residual after quantization, and further quantizingit using a second codebook, and so forth.
>
> (RVQ 通过计算量化后的残差，并使用第二个码本对残差进一步量化，不断重复这个过程来细化量化结果。)

3.2.3 HiFiCodec: Group-residual Vector quantization for High Fidelity Audio Codec

> arxiv.org/pdf/2305.02765
>
> 较好的效果需要很多层codebooks, 这会增加生成模型的难度。本文提出分组残差矢量量化（GRVQ）技术，并只使用了4个codebook就开发了高质量的音频编码模型。

1. motivation：第一层codebook能存储更多的信息，因而我们分组,从而可以使用更多codebooks应用在第一层，让这些codebooks在压缩方面能发挥更大的作用，从而减少codebooks的数量。
2. Methods：

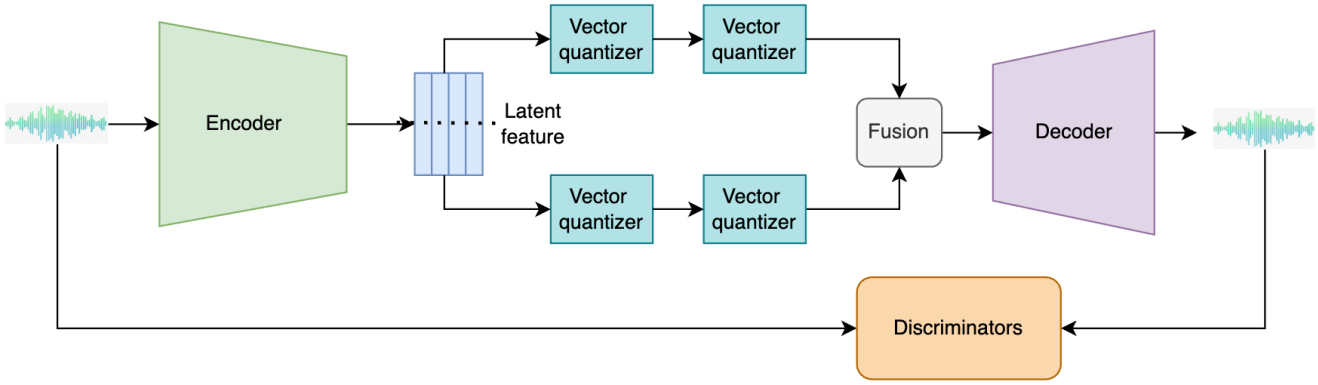


Fig. 1: The overview of HiFi-Codec model.

3.2.4 DAC (Descript Audio Codec (.dac): High-Fidelity Audio Compression with Improved RVQGAN)

> NeurIPS 2023 [arXiv Paper: High-Fidelity Audio Compression with Improved RVQGAN]
>
> Our model is **built on the framework of VQ-GANs, following the same pattern as SoundStream and EnCodec.**

1. Motivation: Vanilla VQ-VAEs struggle from **low codebook usage** due to poor initialization, leading to a significant portion of the code-book being unused. This reduction in effective code-book size leads to an implicit reduction in target bitrate, which translates to poor reconstruction quality.

> Recent audio codec methods use k-means clustering to initialize the codebook vectors, and manually employ randomized restarts when certain codebooks are unused for several batches.However, we find that the EnCodec model trained at 24kbps target bitrate, as well as our proposed model with the same codebook learning method (Proposedw/ EMA) still suffers from codebook under-utilization.

2. Methods: factorized codes and L2-normalized codes.(from Improved VQGAN image model[2110.04627])
    - Factorization decouples code lookup and code embedding, by performing code lookup in a low-dimensional space(8d or 32d) whereas the code embedding resides in a high dimensional space (1024d). Intuitively,this can be interpreted as a code lookup using only the principal components （PCA） of the input vector that maximally explain the variance in the data.
    - The L2-normalization of the encoded and codebookvectors converts euclidean distance to cosine similarity, which is helpful for stability and quality.

3.2.5 WavTokenizer: an Efficient Acoustic Discrete Codec Tokenizer for Audio Language Modeling

> [jishengpeng/WavTokenizer: SOTA discrete acoustic codec models with 40 tokens per second for audio language modeling (github.com)]
>
> Our model is built on the framework of VQ-GANs, **following the same pattern as SoundStream (Zeghidour et al., 2021) and EnCodec (D´efossez et al., 2022).** Specifically, WavTokenizer passes the raw audio X through three modules. 1) a full convolution encoder network that takes the input audio and generates a latent feature representation Z; 2) A single quantizer discretizes Z to generate a discrete representation Zq . 3) an improved decoder that reconstructs the audio signal ˜X from the compressed latent representation Zq . The model is trained end-to-end, optimizing a reconstruction loss applied over both time and frequency domains, along with a perceptual loss in the form of discriminators operating at different resolutions.

3.3 hybrid

3.3.1 SpeechTokenizer

> Our model is built on the framework of RVQ-GANs, following the same pattern as SoundStream and EnCodec.To achieve a hierarchical modeling of diverse information across different RVQ layers, we employ the 9th layer HuBERT representation or the average representation across all HuBERT layers as semantic teachers as semantic teacher to continuous representation distill for the first quantizer, enabling it to capture content information. Leveraginga residual structure enables the subsequent quantizers to complement the remaining paralinguisticinformation.

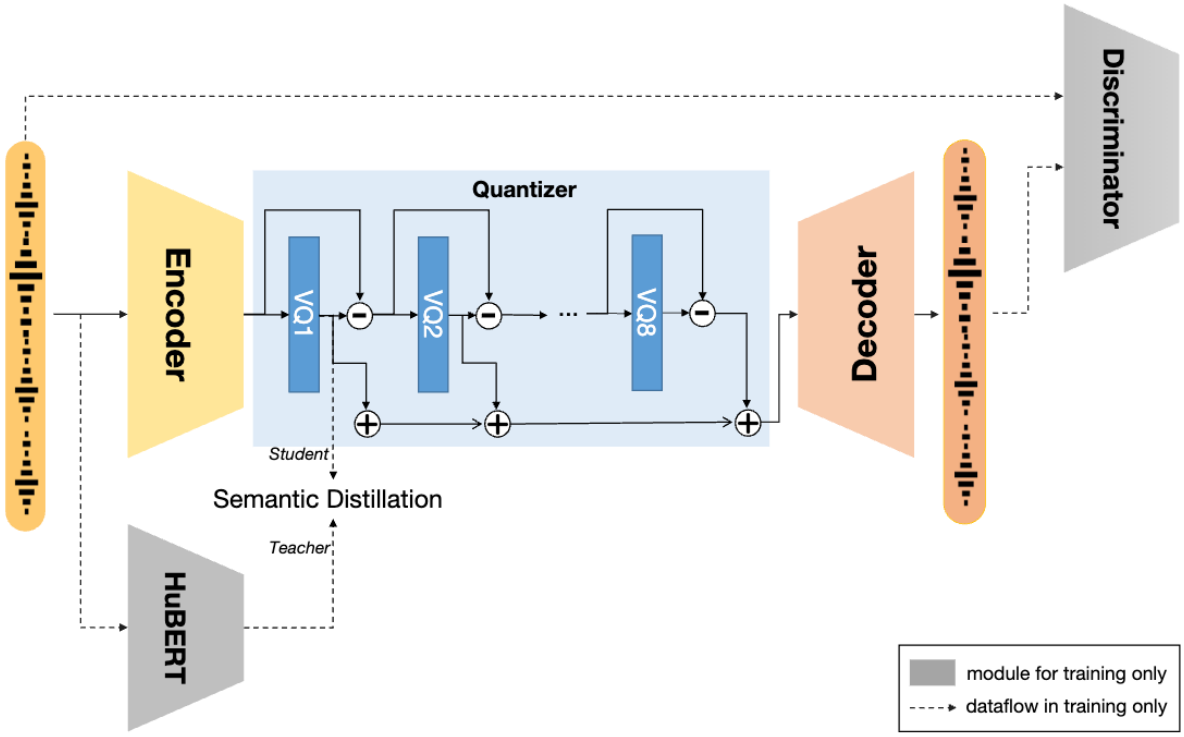


Figure 2: Illustration of SpeechTokenizer framework.

4. REF

1. 语音预训练模型 - 知乎 (zhihu.com)
2. 大模型时代下的语音合成之路——Neural Audio Codec - 知乎 (zhihu.com)
3.