

# マルチモーダル 大規模言語モデル入門

音学シンポジウム 2025

小松亮太(東京科学大学)

# 自己紹介

- 経歴

- 2023年 東京工業大学 修士(工学)
- 2023年～2025年 株式会社日立製作所 研究開発グループ
- 2025年～ 東京科学大学 博士課程

- 研究分野

- 音声表現学習
- 音声言語モデリング

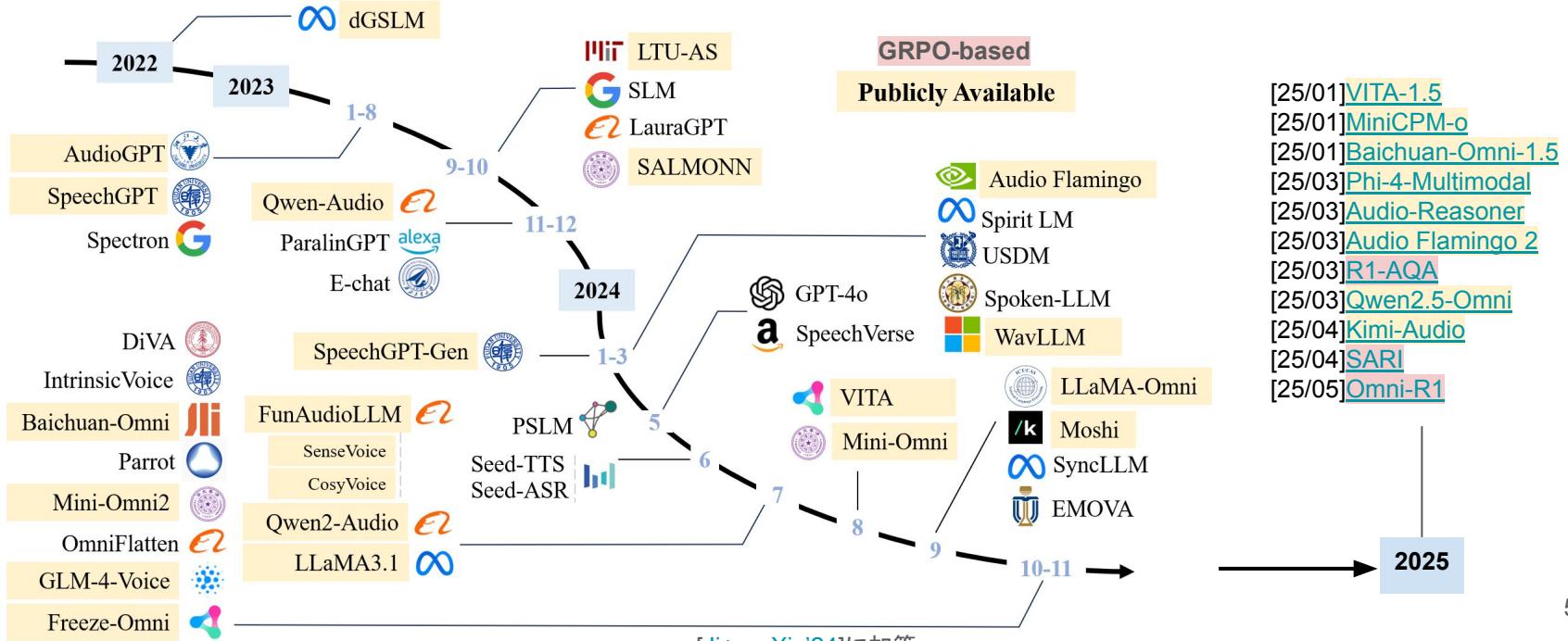
# アジェンダ

1. 最新の研究動向
2. モダリティ情報の表現形式
3. トーケン系列生成手法
4. 学習手法
5. データセット・ベンチマーク
6. 演習
7. まとめ

# 1. 最新の研究動向

# 音声・音響分野におけるマルチモーダルLLMの開発年表

- 学術機関が学習パラダイムの研究、産業コミュニティが基盤モデル開発を担う傾向
- 2025年5月時点では、強化学習アルゴリズム GRPO[[Shao+, arXiv'24](#)]ベースの手法がSOTA



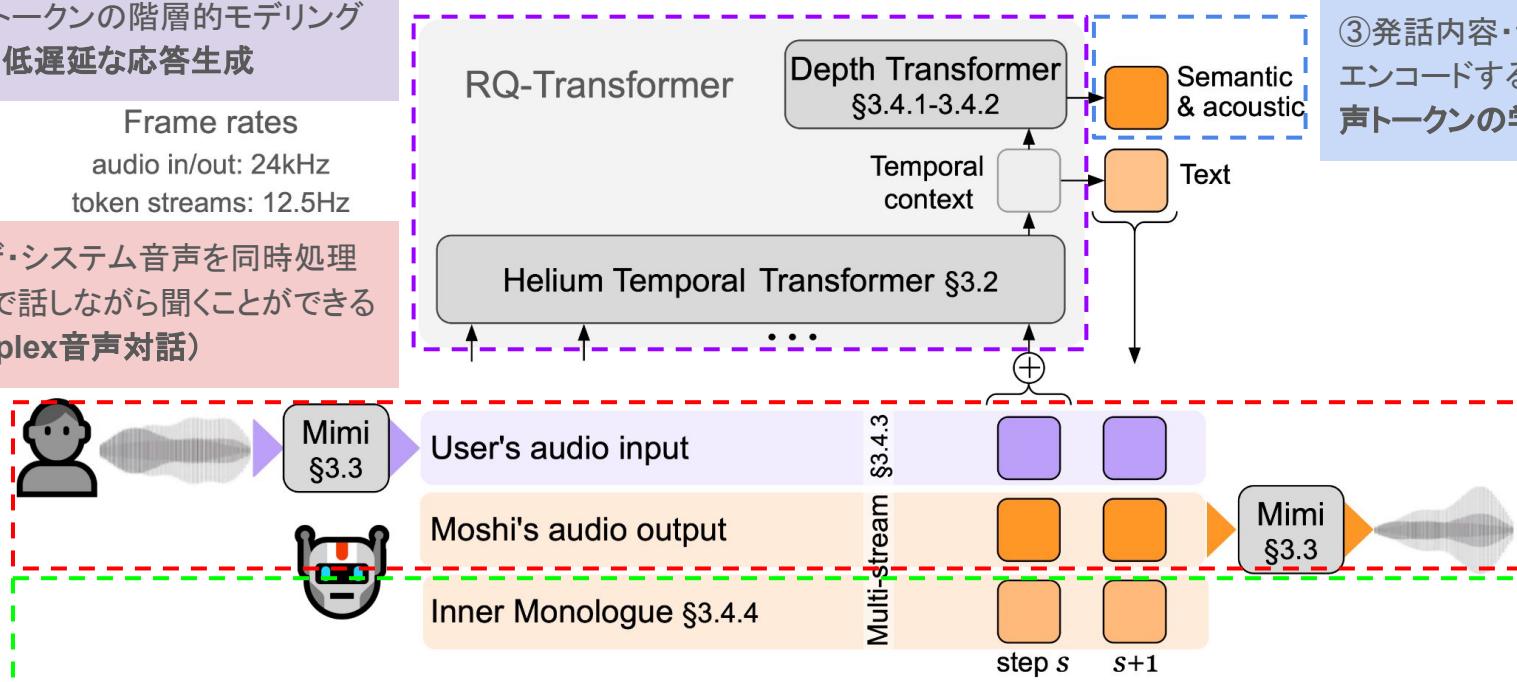
# Moshiにみる最近の技術的進展 [Défossez+, arXiv'24]

④音声トークンの階層的モデリングによる、低遅延な応答生成

Frame rates  
audio in/out: 24kHz  
token streams: 12.5Hz

①ユーザ・システム音声を同時処理することで話しながら聞くことができる  
(full-duplex音声対話)

③発話内容・音響信号をエンコードする離散的な音声トークンの学習

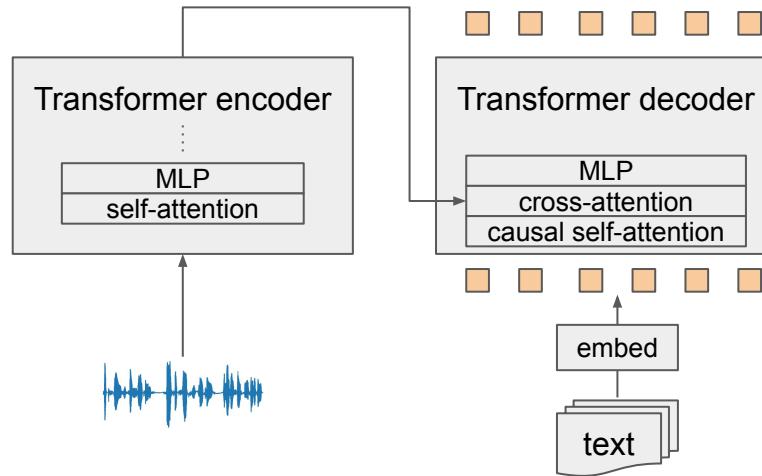


②LLMの論理的推論能力(reasoning capability)を効果的に活用することによる、応答精度の向上

# Transformer言語モデル

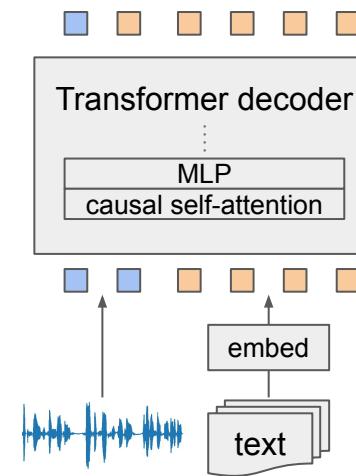
## Encoder-decoder

- Cross-attentionで接続される sequence-to-sequenceモデル (e.g., Whisper [[Radford+, ICML'23](#)])



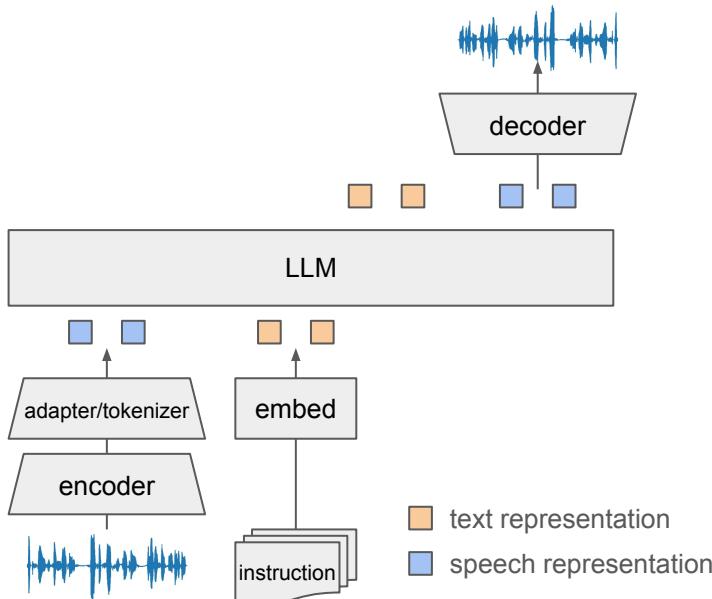
## Decoder-only

- GPTに代表される自己回帰型言語モデル
- 本講演では、マルチモーダル LLMにおいて主流であるこちらを扱います



# 音声テキスト言語モデルの分類

- 従来ではASR, LLM, TTSを組み合わせたカスケード型のシステムが用いられたが、現在ではend-to-endにシステム全体を最適化するアプローチが一般的
- 音声テキスト言語モデルは入出力形式の観点から下表に示す 2種類に分類できる



	speech+text LM	speech-aware text LM
構成	encoder+tokenizer+LLM+decoder	encoder+adapter+LLM(+TTS)
入力	音声・テキスト	音声・テキスト
出力	音声・テキスト	テキスト
音声表現※	離散	連続
例	<a href="#">SpeechGPT</a> , <a href="#">AudioPaLM</a> , <a href="#">AnyGPT</a> , <a href="#">PSLM</a> , <a href="#">Mini-Omni</a> , <a href="#">Moshi</a> , <a href="#">SpiRit-LM</a> , <a href="#">GLM-4-Voice</a> , <a href="#">Kimi-Audio</a>	<a href="#">WavPrompt</a> , <a href="#">LTU-AS</a> , <a href="#">SLM</a> , <a href="#">SALMONN</a> , <a href="#">SpeechVerse</a> , <a href="#">AudioChatLlama</a> , <a href="#">Qwen2-Audio</a> , <a href="#">Llama3</a> , <a href="#">Phi-4-Multimodal</a>

※LLMで音声表現を出力する場合、音声表現は離散である方が言語モデリングの枠組みに当てはめやすいが、LLMで直接音声表現を出力しない場合、連続表現の方が勾配伝播でend-to-endにadapterを最適化しやすい

## 2. モダリティ情報の表現形式

# 連続・離散音声表現の性質

- 連続表現は言語情報に加え、話者性・感情などの非言語・パラ言語情報も含むため下流タスク全般に有用
- 離散表現では情報ボトルネックにより非言語(+わずかな言語情報)が削ぎ落とされるが[Niekerk+, Interspeech'21], [Niekerk+, ICASSP'22], [Kharitonov+, NAACL'22]、音素・単語などの言語単位と強く相関するため発話の言語的内容のモデリングに適する[Nguyen+, JSTSP'22]

		Accuracy (%)		
Standardized	Clustered	Phone	Speaker	Gender
Linear classifiers:				
✗	✗	75.7	93.4	96.7
✓	✗	77.0	14.8	55.3
✗	✓	46.6	3.4	53.5
✓	✓	48.5	3.1	50.9
Non-linear classifiers:				
✗	✗	80.1	99.5	99.8
✓	✗	79.7	89.0	98.1

CPC[Oord+, arXiv'18]潜在表現における音素・話者・性別識別精度[Niekerk+, Interspeech'21]。連続表現では話者・性別識別率90%↑。クラスタリングした離散表現では話者・性別識別率はほぼチャンスレベルであり、音素識別精度も低減。

id	input	target	loss	語彙理解		
				sWUGGY↑	sBLIMP↑	wSIM↑
discrete input, discrete target						
1	disc.	disc.	NLL-1	79.28	59.71	6.32
2	disc.	disc.	NLL-e	80.02	59.86	7.87
continuous input, discrete target						
3	cont.	disc.	NLL-1	60.36	53.23	8.39
4	cont.	disc.	NLL-e	60.20	52.78	9.49
continuous input, continuous target						
5	cont.	cont.	NCE	56.84	52.62	9.16
6	cont.	cont.	L1	59.23	53.12	7.85
7	cont.	cont.	L2	60.56	53.33	6.55
discrete input, continuous target						
8	disc.	cont.	NCE	65.69	57.24	9.33
9	disc.	cont.	L1	73.93	56.02	10.69
10	disc.	cont.	L2	74.22	55.75	5.97

連続・離散CPC音声表現を用いたBERTによる語彙・構文・意味理解性能[Nguyen+, JSTSP'22]。入力表現は離散である方が理解性能は高い。

# 音声トークナイザの分類

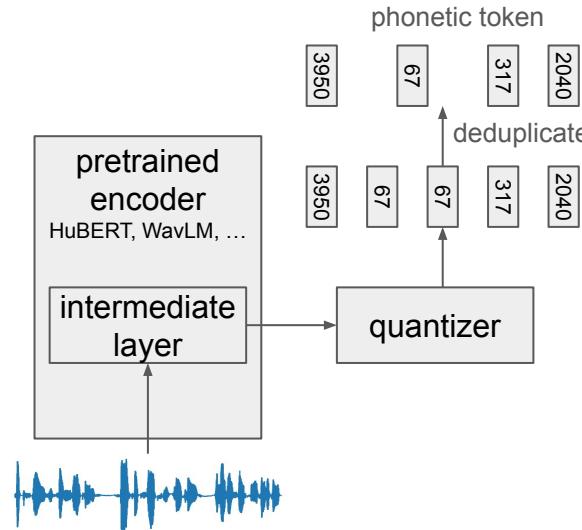
離散音声トークンは主に2種類に分類できる

メインカテゴリ	サブカテゴリ	例
Phonetic token※	Self-supervised	<a href="#">HuBERT</a> , <a href="#">WavLM</a> , <a href="#">BEST-RQ</a> , <a href="#">data2vec</a>
	Perturbation invariant	<a href="#">ContentVec</a> , <a href="#">Spin</a> , <a href="#">NAST</a>
	Subword/syllabic	<a href="#">SD-HuBERT</a>
Acoustic token	Supervised	<a href="#">S<sup>3</sup>tokenizer</a> , <a href="#">GLM-4-Voice-Tokenizer</a>
	General	<a href="#">SoundStream</a> , <a href="#">EnCodec</a> , <a href="#">DAC</a> , <a href="#">WavTokenizer</a>
	Phonetic distillation	<a href="#">SpeechTokenizer</a> , <a href="#">Mimi</a>
	Semantic	<a href="#">BEATs</a>

※semantic tokenとも呼ばれるが[[Borsos+](#), TASLP'23], 実際にはsemanticというよりphoneticである[[Choi+](#), Interspeech'24]

# Phonetic token(離散ユニット)

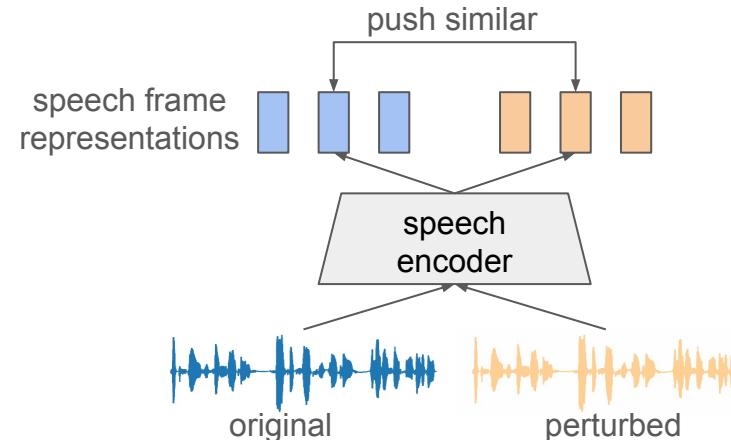
- 音声エンコーダ(HuBERT, WavLM等)の中間Transformer層から抽出した潜在音声フレーム表現を量子化
- 言語単位(e.g., 音素)と強く相関する **phonetic**なトークンであるため、テキストと同様に言語モデルを構築可能
- 通常、言語モデリングの性能向上のため、連続する同一ユニットから重複を除去[Hsu+, ACL'21]
- クラスタ数は大きい方が合成音声の質が高いが、大きすぎると言語モデリング精度が劣化[Lakhotia+, TACL'21], [Maiti+, ICASSP'24]
- textless-lib[Kharitonov+, NAACL'22]から学習済みHuBERT・CPCトークナイザを簡単に利用可能



# 話者・ノイズに対する摂動不変ユニット

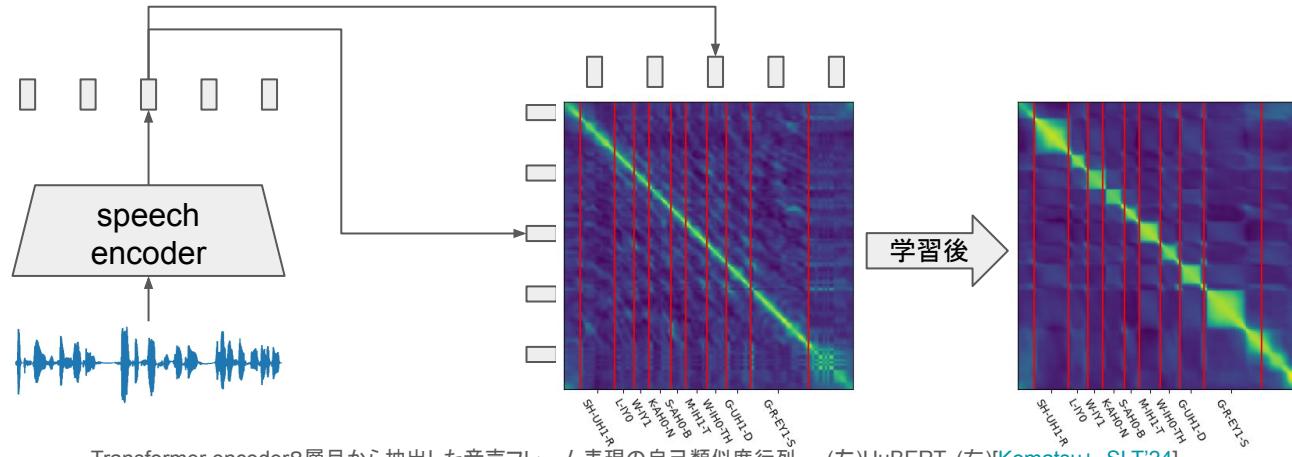
元音声・摂動付与音声それぞれから一貫した特徴量を抽出するように学習することで摂動に頑健なユニットを獲得

	<u>ContentVec</u>	<u>Spin</u>	[Gat+, IWSLT'23]	<u>NAST</u>
ピッチ・フォルマントシフ <small>[Choi+, NeurIPS'21]</small>	✓	✓	✓	✓
話速調整 <small>[Ko+, Interspeech'15]</small>			✓	✓
ノイズ付加			✓	✓
残響付与			✓	✓



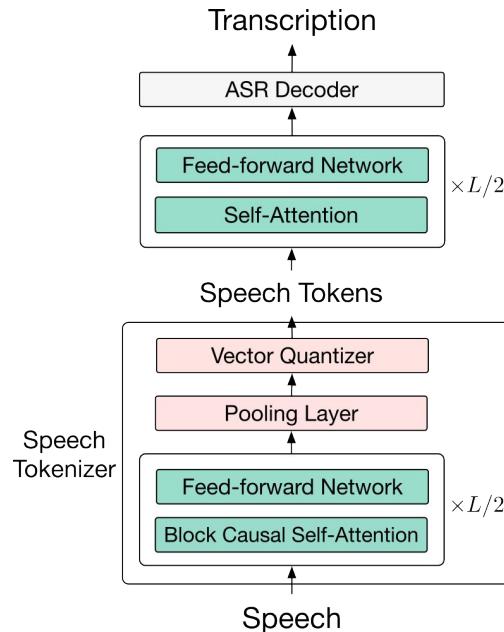
# サブワード・音節ユニット

- 50HzのHuBERT unitでは1分の音声でも系列長が3000となり計算量が大きいため、粒度を粗くしたい
- サブワードユニット [[Li+, Interspeech'24](#)], [[Shen+, ICASSP'24](#)], [[Dekel+, Interspeech'24](#)]
  - NLP同様、HuBERTユニットに対してbyte-pair encoding (BPE)を適用することでサブワード化
- 音節ユニット [[Cho+, ICASSP'24](#)], [[Komatsu+, SLT'24](#)], [[Baade+, ICLR'25](#)], [[Cho+, ICLR'25](#)]
  - 事前学習された音声エンコーダをteacher-student学習で自己教師ありファインチューニングすることにより、**中間Transformer層表現が音節単位で組織化**
  - 音節表現をセグメンテーション・プーリング・量子化することで**4–6Hz**の音節ユニットを得る



# GLM-4-Voice [Zeng+, ICLR'25], [Zeng+, arXiv'24]

- Whisper-large-v3エンコーダの中間にプーリングおよび単一ベクトル量子化を追加し, ASRで教師あり学習
- ベクトル量子化以前のself-attentionをblock causal self-attentionに変更することで, ストリーミング量子化を可能に
- 再合成音声の質・トークン効率において, MoshiのトークナイザであるMimiを上回る



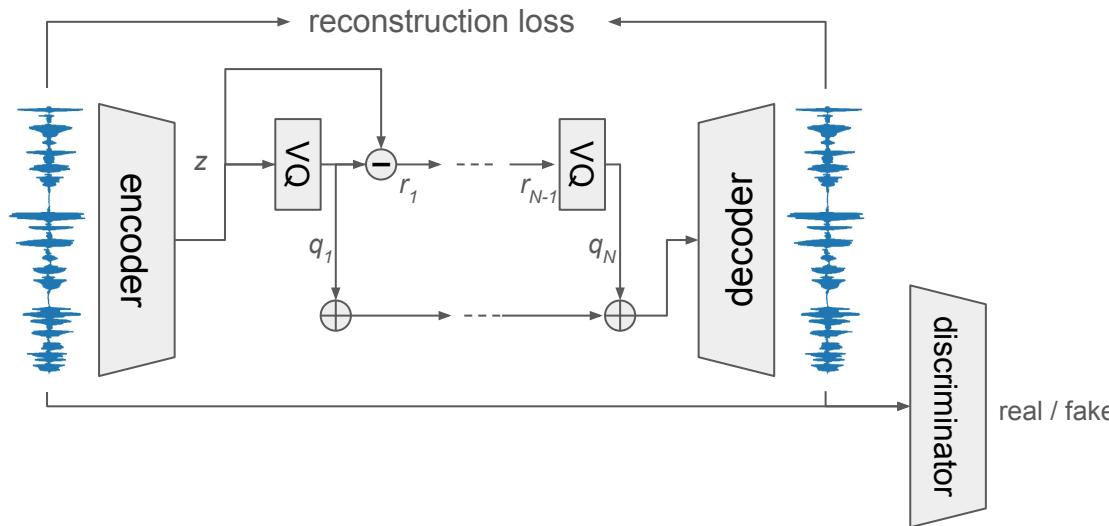
[Zeng+, arXiv'24]

Model	Frame Rate	Bitrate	Causal	LibriSpeech		
				WER↓	VisQOL↑	MOSNet↑
Ground Truth	-	-	-	4.62	-	3.27
RVQGAN	75Hz	1.50K	✗	-	1.74	2.74
SemantiCodec	50Hz	1.30K	✗	-	2.43	3.12
SpeechTokenizer	50Hz	1.50K	✗	-	1.53	2.67
SpeechTokenizer	50Hz	4.00K	✗	-	3.07	3.10
Spirit-Base	25Hz	225.0	✗	11.66	-	-
Spirit-Expressive	38.5Hz	307.0	✗	10.60	-	-
Moshi (Mimi)	12.5Hz	1.10K	✓	8.36	2.82	2.89
Ours	50Hz	600	✓	6.24	2.67	3.38
	25Hz	300	✓	6.80	2.60	3.33
	12.5Hz	175	✓	8.43	2.52	3.39
	6.25Hz	100	✓	14.41	2.34	3.24

音声再合成結果 [Zeng+, ICLR'25]

# Acoustic token

- Neural audio codecで音声波形を再構成するように学習
- Residual Vector Quantizer (RVQ)で残差( $r=z-q$ )を階層的に量子化
  - SoundStream[Zeghidour+, TASLP'21], EnCodec[Défossez+, TMLR'23], DAC[Kumar+, NeurIPS'23]
- 最近では單一コードブックのモデルも登場
  - Single-Codec[Li+, Interspeech'24], WavTokenizer[Ji+, ICLR'25], BigCodec[Xin+, arXiv'24], TS3-Codec[Wu+, arXiv'24]



# Codebook collapse

- コードが高次元である場合、コードブックのうち一部のコードのみしか活性化しない codebook collapseがしばしば起こる
- モデルのエンコード容量を制限してしまうため、再構成精度およびエンコード効率を低下させる
- コードブックを満遍なく使うために有効な手法が提案されている
  - 活性化しないコードをランダムベクトルで置換 [[Dhariwal+](#), arXiv'20], [[Zeghidour+](#), TASLP'21], [[Ji+](#), ICLR'25]
  - k-meansでコードを初期化することで、コード分布を潜在表現分布に近づける [[Zeghidour+](#), TASLP'21], [[Ji+](#), ICLR'25]
  - 潜在表現・コードを低次元へ射影してコード参照 [[Chiu+](#), ICML'22], [[Yu+](#), ICLR'22], [[Kumar+](#), NeurIPS'23]
  - 潜在表現・コードを  $\ell_2$  正規化して距離を算出 [[Chiu+](#), ICML'22], [[Yu+](#), ICLR'22], [[Kumar+](#), NeurIPS'23]

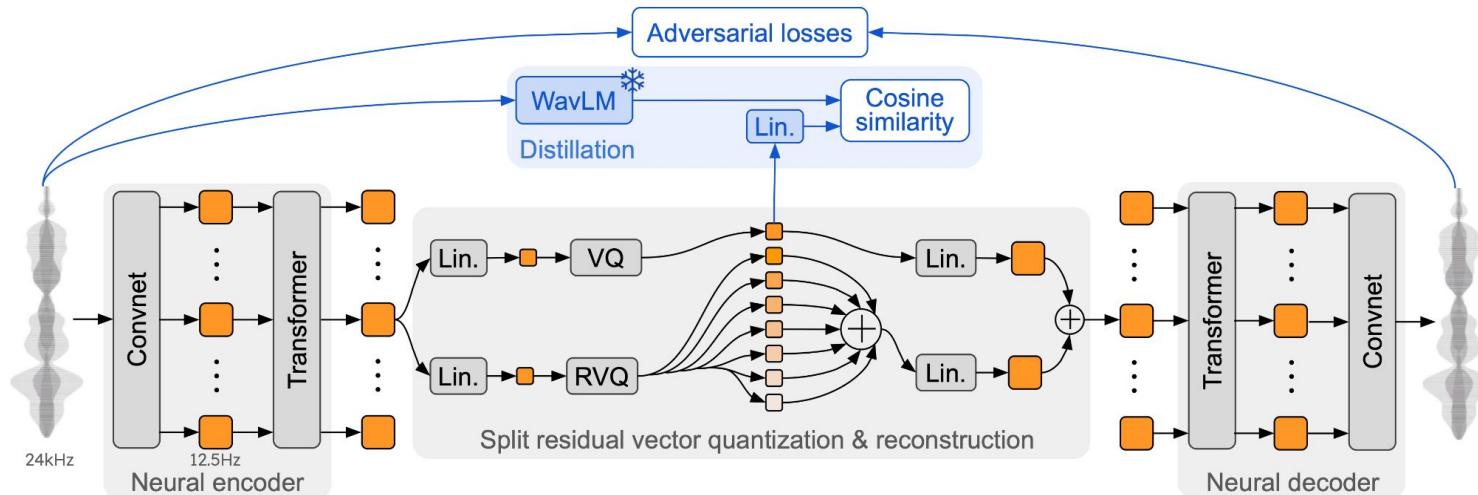
# Phonetic tokenとacoustic tokenにおけるトレードオフ

	Phonetic token	Acoustic token
対象	音声	音声・音響信号・音楽
粒度	predictiveな目的関数で最適化されるため、言語単位を識別する情報を含んでおり、粗い	元の音声波形を再構成するように最適化されるため、詳細な音響情報まで含んでおり、細かい
圧縮率	高	低
合成音声の質	低	高
音声合成器の要否	別途学習が必要	デコーダも同時に得ることができる
言語モデリング容易性	プロンプト無し(unconditional)でも一貫した発話内容を生成可能 [ <a href="#">Lakhotia+, TACL'21</a> ]	babblingを生成してしまうことがある [ <a href="#">Borsos+, TASLP'23</a> ]
解釈性	言語単位との対応が容易	困難

# Phonetic distillation

[Zhang+, ICLR'24]

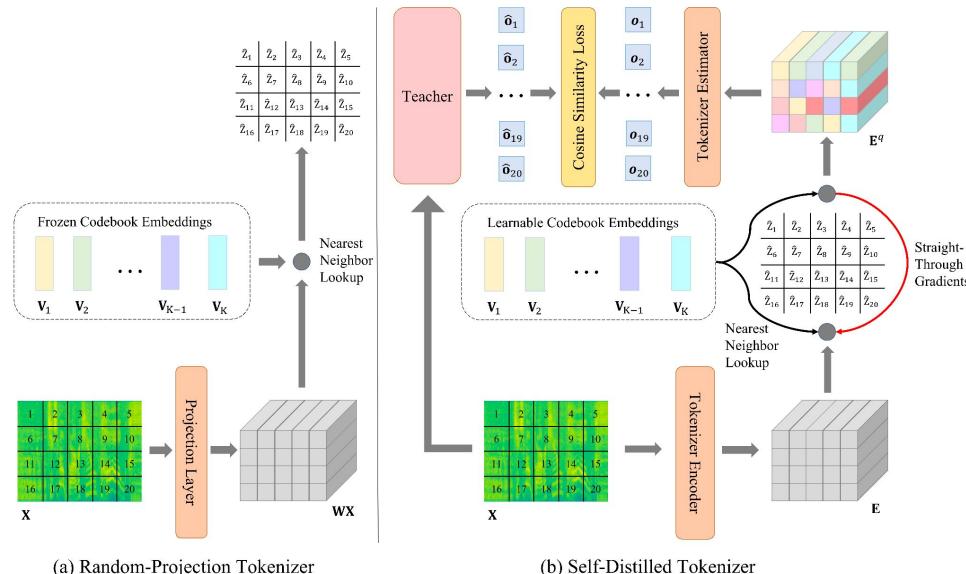
- RVQ1層目にて音声エンコーダからphonetic tokenへ言語情報を蒸留し, 2層目以降でacoustic tokenを得る
- Phonetic tokenとacoustic tokenで**相補的に言語・音響情報を抽出**
- Mimi[[Défossez+, arXiv'24](#)]では, 畳み込み層にleft padding, self-attentionにcausal maskを導入することでストリーミング処理を可能に



[Défossez+, arXiv'24]

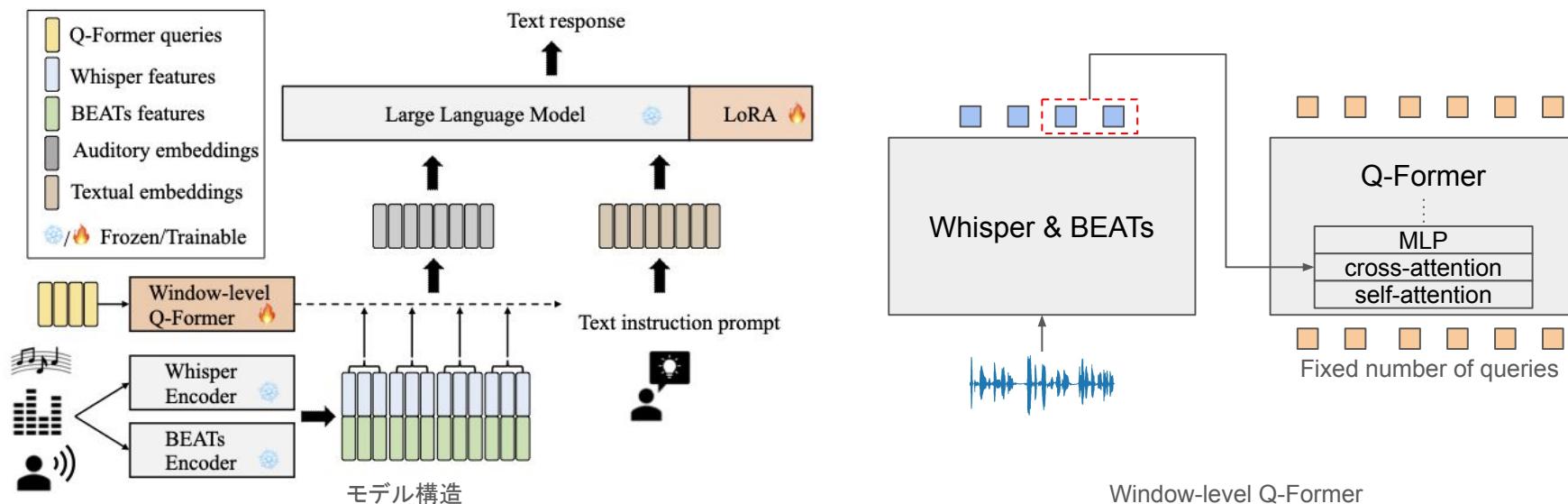
# BEATs [Chen+, ICML'23]

- 再構成ベースの目的関数では、時間・周波数に関する詳細な情報までエンコードされる
- 一方で、音響理解タスクでは意味的な特徴量を抽出したい
  - 例えば、トーンが異なっていても、鳥の鳴き声は同じ離散トークンに分類したい
- そこで、離散トークンの予測を通じて意味的な音響トークンを学習
- 最初の反復ではトークナイザをランダムに初期化 [Chiu+, ICML'22]、以降の反復では学習した表現で自己蒸留



# SALMONN [Tang+, ICLR'24]

- Whisper音声特徴量とBEATs音響特徴量をQ-Former[Li+, ICML'23]で集約することで、**音声・音響・音楽を統合的に理解**
- 固定長クエリとのcross-attentionで固定長表現を抽出するQ-Formerを、窓レベルで適用することで可変長出力
- Vicuna 7/13BをASR・audio captioningにて事前学習した後、全12タスクからなる4400時間の音声・音響・音楽データセットでマルチタスクinstruction tuning

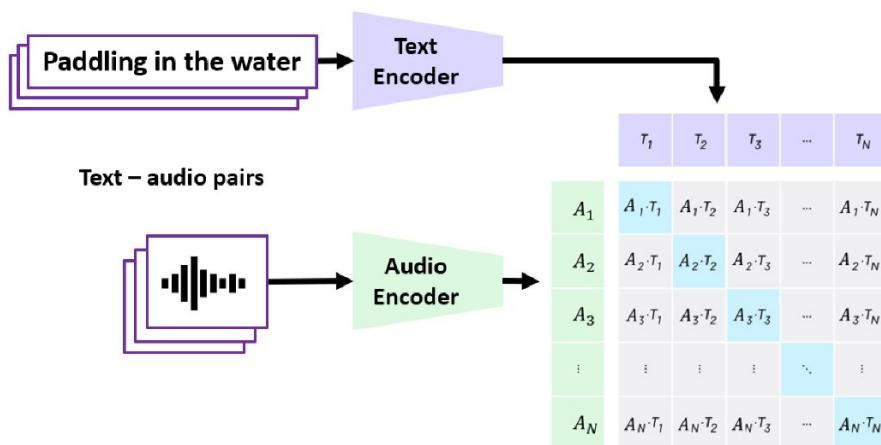


# CLAP

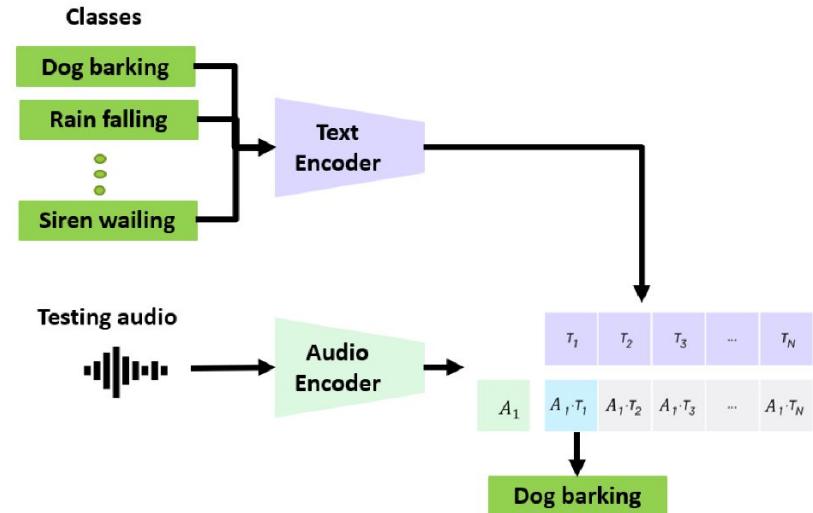
[Elizalde+, ICASSP'23], [Wu+, ICASSP'23], [Elizalde+, ICASSP'24], [Ghosh+, arXiv'25]

- テキスト・オーディオのペアで対照的に類似度を学習
- Audio Flamingo 2[Ghosh+, arXiv'25]では、8Mのペアデータで学習したCLAPとLLMを統合することで、特に音楽理解においてSOTAレベルの性能を達成 [Rouditchenko+, arXiv'25]

## 1. Contrastive Pretraining



## 2. Use pretrained encoders for zero-shot prediction in a new dataset or task



# 連続音声表現のためのAdapter

- Speech-aware text LMでは, adapterによって音声エンコーダが抽出した連続表現をLMへの入力埋め込みに変換
- サブワードのテキストトークンと比較して音声フレーム表現が高周波である場合adapterによって粒度を揃える

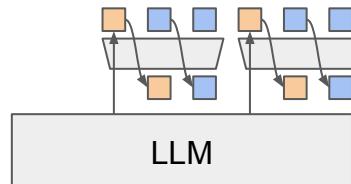
Adapter	実装	採用モデル
downsampling + MLP	$k$ 連続フレームをチャネル次元に沿って連結した後, MLP	<a href="#">LLaMA-Omni</a>
CNN / pooling	ストライドでダウンサンプリング	<a href="#">SpeechVerse</a> , <a href="#">Qwen2-Audio</a>
CNN + Transformer + Linear	ストライドでダウンサンプリング	<a href="#">Llama3</a>
CTC[ <a href="#">Graves+, ICML'06</a> ]	連続する非blank表現をpoolingで平均した後, blank表現除去	<a href="#">Speech-LLaMA</a>
Q-Former[ <a href="#">Li+, ICML'23</a> ]	固定長クエリとのcross-attentionで音声フレーム表現を固定長に	<a href="#">SALMONN</a> , <a href="#">DeSTA</a>
AlignFormer[ <a href="#">Fan+, arXiv'24</a> ]	CTC + Q-Former	<a href="#">AlignFormer</a>

### 3. トークン系列生成手法

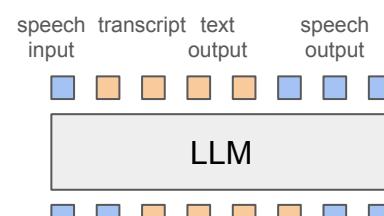
# Speech+text LMにおけるトークン系列生成

- テキストのみ出力する speech-aware text LMとは異なり、音声とテキストどちらも出力する speech+text LMでは設計上選択肢がある
- 高応答精度や低遅延を実現するために主に 4種類の系列生成手法が提案されている

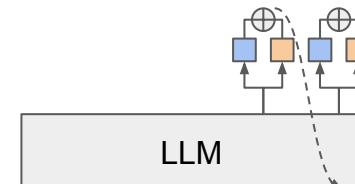
手法	生成手順	採用モデル
Coarse-to-fine generation	text→phonetic token→acoustic token	<a href="#">UniAudio</a> , <a href="#">Moshi</a>
Chain-of-Modality	speech input→(transcript)→text output→speech output	<a href="#">SpeechGPT</a> , <a href="#">GLM-4-Voice</a>
Parallel speech-text generation	音声・テキストトークンを同時生成し、足し合わせて次時刻へ	<a href="#">PSLM</a> , <a href="#">Mini-Omni</a> , <a href="#">SLAM-Omni</a>
Text-driven generation	テキスト出力LLMの隠れ状態から音声トークン生成	<a href="#">LLaMA-Omni</a> , <a href="#">Freeze-Omni</a>



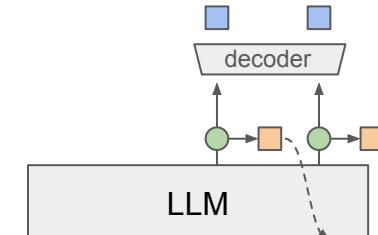
(a) Coarse-to-fine generation



(b) Chain-of-Modality



(c) Parallel speech-text generation

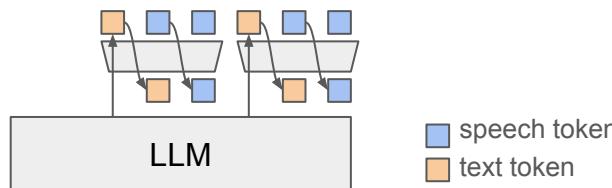


(d) Text-driven generation

■ speech token   ■ text token   ● hidden states

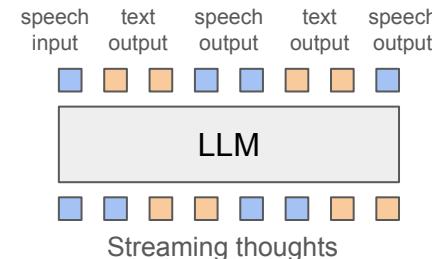
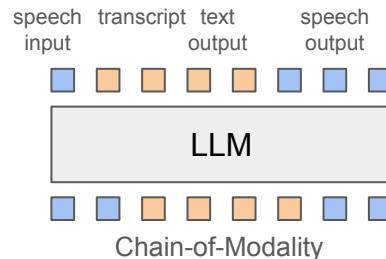
# Coarse-to-fine generation [[Borsos+, TASLP'23](#)]

- Phonetic tokenのみでは合成音声の質は低いが、 acoustic tokenを自己回帰的に生成すると babblingが起きやすい問題がある
- テキストから自己回帰的に生成した phonetic tokenを経由して acoustic tokenを予測することで， babblingを回避しつつ合成音声の質を向上
- またテキストから直接 acoustic tokenを予測する場合と比較して、 phonetic tokenを経由することで予測の複雑度が下がるため、 少量のラベル付きデータでも TTSモデルを学習可能 [[Kharitonov+, TACL'23](#)]
- Moshiでは、 時間方向LLMとチャネル方向の小規模TransformerからなるRQ-Transformer [[Lee+, CVPR'22](#)]でテキスト・音声トークンを階層的に予測



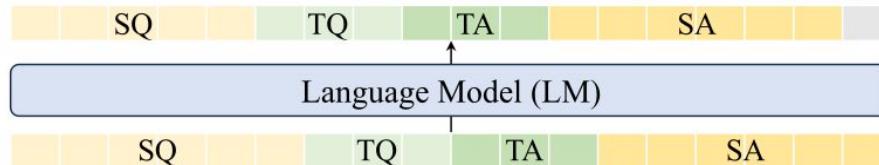
# Chain-of-Modality (CoM) [[Zhang+, EMNLP'23](#)]

- HuBERTユニットでテキスト語彙空間を拡張
- ASR→LM→TTSのプロセスのように<入力音声ユニット, 書き起こし, 出力テキスト, 出力音声>の順にトークン生成
- まずテキストで応答生成することで **LLMの論理的推論能力を活用** できるため応答精度が高い
- 一方で、出力音声の前に入出力テキストを生成する必要があるため、遅延が大きい
- GLM-4-Voice[[Zeng+, arXiv'24](#)]では、応答テキスト・音声を交互に生成する **streaming thoughts** によって、応答精度と低遅延を両立

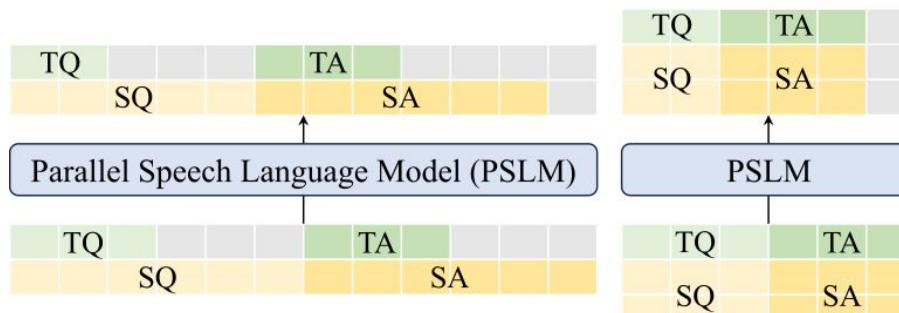


# Parallel speech-text generation [Mitsui+, EMNLP'24]

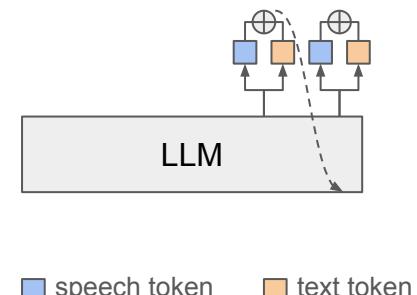
- CoMでは入力音声をテキストに書き起こし、テキストで応答生成した後、音声合成するため遅延大
- 共有LLMから抽出した潜在表現に対して、音声・テキストそれぞれの head で並列に次トークン予測
- テキスト予測によるガイダンスで **応答の質を保ちつつ**、並列生成で**低遅延を実現**



(a) Single stream decoding with Chain-of-Modality prompting



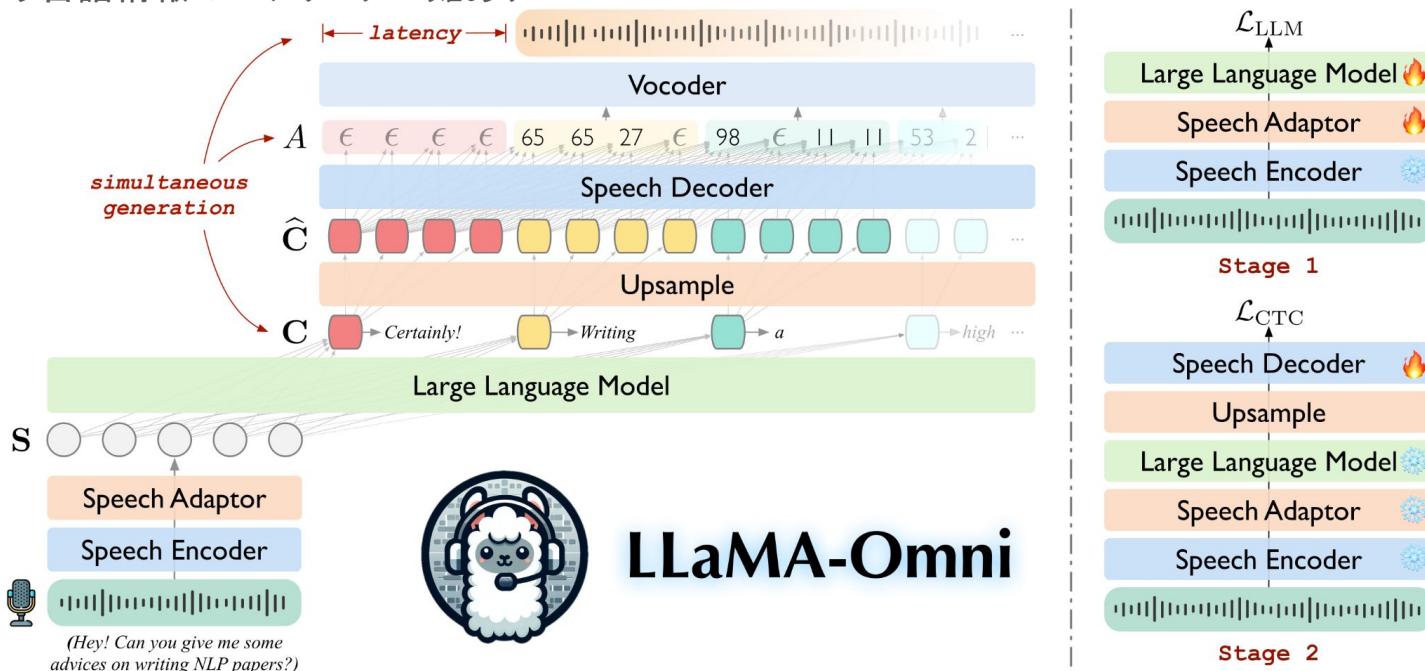
(b) Parallel text and speech decoding



(c) Multiple speech streams

# Text-driven generation [Fang+, ICLR'25]

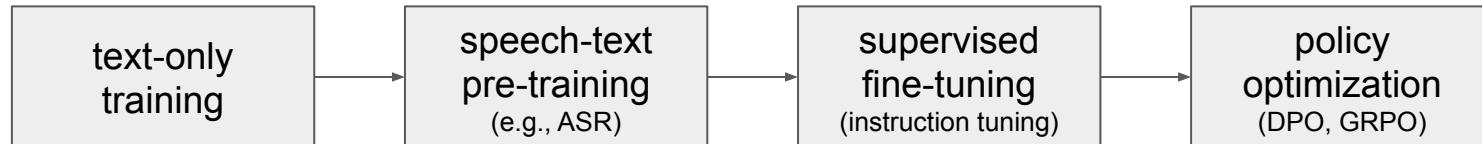
- テキストLLMの隠れ状態からHuBERTユニットをCTCで予測し、音声合成
- LLMの構造をテキスト出力のままで維持するため論理的推論能力を保持しやすいが、感情・韻律などのパラ言語情報のモデリングに難あり



## 4. 学習手法

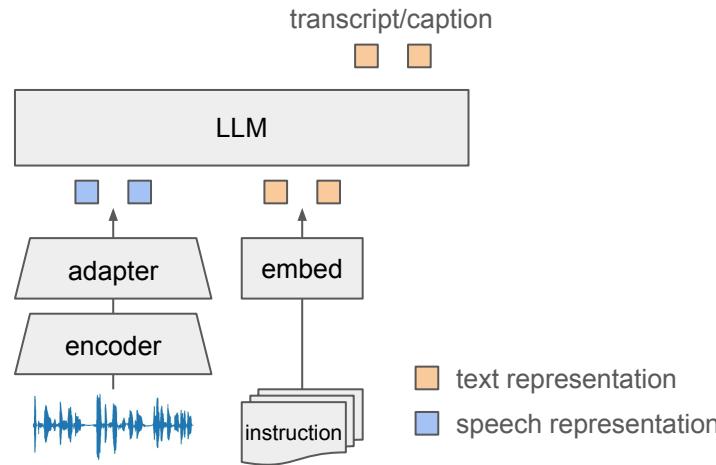
# 学習プロセス

- LLMの論理的推論能力を活用するために、まず ASRでの事前学習などにより、埋め込み空間において音声とテキストのアライメントをとる
- その後、翻訳・QA・要約などの複雑な内容理解や、話者性・感情などの非言語・パラ言語理解を要するマルチタスクでのカリキュラム学習 [[Tang+, ICLR'24](#)], [[Hu+, EMNLP'24](#)], [[Microsoft, arXiv'25](#)]
- Moshiなどの一部モデルを除き、Llamaに代表されるオープンな事前学習済み LLMを活用する場合、text-only trainingをスキップ
- Qwen2-Audio[[Chu+, arXiv'24](#)], Qwen2.5-Omni[[Xu+, arXiv'25](#)], R1-AQA[[Li+, arXiv'25](#)], SARI[[Wen+, arXiv'25](#)]、および2025年5月時点でのSOTAであるOmni-R1[[Rouditchenko+, arXiv'25](#)]は最後に方策の最適化まで実施



# Speech-aware text LMにおけるspeech-text pre-training①

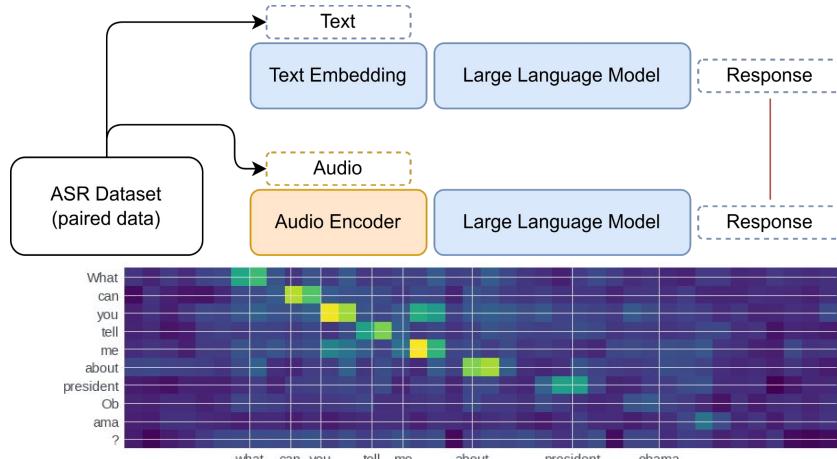
- ASR・audio captioningデータセットを用いて adapter(+事前学習済み音声エンコーダ)を学習 [[Tang+, ICLR'24](#), [Das+, arXiv'24](#), [Microsoft, arXiv'25](#)]
- 一方でLLMパラメータを凍結あるいはLoRAのみ更新することで、テキストでの論理的推論能力を保持
- LLMパラメータを凍結したとしてもzero-shotで未知タスクに対して汎化することが示されている [[Wang+, ASRU'23](#), [Meta, arXiv'24](#), [Fan+, arXiv'24](#)]



# Speech-aware text LMにおけるspeech-text pre-training②

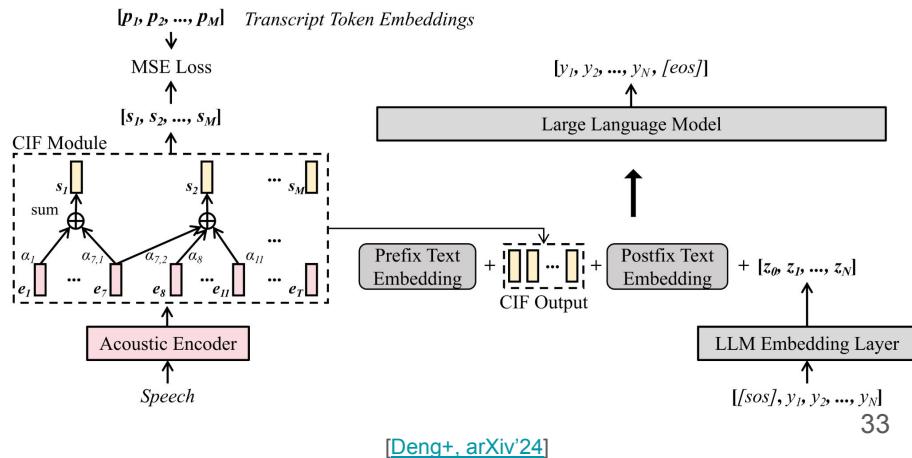
## 暗黙的アライメント [\[Fathullah+, NAACL'24\]](#), [\[Wang+, arXiv'23\]](#)

- ASRデータの書き起こしから後続文生成
- ペアとなる音声からも同一後続文を生成するように音声エンコーダを学習(LLMは凍結)
- テキスト埋め込みと対応する音声フレーム表現間で自然に類似度が高くなる



## 明示的アライメント [\[Deng+, arXiv'24\]](#), [\[Held+, arXiv'24\]](#)

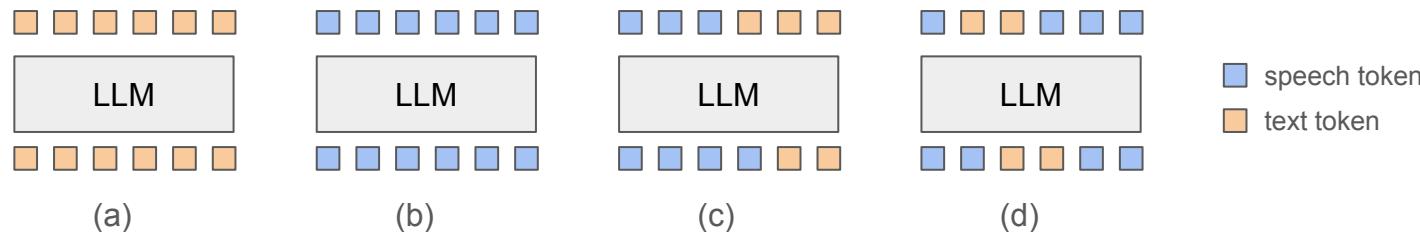
- ASRデータの書き起こしの埋め込みと音声フレーム表現との間のユークリッド距離を最小化
- LLMは凍結
- 翻訳、質問応答などのタスクでASR+LLMカスケードと同等のzero-shot性能



# Speech+text LMにおけるspeech-text pre-training

テキストと離散ユニットの混合データで学習することで、破滅的忘却(※)を軽減しつつ LLMを音声に適応

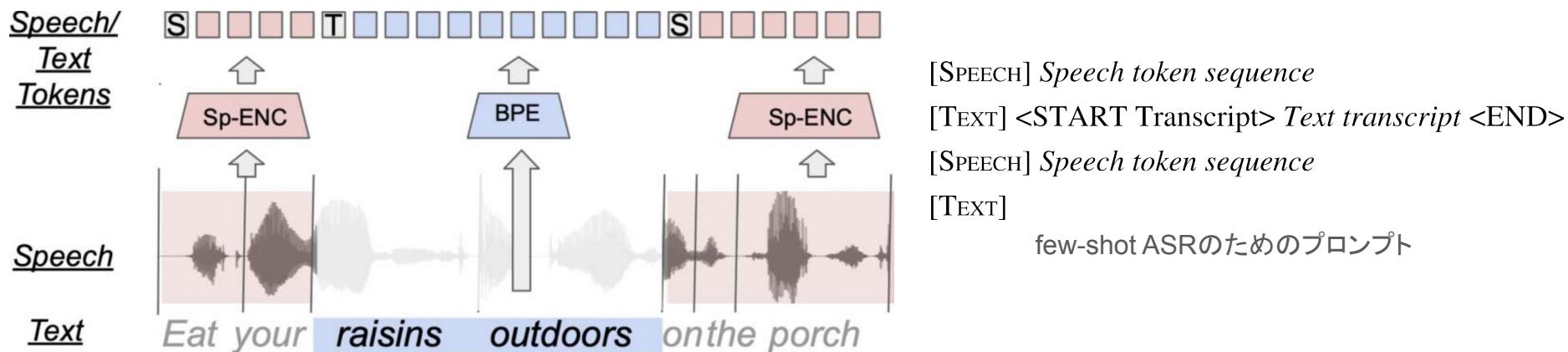
	<u>SpeechGPT</u>	<u>Moshi</u>	<u>SpiRit-LM</u>	<u>GLM-4-Voice</u>
Next text token prediction		✓		✓
Next speech token prediction	✓	✓		✓
Unit-to-text / text-to-unit prediction (ASR/TTS)	✓			✓
Word-level speech-text interleaving			✓	✓



※事前学習とは異なるタスクで継続学習すると事前学習タスクでの精度が低下する問題 [[Kirkpatrick+](#), PNAS'17]

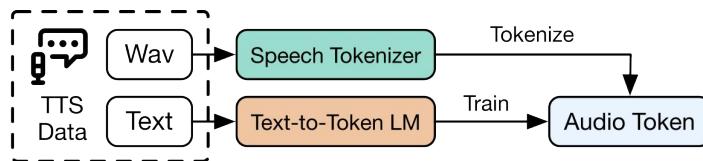
# SpiRit-LM [Nguyen+, TACL'25]

- テキストとHuBERTユニットを単語単位で交互挿入 したデータでLlama2-7Bをファインチューニング
- 推論時には、学習に用いていない ASR, TTS, intent classificationをfew-shot learningで実現可能

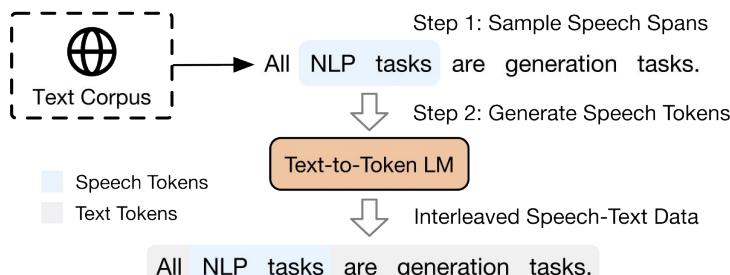


# GLM-4-Voice [Zeng+, ICLR'25]

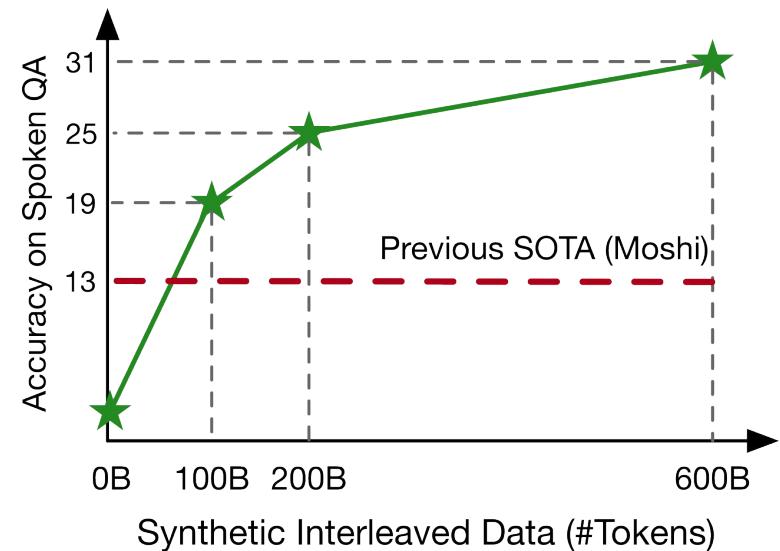
- 音声・テキストを交互配置したinterleaved dataでGLM-4-9B-Base[Team GLM, arXiv'24]を継続学習
- TTSデータを用いて、テキストからphonetic tokenを予測するtext-to-token言語モデルを学習
- Text-to-Token言語モデルを用いてテキストから **interleaved data**を合成することでデータを1兆トークンにスケール
- 訓練データ量に関してスケーリングを示し、音声質問応答においてMoshiを上回る



(a) Train a Text-to-Token Model using TTS data



(b) Construct Interleaved Speech-Text Data From Text Corpus



# Supervised Fine-Tuning (SFT)

マルチタスク学習によって音声・音響信号・音楽の統合的内容理解を実現

## タスク指定方法

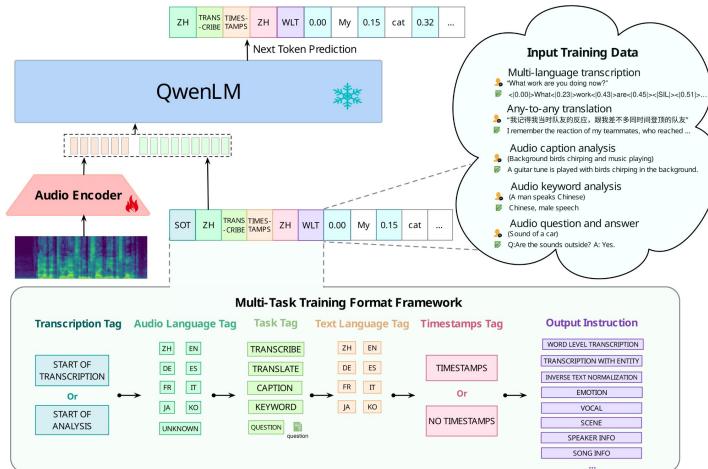
Whisper-likeな階層的タグでタスク指定

自然言語でタスク指示 (instruction tuning)

## 採用モデル

[Qwen-Audio](#)

[SpeechGPT](#), [SALMONN](#), [Qwen2-Audio](#), [Phi-4-Multimodal](#)



Task	Instruction
Speech recognition	Recognize the speech and give me the transcription.
Speech translation	Listen to the speech and translate it into German.
Audio captioning	Please describe the audio.
Audio event detection	Please list each event in the audio in order.
Speaker recognition	How many speakers did you hear in this audio? Who are they?
Emotion recognition	Describe the emotion of the speaker.
Question answering	Please answer the question in detail.
Music titling	Give me a title of the music based on its rhythm and emotion.

instruction tuningプロンプト例 [[Tang+](#), ICLR'24]

# 破滅的忘却の防止

[[Peng+, NAACL'25](#)]

- 破滅的忘却を緩和するために full fine-tuning の代わりに LoRA が用いられてきたが [[Gong+, ASRU'23](#)], [[Das+, arXiv'24](#)], LoRA でも訓練タスクへの過学習 [[Tang+, ICLR'24](#)], あるいはテキストタスクにおいて破滅的忘却が起こる
- 一方で、LLM パラメータを凍結すると、LoRA ありと比較して音声タスクの精度が低い
- Cross-modal instruction tuning サンプルとテキストのみの instruction tuning サンプルの **混合データセット** で fine-tuning することで [[Lin+, CVPR'24](#)], テキストでの推論能力を維持しつつ音声でも高い性能

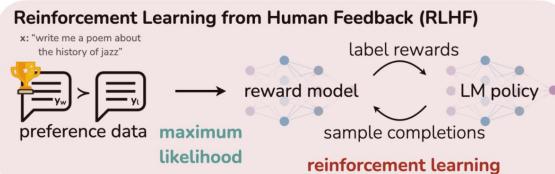
# SFTプロンプトに関する知見

- 音理解や音声翻訳において、まずキャプションあるいは書き起こしを生成するよう指示するChain-of-Thoughtによって精度が向上 [[Hu+, ICASSP'25](#)], [[Microsoft, arXiv'25](#)], [[Deshmukh+, AAAI'25](#)]
- Qwen-AudioではWhisper-likeな階層タグで事前学習した後に自然言語で instruction tuningを行ったが、Qwen2-Audioでは事前学習でも自然言語プロンプトを採用した方が、指示に従う能力が向上すると報告されている [[Chu+, arXiv'24](#)]

# Policy optimization

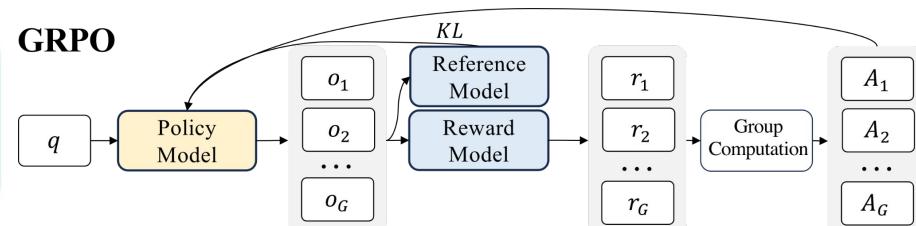
Direct preference optimization (DPO)[\[Rafailov+, NeurIPS'23\]](#)

- AIフィードバックで人間が好む応答にアライメント
- 好まれる応答の相対的な対数尤度を明示的に最大化することによって, RLHF[\[Christiano+, NIPS'17\]](#)での報酬モデルの学習を不要に



Group relative policy optimization (GRPO)[\[Shao+, arXiv'24\]](#)

- 論理的推論能力を向上
- 入力質問に対してG個の出力 $o_i$ を生成し, ルールベースの報酬モデルで正解かどうかのバイナリ報酬 $r_i$ を割当て
- 報酬間での相対的なアドバンテージ  $A_i$ に基づき方策(LLM)を最適化
- オーディオQAでSFTを上回る[\[Li+, arXiv'25\]](#)



$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[ \log \sigma \left( \beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

## 5. データセット・ベンチマーク

# SFTデータセット構築

- 主に2つのアプローチ
  - テキストinstruction tuningデータセットにおいて、入力テキストを音声合成
  - オープン音声データセットに対して、テキスト instructionをLLMで生成
    - オリジナルデータセットよりChatGPTで生成したQAで学習した方が良い性能 [[Rouditchenko+ arXiv'25](#)]

オープンなinstruction tuningデータセット

Instruction tuningデータセット	音声・音響データセット	Example数
AVQA[ <a href="#">Yang+, ACM MM'22</a> ]	VGG-Sound	38K
OpenASQA[ <a href="#">Gong+, ASRU'23</a> ]	IEMOCAP, LibriTTS, VoxCeleb2, MOSEIを含む13データセット	9.6M
Audio-FLAN[ <a href="#">Xue+, arXiv'25</a> ]	52のオープンデータセット	100M
SIFT-50M[ <a href="#">Pandey+, arXiv'25</a> ]	Multilingual Librispeech, Common Voice, VCTK	50M

# ベンチマーク

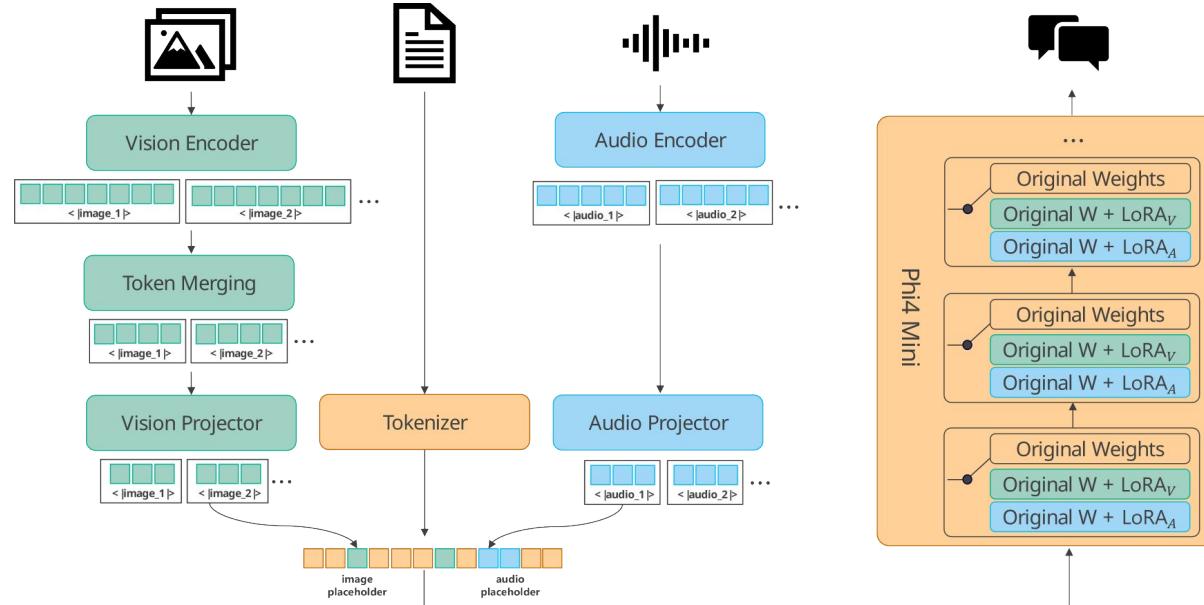
- 音声・音響・音楽タスクを含む包括的なベンチマークが整備されつつある

ベンチマーク	タスク
AIR-Bench[ <a href="#">Yang+, ACL'24</a> ]	音声, 音響, 音楽, open-end QAチャット
AudioBench[ <a href="#">Wang+, arXiv'24</a> ]	音声理解, 声理解(アクセント・性別・感情), 音響シーン理解
MMAU[ <a href="#">Sakshi+, ICLR'25</a> ]	複雑な論理的推論を要する4択問題

# 6. 演習

# Phi-4-Multimodalで音声翻訳 [Microsoft, arXiv'25]

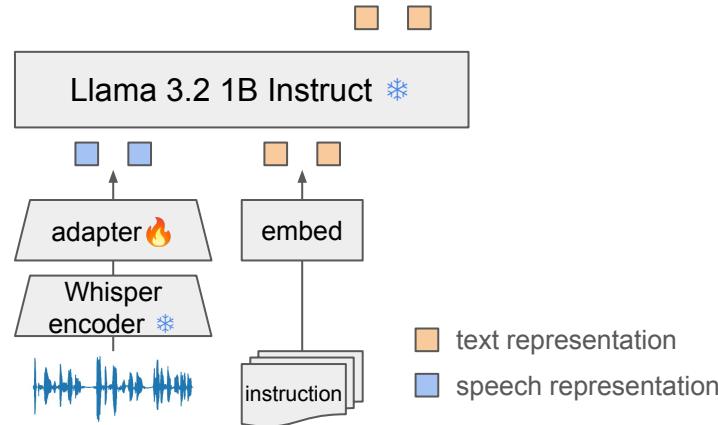
- 3.8BパラメータのPhi-4-Miniを200万時間のASRデータおよび1億のsupervised fine-tuning(SFT)サンプルで学習
- 音声・画像埋め込みをプレースホルダへ挿入し、モダリティ毎にLoRAで処理することでモダリティを干渉なく統合
- 2025年4月時点で、多言語音声認識ベンチマーク[Open ASR Leaderboard](#)においてSOTAを達成



demo: <https://colab.research.google.com/github/ryota-komatsu/slp2025/blob/main/demo1.ipynb>

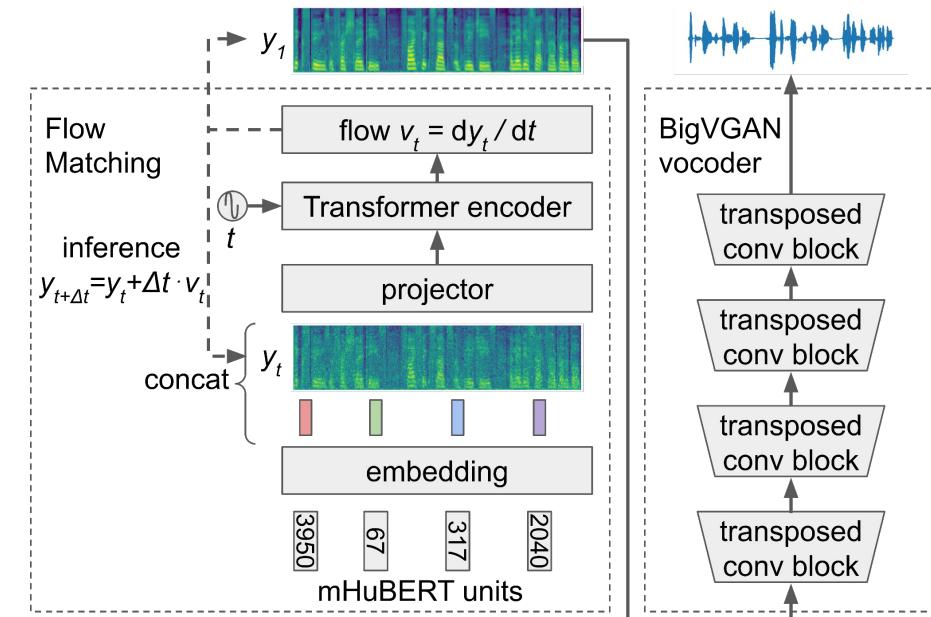
# Llama 3.2とWhisperでzero-shot instruction following

- モデル
  - Llama 3.2 1B InstructとWhisper small encoderを2層MLPからなるadapterで接続
  - LlamaおよびWhisperパラメータを凍結し, adapterのみ更新
- 学習
  - まず, LibriSpeech 960hでのASR, およびClothoでのaudio captioningによって事前学習
  - alpaca[[Taori+, GitHub'23](#)]データセットにおいて入力テキストをVITS[[Kim+, ICML'21](#)]で音声合成し, SFT

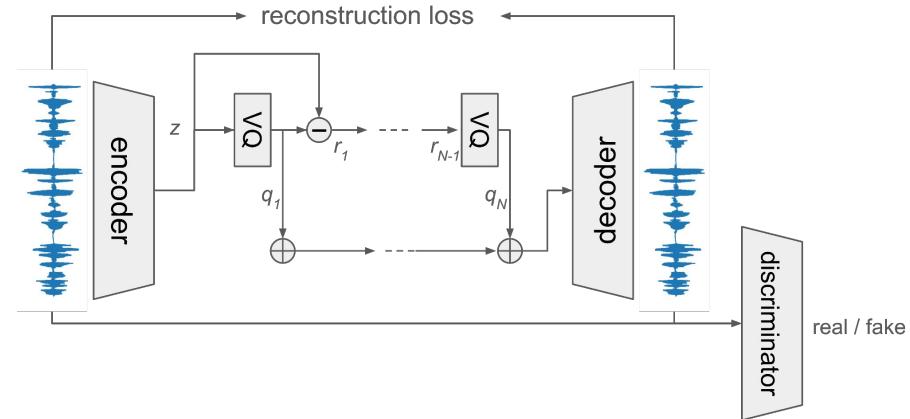


# Phonetic tokenとacoustic tokenとで再合成音声を比較①

Phonetic token: mHuBERT[[Hsu+, CVPR'23](#)]



Acoustic token: DAC[[Kumar+, NeurIPS'23](#)]



demo: [https://ryota-komatsu.github.io/speech\\_resynth/](https://ryota-komatsu.github.io/speech_resynth/)

# Phonetic tokenとacoustic tokenとで再合成音声を比較②

	Phonetic token	Acoustic token
Encoder	mHuBERT[ <a href="#">Hsu+, CVPR'23</a> ]	DAC[ <a href="#">Kumar+, NeurIPS'23</a> ]
Decoder	Flow matching[ <a href="#">Le+, NeurIPS'23</a> ] + BigVGAN vocoder[ <a href="#">Lee+, ICLR'23</a> ]	DAC
#codebooks	1	12
Codebook size	2000	1024
Bitrate [bit/s]	548	6000

- Acoustic tokenでは圧縮率は低いが、元音声を忠実に再現するため、音声合成に適する
- Phonetic tokenでは圧縮率が高く、発話内容のみ保存するため、内容理解タスクに適する

※flow matchingは非決定的であるため、推論を2回試行

demo: [https://ryota-komatsu.github.io/speech\\_resynth/](https://ryota-komatsu.github.io/speech_resynth/)

# まとめ

1. 音声テキストLLMは、1)テキスト・音声トークンの同時分布をモデル化するspeech+text LMと、2)LLMと音声エンコーダを連続表現で接続したspeech-aware text LMに分類される
2. Phoneticな表現とacousticな表現との間にはトレードオフがあり、双方を相補的に用いることで**音声・音響・音楽の統合的理**解を実現
3. Speech+text LMでは、テキスト・音声トークンを階層的あるいは交互/並列に生成することで、LLMの論理的推論能力を活用して**高精度かつ低遅延に応答生成**
4. 学習では、埋め込み空間において**音声とテキストのアライメント**をとった後、複雑なマルチタスクSFTでカリキュラム学習することでテキストでの推論能力を保持しつつ音声タスクでも高精度を実現

# 参考文献

1. Ji+, “[WavChat: A Survey of Spoken Dialogue Models](#)”, 2024, arXiv
2. Guo+, “[Recent Advances in Discrete Speech Tokens: A Review](#)”, 2025, arXiv
3. Arora+, “[On The Landscape of Spoken Language Models: A Comprehensive Survey](#)”, 2025, arXiv