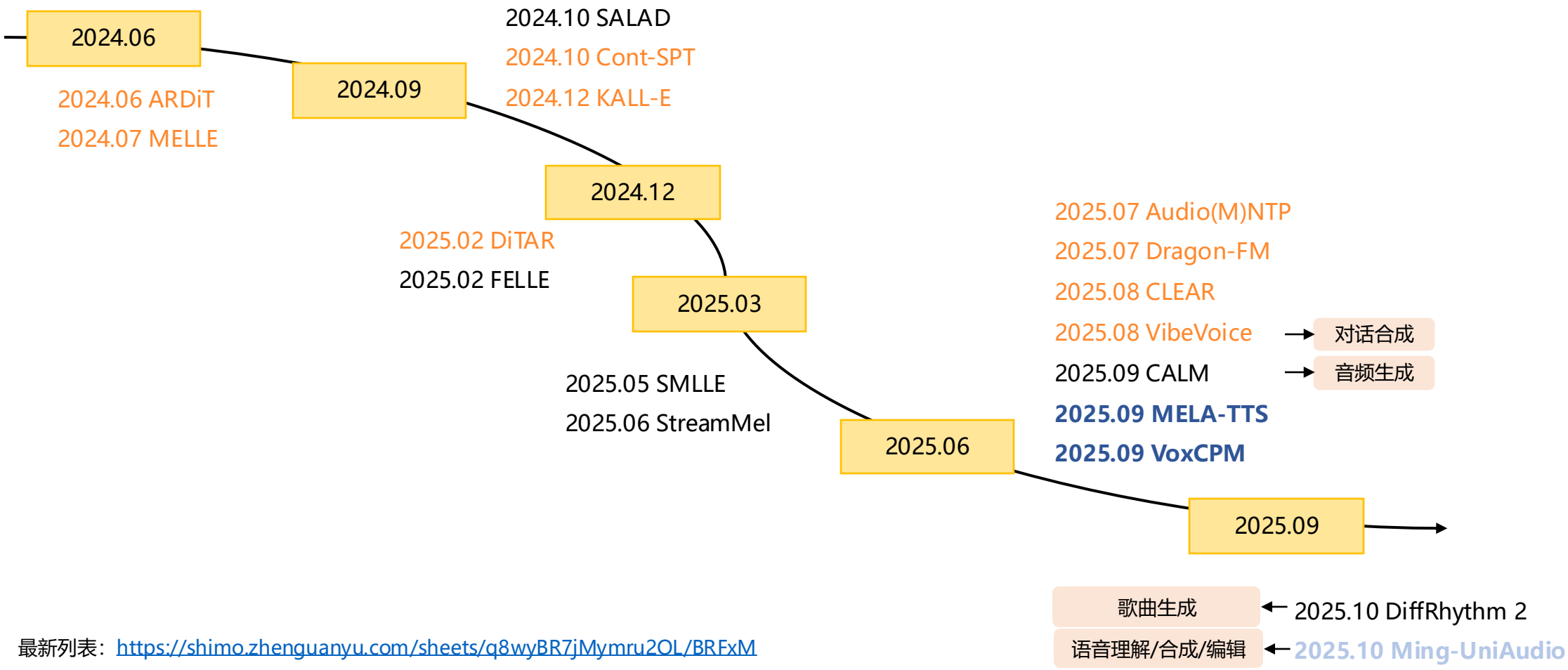
Recent Advances in Autoregressive Diffusion Models for Speech Generation

2025-11-3

Overview: Audio Models based on Autoregressive Diffusion Models



最新列表：https://shimo.zhenguanyu.com/sheets/q8wyBR7jMymru2OL/BRFxM

* 字体为橙色的表示之前有同学在大/小组会上分享过

MELA-TTS: Joint Transformer-Diffusion Model with Representation Alignment for Speech Synthesis

https://arxiv.org/pdf/2509.14784

Keyu An⋆, *Zhiyu Zhang*⋆†, *Changfeng Gao*⋆, *Yabin Li*⋆, *Zhendong Peng*⋆,
Haoxu Wang⋆, *Zhihao Du*⋆, *Han Zhao*⋆, *Zhifu Gao*⋆, *Xiangang Li*⋆

⋆ Alibaba group † National Mobile Communications Research Laboratory, Southeast University
ankeyu.aky@alibaba-inc.com, zhiyuzhang@seu.edu.cn

MELA-TTS: TTS based on Autoregressive Diffusion Models

Autoregressive Diffusion Models (ARDMs)

1. 出发点（连续表征 + 端到端建模）

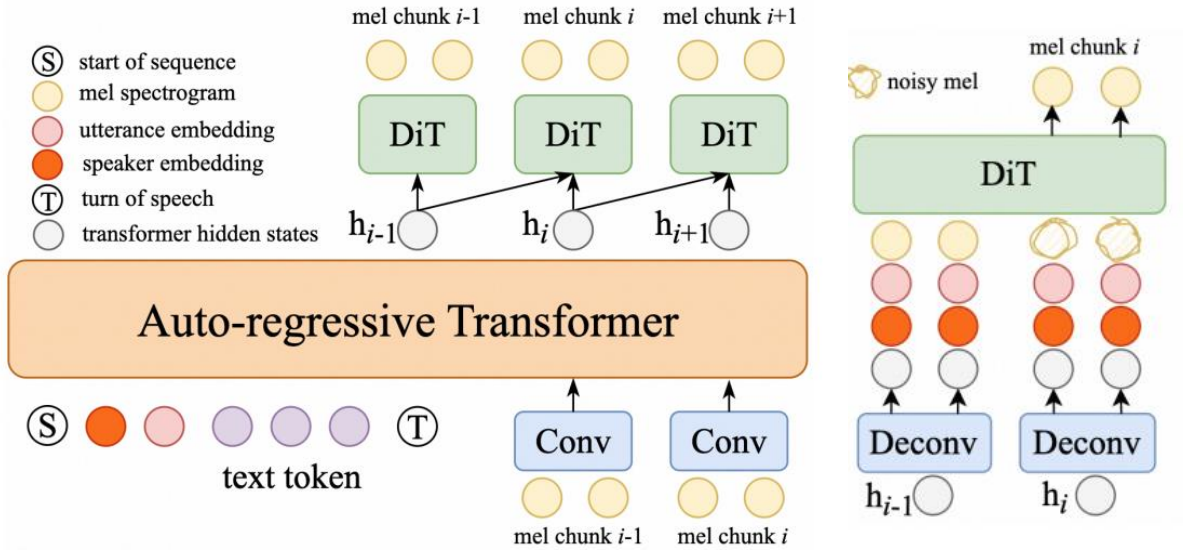
- 离散化表征导致信息损失，在恢复声学细节上难度大 → **连续表征**几乎认为是无损/极少损的压缩形式
- LLM + Flow Matching 两个模块级联的方案，存在误差累积问题 → **端到端建模**一般能够提升模型上限

2. 三要素

- 要素一：连续表征（Continuous Token）
  - 直接在**梅尔特征（50Hz）**上建模，不引入额外 WaveVAE 建模
- 要素二：模型结构（LLM with Local DiT）
  - LLM：Qwen2-0.5B (文本是 BPE tokenizer)
- 要素三：训练目标（CFM Loss）
  - 使用了 CFG，推理使用 DDIM 采样，每步采样 10 次

3. 其他设计

- **【上下采样】**梅尔 chunk 做卷积下采样，时序压缩降低 LLM 帧率
  - chunk_size=8，LLM 实际帧率是 **50/8=6.25Hz**
- **【停止预测】**LLM 每步输出的 h 上，额外增加 stop predictor（二分类）任务
- **【历史 patch】**DiT 预测当前 patch 时，会使用上一个 chunk 的信息
  - DiT 的输入长度为 8 + 8 = 16 帧

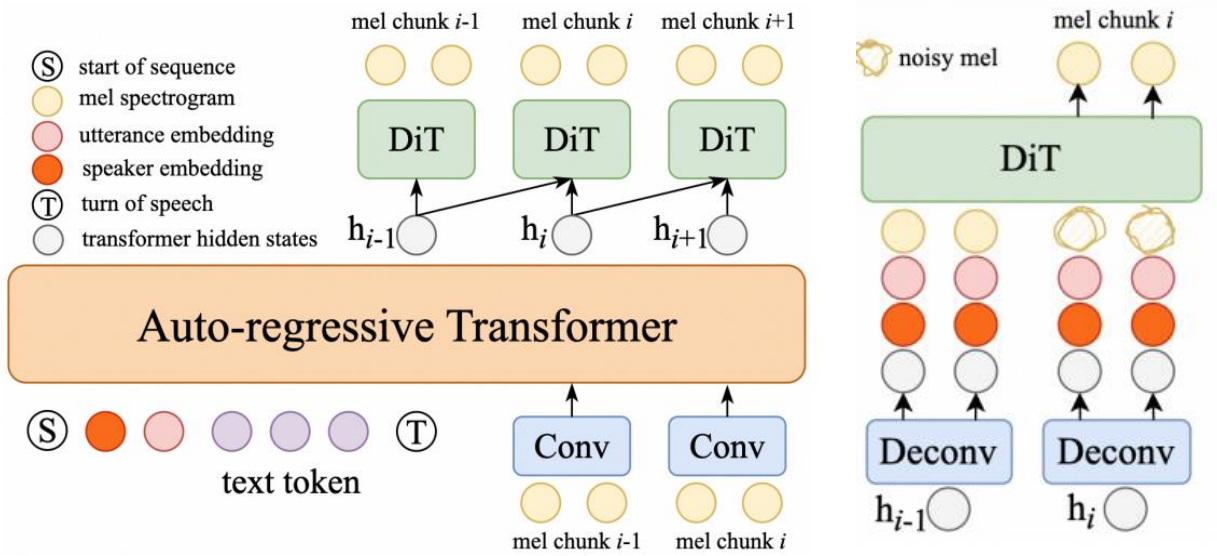


- LLM 输入始终是真实梅尔特征，不牵扯 CFM 的 noise 采样
- 可以将整个 LLM 看做一个因果 Encoder，提取特征作为 DiT 额外输入条件
- 训练阶段与普通 LLM 的 Teacher-Forcing 训练相同**（自回归架构）**

MELA-TTS: Training

训练伪代码

```
def forward(input_text, audio_waveform):
    # 1. 提取语音 latents
    latents = ExtractAudioFeature(audio_waveform)

    # 2. 将语音特征插入文本embedding序列
    multimodal_embeddings = InsertAudioIntoTextEmbeddings(input_text, latents)

    # 3. 前向通过语言模型，得到隐藏表示
    hidden_states = LLM(multimodal_embeddings)

    # 4. 从隐藏表示中提取语音条件
    flow_condition = ExtractCondition(hidden_states)

    # 5. 构造Flow Matching的输入与目标
    latent_last_patch = BuildLatentHistory(latents)
    latent_target = BuildLatentTarget(latents)

    # 6. 计算Flow Matching损失
    flow_loss = ComputeFlowLoss(flow_condition, latent_last_patch, latent_target)

    # 7. 计算停止信号损失
    stop_loss = ComputeStopLoss(hidden_states)

    # 8. 返回结果
    return { flow_loss, stop_loss }
```

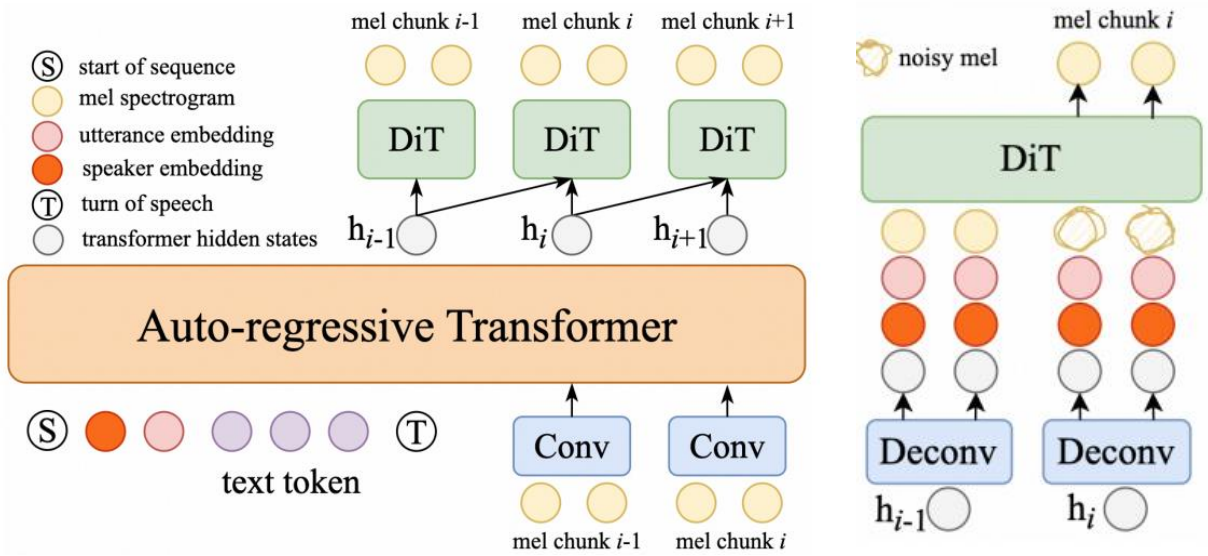


- LLM 输入始终是真实梅尔特征，不牵扯 CFM 的 noise 采样
- 可以将整个 LLM 看做一个因果 Encoder，提取特征作为 DiT 额外输入条件
- 训练阶段与普通 LLM 的 Teacher-Forcing 训练相同 **（自回归架构）**

MELA-TTS: Inference

推理伪代码

```python
def inference(text_input, audio_features):
    # 1. 编码阶段：将文本和音频特征分别编码
    text_embedding  = EncodeText(text_input)
    audio_embedding = EncodeAudio(audio_features)
    # 2. 投影到语言模型输入空间并融合
    combined_embedding = MergeTextAudio(text_embedding, audio_embedding)
    # 3. 初始化语言模型上下文与缓存
    lm_hidden, lm_cache = BaseLanguageModel(combined_embedding)
    # 4. 初始化生成条件（取音频前缀的最后一帧）
    prefix_condition = None
    generated_sequence = []

    # 5. 自回归生成循环
    for step in range(max_len):
        # （1）将语言模型隐藏状态映射到扩散模型空间
        diffusion_condition = ProjectToDiffusionSpace(lm_hidden)
        # （2）通过扩散模型生成下一个音频特征 patch
        predicted_patch = DiffusionDecoder(
            condition = diffusion_condition,
            prefix = prefix_condition,
            steps = inference_timesteps,
            guidance_scale = cfg_value
        )

        # （3）将生成的 patch 编码回语言模型输入空间
        curr_embed = EncodeGeneratedFeature(predicted_patch)
        # （4）使用 KV cache 增量更新语言模型状态
        lm_hidden = BaseLanguageModel.ForwardStep(curr_embed, cache=lm_cache)
        # （5）更新前缀与生成序列
        prefix_condition = predicted_patch
        generated_sequence.append(predicted_patch)
        # （6）判断停止条件
        stop_flag = StopHead(lm_hidden)
        if step > min_len and stop_flag == 1:
            break

    # 6. 返回完整生成结果
    return generated_sequence
```



- LLM 输入始终是真实梅尔特征，不牵扯 CFM 的 noise 采样
- 可以将整个 LLM 看做一个因果 Encoder，提取特征作为 DiT 额外输入条件
- 训练阶段与普通 LLM 的 Teacher-Forcing 训练相同**（自回归架构）**

MELA-TTS: Speaker Condition

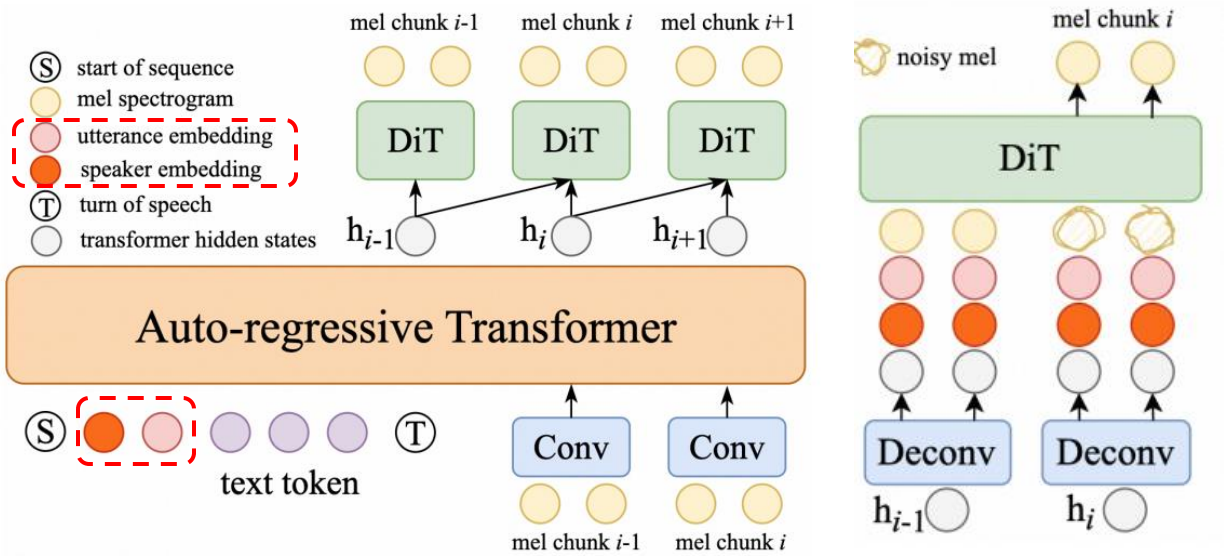
改进点一：speaker & utterance embedding

speaker embedding

- 从预训练的 speaker embedding 模型中提取

utterance embedding

- 模型结构：Transformer encoder + Pooling
  - 参数与整个模型一起更新训练
  - 相当于 Learnable Utterance/Speaker Encoder
- 训练阶段输入：随机从输入音频中截取片段
  - 随机片段，应该是为了避免语义信息泄露
- 以上两个 embedding 也作为 DiT 的输入条件



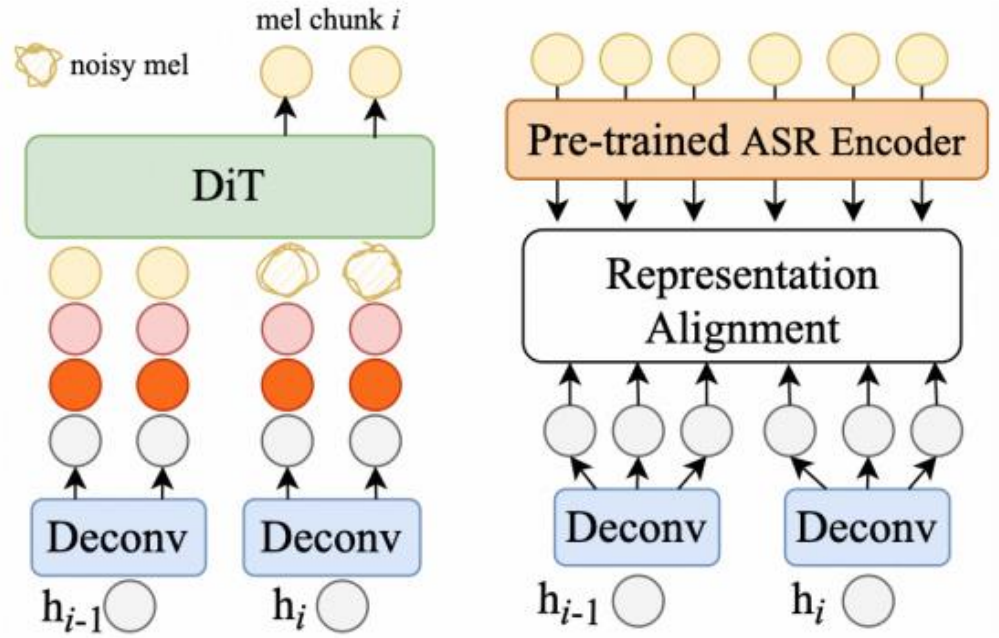
MELA-TTS: Representation Alignment

改进点二：表征对齐（Representation Alignment）

出发点

- ARDM 模型使用 AR 建模，但没有显式引导到语义空间，建模难度大
  - 现象：发音不稳定，尤其是长文本/hard文本上合成效果差
- 探索：将 LLM 输出的表征对齐到语义（semantic）空间

思路和 CosyVoice 系列类似
- LLM 更侧重负责**语义层面**的生成任务（WaveVAE 做不到）
- 语义到声学（梅尔特征）由 DiT 负责
- 相比 CosyVoice 的优点是：端到端可微分训练



表征对齐方案

- 预训练 ASR Encoder：SenseVoice-Large Encoder
  - 输入梅尔特征帧率是 100Hz，下采样到 25Hz 得到 h_{asr}
- LLM 输出 h 帧率是 6.25Hz
  - TAM（Time Alignment Module）上采样到 25Hz
- 对齐 Loss（不需要数值大小一致，只要方向尽可能一致）

$$\mathcal{L}_{\text{align}} = \text{CosineSimilarity}(\text{TAM}(\mathbf{h}), \mathbf{h_{asr}})$$

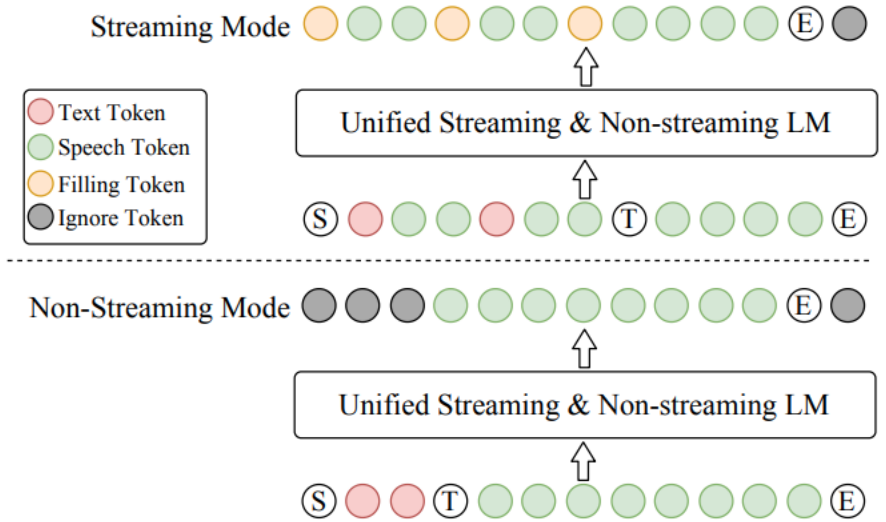
模型整体 loss $\mathcal{L} = \mathcal{L}_{\text{diff}} + \mathcal{L}_{\text{stop}} + \mathcal{L}_{\text{align}}$

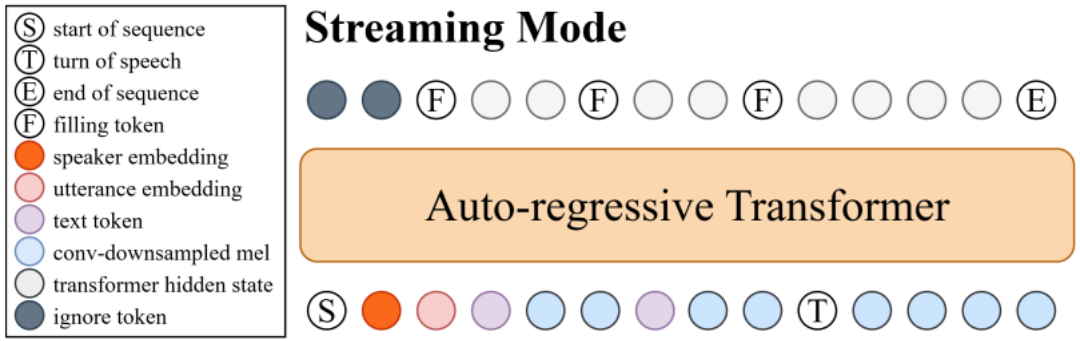
MELA-TTS: Unified Streaming and Non-Streaming Input

改进点三：单流式/双流式统一

思路：CosyVoice2 的 N:M 方案，复用到 ARDM 上

- 文本 token 与语音 token 交错排列（Interleave）
- 文本输入完后，Turn-of-Speech token 之后只输入语音 token
  - 文本的实际帧率小于语音的 6.25Hz 帧率
- 训练时两种模式同时训练，统一到同一个模型中
- Filling token 不参与训练阶段 loss 计算





MELA-TTS: Experiments

训练数据

- CosyVoice/CosyVoice2 的 17 万小时数据：中文 13 万，英文 3 万，其他 1 万小时
- SS1/SS2 代表两个不同的 speaker embedding 模型计算的音色相似度
- **音色相似度差异：**（数据基本可比的实验只有 CosyVoice 系列，MELA-TTS 音色相似度略低）
  - 论文猜测：MELA-TTS 的 DiT 输入是 Local context 级别的，而 CosyVoice 先将 token 全部输出完、再通过 DiT 推理，序列信息更全

Table 2. Zero-shot TTS performance comparison between MELA-TTS and results from literature on seed-tts-eval. † indicates that the model is trained using the same data, so the results are comparable.

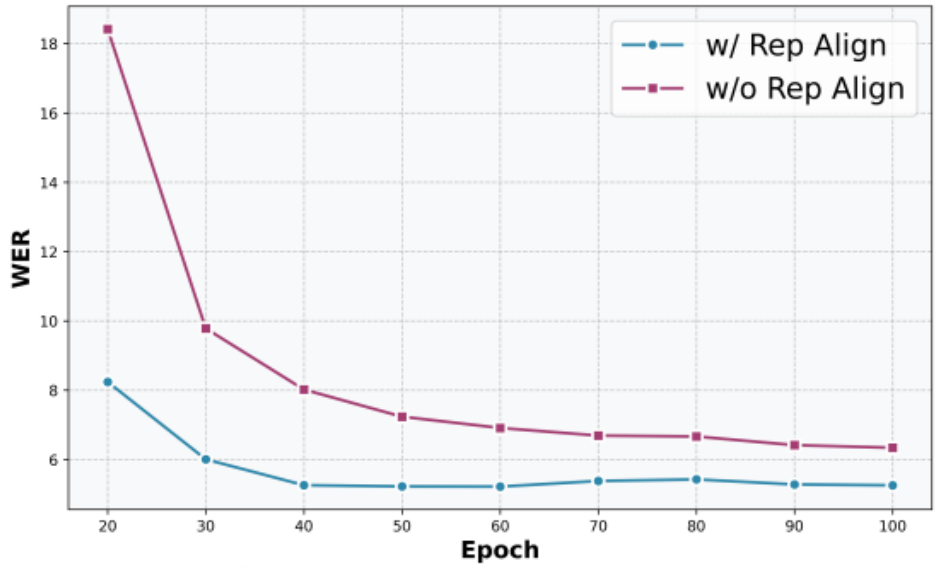
Model	*test-zh*			*test-en*			*test-hard*		
	CER ↓	SS1 ↑	SS2 ↑	WER ↓	SS1 ↑	SS2 ↑	CER ↓	SS1 ↑	SS2 ↑
Human	1.3	0.76	0.78	2.1	0.73	0.74	-	-	
Non-autoregressive Models									
F5-TTS [10]	1.6	0.74	0.80	1.8	0.65	0.74	8.7	0.71	0.76
MaskGCT [11]	2.3	0.77	0.75	2.6	0.71	0.73	10.3	0.75	0.72
Autoregressive Models									
Seed-TTS [12]	1.1	**0.80**	-	2.3	**0.76**	-	7.6	**0.78**	-
DiTAR [4]	1.0	0.75	-	**1.7**	0.74	-	-	-	-
CosyVoice [9] †	3.6	0.72	0.78	4.3	0.61	0.70	11.8	0.71	0.76
CosyVoice 2.0 [13] †	1.5	0.75	0.81	2.6	0.65	0.74	**6.8**	0.72	0.78
CosyVoice 3.0-0.5B [2] †	1.3	0.75	**0.81**	2.5	0.65	**0.75**	7.0	0.72	**0.79**
MELA-TTS									
w/o rep align †	1.2	0.74	0.79	4.0	0.60	0.68	10.9	0.72	0.78
w/ rep align †	**0.9**	0.72	0.77	2.4	0.59	0.68	7.6	0.71	0.76
streaming mode w/ rep align †	**0.9**	0.72	0.78	2.5	0.59	0.68	7.7	0.71	0.77

MELA-TTS: Experiments

消融实验：LibriTTS

Table 1. Ablation study of streaming synthesis, utterance embedding (Utt Emb), and representation alignment (Rep Align). ∗ indicates using mel-spectrogram instead of the pretrained ASR encoder output as the representation alignment target.

Exp ID	Streaming	Utt Emb	Rep Align	WER ↓	SS1 ↑	SS2 ↑
0	✗	✗	✗	6.3	0.46	0.55
1	✗	✗	✓	5.3	0.46	0.54
2	✗	✗	✓ ∗	6.7	0.41	0.48
3	✗	✓	✗	6.0	0.47	0.57
4	✗	✓	✓	5.2	**0.48**	**0.58**
5	✓	✗	✗	6.6	0.46	0.55
6	✓	✓	✓	**5.0**	**0.48**	**0.58**



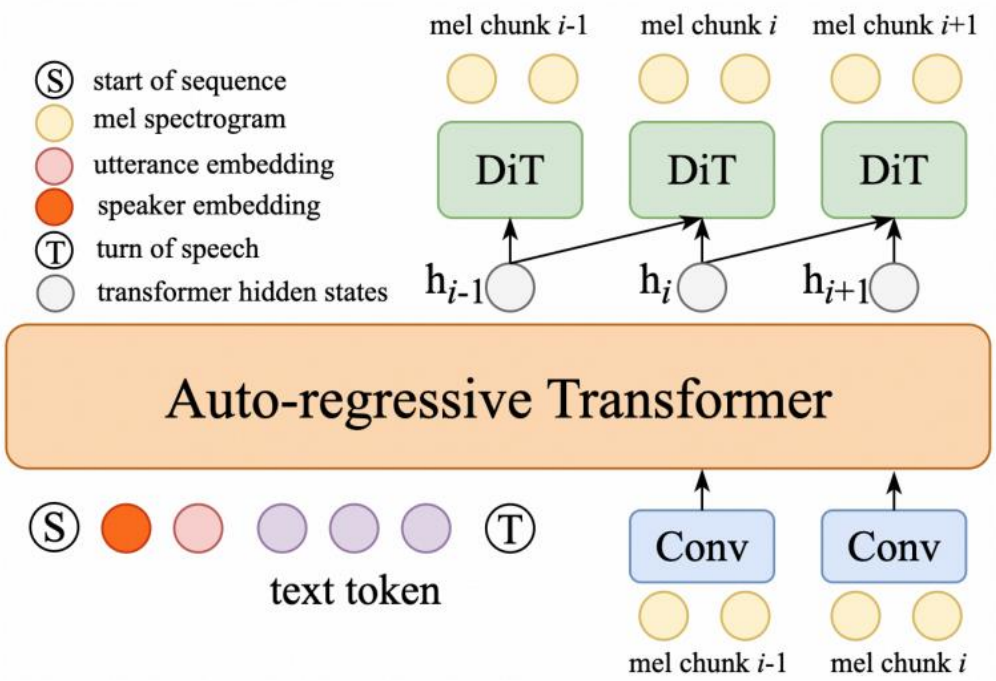
实验对比结论

- 0 vs 1: 表征对齐之后，合成发音错误率显著降低
- 1 vs 2: 表征需要对齐到语义空间而不是声学梅尔特征
- 0 vs 3: 可学习的 utt 表征，对发音准确度和音色相似度都有正向作用
  - **CosyVoice 也可以复用这个思路**
- 0 vs 5 | 4 vs 6: 流式方案在发音层面会略有变差

图示不仅训练收敛速度更快，而且最终效果会更好

MELA-TTS: Takeaways

1. pretrained 和 learnable 的 speaker embedding 可以一起用，不冲突
2. DiT 输入表征对齐到语义空间，提升 ARDM 发音准确性
    - 大小参数量级实验上都验证有效
    - 低帧率的 LLM 更适合语义层面的建模
3. 同样 17 万小时数据 + SenseVoice-Large Encoder 的不同用法
    - MELA-TTS 是与 CosyVoice 可比性最高的 ARDM 模型
4. ARDM 基于 Local DiT 的方案，目前在音色相似度上可能存在固有短板



ModelBest

OpenBMB

VoxCPM: Tokenizer-Free TTS for Context-Aware Speech Generation and True-to-Life Voice Cloning

https://arxiv.org/pdf/2509.24650

VoxCPM Team

Project: https://github.com/OpenBMB/VoxCPM/
Model: https://huggingface.co/openbmb/VoxCPM-0.5B
Demo: https://huggingface.co/spaces/openbmb/VoxCPM-Demo
Samples: https://openbmb.github.io/VoxCPM-demopage/

Background: Semantic & Acoustic Dilemma

背景

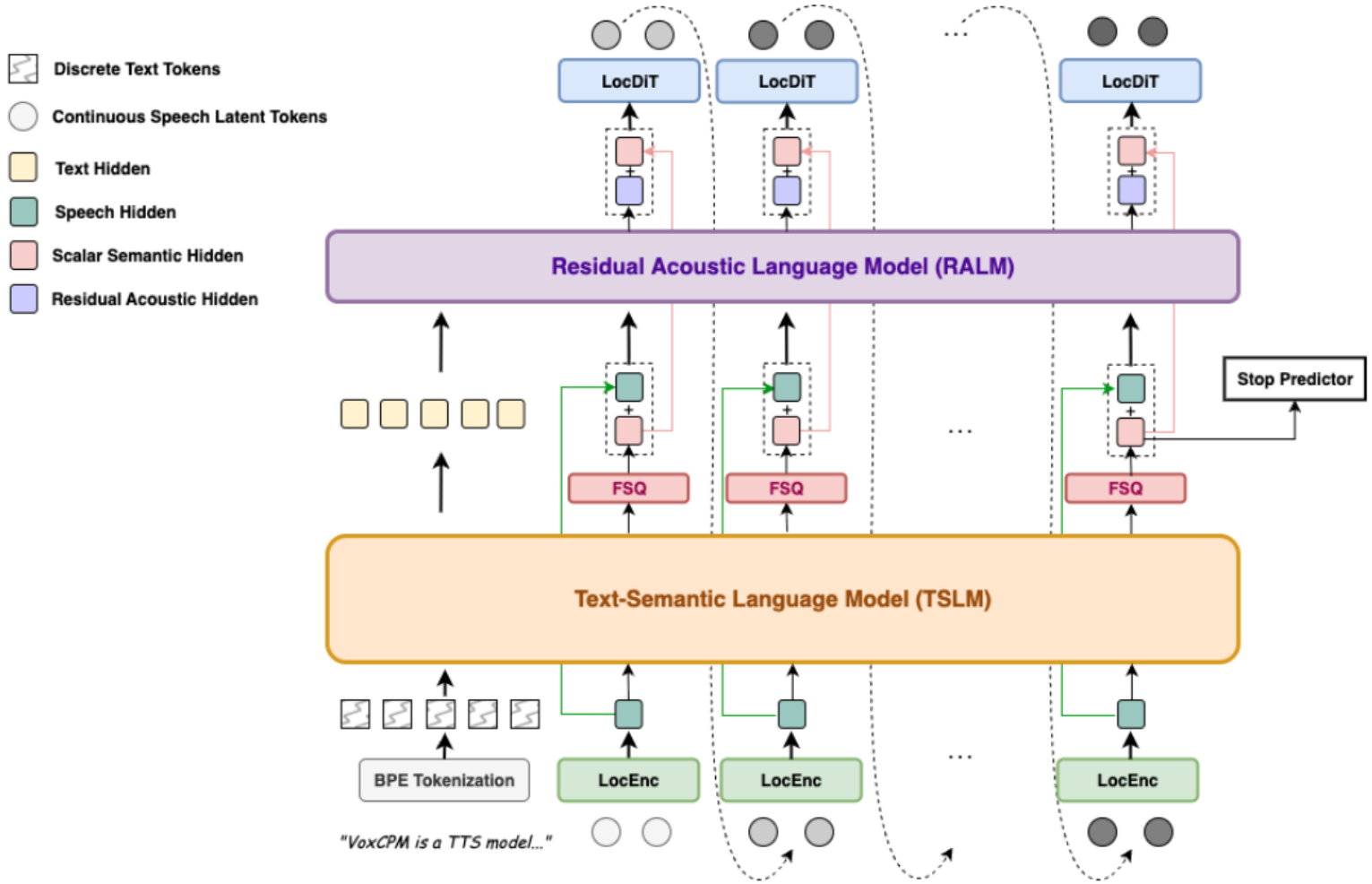
- CosyVoice/IndexTTS 系列工作：先 LLM 建模语义 token，再用 Flow Matching 还原梅尔声学特征
  - 优点：将任务拆解为语义建模（Semantic） → 声学建模（Acoustic），将**学习难度拆分到不同阶段**
  - 缺点：LLM 和 DiT 是级联系统，导致**误差累积；**模块分离不是端到端优化，限制模型表现力效果上限

- DiTAR 方案：WaveVAE 端到端恢复波形，语义和声学信息是耦合的
  - 语义和声学信息表征/建模任务过于耦合，**模型的学习难度更大** → 模型不稳定
  - 具体表现：模型发音不稳定，发现在更长的序列会有较差的表现

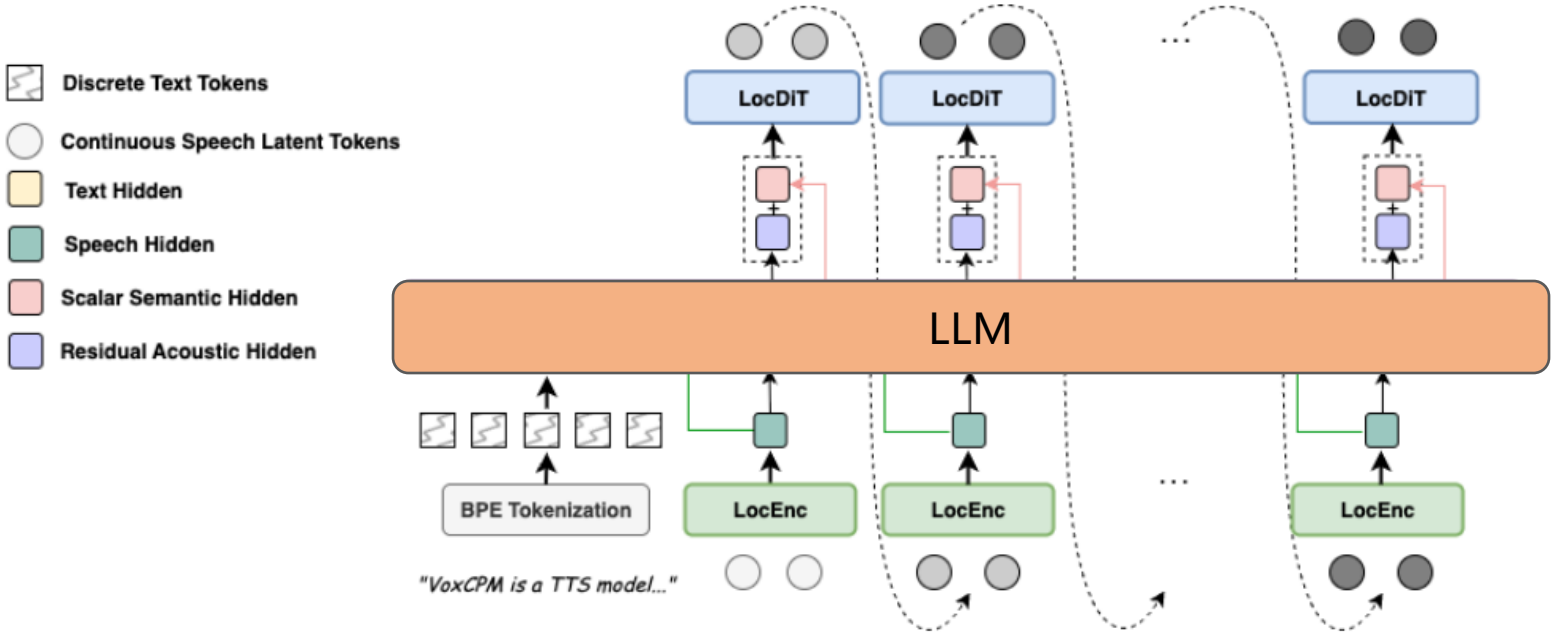
VoxCPM 出发点

- 结合以上两种方案各自的优势
  - 将 ARDM 与 Semantic → Acoustic 的层次化表征结合，降低学习难度
  - 同时保留 ARDM 的端到端建模优势

VoxCPM: Hierarchical ARDMs



VoxCPM: Hierarchical ARDMs



WaveVAE 特征 $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_M\}$

$\mathbf{T} = \{t_1, ..., t_N\}$

模型建模过程可以表示为

$$p(\mathbf{Z}|\mathbf{T}) = \prod_{i=1}^{M} p(\mathbf{z}_i|\mathbf{T}, \mathbf{Z}_{<i})$$

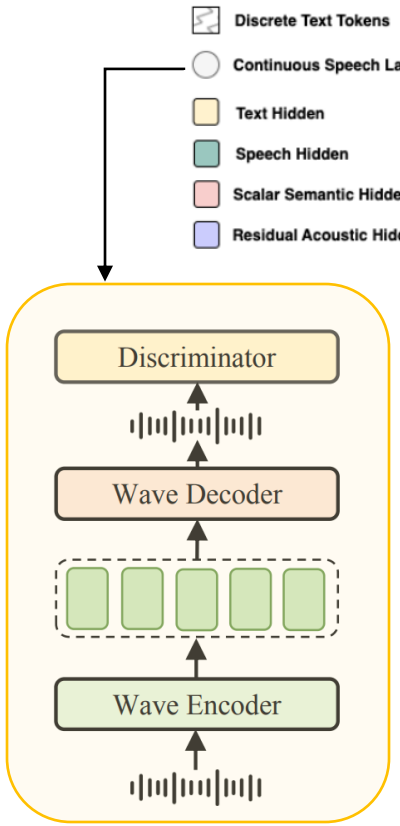
$\mathbf{z}_i \in \mathbb{R}^{P \times D}$ P 是一个 patch 所包含的帧数（图示 P=2）

VoxCPM: Hierarchical ARDMs – Semantic Modeling

TSLM（Text-to-Semantic LM）

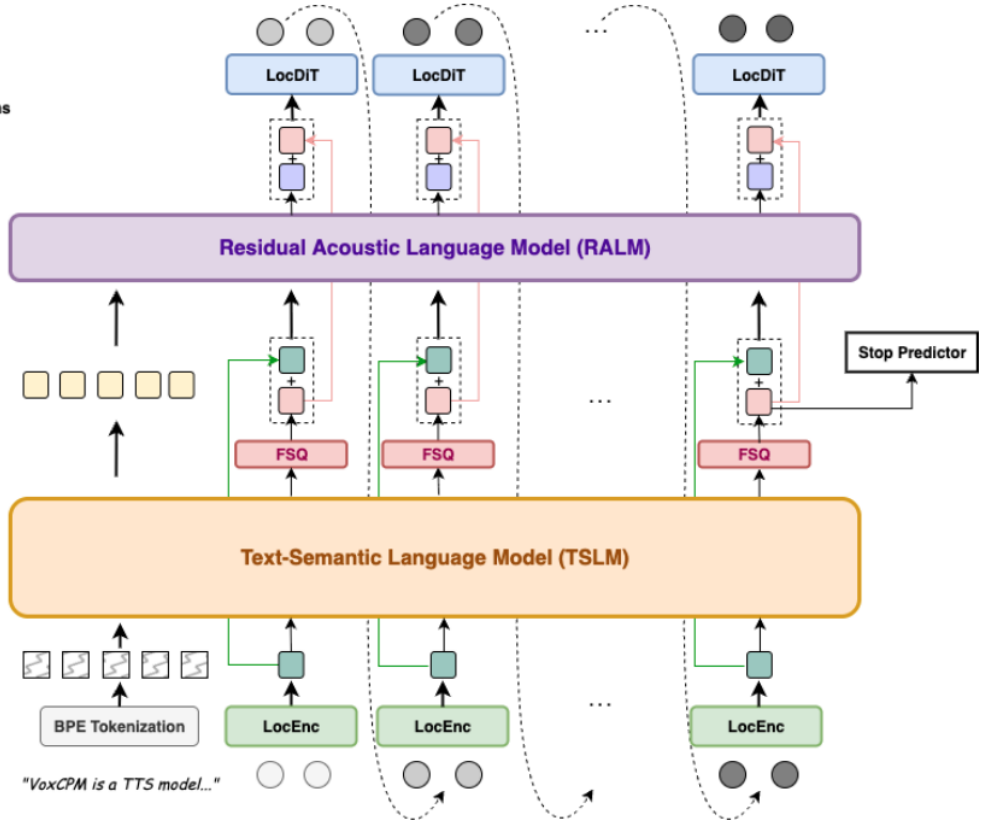
1. 连续表征：WaveVAE

- 模型结构 DAC
  - 波形 16kHz 下采样 640 → 帧率 25Hz
  - Encoder 和 Decoder 都是**因果卷积结构**
- 损失函数
  - Mel 特征的重建 Loss；波形 GAN loss
  - KL 散度 regularization (系数为 5e-5)



2. Patch Encoder：LocEnc

- 结构：4 层 Transformer + Pooling
- Patch size = 2，两个 LLM 实际是 12.5Hz



3. Stop-Predictor

$$\mathcal{L}_{\text{Stop}} = \mathbb{E}_{i\sim\text{sequence}}\left[\text{BCE}\left(s_\theta(\mathbf{h}_i^{\text{FSQ}}), \mathbb{1}[\text{token } i \text{ is the last}]\right)\right]$$

VoxCPM: Hierarchical ARDMs – Semantic Modeling

TSLM（Text-to-Semantic LM）

4. TSLM 主体结构

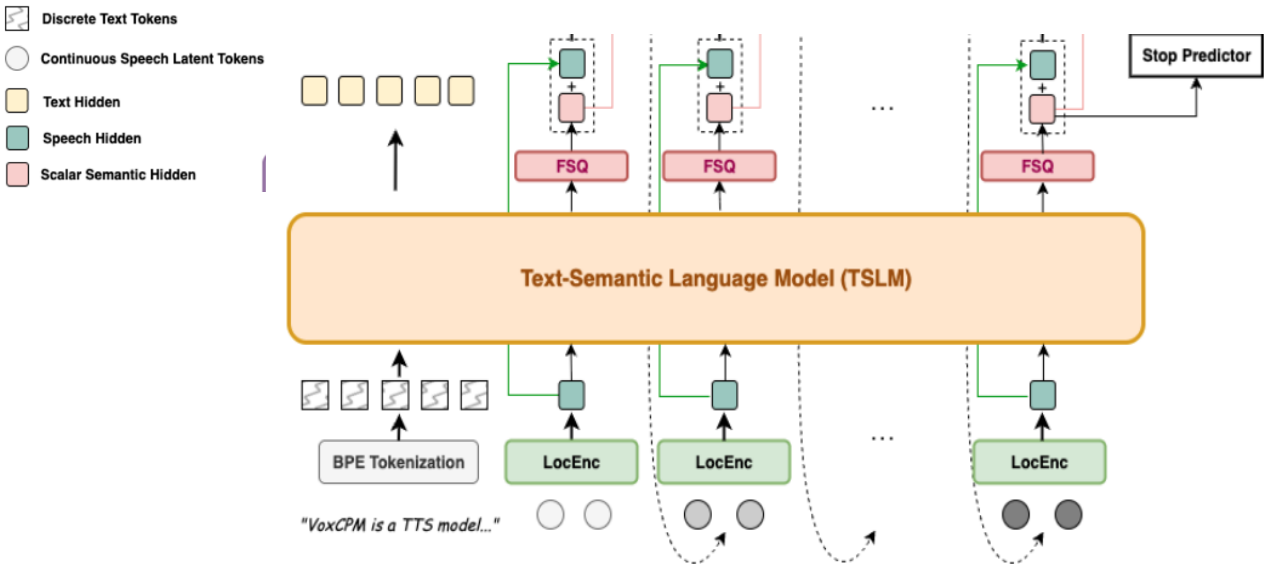
- LLM 基座使用 MiniCPM-4 **0.5B**
- 文本建模采用 BPE（中文只使用汉字，避免稀疏性问题）
- 训练阶段文本**随机替换成音素**，支持修正错误

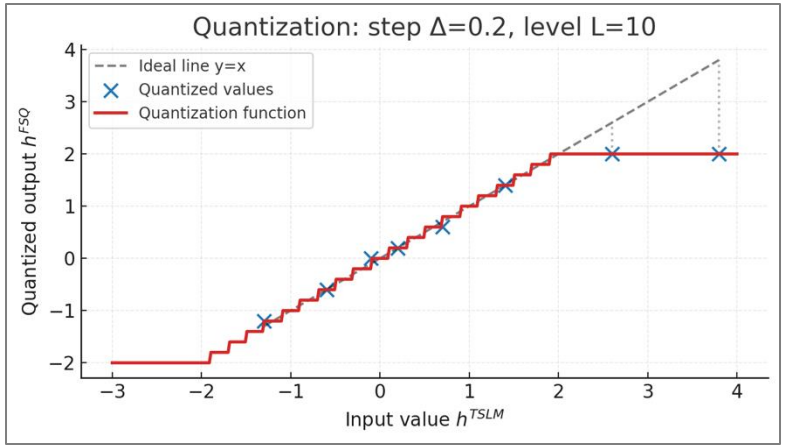
5. TSLM 输出增加 FSQ 离散化限制

- 限制只保留语义韵律（semantic-prosodic）特征
  - FSQ 采用 bottleneck，只关注关键核心信息
  - TSLM 先不建模声学细节，减轻学习难度，只需保证语义正确性

$$\mathbf{h}_{i,j}^{\text{FSQ}} = \Delta \cdot \text{clip}\left(\text{round}\left(\frac{\mathbf{h}_{i,j}^{\text{TSLM}}}{\Delta}\right), -L, L\right)$$

- 采用 **STE (Straight Through Estimator)**，保证模型整体可微分训练





VoxCPM: Hierarchical ARDMs – Acoustic Modeling

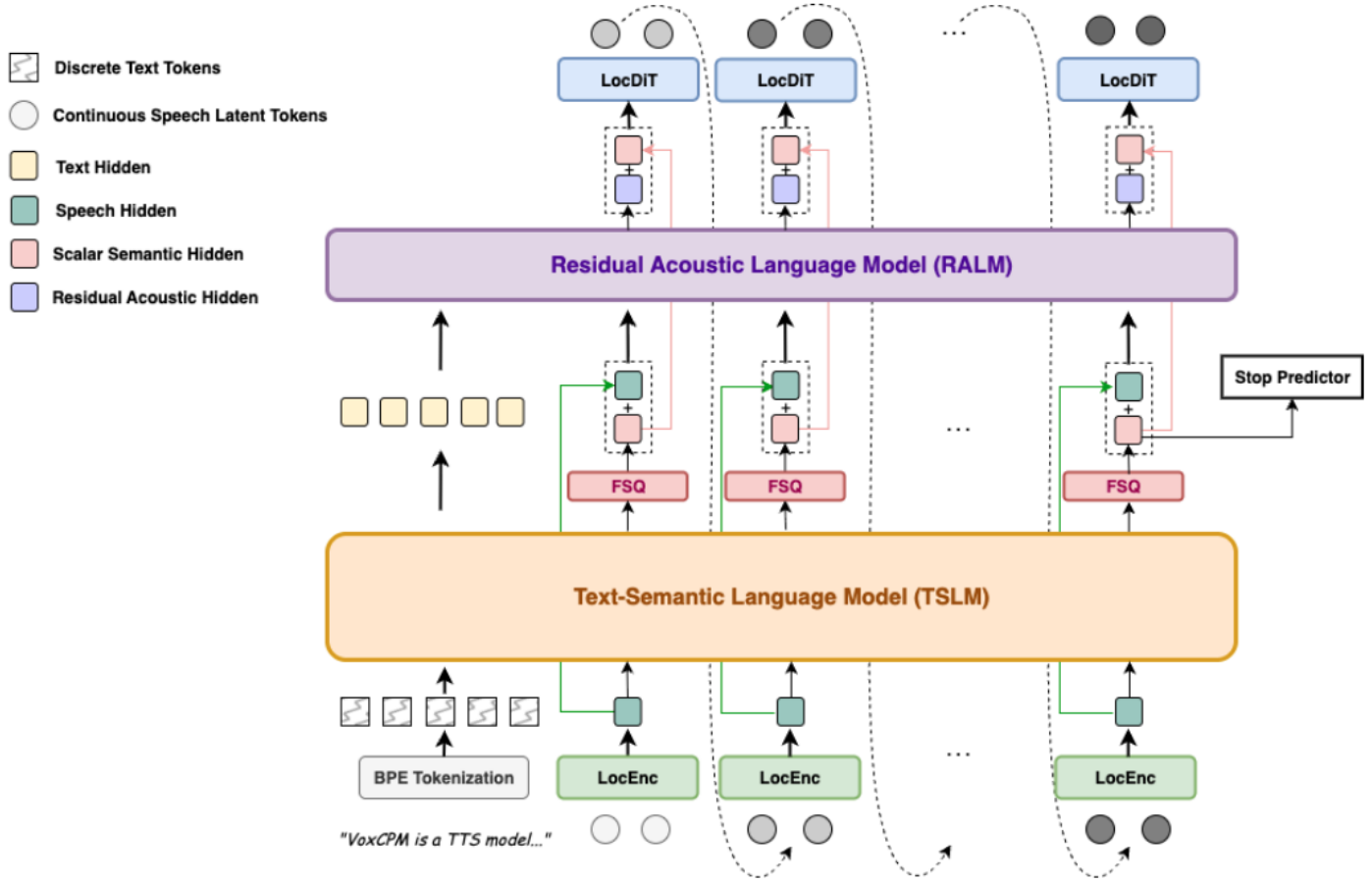
RALM（Residual Acoustic LM）

目标：补充建模 TSLM 之外的声学细节特征

$$\mathbf{E}_{<i} = \mathrm{LocEnc}(\mathbf{Z}_{<i})$$

$$\mathbf{h}_i^{\text{residual}} = \mathrm{RALM}(\mathbf{H}_{\text{text}}^{\text{TSLM}}, \mathbf{H}_{<i}^{\text{FSQ}} \oplus \mathbf{E}_{<i})$$

$$\mathbf{h}_i^{\text{final}} = \underbrace{\mathrm{FSQ}(\mathrm{TSLM}(\mathbf{T}, \mathbf{E}_{<i}))}_{\text{stable skeleton}} + \underbrace{\mathrm{RALM}(\cdot)}_{\text{residual details}}$$



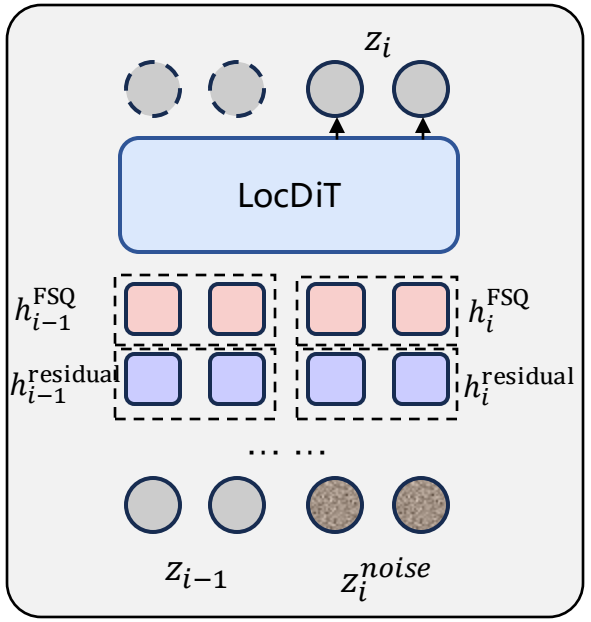
VoxCPM: Hierarchical ARDMs – LocDiT

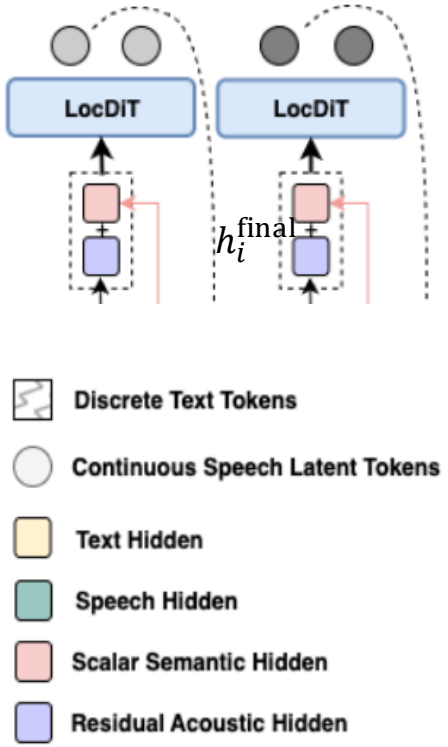
LocDiT

$$\mathbf{h}_i^{\text{final}} = \mathbf{h}_i^{\text{FSQ}} + \mathbf{h}_i^{\text{residual}}$$

$$\mathbf{z}_i \sim \text{LocDiT}(\mathbf{h}_i^{\text{final}})$$

- 输入特征：离散化的语义表征+连续的残差声学表征
- 只依赖于前一个 patch 的 z_{i-1}作为额外的 context





☆ **整个模型是端到端训练的**

$$\mathcal{L} = \mathcal{L}_{\text{FM}} + \lambda \mathcal{L}_{\text{Stop}}$$

以上关于 semantic/acoustic 的层次化定义，只是对各模块功能的一种直观解释，模型实际是端到端的

VoxCPM: Experiments

训练数据信息

- 小规模实验：Emilia 开源的 10 万小时中英文数据
- 大规模实验：中英文共计 180 万小时数据，包含：有声书、播客、访谈、广播剧等

SeedTTS 测试集

Table 3: Performance on Seed-TTS-eval Benchmark

Model	Params	Open-Source	EN		ZH		Hard	
			WER ↓	SIM ↑	CER ↓	SIM ↑	CER ↓	SIM ↑
MegaTTS3 (Jiang et al., 2025)	0.5B	✗	2.79	77.1	1.52	79.0	-	-
DiTAR (Jia et al., 2025)	0.6B	✗	1.69	73.5	1.02	75.3	-	-
CosyVoice3 (Du et al., 2025)	0.5B	✗	2.02	71.8	1.16	78.0	6.08	75.8
CosyVoice3 (Du et al., 2025)	1.5B	✗	2.22	72.0	1.12	78.1	5.83	75.8
Seed-TTS (Anastassiou et al., 2024)	-	✗	2.25	76.2	1.12	79.6	7.59	77.6
MiniMax-Speech (Zhang et al., 2025)	-	✗	1.65	69.2	0.83	78.3	-	-
F5-TTS (Chen et al., 2024)	0.3B	✓	2.00	67.0	1.53	76.0	8.67	71.3
MaskGCT (Wang et al.)		✓	2.62	71.7	2.27	**77.4**	-	-
CosyVoice (Du et al., 2024a)	0.3B	✓	4.29	60.9	3.63	72.3	11.75	70.9
CosyVoice2 (Du et al., 2024b)	0.5B	✓	3.09	65.9	1.38	75.7	**6.83**	72.4
SparkTTS (Wang et al., 2025b)	0.5B	✓	3.14	57.3	1.54	66.0	-	-
FireRedTTS (Guo et al., 2024)	0.5B	✓	3.82	46.0	1.51	63.5	17.45	62.1
FireRedTTS-2 (Xie et al., 2025)		✓	1.95	66.5	1.14	73.6	-	-
Qwen2.5-Omni (Xu et al., 2025)	7B	✓	2.72	63.2	1.70	75.2	7.97	74.7
OpenAudio-s1-mini (OpenAudio, 2024)	0.5B	✓	1.94	55.0	1.18	68.5	23.37	64.3
IndexTTS 2 (Zhou et al., 2025)	1.5B	✓	2.23	70.6	1.03	76.5	7.12	**75.5**
VibeVoice (Peng et al., 2025)	1.5B	✓	3.04	68.9	1.16	74.4	-	-
HiggsAudio-v2 (BosonAI, 2025)	3B	✓	2.44	67.7	1.50	74.0	55.07	65.6
VoxCPM-Emilia	0.5B	✓	2.34	68.1	1.11	74.0	12.46	69.8
VoxCPM	0.5B	✓	**1.85**	**72.9**	**0.93**	77.2	8.87	73.0

VoxCPM: Experiments

C3-Eval 测试集

关注：hard 测试集上的音色相似度客观指标，比 CosyVoice2/3 差不少（原因可能和 MELA-TTS 的 LocDiT 类似）

Table 4: Performance on CV3-eval Benchmark. *denotes close-sourced systems.

Model	CV3-EVAL		CV3-Hard-ZH			CV3-Hard-EN		
	ZH-CER ↓	EN-WER ↓	CER ↓	SIM ↑	DNSMOS↑	WER ↓	SIM ↑	DNSMOS↑
F5-TTS	5.47	8.90	-	-	-	-	-	-
SparkTTS	5.15	11.0	-	-	-	-	-	-
GPT-Sovits	7.34	12.5	-	-	-	-	-	-
CosyVoice2	4.08	6.32	12.58	72.6	**3.81**	11.96	66.7	**3.95**
OpenAudio-s1-mini	4.00	5.54	18.1	58.2	3.77	12.4	55.7	3.89
IndexTTS2	3.58	4.45	12.8	**74.6**	3.65	8.78	**74.5**	3.80
HiggsAudio-v2	9.54	7.89	41.0	60.2	3.39	10.3	61.8	3.68
CosyVoice3-0.5B*	3.89	5.24	14.15	78.6	3.75	9.04	75.9	3.92
CosyVoice3-1.5B*	3.91	4.99	9.77	78.5	3.79	10.55	76.1	3.95
VoxCPM-Emilia	4.47	5.23	22.2	62.6	3.47	10.00	62.6	3.68
VoxCPM	**3.40**	**4.04**	**12.9**	66.1	3.59	**7.89**	64.3	3.74

Table 5: Subjective Evaluations in terms of Naturalness and Speaker Similarity.

Model	ZH		EN	
	N-MOS	S-MOS	N-MOS	S-MOS
MaskGCT	3.20 ± 0.11	3.77 ± 0.11	3.84 ± 0.11	4.00 ± 0.10
CosyVoice 2	3.38 ± 0.12	4.01 ± 0.10	$\mathbf{4.14 \pm 0.09}$	3.97 ± 0.10
IndexTTS 2	$\mathbf{4.25 \pm 0.09}$	4.05 ± 0.09	4.03 ± 0.10	4.16 ± 0.09
VoxCPM-Emilia	3.79 ± 0.12	3.99 ± 0.11	3.91 ± 0.10	4.10 ± 0.09
VoxCPM	4.10 ± 0.10	$\mathbf{4.11 \pm 0.10}$	4.11 ± 0.09	$\mathbf{4.18 \pm 0.09}$

主观评测

VoxCPM: Experiments

消融实验 – 模型结构设计

- ②③ TSLM → DiT：与 DiTAR 类似
- ④ RALM 没有绿色信息，更难恢复出残差的声学细节
- ⑤ 直接用 FSQ 输出作为 LocDiT 的输入
  - 限制语义表征是有必要的，但是音色相似度也是最差的

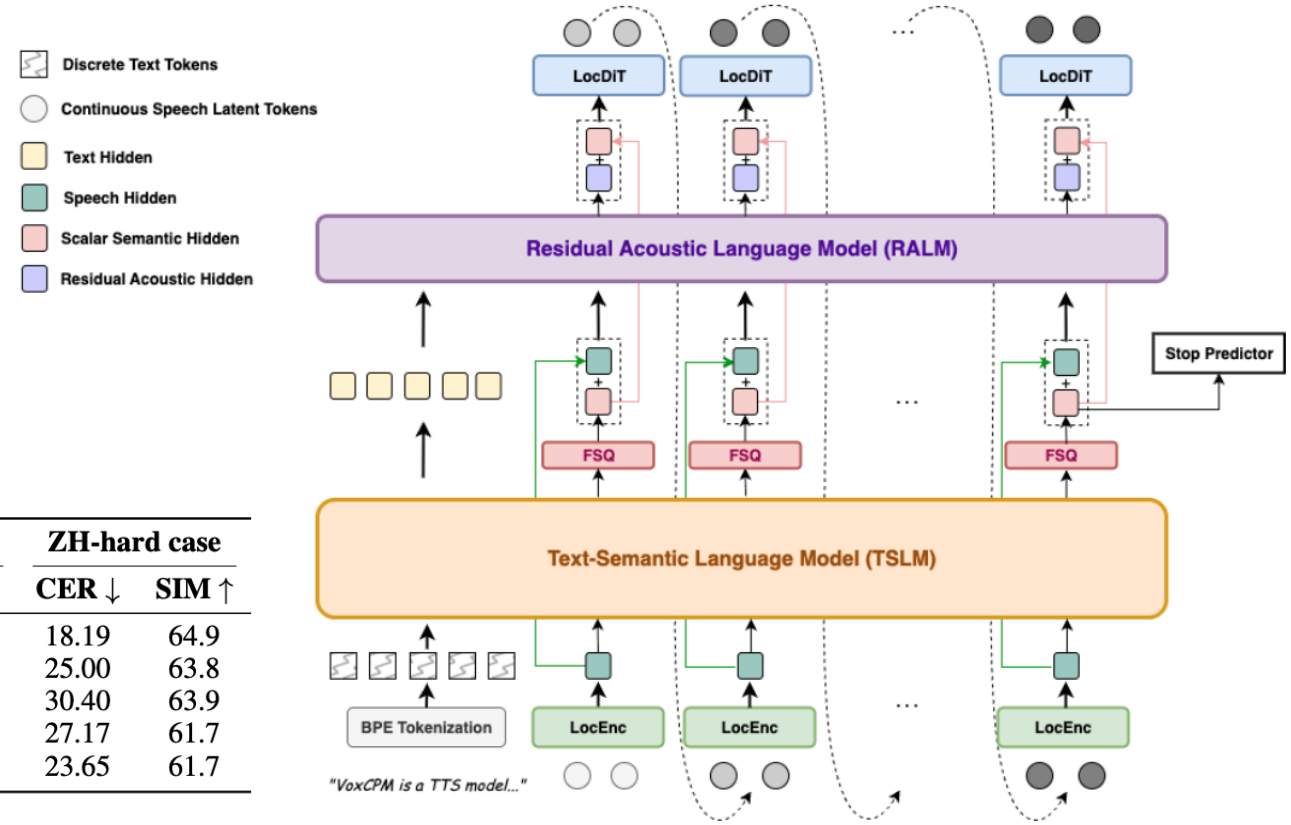


Table 7: Ablation Studies about core architecture designs.

	Model Setting	EN		ZH		ZH-hard case	
		WER ↓	SIM ↑	CER ↓	SIM ↑	CER ↓	SIM ↑
①	**default setting**	2.98	62.6	1.77	70.4	18.19	64.9
②	w/o RALM: TSLM (24 layers) → LocDiT	4.34	61.8	3.05	69.4	25.00	63.8
③	w/o RALM: TSLM (30 layers) → LocDiT	5.35	62.6	3.46	69.8	30.40	63.9
④	w/o $E_{<i}$ in RALM: TSLM → ALM → LocDiT	4.91	60.9	4.94	68.1	27.17	61.7
⑤	w/o h^{residual} in condition: TSLM → FSQ → LocDiT	3.86	58.3	3.05	67.6	23.65	61.7

VoxCPM: Experiments

消融实验 – FSQ 参数

- s 表示将数值表示范围离散化的个数
- d 代表 FSQ 的 embedding 维度
- 缺点：模型效果对模型超参数设计有点敏感

Table 6: FSQ dimension selection study on the Emilia dataset. *Note:* The 256-dim was selected for the final VoxCPM configuration, with the understanding that larger training datasets needs more powerful modeling capabilities.

Model Setting	EN		ZH		ZH-hard case	
	WER ↓	SIM ↑	CER ↓	SIM ↑	CER ↓	SIM ↑
w FSQ: d4s9	5.18	59.3	4.05	68.0	19.55	62.3
w FSQ: d16s9	3.22	60.4	1.87	70.5	**14.42**	**66.2**
w FSQ: d64s9	3.22	61.1	2.14	69.8	17.48	65.1
w FSQ: d128s9	3.43	62.2	**1.67**	**70.7**	16.76	65.7
w FSQ: d256s9	**2.98**	**62.6**	1.77	70.4	18.19	64.9
w FSQ: d1024s9	3.07	62.0	2.38	69.8	20.38	64.7
w/o FSQ: d1024s∞	3.67	62.1	2.30	69.6	24.92	63.5

消融实验 – CFG

- 没有 CFG 时，推理效果**很差**
- CFG 参数在 1.5-2.0 最优（WER/SIM 一致更好）

Table 9: Effect of LM guidance on LocDiT, tested with **VoxCPM**.

CFG Value	EN		ZH		ZH-hard case	
	WER ↓	SIM ↑	CER ↓	SIM ↑	CER ↓	SIM ↑
1.0 (w/o CFG)	16.32	55.1	14.47	61.5	56.87	43.0
1.5	1.86	72.1	1.16	77.0	9.60	73.9
2.0	1.85	72.9	0.93	77.2	8.87	73.0
3.0	2.16	71.4	1.12	74.7	13.22	65.0
5.0	12.78	60.7	17.23	59.4	48.46	39.9

VoxCPM: Experiments

模型训练技巧

Warmup-Stable-Decay (WSD)

- MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies: https://arxiv.org/pdf/2404.06395

$$WSD(T;s) = \begin{cases} \frac{s}{W}\eta, & s < W \\ \eta, & W < s < T \\ f(s-T)\eta, & T < s < S \end{cases}$$

$$0 < f(s-T) \leq 1$$

单调递减函数，论文用的指数衰减

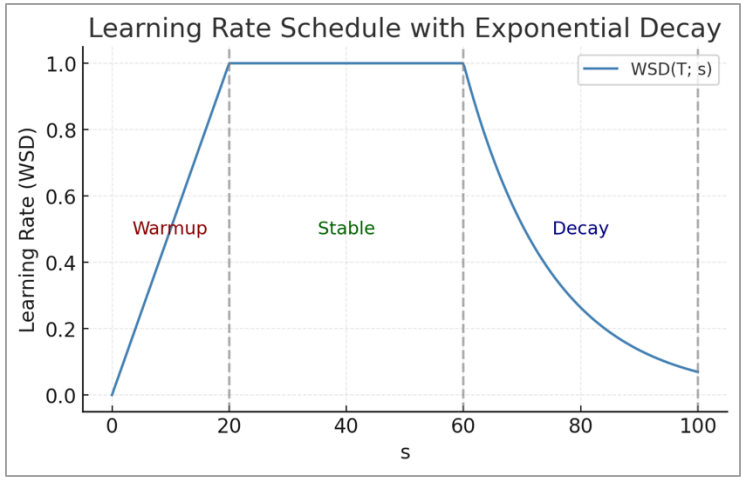


Table 2: Training configurations for VoxCPM variants.

Model	Phase	Learning Rate	Tokens/Batch	Iterations	GPUs
VoxCPM	Stable	1×10^{-4}	4,096	400K	40 × H100
VoxCPM	Decay	$1 \times 10^{-4} \rightarrow 5 \times 10^{-6}$	8,192	100K	40 × H100
VoxCPM-Emilia	Stable	1×10^{-4}	4,096	150K	24 × H100
VoxCPM-Emilia	Decay	$1 \times 10^{-4} \rightarrow 5 \times 10^{-6}$	8,192	50K	24 × H100
VoxCPM-ablation	Stable	1×10^{-4}	4,096	200K	8 × H100

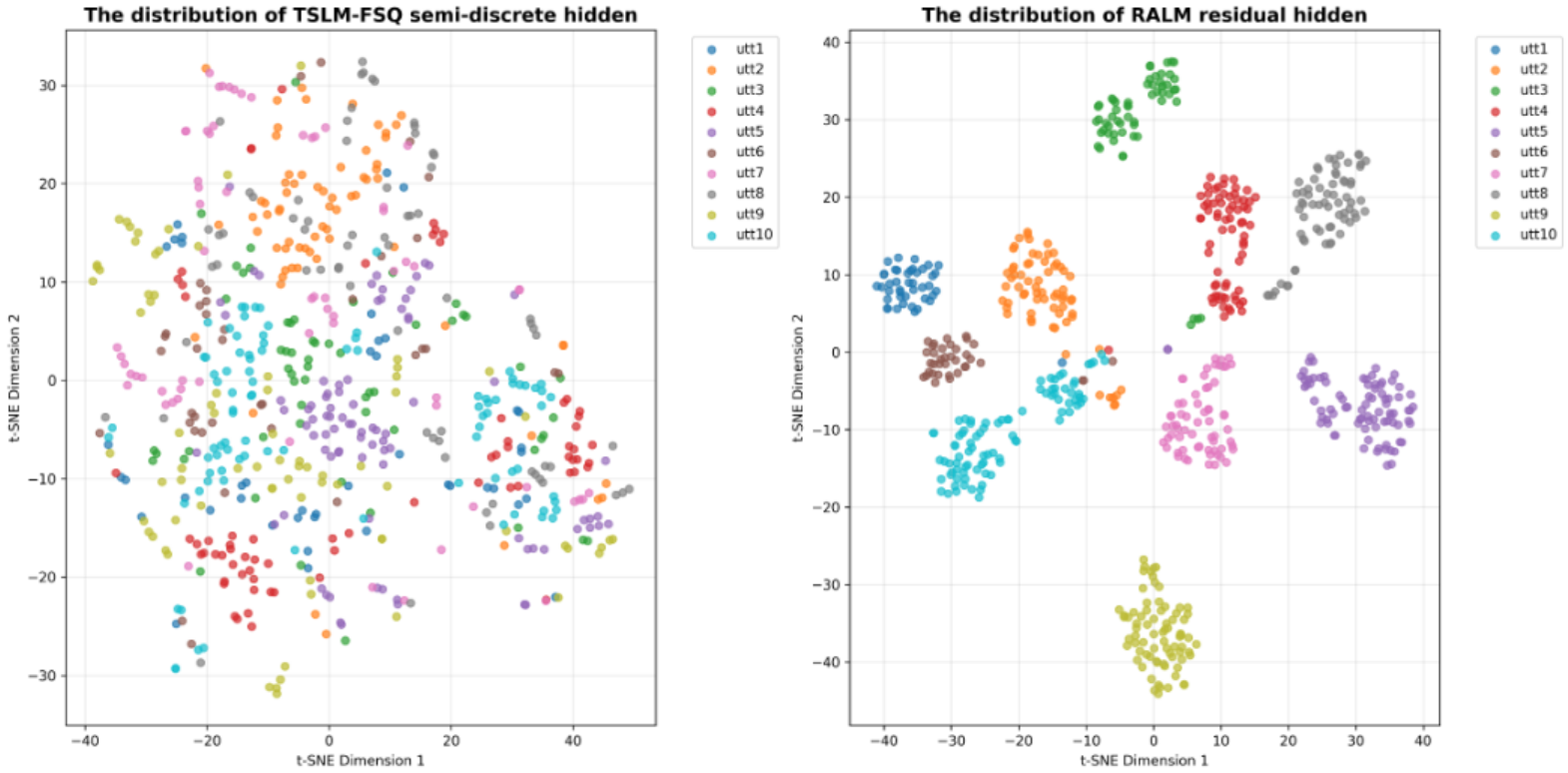
Table 8: Performance across training phases.

Phase	EN		ZH		ZH-Hard Case	
	WER ↓	SIM ↑	CER ↓	SIM ↑	CER ↓	SIM ↑
Stable	2.05	69.7	0.99	75.1	13.22	68.6
Decay	1.85	72.9	0.93	77.2	8.87	73.0

VoxCPM: Hierarchical Representations Visualization

可视化配置一：10条 prompt 音频，来自不同的 speaker

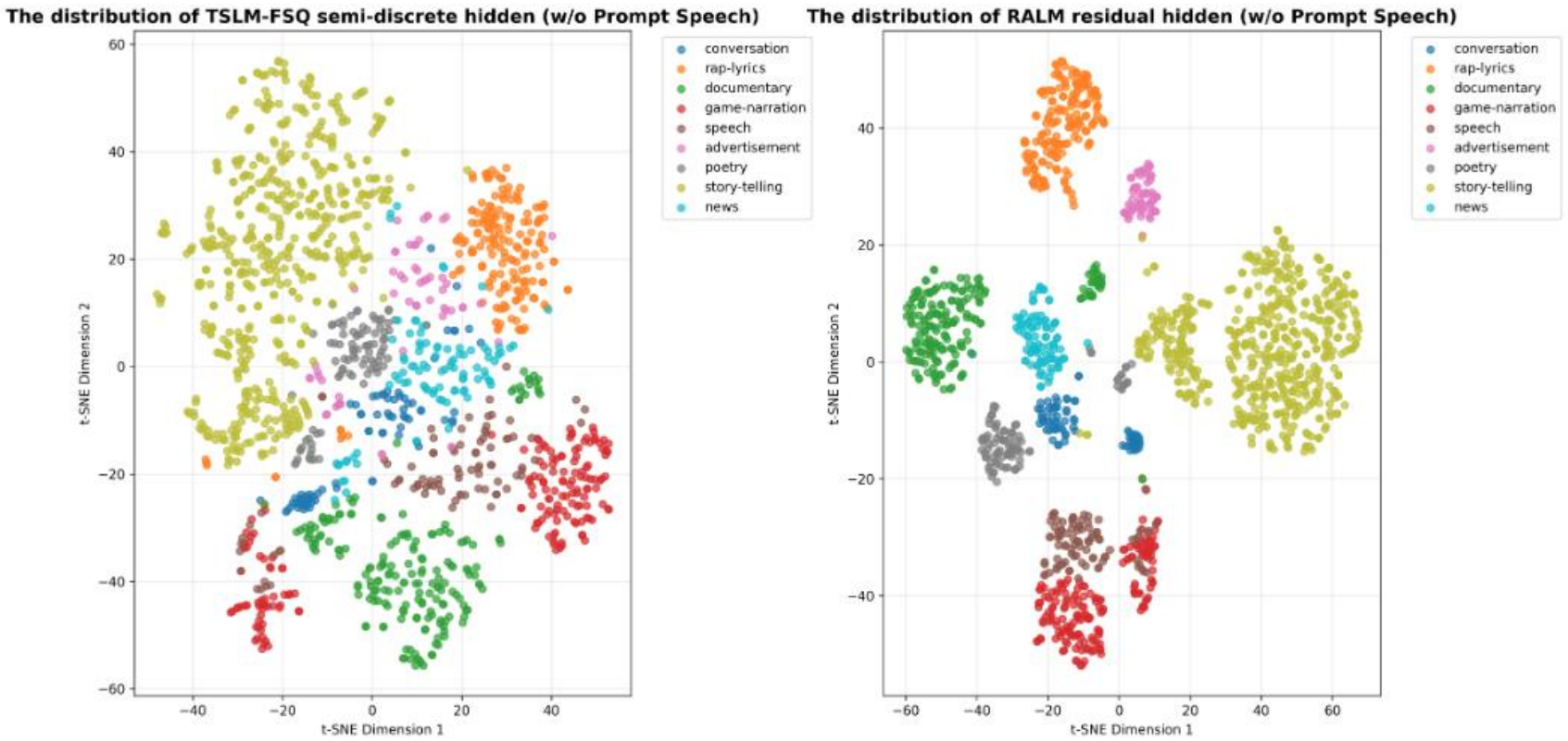
现象一：TSLM embedding 混杂在一起，但RALM 按照 speaker 明显聚类在一起（声学信息）



VoxCPM: Hierarchical Representations Visualization

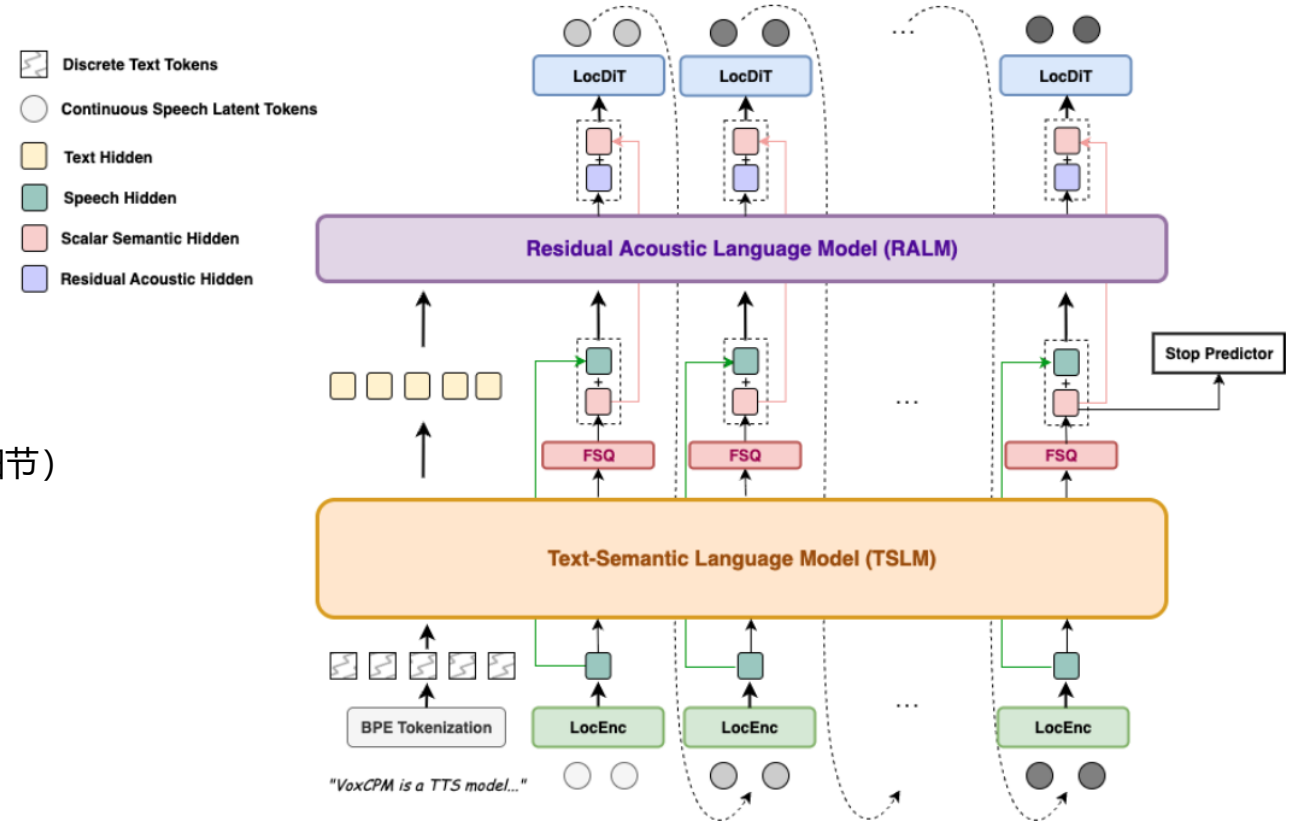
可视化配置二：不使用 prompt，推理不同风格领域的文本（不同语义韵律类别）

现象二：TSLM/RALM 均体现出明显的聚集性，RALM 比 TSLM 的类间差距更明显



VoxCPM: Takeaways

Demo: https://openbmb.github.io/VoxCPM-demopage/



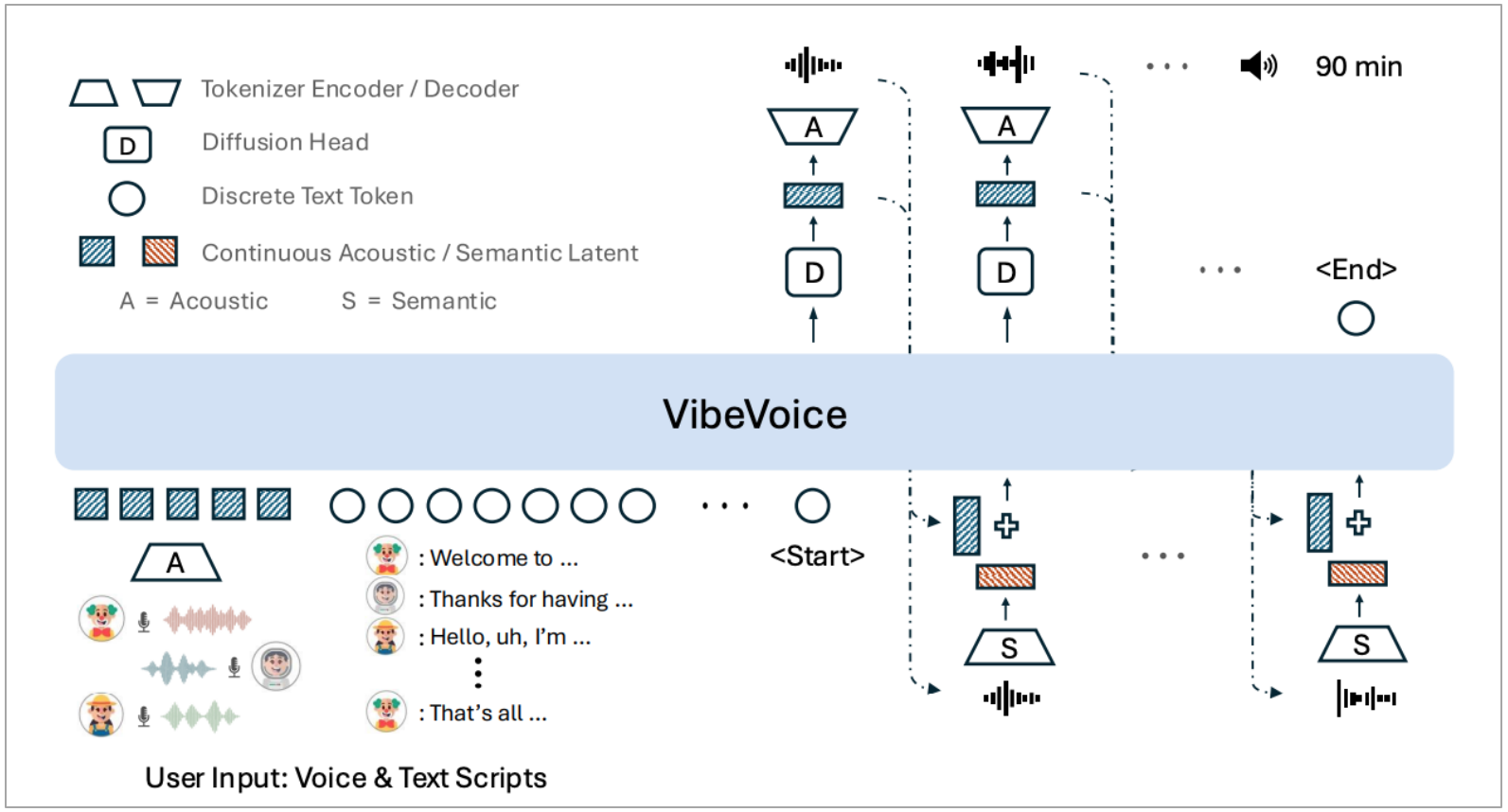
1. 引导模型按照语义/声学进行隐式的分层级建模

- FSQ bottleneck：引导模型有效保留语义/韵律层面的信息
- 远程残差：除了语义表征之外的信息，被吸收到残差中（声学细节）

2. 小参数量模型 WSD Scheduler 训练，速度更快效果更好

VoxCPM: Comparison with VibeVoice

VibeVoice Technical Report: https://arxiv.org/pdf/2508.19205



同样采用 Semantic 和 Acoustic 两种表征

- 都是预训练的 Tokenizer 模型
- Acoustic Tokenizer 是 WaveVAE
- Semantic Tokenizer 是 ASR 模型
- Tokenizer 帧率都是 **7.5Hz**

直接用预训练语音 Token，简化了 VoxCPM 通过 TSLM 学习 FSQ 语义特征的过程（类似 RALM 模块）

- 推理阶段，需要将 Acoustic Latent 还原到波形并重新编码为 Semantic Latent

VoxCPM: Comparison with VibeVoice

VibeVoice Technical Report: https://arxiv.org/pdf/2508.19205

Model	Frame Rate	test-zh		test-en	
		CER(%) ↓	SIM ↑	WER(%) ↓	SIM ↑
MaskGCT [WZL+24]	50	2.27	0.774	2.62	0.714
Seed-TTS [ACC+24b]	-	1.12	0.796	2.25	0.762
FireRedTTS [GLS+24]	25	1.51	0.635	3.82	0.460
CosyVoice 2 [DWC+24b]	25	1.45	0.748	2.57	0.652
Spark TTS [WJM+25]	50	1.20	0.672	1.98	0.584
VIBEVOICE-1.5B	7.5	1.16	0.744	3.04	0.689

Table 2: Results on the SEED test sets.

Tokenizer	N_q	Token Rate	test-clean			test-other		
			PESQ	STOI	UTMOS	PESQ	STOI	UTMOS
Ground-Truth	-	-	-	-	4.056	-	-	3.483
Encodec [DCSA22]	8	600	2.72	**0.939**	3.04	2.682	**0.924**	2.657
DAC [KSL+23]	4	400	2.738	0.928	3.433	2.595	0.908	2.945
Encodec [DCSA22]	4	300	2.052	0.901	2.307	2.052	0.884	2.088
SpeechTokenizer [ZZL+23]	4	300	1.931	0.878	3.563	1.737	0.837	3.018
DAC [KSL+23]	1	100	1.246	0.771	1.494	1.245	0.751	1.499
WavTokenizer [JJW+25]	1	75	2.373	0.914	4.049	2.261	0.891	3.431
WavTokenizer [JJW+25]	1	40	1.703	0.862	3.602	1.662	0.834	3.055
Ours (Acoustic)	1	7.5	**3.068**	0.828	**4.181**	**2.848**	0.823	**3.724**

Table 3: Objective evaluation of speech tokenizer's reconstruction quality on the LibriTTS test-clean and test-other datasets. N_q denotes the number of quantizers (VAE for us). Token Rate indicates the number of tokens/frames generated per second of audio. Higher PESQ, STOI, and UTMOS scores indicate better performance. Best results are in **bold**.

VoxCPM: Comparison with VibeVoice

VibeVoice Technical Report: https://arxiv.org/pdf/2508.19205

Model	Subjective				Objective		
	Realism	Richness	Preference	Average	WER (Whisper)	WER (Nemo)	SIM
Nari Labs Dia [Nar25]	-	-	-	-	11.96	10.79	0.541
Mooncast [JYY+25]	-	-	-	-	2.81	3.29	0.562
SesameAILabs-CSM [Ses25]	2.89 ±1.15	3.03 ±1.11	2.75 ±1.08	2.89 ±1.12	2.66	3.05	0.685
Higgs Audio V2 [Bos25]	2.95 ±1.13	3.19 ±1.06	2.83 ±1.16	2.99 ±1.13	5.94	5.97	0.543
Elevenlabs v3 alpha [Ele]	3.34 ±1.11	3.48 ±1.05	3.38 ±1.12	3.40 ±1.09	2.39	2.47	0.623
Gemini 2.5 pro preview tts [Goo]	3.55 ±1.20	3.78 ±1.11	3.65 ±1.15	3.66 ±1.16	1.73	2.43	-
VIBEVOICE-1.5B	3.59 ±0.95	3.59 ±1.01	3.44 ±0.92	3.54 ±0.96	**1.11**	**1.82**	0.548
VIBEVOICE-7B	**3.71** ±0.98	**3.81** ±0.87	**3.75** ±0.94	**3.76** ±0.93	1.29	1.95	**0.692**

Table 1: Human subjective and objective evaluation results. For all subjective metrics and SIM-O, higher scores are better. For WER, lower scores are better. Best results are in **bold**.

Ming-UniAudio: Speech LLM for Joint Understanding, Generation and Editing with Unified Representation

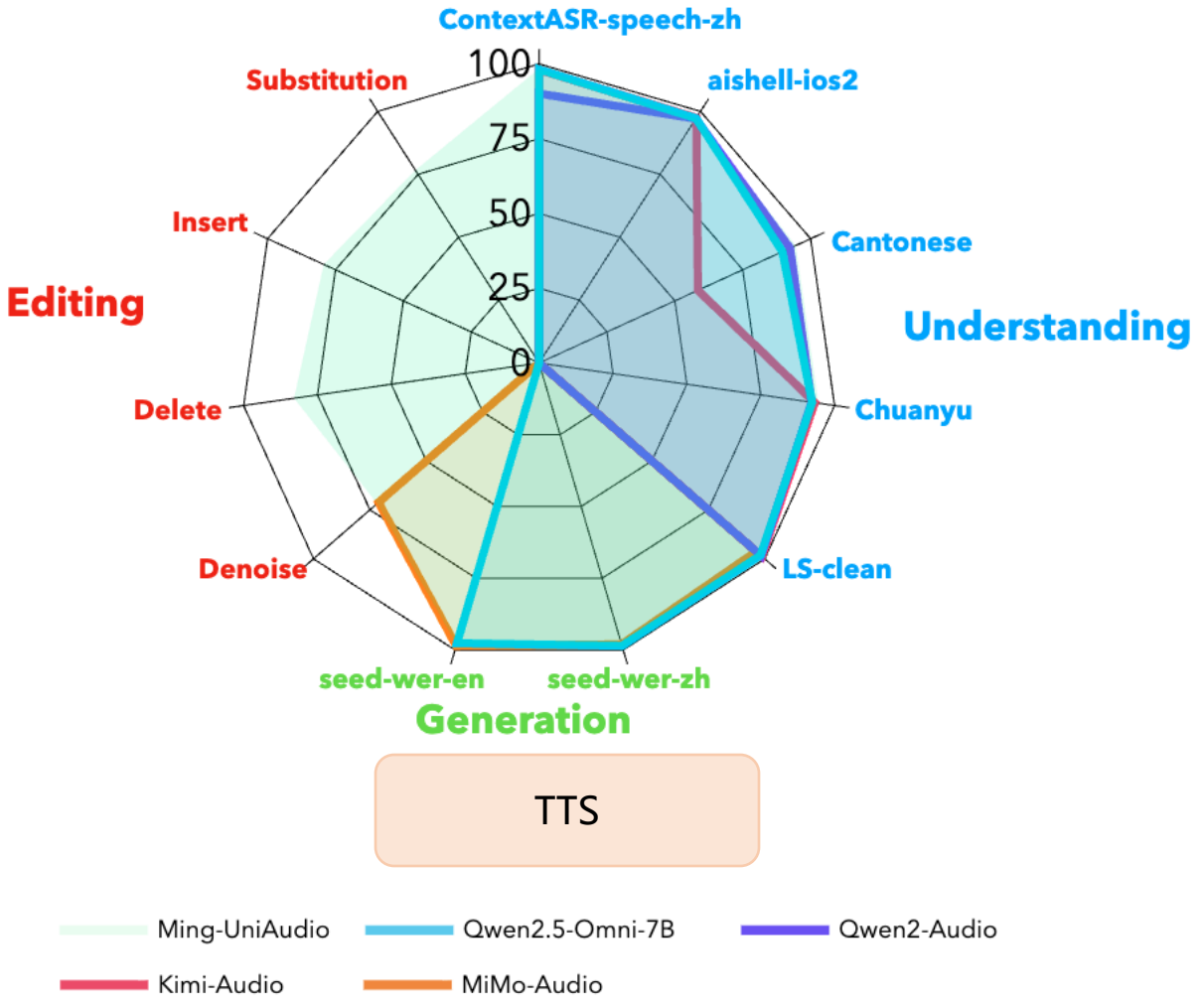
https://mdn.alipayobjects.com/cto_asrtts/uri/file/as/TR-Ming-UniAudio.pdf

Authors are listed **alphabetically by the first name**.

Ant Inclusion AI
Canxiang Yan
Chunxiang Jin
Dawei Huang
Haibing Yu
Han Peng
Hui Zhan
Jie Gao
Jing Peng
Jingdong Chen
Jun Zhou
Kaimeng Ren
Ming Yang
Mingxue Yang
Qiang Xu
Qin Zhao
Ruijie Xiong
Shaoxiong Lin
Xuezhi Wang
Yi Yuan
Yifei Wu
Yongjie Lyu
Zhengyu He
Zhihao Qiu
Zhiqiang Fang
Ziyuan Huang

Ming-UniAudio: Multi-Task Benchmark

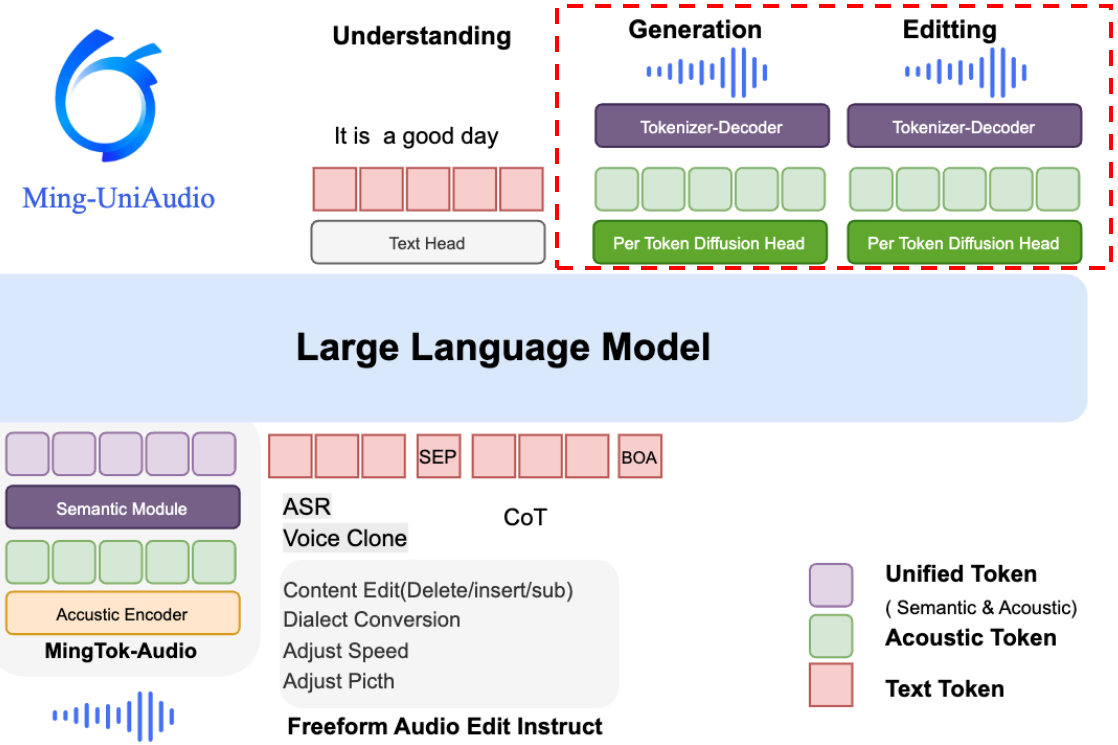
语音编辑



ASR/Context-ASR

TTS

Ming-UniAudio: Model Architecture

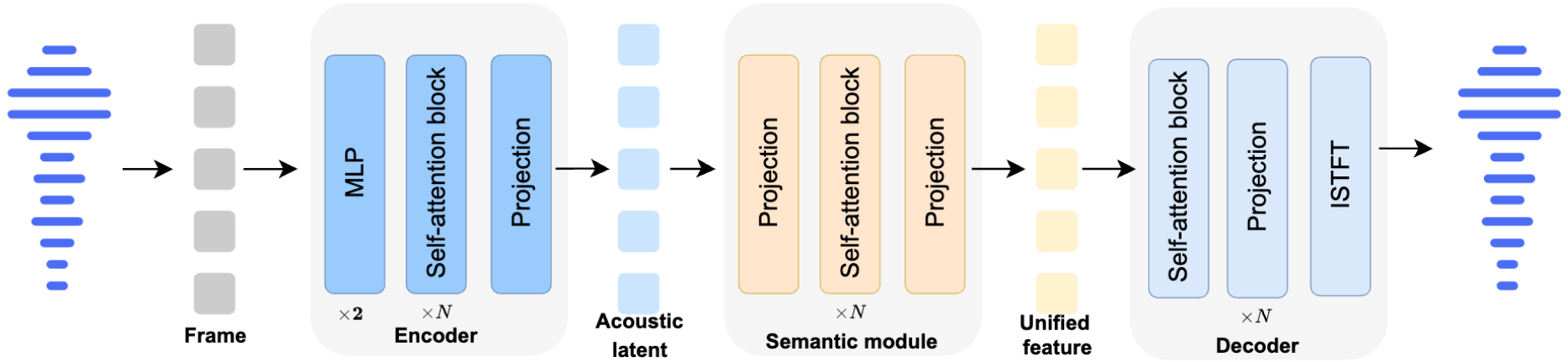


输出侧一般用离散 Token，便于自回归语音生成（各类 Omni 论文），但近期 DiTAR/VibeVoice 证明了 ARDM 的可行性，表明输出侧用连续表征也是可行的

语音理解模型输入侧一般采用连续表征（或者 Kimi-Audio 离散+连续都用），减少语音信息的损失

Ming-UniAudio: Unified Representation

MingTok-Audio：连续 Tokenizer

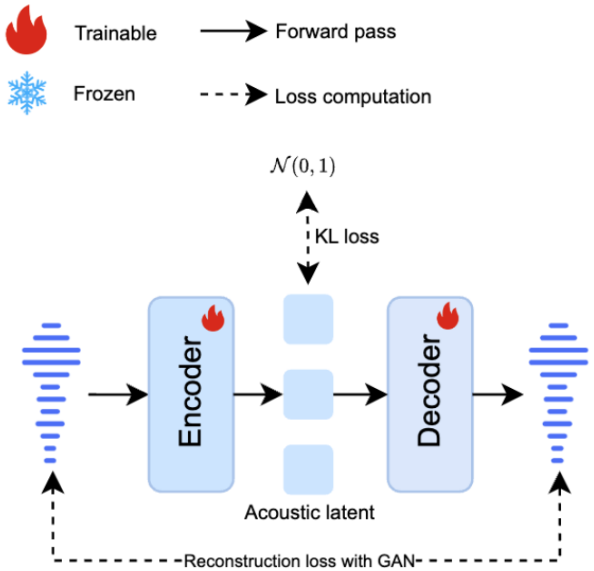


(a) The architecture of MingTok-Audio.

Ming-UniAudio: Unified Representation

MingTok-Audio：连续 Tokenizer

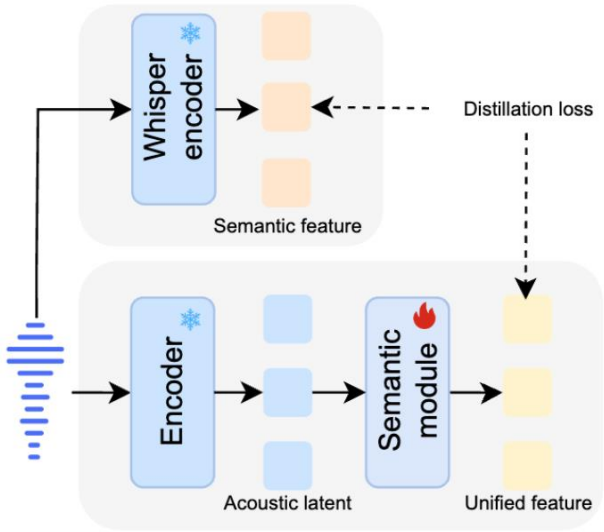
0. 常规的 WaveVAE 训练



S0: Acoustic reconstruction training

$$\mathcal{L}_G = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{fm}}\mathcal{L}_{\text{fm}} + \lambda_{\text{kl}}\mathcal{L}_{\text{KL}}$$

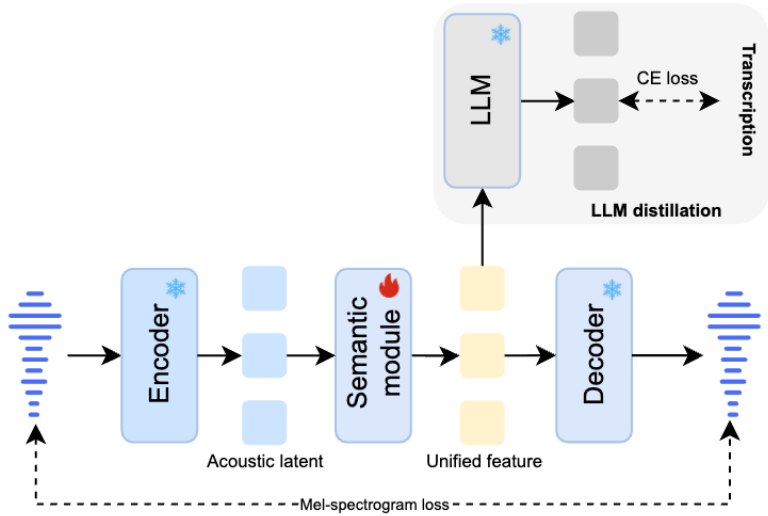
1. 语义对齐训练 Semantic 模块



S1: Semantic feature distillation

$$\mathcal{L}_{\text{distill}} = \text{MSE}(Z_{\text{uni}}, Z_{\text{semantic}})$$

2. LLM ASR 对齐 + 重建

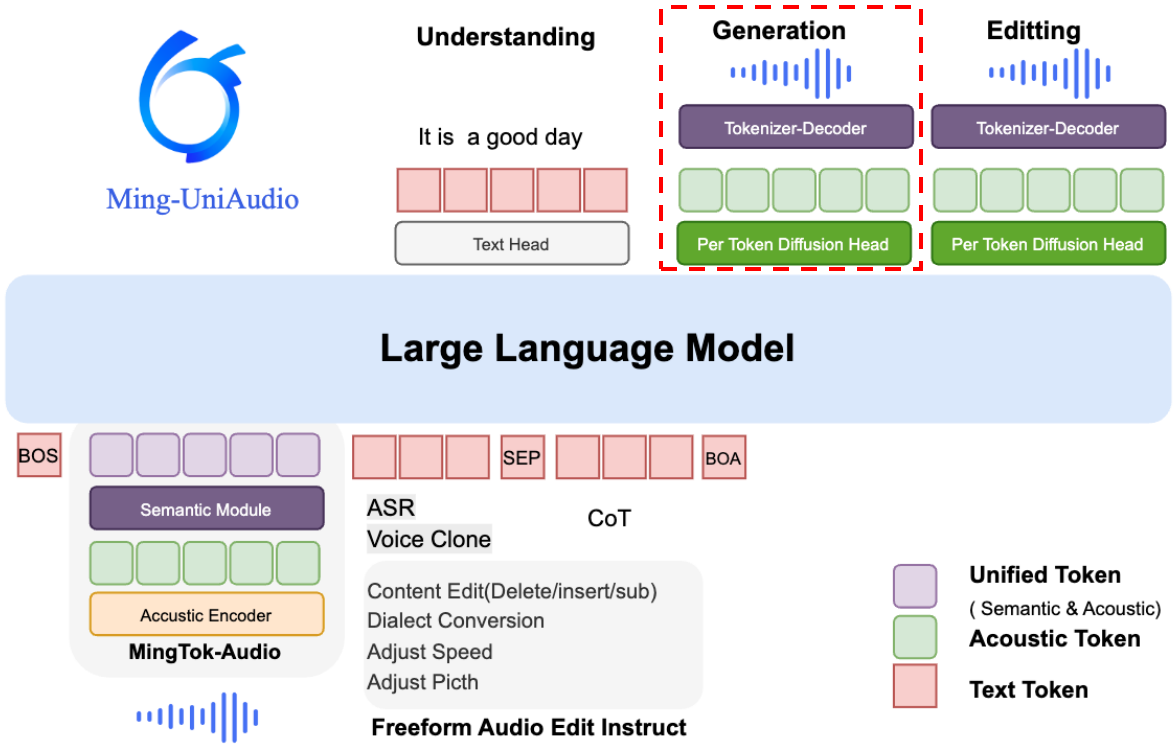


S2: Unified tokenizer training

$$\mathcal{L}_{\text{align}} = -\sum_{t=1}^{T} \log P(y_t | Z_{\text{uni}}, y_{<t})$$

$$\mathcal{L} = \lambda_{\text{align}}\mathcal{L}_{\text{align}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}$$

Ming-UniAudio: Model Architecture



Ming-UniAudio: Experiments (Speech Generation)

System	Seed-zh WER(%)	Seed-zh SIM	Seed-en WER(%)	Seed-en SIM
Seed-TTS Anastassiou et al. (2024b)	1.12	**0.80**	2.25	**0.76**
Ming-Omni-Lite Inclusion et al. (2025)	1.69	0.68	4.31	0.51
MingTok-Audio-TTS	**1.04**	0.75	**1.54**	0.68

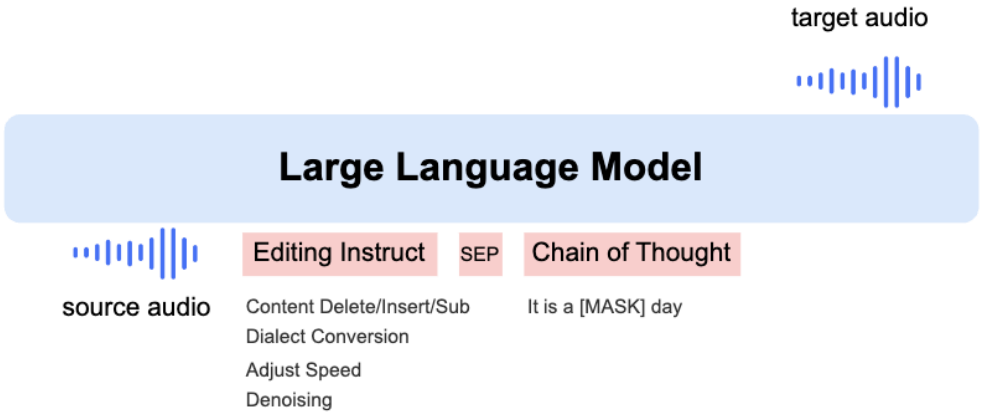
Table 3 Performance of MingTok-Audio on the downstream TTS task.

Datasets	Model	Performance			
		Seed-zh WER(%)	Seed-zh SIM	Seed-en WER(%)	Seed-en SIM
Generation	Seed-TTS Anastassiou et al. (2024b)	1.12	**0.80**	2.25	**0.76**
	FireRedTTS Guo et al. (2024)	1.51	0.65	3.82	0.53
	FireRedTTS-2 Xie et al. (2025)	1.14	0.736	1.95	0.65
	DiTAR Jia et al. (2025)	1.02	0.753	1.69	0.74
	F5-TTS Chen et al. (2024)	1.56	0.74	1.83	0.65
	CosyVoice 2 Du et al. (2024c)	1.45	0.75	2.57	0.65
	CosyVoice 3-1.5B Du et al. (2025)	1.12	0.78	2.21	0.72
	MiMo-Audio Xiaomi (2025)	1.96	-	5.37	-
	Qwen2.5-Omni-7B$_{RL}$ Xu et al. (2025a)	1.42	0.75	2.33	0.64
	Qwen3-Omni-30B-A3B-Instruct Xu et al. (2025b)	1.07	-	**1.39**	-
	Ming-UniAudio	**0.95**	0.70	1.85	0.58

Table 13 Performance comparison on speech generation benchmark datasets. The best results are in **bold**.

Ming-UniAudio: Experiments (Speech Editing)

Demo: https://xqacmer.github.io/Ming-Unitok-Audio.github.io



$$\texttt{SourceSpeech}_{z_{\text{uni}}} + \texttt{Instruction}_{\text{text}} \rightarrow \texttt{CoT}_{\text{text}} + \texttt{TargetSpeech}_{z_{\text{latent}}}$$

Datasets	Model	Performance			
		WER(%) zh | en	**ACC zh | en**	**SIM zh | en**	**no-edit WER(%) zh | en**
Deletion-basic	Ming-UniAudio	11.89 | 14.85	100 | 82.22	0.78 | 0.76	11.49 | 24.26
Deletion-full		22.92 | 27.60	82.92 | 85	0.81 | 0.74	17.50 | 35.21
		WER(%) zh | en	**ACC zh | en**	**SIM zh | en**	**no-edit WER(%) zh | en**
Insertion-basic	Ming-UniAudio	3.42 | 6.63	80 | 71.43	0.83 | 0.79	3.52 | 17.70
Insertion-full		3.89 | 7.592	79.31 | 62.31	0.83 | 0.79	4.10 | 18.84
		WER(%) zh | en	**ACC zh | en**	**SIM zh | en**	**no-edit WER(%) zh | en**
Substitution-basic	Ming-UniAudio	4.52 | 8.99	78.62 | 59.78	0.82 | 0.78	4.63 | 19.28
Substitution-full		4.56 | 7.64	76.62 | 65.62	0.83 | 0.77	4.75 | 18.39
		WER(%)	**ACC**	**SIM**	
Dialect conversion	Ming-UniAudio	8.93	0.50	0.66	
		WER(%) zh | en	**SIM zh | en**	**RDE(%) zh | en**	
Speed Alteration	Ming-UniAudio	5.88 | 17.53	0.66 | 0.57	6.36 | 5.92	
		WER(%) zh | en	**SIM zh | en**		
Pitch Alteration	Ming-UniAudio	7.45 | 13.37	0.36 | 0.24		
		WER(%) zh | en	**SIM zh | en**	**RAE(%) zh | en**	
Volume Alteration	Ming-UniAudio	1.71 | 1.35	0.86 | 0.80	14.9 | 11.7	
		WER(%)	**SIM**	**DNSMOS OVRL**	
Add Sound	GroundTruth	3.68	-	2.12	
	Ming-UniAudio	3.59	0.78	2.46	
		Model Type	**DNSMOS OVRL**	**DNSMOS SIG**	**DNSMOS BAK**
DNS Challenge	FullSubNet Hao et al. (2021)	specialized	3.26	3.54	4.04
	Inter-Subnet Chen et al. (2023)		3.24	3.54	4.00
	CDiffuSE Lu et al. (2022)		1.31	1.89	1.27
	SGMSE Richter et al. (2023)		3.28	**3.59**	3.97
	StoRM Lemercier et al. (2023)		3.23	3.56	3.92
	MiMo-Audio Xiaomi (2025)	general	**3.30**	3.56	**4.10**
	Ming-UniAudio		3.26	**3.59**	3.97

Table 14 Performance comparison on Ming-Freeform-Audio-Edit. The best results are in **bold**.

Ming-UniAudio: Experiments (Speech Understanding)

Datasets	Model	Performance					
		aishell2-ios	LS-clean	Hunan	Guangyue	Chuanyu	Shanghai
Understanding ASR	Kimi-Audio KimiTeam et al. (2025)	**2.56**	**1.28**	31.93	41.49	6.69	60.64
	Qwen2.5 Omni Xu et al. (2025a)	2.75	1.80	29.31	10.39	7.61	32.05
	Qwen2 Audio Chu et al. (2023)	2.92	1.60	25.88	7.59	7.77	31.73
	Ming-UniAudio	2.84	1.62	**9.80**	**5.51**	**5.46**	**14.65**

Table 11 Comparison of ASR performance on various audio benchmark datasets. The best results are in **bold**.

Datasets	Model	Performance			
		Speech-English	Dialogue-English	Speech-Mandarin	Dialogue-Mandarin
		WER | NE-WER | NE-FNR	WER | NE-WER | NE-FNR	WER | NE-WER | NE-FNR	WER | NE-WER | NE-FNR
Understanding Context ASR Wang et al. (2025b)	Qwen2-Audio Chu et al. (2023)	11.49 | 27.27 | 35.08	13.99 | 33.02 | 32.92	9.92 | 24.10 | 30.02	7.00 | 22.76 | 26.17
	Baichuan-Audio Li et al. (2025a)	7.52 | 5.87 | 4.55	5.66 | 10.01 | 3.64	2.16 | 6.65 | **2.35**	2.96 | 11.48 | 3.94
	Kimi-Audio KimiTeam et al. (2025)	**2.90** | 6.68 | 8.01	**4.67** | 13.50 | 11.31	1.95 | 11.13 | 15.28	2.90 | 15.91 | 16.68
	Baichuan-Omni-1.5 Li et al. (2025c)	8.16 | 7.69 | 6.53	9.91 | 14.40 | 5.54	2.98 | 8.39 | 4.71	5.00 | 16.83 | 7.84
	Qwen2.5-Omni-3B Xu et al. (2025a)	3.99 | 7.80 | 9.69	4.83 | 14.36 | 12.85	2.13 | 10.55 | 14.11	3.12 | 15.07 | 15.17
	Qwen2.5-Omni-7B Xu et al. (2025a)	3.96 | 7.38 | 8.72	5.32 | 11.83 | 9.24	1.84 | 9.80 | 12.19	**2.40** | 14.06 | 13.17
	Ming-UniAudio	4.00 | **3.56** | **3.69**	5.34 | **8.73** | **2.53**	**1.58** | **5.98** | 2.40	3.04 | **9.50** | **1.48**