LongCat-AudioDiT: High-Fidelity Diffusion Text-to-Speech in the Waveform Latent Space

demo: https://aria-k-alethia.github.io/LongCat-AudioDiT-demo

Table of Contents

Background: Latent Representations in DiT-NAR TTS

基于 Diffusion/CFM 的 TTS 模型

论文	发表时间	特征类型	维度	帧率	采样率	总压缩比	Vocoder
VoiceBox	2023.06	log Mel spectrogram	80	100 Hz	16 kHz	160x	HiFi-GAN
Matcha-TTS	2023.09	Mel spectrogram	80	~86.1 Hz	22.05 kHz	256x	HiFi-GAN
Seed-TTS DiT	2024.06	Vocoder latent	未公开	未公开	未公开	—	自有 Vocoder
DiTTo-TTS	2024.06	Mel-VAE latent	512	10.76 Hz	22.05 kHz	~2049x	Mel-VAE → BigVGAN
E2-TTS	2024.06	log mel-filterbank	100	~93.75 Hz	24 kHz	256x	BigVGAN
F5-TTS	2024.10	log mel-filterbank	100	~93.75 Hz	24 kHz	256x	Vocos
Mega-TTS 3	2025.02	WaveVAE latent	32	25 Hz	24 kHz	960x	WaveVAE Decoder
ZipVoice	2025.06	Mel spectrogram (同 F5)	100*	~93.75 Hz*	24 kHz*	256x*	Vocos
LongCat-AudioDiT	2026.03	Wav-VAE latent	64	11.72 Hz	24 kHz	~2048x	Wav-VAE Decoder

> ** ZipVoice 论文未显式说明，参数与 F5-TTS 一致。*

建模目标的三种选择

方案 1：Mel-Spectrogram

- 扩散模型直接生成 mel，再通过独立 vocoder 转波形
- 问题一：mel 丢失相位信息和高频细节，音质上限更低
- 问题二：特征帧率明显更高，模型训练 / 推理成本高
- 代表工作：**VoiceBox, E2-TTS, F5-TTS, ZipVoice**

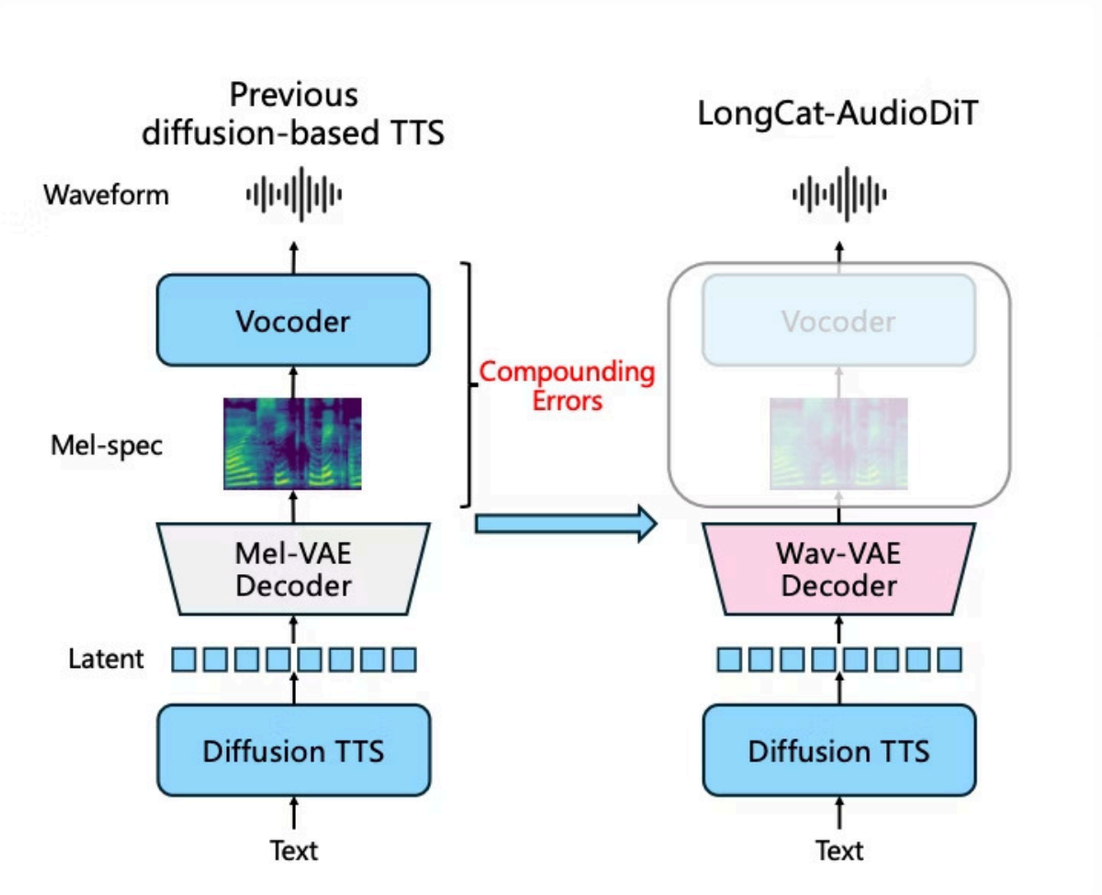
方案 2：Mel-VAE

- 将 mel 压缩到更低维 latent 和更低帧率，加速训练/推理
- 问题：仍依赖 mel 中间表征，级联误差仍存在甚至更严重
- 代表工作：**DiTTo-TTS**

方案 3：Wav-VAE

- DiT 采样后由 VAE 解码器直接输出波形，绕过 mel，消除级联误差
- 特征帧率显著更低，训练 / 推理效率更高，音质上限更高
- 代表工作：**Seed-TTS DiT, Mega-TTS 3, LongCat-AudioDiT**

LongCat-AudioDiT Architecture



整体架构简洁，包含两个组件：

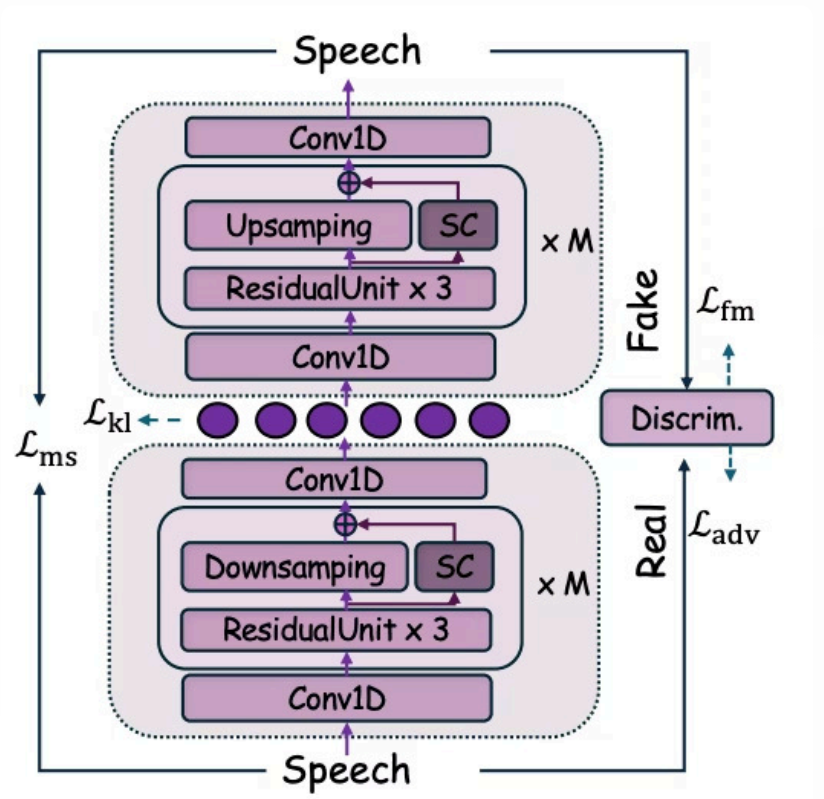
- Wav-VAE (157M)：全卷积自编码器，将 24kHz 波形压缩为 64 维 / 11.72 Hz 连续 latent
- Diffusion Transformer (1B / 3.5B)：在 Wav-VAE latent 表征上进行建模
  - 训练：VAE encoder 编码波形 → DiT 在 latent 空间学习
  - 推理：DiT 从噪声采样 → VAE decoder 直接输出波形

Wav-VAE: Waveform Variational Autoencoder

设计目标

将原始波形直接压缩为紧凑的连续表征，为 DiT 提供更高质量的建模目标

图片引用自论文 CLEAR: **https://arxiv.org/pdf/2508.19098**

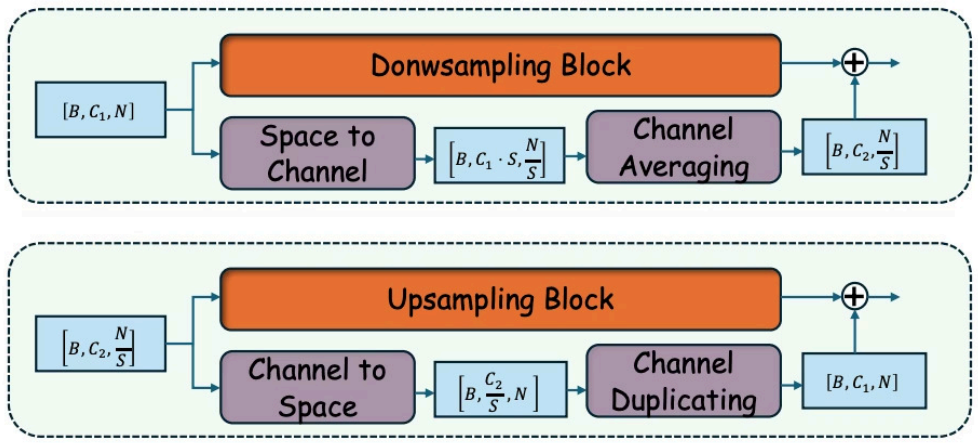


Encoder 架构

- 输入 x ∈ R^(1×T) 的波形，通过 1D Conv 映射到高维特征
- N 个级联 Oobleck block 逐步下采样到目标帧率
- 每个 block 使用膨胀卷积+残差+1维下采样，膨胀卷积捕获多尺度时间依赖（使用 Snake 激活函数）

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + \mathrm{Conv}_{1\times1}\Big(\sigma\big(\mathrm{Conv}_{k,d}\big(\sigma(\boldsymbol{h})\big)\big)\Big)$$

- **无参数 shortcut**：
  - space-to-channel reshape + channel-wise averaging
  - 作用：在多层上/下采样过程中，保持训练的稳定性



Decoder

- 编码器的镜像结构，逐步上采样 + 无参数 shortcut
- channel-to-space reshape + channel-wise replication

最终配置：D=64, 帧率 11.72 Hz（2048 倍下采样）, 模型总参数 157M

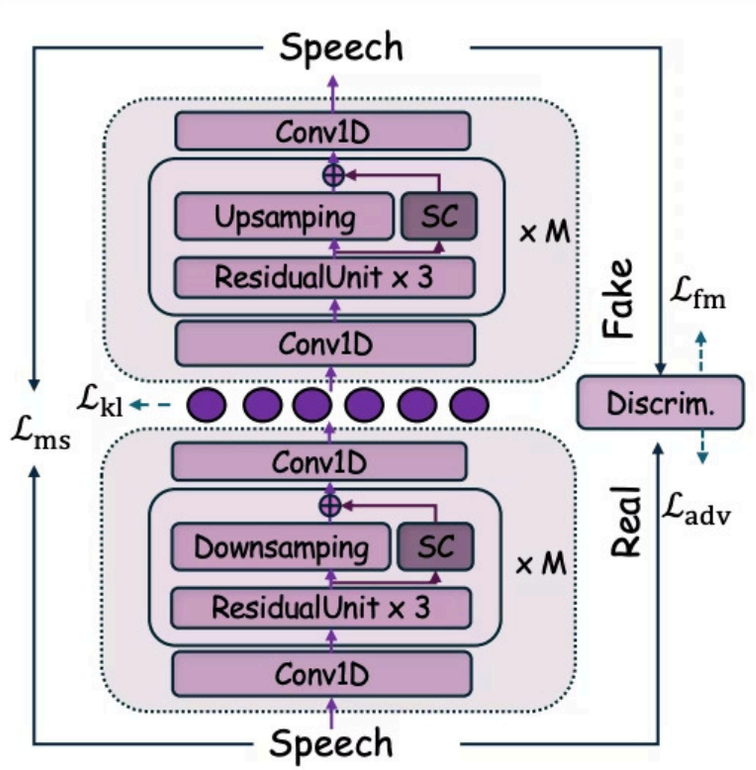
Wav-VAE: Waveform Variational Autoencoder

VAE 瓶颈

- Encoder 每帧实际输出 128 维，生成 μ 和 $\log \sigma^2$
  - 与标准高斯分布计算 KL 散度
- VAE 训练阶段，重参数化采样 $z = \mu + \sigma \odot \varepsilon$，作为 latent
- 最终的 latent 实际是 D=64 维

```python
def forward(self, audio: torch.Tensor) -> dict[str, torch.Tensor]:
    # --- Encode ---
    # encoder 输出 (batch, encoder_latent_dim=128, num_frames)
    # 128 = latent_dim×2, 前半是均值, 后半是尺度参数
    latents = self.encoder(audio)
    mean, scale_param = latents.chunk(2, dim=1)
    # softplus(x) = log(1 + exp(x)), 保证输出恒正; +1e-4 防止数值退化为零
    stdev = F.softplus(scale_param) + 1e-4
    # --- Reparameterization ---
    # z = μ + σ·ε, ε~N(0,I), 使梯度可通过 μ 和 σ 回传
    z = torch.randn_like(mean) * stdev + mean
    # --- Decode 解码为波形 ---
    recon = self.decoder(z)
    return {
        "recon": recon,
        "mean": mean,
        "stdev": stdev,
        "latent": z,
    }
```

图片引用自论文 CLEAR: **https://arxiv.org/pdf/2508.19098**



Wav-VAE: Training Objective

训练目标

- 整体采用 VAE+GAN 的训练方式
- 生成器损失包含六项，覆盖频域、时域、KL 正则化和对抗四个维度：
- L_spec：频域，多分辨率 STFT 损失
- L_mel：频域，多尺度 mel 频谱损失
- L_time：时域，采样点级别的 L1 损失
- L_KL：KL 散度正则化，确保 latent 表征空间接近标准高斯，分布更平滑连续
- L_adv + L_fm：多尺度 STFT 判别器的对抗损失 + 特征匹配损失

关键训练技巧

初始 warmup 阶段禁用 L_adv 和 L_fm，让 autoencoder 先建立稳定的重建映射，再引入对抗训练

这一策略在 audio codec 训练中也被反复验证有效：

- 让编码器先不过多关注对抗任务的细节学习，避免训练初期 GAN 训练的梯度不稳定

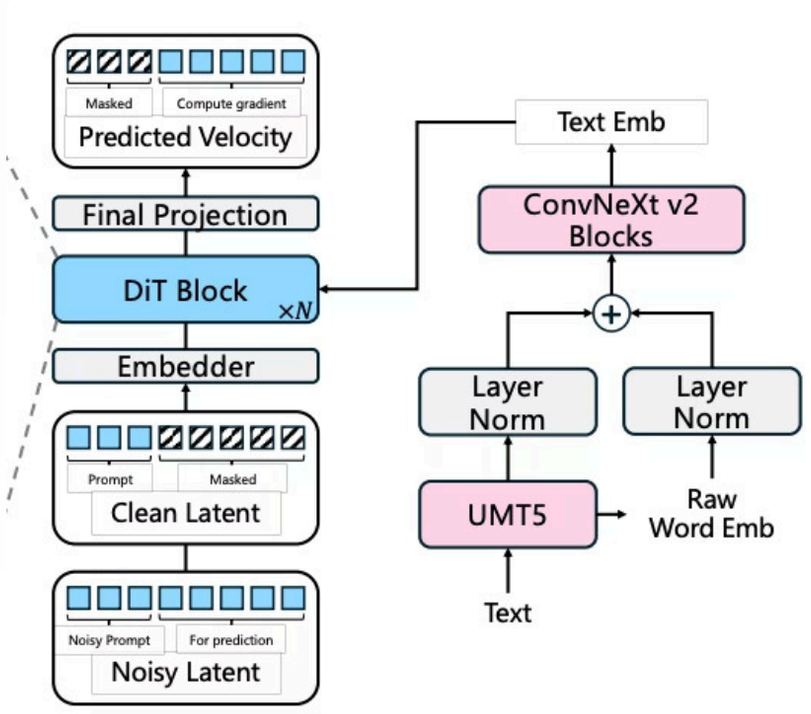
DiT Backbone: Conditional Flow Matching

DiT 基础

训练目标

$$\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t,m,z_0,z_1}\left[\left\|(1-m)\odot\left((z_1-z_0)-v(z_t,t,z_{ctx},q;\theta_{\mathrm{CFM}})\right)\right\|^2\right],$$

$$z_t = (1-t)z_0 + tz_1.$$



Masked Modeling

- 参考 VoiceBox/F5-TTS，随机保留靠前一部分作为 prompt
- 目的：天然适配 zero-shot 声音复刻推理
- 做法：仅在 masked 区域计算损失

CFG 训练策略

- 以 10% 概率同时丢弃 z_ctx 和文本条件 q
- 使模型学会无条件分布，推理时支持 CFG

推理方案：Euler ODE solver，16 步 NFE

DiT Network Architecture

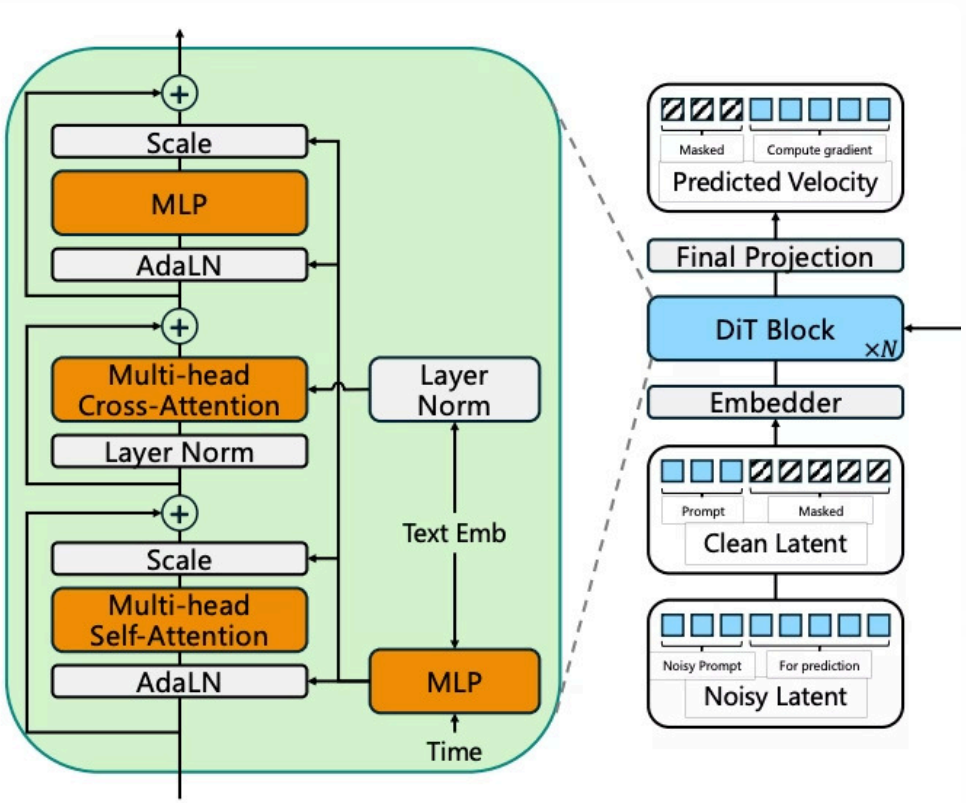
DiT 模型结构

核心设计

- 时间步 t 注入：使用 Adaptive Layer Normalization 注入
- QK-Norm：加入 QK-Norm 稳定训练（使用 RMSNorm）
- RoPE 位置编码：所有注意力层使用旋转位置编码

结构优化

- Long-skip Connection：网络输入直接加到最终层隐藏状态，带来一致性的略微提升
- Global AdaLN：全局共享 AdaLN 替代每层独立，减少参数

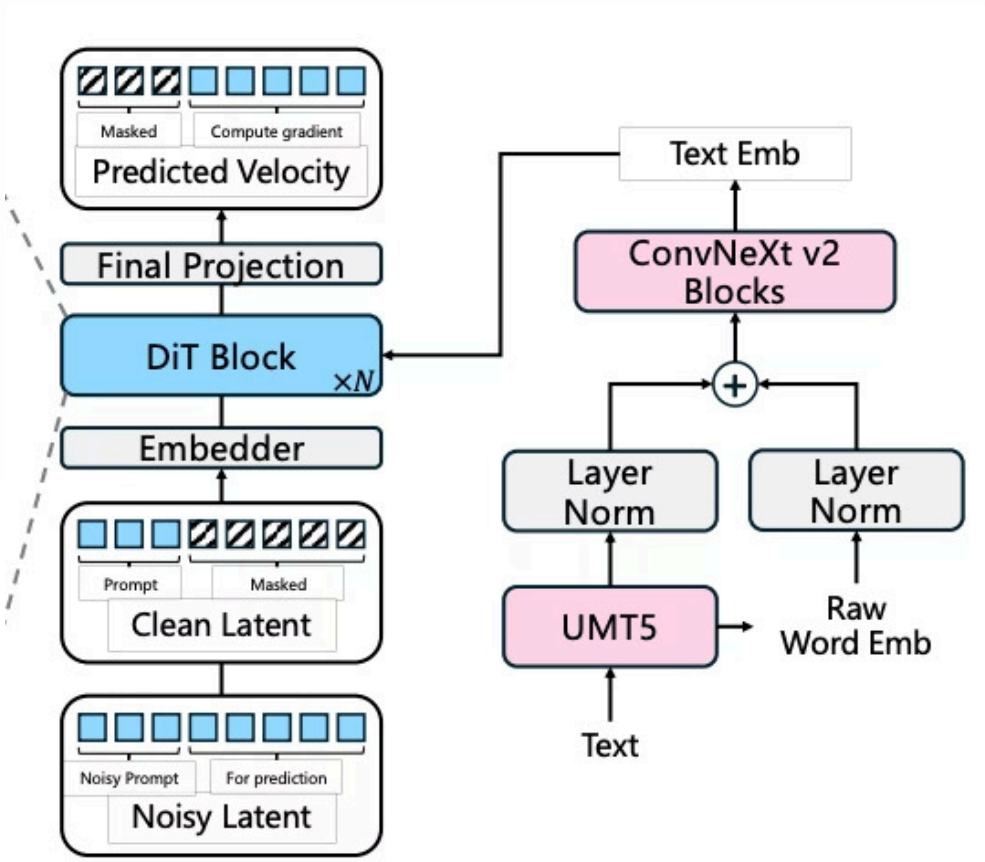


DiT Network Architecture

DiT 引入文本信息

1. 文本编码器选型

- 从零训练：成本高，难扩展到新语言
- ByT5：byte 级 tokenization 导致中文序列过长，对齐困难
- UMT5（最终选择）：支持 107 种语言，subword tokenizer 保持合理序列长度



2. 文本 Embedding 使用方式

- 问题：仅用 last hidden state 做 TTS，可懂度差
- 原因：last hidden state 是高层语义表征，丢失了声学映射所需的底层词汇和音素信息
- 方案：融合 raw word embedding（subword 级信息）+ last hidden state（语义信息）

$$q = \mathrm{LayerNorm}(\mathrm{last\_hidden\_state}) + \mathrm{LayerNorm}(\mathrm{raw\_word\_embedding}).$$

- 补充一：LayerNorm 平衡两个表征空间的尺度差异
- 补充二：ConvNeXt V2 增加局部信息，加速对齐收敛

3. 文本引入方式：Cross-Attention

- 借鉴 DiTTo-TTS，用 cross-attention 隐式学习文本-语音对齐
- 无需 phoneme 级 duration，仅需整句级 duration

DiT Network Architecture

DiT 引入文本信息

3. 文本引入方式：Cross-Attention

- 借鉴 DiTTo-TTS，用 cross-attention 隐式学习文本 - 语音对齐
- 无需 phoneme 级 duration，仅需整句级 duration

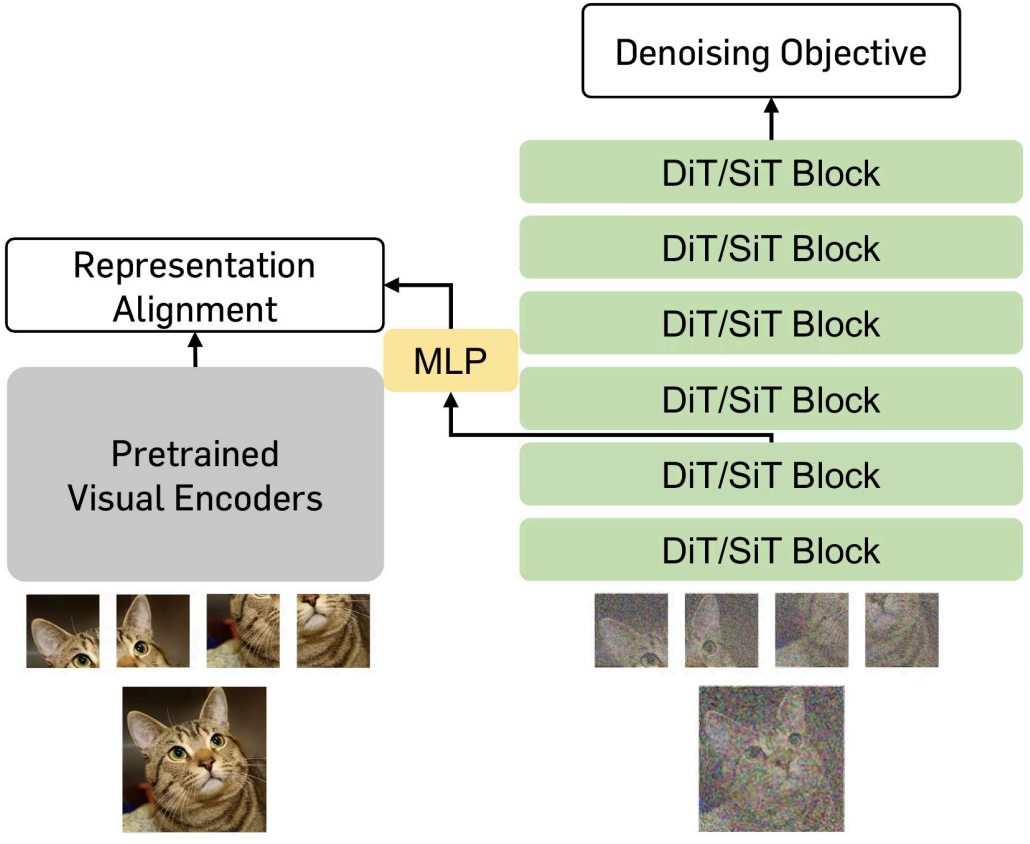
Alignment 类型	论文	发表时间	Duration 预测粒度
显式	VoiceBox	2023.06	phoneme 级
	Matcha-TTS	2023.09	phoneme 级
	Mega-TTS 3	2025.02	phoneme 级
隐式 — Filler padding + Self-attn	E2-TTS	2024.06	整句级
	F5-TTS	2024.10	整句级
隐式 — Cross-attention	Seed-TTS DiT	2024.06	整句级
	DiTTo-TTS	2024.06	整句级
	LongCat-AudioDiT	2026.03	整句级
隐式 — Average upsampling	ZipVoice	2025.06	整句级

DiT Network Architecture

DiT 训练策略

REPA（Representation Alignment）

- 做法：用预训练 mHuBERT 的特征监督 DiT 第 8 层输出
  - 损失函数：L1 Loss
- 目标：将 DiT 中间表征对齐到语义表征空间
- 结论：不提升生成质量，但大幅加速训练收敛



Core Innovation: Training-Inference Mismatch (1/2)

核心问题：训练与推理不匹配

> 这是一个被所有前作忽视的推理漂移问题

前提：将 z_t 沿时间轴划分为两部分：prompt（条件）区域 + 目标生成区域

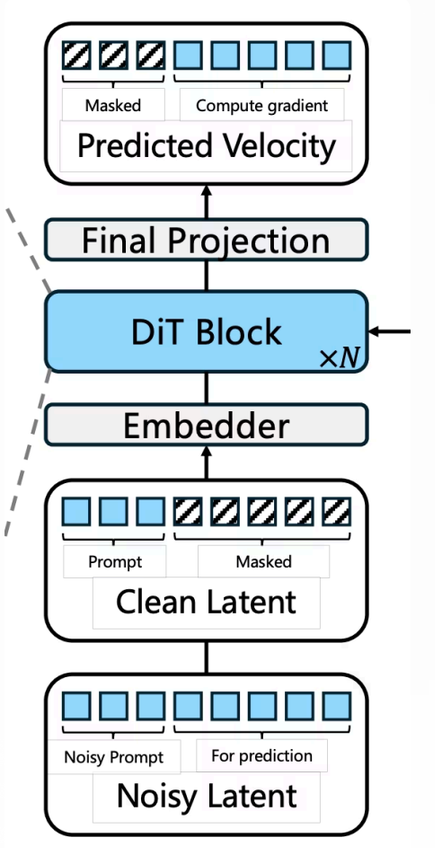
Prompt 区域：训练时未加入 loss 计算 → 速度预测结果无法保证 → 累积更新导致偏离真实轨迹

目标生成区域：

- 训练过程中，prompt 条件区域用的是真实 z_t
- 推理过程中，无法保证 prompt 条件区域的 z_t 合理性

后果：prompt 区域逐步偏离 GT 轨迹，引入训练-推理不匹配，影响目标生成区域的合成质量

这个问题在 VoiceBox、F5-TTS、E2-TTS 等所有架构中都存在，但从未被识别和关注



Core Innovation: Training-Inference Mismatch (2/2)

解决方案：每步迭代，覆写 prompt 噪声为 GT 值

- 在每个推理步骤，强制将 z_t^ctx 覆写为其理论值：

$$z_t^{ctx} \leftarrow t z^{ctx} + (1-t) z_0^{ctx},$$

- 其中 z_0^ctx 是 prompt 部分的初始高斯噪声

修改后引发的问题：推理阶段条件信息泄漏

- 真正的 unconditional 速度估计，仅丢弃 z_ctx 是不够的
- z_t^ctx 本身已经包含了 prompt 的声学信息
- 因此计算 unconditional 输出时，必须一同丢弃 z_t^ctx

> 补充说明
>
> F5-TTS：代码存在这个问题
>
> CosyVoice 系列 DiT：不存在这个问题
>
> - CosyVoice 的 DiT 直接是计算的完整序列上的 loss
> - Prompt 和非 prompt 区域，只有在输入上有差别（是否提供了 prompt mel）

Experiment	CER (%) ↓	SIM ↑	UTMOS ↑	DNSMOS ↑
LongCat-AudioDiT-1B	**1.18**	**0.812**	**3.16**	**3.40**
training-inference mismatch	1.21	0.769	2.83	3.34
w/o APG	**1.18**	**0.812**	3.06	3.38

Core Innovation: Adaptive Projection Guidance

CFG 的过饱和问题

标准 CFG 通过放大条件 / 无条件预测差异来提升质量：

$$v_t^{\mathrm{CFG}} = v_t + \alpha(v_t - v_t^u),$$

实验观察：

- CFG 有效提升了合成质量，但可能会引入可听伪影
- 增大引导强度 α 进一步加剧伪影→「过饱和」现象

APG 的核心思路

将引导残差在样本域几何分解为两个正交分量：

$$v_t - v_t^u$$

- 平行分量 Δμ∥（与条件预测同向）→ 导致过饱和的主因
- 正交分量 Δμ⊥（与条件预测垂直）→ 有益的引导信号

APG 选择性抑制平行分量，保留正交分量，消除过饱和问题

Table 4 | Objective evaluation results of the ablation studies on noise-prompt dual masking and APG on the Seed-ZH benchmark [Anastassiou et al., 2024]. **Bold** indicates the best score.

Experiment	CER (%) ↓	SIM ↑	UTMOS ↑	DNSMOS ↑
LongCat-AudioDiT-1B	**1.18**	**0.812**	**3.16**	**3.40**
training-inference mismatch	1.21	0.769	2.83	3.34
w/o APG	**1.18**	**0.812**	3.06	3.38

超参数配置

- α = 4.0（引导强度，与 CFG 相同）
- η = 0.5（抑制平行分量的系数）

→ APG 有效消除过饱和伪影，提升感知自然度

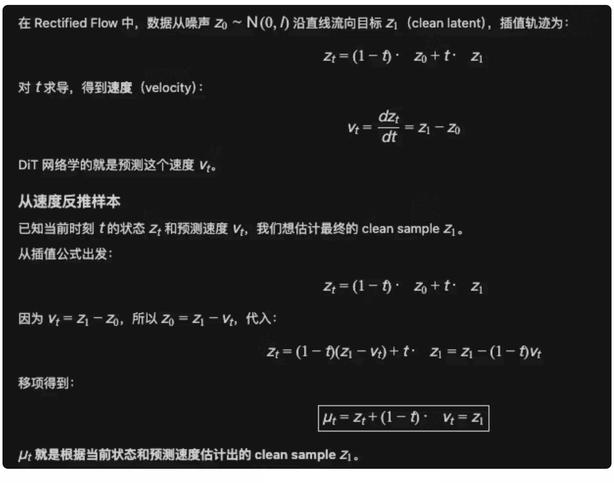
APG 步骤

- 将速度域预测转换到样本域：

$$\mu_t = z_t + (1-t)v_t$$

为什么 APG 要转换到样本域?

APG 需要对比"有条件引导"和"无条件引导"预测的**目的地差异**，然后将这个差异分解为平行/正交分量。在样本域中做这个分解更有几何意义。



- 计算引导残差：

$$\Delta\mu_t = \mu_t - \mu_t^u$$

- 分解为平行和正交分量：

$$\Delta\mu_t^{\parallel} = \frac{\langle \Delta\mu_t, \mu_t \rangle}{\langle \mu_t, \mu_t \rangle}\mu_t$$

$$\Delta\mu_t^{\perp} = \Delta\mu_t - \Delta\mu_t^{\parallel}$$

- 选择性抑制平行分量：

$$\mu_t^{\mathrm{APG}} = \mu_t + \alpha\Delta\mu_t^{\perp} + \eta\Delta\mu_t^{\parallel},$$

- 映射回速度域继续 ODE 求解：

$$v_t^{\mathrm{APG}} = \frac{\mu_t^{\mathrm{APG}} - z_t}{1-t}.$$

- 额外 trick：负动量 negative momentum

$$\overline{\Delta\mu}_t \leftarrow \Delta\mu_t + \beta\cdot\overline{\Delta\mu}_t$$

$$\overline{\Delta\mu}_t^{\mathrm{new}} = \Delta\mu_t - 0.3\cdot\overline{\Delta\mu}_t^{\mathrm{old}}$$

Training Configuration & Data

训练数据

- Wav-VAE：200K 小时中英语音，音频切片约 3 秒
- DiT 基线/消融：10 万小时中英语音
- DiT 大规模实验：100 万小时中英语音
- 采样率 24kHz，最大音频时长 60 秒（TTS）
- 训练数据文本由 ASR 模型生成（非人工标注）

模型规模与训练配置

- Wav-VAE: 157M params, 32×H800, batch=384
- DiT-1B: 16×GPU, batch=256
- DiT-3.5B: 64×GPU, batch=1024
- AdamW (β_1=0.9, β_2=0.95), LR: 1e-4 → 1e-5 线性衰减
- 1K warmup steps, 推理 16 步 Euler ODE

评测指标

Wav-VAE 重建

- PESQ（感知质量）、STOI（可懂度）、UTMOS（自然度）

TTS 合成（Seed Benchmark）

- CER/WER：可懂度（Whisper / Paraformer 转写）
- SIM：说话人相似度（cosine similarity of WavLM embeddings）
- UTMOS：自然度（neural MOS approximation）
- DNSMOS：整体声学质量

Experiment: SOTA Results on Seed-TTS Benchmark

Model	ZH CER (%) ↓	ZH SIM ↑	EN WER (%) ↓	EN SIM ↑	ZH-Hard CER (%) ↓	ZH-Hard SIM ↑
GT	1.26	0.755	2.14	0.734	-	-
NAR Models						
Seed-DiT [Anastassiou et al., 2024]	1.18	0.809	1.73	**0.790**	-	-
MaskGCT [Wang et al., 2024]	2.27	0.774	2.62	0.714	10.27	0.748
E2 TTS [Eskimez et al., 2024b]	1.97	0.730	2.19	0.710	-	-
F5 TTS [Chen et al., 2024a]	1.56	0.741	1.83	0.647	8.67	0.713
F5R-TTS [Sun et al., 2025]	1.37	0.754	-	-	8.79	0.718
ZipVoice [Zhu et al., 2025]	1.40	0.751	1.64	0.668	-	-
AR/Hybrid Models						
Seed-ICL [Anastassiou et al., 2024]	1.12	0.796	2.25	0.762	7.59	0.776
SparkTTS [Wang et al., 2025]	1.20	0.672	1.98	0.584	-	-
Qwen2.5-Omni [Xu et al., 2025]	1.70	0.752	2.72	0.632	7.97	0.747
CosyVoice [Du et al., 2024a]	3.63	0.723	4.29	0.609	11.75	0.709
CosyVoice2 [Du et al., 2024b]	1.45	0.748	2.57	0.652	6.83	0.724
FireRedTTS-1S [Guo et al., 2025]	1.05	0.750	2.17	0.660	7.63	0.748
CosyVoice3-1.5B [Du et al., 2025]	1.12	0.781	2.21	0.720	5.83	0.758
IndexTTS2 [Zhou et al., 2025a]	1.03	0.765	2.23	0.706	7.12	0.755
DiTAR [Jia et al., 2025]	1.02	0.753	1.69	0.735	-	-
MiniMax-Speech [Zhang et al., 2025]	0.99	0.799	1.90	0.738	-	-
VoxCPM [Zhou et al., 2025b]	0.93	0.772	1.85	0.729	8.87	0.730
MOSS-TTS [SII-OpenMOSS, 2026]	1.20	0.788	1.85	0.734	-	-
Qwen3-TTS [Hu et al., 2026]	1.22	0.770	**1.23**	0.717	6.76	0.748
CosyVoice3.5	**0.87**	0.797	1.57	0.738	**5.71**	0.786
LongCat-AudioDiT-1B	1.18	0.812	1.78	0.762	6.33	0.787
LongCat-AudioDiT-3.5B	1.09	**0.818**	1.50	0.786	6.04	**0.797**

- LongCat-3.5B 在 Seed-ZH 刷新 SIM SOTA: 0.818 (vs Seed-DiT 0.809, +0.009)
- Seed-Hard SIM SOTA: 0.797 (vs Seed-ICL 0.776, +0.021)，大幅超越所有 AR/Hybrid 模型
- 全面超越所有 mel 基扩散模型：vs F5-TTS SIM 0.741 → 0.812 (+0.071)，有力验证 Wav-VAE 建模消除级联误差
- 可懂度竞争力强（CER 1.09%），略逊于 CosyVoice3.5 等依赖多阶段训练+人工标注的系统

Experiment: Wav-VAE Reconstruction Quality

Model	N_q	FPS	PESQ ↑	STOI ↑	UTMOS ↑
GT	–	–	4.644	1.0	4.056
Discrete Codecs					
DAC [Kumar et al., 2023]	9	900	3.908	0.970	3.910
Encodec [Défossez et al., 2022]	8	600	2.720	0.939	3.040
Vocos [Siuzdak, 2023]	8	600	2.807	0.943	3.695
WavTokenizer [Ji et al., 2024]	1	75	2.373	0.914	4.049
BigCodec [Xin et al., 2024]	1	80	2.697	0.939	4.097
Continuous VAEs					
VibeVoice [Peng et al., 2025]	1	7.50	3.068	0.828	**4.181**
Ours Wav-VAE	1	7.81	3.089	0.963	4.116
Ours Wav-VAE	1	11.72	**3.237**	**0.967**	4.013

Wav-VAE 重建质量分析

- 在极低帧率（11.72 FPS）下，PESQ 3.237 和 STOI 0.967 均超过大多数离散 codec
- 相比 DAC（9 codebook, 900 FPS），Wav-VAE 用 1/77 的序列长度实现更高的 STOI
- 相比同类连续 VAE（VibeVoice），PESQ +0.169, STOI +0.139，在可比帧率下全面领先
- 充分展示了连续表征相比离散 token 在表达效率和重建保真度上的优势

Ablation RQ1: Wav-VAE vs. Mel-VAE

核心验证：Wav-VAE vs. Mel-VAE

Table 3 | Objective evaluation results of TTS models based on Wav-VAE and Mel-VAE on the Seed benchmark [Anastassiou et al., 2024]. **Bold** indicates the best score.

TTS Latent Model	ZH		EN		ZH-Hard	
	CER (%) ↓	SIM ↑	WER (%) ↓	SIM ↑	CER (%) ↓	SIM ↑
Mel-VAE	1.29	0.706	2.20	0.714	7.70	0.696
Wav-VAE	**1.18**	**0.812**	**1.78**	**0.762**	**6.33**	**0.787**

- SIM 从 0.706 跃升到 0.812（+0.106），提升非常明显
- CER 从 1.29% 降到 1.18%，WER 从 2.20% 降到 1.78%，可懂度也有显著改善

原因分析

- Mel-VAE pipeline 中，latent → mel → waveform 的级联转换会不可逆地丢失这些细节（尤其是相位信息）
- **验证了论文的核心假设：直接在 Wav-VAE 建模能消除/缓解级联误差**

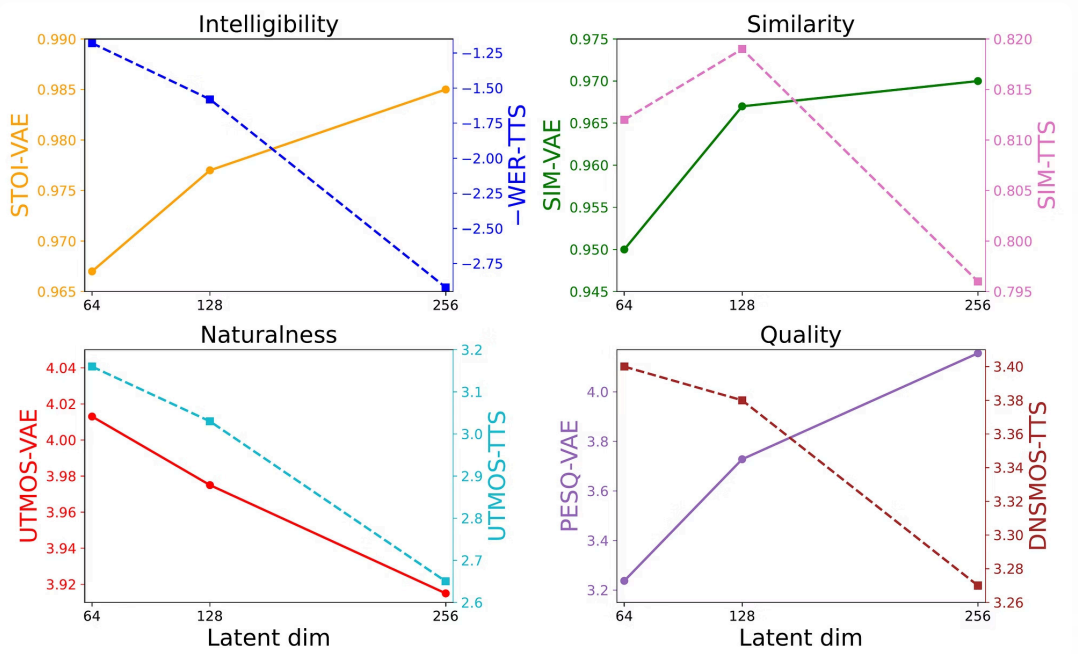
Ablation RQ2: Dimension-Capacity Trade-off

反直觉发现：VAE 重建越好，TTS 不一定越好

实验设计：固定帧率 20Hz，变化 Wav-VAE 维度 {64, 128, 256}，分别训练 Wav-VAE 和对应的 1B TTS 模型

发现 1: 维度-容量权衡

- 增大维度 (64→128→256) 一致提升 Wav-VAE 重建质量
- 但同时降低 TTS 生成质量，违反直觉假设
- 即使 TTS 扩展到 3.5B，128 维仍不如 64 维



解释：过高维度的 Latent 给 DiT 带来了严重的建模负担

进一步验证实验：单纯扩大参数量无法弥补

Ablation RQ2: Frame Rate Sweet Spot

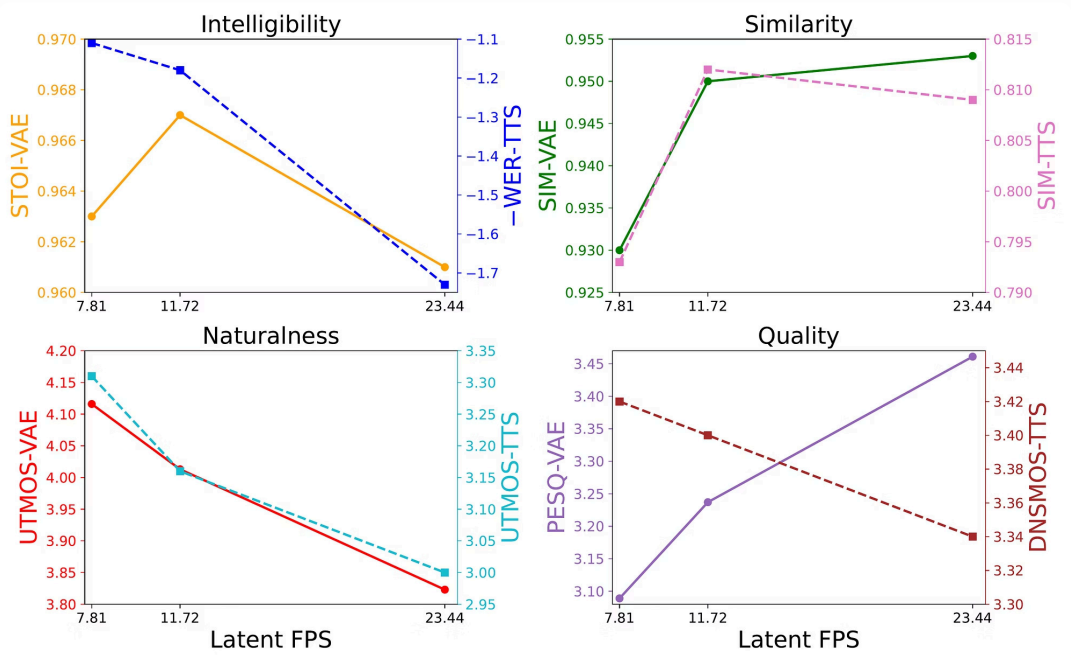
帧率 Sweet Pot：VAE 和 TTS 的最优点不相同

实验设计：固定维度 64，变化帧率 {7.81, 11.72, 23.44} Hz

> **发现 2: 帧率对 VAE 和 TTS 的影响不同**
>
> - 低 FPS 的 VAE：
> - 可懂度和自然度反而更好（全局结构保留）
> - 但相似度较差（高频丢失）
> - 低 FPS 的 TTS：整体合成质量显著提升
> - 原因：DiT 难以准确建模高帧率 latent 中复杂的时域动态变化

最终配置：64 维 / 11.72 Hz

在 VAE 重建和 TTS 生成之间取得最佳平衡



Ablation RQ3: Effectiveness of Inference Techniques

Experiment	CER (%) ↓	SIM ↑	UTMOS ↑	DNSMOS ↑
LongCat-AudioDiT-1B	**1.18**	**0.812**	**3.16**	**3.40**
training-inference mismatch	1.21	0.769	2.83	3.34
w/o APG	**1.18**	**0.812**	3.06	3.38

训练 - 推理不匹配的影响

- SIM: 0.812 → 0.769 (-0.043)
- UTMOS: 3.16 → 2.83 (-0.33)
- DNSMOS: 3.40 → 3.34 (-0.06)

影响极其显著，验证了问题的存在和修正方案的有效性。

APG vs. 标准 CFG

- CER/SIM 完全持平（1.18% / 0.812）
- UTMOS: 3.06 → 3.16 (+0.10)
- DNSMOS: 3.38 → 3.40 (+0.02)

APG 不影响可懂度和相似度，但有效消除 CFG 过饱和伪影，提升声学质量

Takeaways

1 Wav-VAE 路线保持跟进

- 对于重视音色克隆保真度/音质更好的场景，直接用 Wav-VAE 建模
- 即使不完全复现 LongCat-AudioDiT，也可考虑将现有 Mel 替换为 Wav-VAE

2 训练 - 推理不匹配的问题修正

- 纯推理阶段改动（每步覆写 prompt 区域噪声），对所有 CFM-based TTS 系统都适用
- 成本为零，潜在收益较大（SIM +0.043）

3 APG 替代 CFG — 纯推理改动

- 如果 CFG 存在伪影问题，直接尝试 APG，超参数 α=4.0, η=0.5 可作为起点

4 文本编码双嵌入融合 & 64 维 / 11.72 Hz 配置参考

- raw word embedding + last hidden state 相加可，零成本改善可懂度
- 设计 audio VAE 时不要盲目追求高维度 / 高帧率，需综合考虑下游生成模型的建模能力

5 VAE 重建能力与 DiT 生成效果之间的 trade-off

6 REPA 能够提高 DiT 模型收敛速度

Limitations & Future Directions

未来方向

- RLHF for TTS：可进一步弥合可懂度差距
- 蒸馏/加速：将推理步数压缩到 1-4 步
- **如何更好的解决重建与生成任务之间的冲突/矛盾?**
- 与 AR 模型结合：Wav-VAE 作为 AR 模型的目标