复现 Qwen3-Omni 的 Talker 语音标签：bf16 下的 Mimi，以及末帧陷阱

Yuxuan Chen香港中文大学（深圳）
yuxuanchen5@link.cuhk.edu.cn

arXiv:2611.32286v1 [cs.SD] 24 Nov 2026

摘要

Qwen3-Omni 通过 Talker 生成 12.5 Hz 音频编解码器的 16 个码本，但生成其训练标签的编码器并未发布，因此无法使用匹配的标签训练新声音。我们根据已发布模型的行为确定了该编码器。白盒回归结果表明，Talker、多令牌预测器和 Code2Wav 解码器中的码本嵌入，与 kyutai 的 Mimi 各层的前 16 个码本相匹配。基于解码器的测试看似令人信服，却无法通过校准：使用已知标签训练的解码器无法区分这些标签。使用 13 个基于已知标签训练的代码语言模型进行校准后，教师强制的 Talker 似然确实能区分不同实现：与我们尝试的所有其他变体相比，Talker 更偏好来自 kyutai 的 moshi 软件包、使用 bf16 权重生成的标签。我们排除了响度归一化，以及对 20% 或更多数据进行带限的可能性。另一个发现是，除非输入长度是 1920 个采样点的倍数，否则 transformers 的 MimiModel 对最后一帧的编码方式与 kyutai 的编码器不同。使用其标签进行 LoRA 微调后，44–55% 的生成句子会以噪声突发结尾；对于基于 Mimi 的 CSM-1B，同一帧会造成 119 纳特的代价。对输入进行填充即可消除这两种问题。我们发布了编码方案、标签检查器和 Talker 微调工具包。

1 引言

Qwen3-Omni（Xu 等人，2025b）分两个阶段生成语音（图 1）。Thinker 是一个专家混合语言模型，负责编写文本；Talker 是一个规模较小的专家混合模型，将 Thinker 的文本和隐藏状态转换为音频编码，每 80 毫秒生成一帧。每帧包含 16 个码本：Talker 预测第一个码本，多令牌预测器（MTP）预测其余 15 个码本，卷积解码器（Code2Wav）则将编码转换为 24 千赫波形。发布的检查点包含全部三个生成器，但不包含将训练音频转换为编码的编码器。没有它，Qwen 之外的任何人都无法生成用于微调 Talker 的标签，而 Talker 决定了模型的声音表现。

这个问题在公开场合已被反复提出，但一直没有官方答复；社区发现，Qwen3-Omni 的编码可以使用 kyutai 的 Mimi 解码器进行解码（ Défossez 等人，2024年），并且已经使用 Mimi 生成标签（第 6).

“使用 Mimi 解码”并未说明这些标签与如今运行 Mimi 编码器时的结果有多接近：使用相同的权重还是经过微调的权重、采用 fp32 还是更低的精度、使用参考 moshi 代码还是 transformers 移植版，以及使用原始音频还是预处理后的音频。在码本 1–4 上，bf16 推理会改变 12% 的编码；使用 moshi 而不是 transformers（两者均采用 bf16）会改变约 15%；将音频带限至 16 kHz 会改变 40%；进行响度归一化则会改变 67%。我们通过测试回答这些问题，并首先在使用已知标签训练的模型上测量这些测试的敏感性。我们的贡献如下：

1. 标签编码器的识别：白盒证据表明，Talker 的码本是 Mimi 的前 16 个码本；此外，校准似然检验也唯一确定了使用 kyutai 的 moshi 软件包和 bf16 权重运行的 Mimi。

2. 一个否定性结果：基于解码器的测试将公开编码器排在首位，但这一结果并无信息量，因为在已知标签上训练的解码器也会得出相同的排序。

3. transformers 的 Mimi 编码器与 kyutai 的编码器之间存在末帧不匹配，导致经过微调的 Qwen3-Omni 和 CSM-1B 生成的句子中，约有一半以噪声突发结尾。

4. 一种通过填充输入来避免这一问题的编码方案、一个标签检查器，以及一个 Talker 微调工具包。

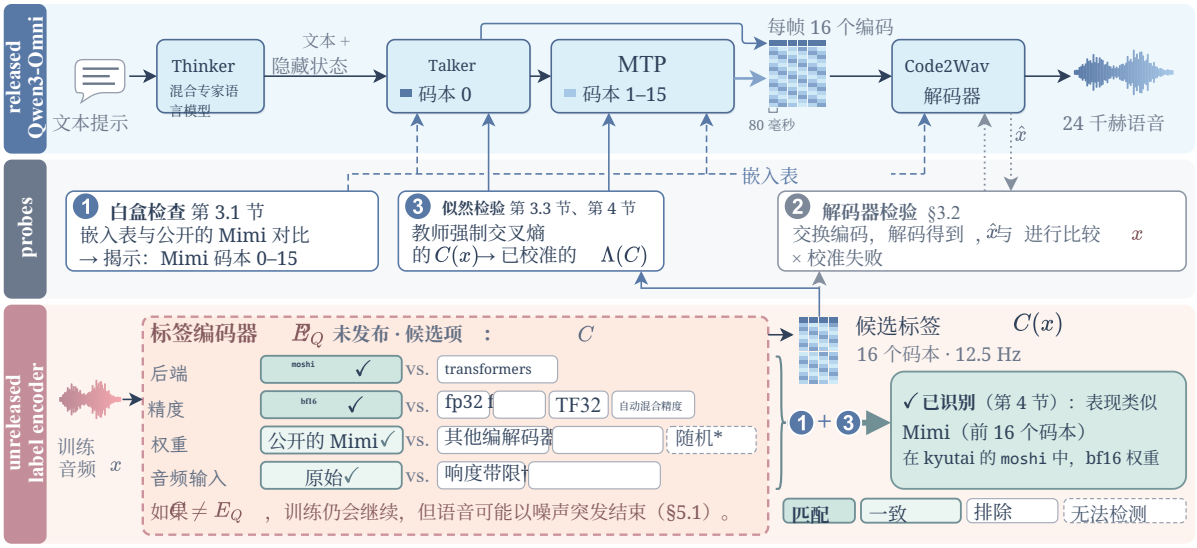

图 1：Qwen3-Omni 发布的语音路径（上方）、我们识别出的未发布标签编码器 E_Q（下方），以及连接两者的三个探针（中间）。为 Talker 和 Code2Wav 制作训练标签的编码器尚未发布；我们用候选编码器 C 替代它进行检验，每行对应一种设置（第 2 节和第 4节）。“一致”标记符合证据，但尚未得到确证。探针 1，即白盒检查（第 3.1节），将 Talker、MTP 和 Code2Wav 的码本嵌入表与公开的 Mimi 码本进行比较，从而排除其他编解码器。探针 2，即解码器检验（第 3.2节），替换单个编码，再用 Code2Wav 将其解码为 $\hat{x}$，并检验候选编码器的量化裕度能否预测 $\hat{x}$ 与音频 x 之间的偏离程度；该检验未通过校准，因此我们不将其作为证据（灰色）。探针 3，即似然检验（第 3.3 节和第 4节），在 Talker（码本 0）和 MTP（码本 1–15）下，使用教师强制对 $C(x)$ 进行评分，从而排除其他普通标记。 *我们只能排除这两种微调方向；随机权重扰动会改变码本 1–4 中 2–10% 的编码，却无法被检测到，因此我们识别的是行为，而非位精确权重。 †只有对 20% 或更多数据进行带限的情况才能被排除。在码网格中，深色顶行是码本 0，其余各行是码本 1–15。

图 2 展示了由此得出的方案。[1] 其中各部分的状态各不相同：编码器及其精度是我们在已发布模型中确认的；尾部填充则是我们的建议，因为最后一帧并不能说明 Qwen 如何进行编码（第 4 节）。这对于 transformers 的 `MimiModel` 很重要，否则它会在编码器内部对隐藏状态进行零填充；由此产生的最后一帧编码通常会解码成噪声突发，而微调后的模型会学着复现这种现象（第 5.1 节）。我们将每段话语填充到其长度的两倍。若将批次填充到其中最长条目的长度，该条目就不会被填充：在批次大小为 8 时，仍有 6% 的话语以噪声突发结尾，而非 1%。在 LoRA 语音适配中，我们的测试表明，精度并不重要，填充才重要。

[1]代码：https://github.com/elpsykongloo/qwen3-omni-talker-finetune（配方、标签_检查、Talker_微调）。数据：https://huggingface.co/collections/mifanbushipeicai/qwen3-omni-talker-finetune-6aec41e50ac465e3f0ef9222（所有变体的测试集编码、Talker 对数似然、13 个校准模型、示例声音）。

2 设置

符号与假设。 E_Q 是 Qwen 用于生成 Talker 标签的编码器（不是 Thinker 的音频编码器）。我们的参考值是公开的 Mimi 编码器（`kyutai/mimi`，取其 32 个码本中的前 16 个），使用 fp32 和 transformers（Wolf 等人，2020年）运行；我们将其输出称为*fp32 参考编码*。我们检验四个假设：（i）相同编码器：E_Q 是公开编码器，但可能采用不同的数值实现（精度、库）运行；（ii）带预处理的相同编码器：（i）加上固定的预处理步骤，例如带限或响度归一化；（iii）修改后的编码器：使用相同的量化器和代码 ID，但编码器网络经过微调或被替换；（iv）不同的分词器：张量形状相同但无关的分词器。

候选编码器。 除了参考值之外，我们还构建了以已知方式接近参考值的编码器（表1，第一列）：对权重进行随机扰动，使码本 1–4 中 2%、5% 或 10% 的编码发生变化（根据目标命名，例如“随机 5%”）；微调点则沿着中文或英文语音中的真实微调方向移动权重，



图 2：与 Qwen3-Omni Talker 匹配的标签配方。绿色标注表示我们在已发布模型中识别出的内容，珊瑚色标注表示我们的建议，蓝色标注表示标签格式。

以及采用 bf16 的 transformers 模型、带限输入和响度归一化输入。第 4 节增加了实现变体。附录 A介绍了构建过程，附录 B 列出了每种变体的生成方式。构建候选项时未参考任何 Qwen 得分。

数据。 校准模型基于 VCTK（Yamagishi 等人，2019年）（mic1）和 AISHELL-3（Shi 等人，2021）中 263 位说话人的数据进行训练，这些数据以 24 千赫切分为 10 秒片段。Talker 测试使用来自 64 位留出说话人的 512 条话语（2–9 秒、0.49 小时、22,295 帧），每位说话人 8 条（留出的是说话人而非文件；附录A）。所有置信区间均为 95% 置信区间，采用说话人级别的自助法（$B = 1000$），在各模型间共享重采样权重，因此差异是配对的。

3 标签编码器是 Mimi 的编码器吗?

3.1 白盒证据

Qwen3-Omni 检查点没有编码器或量化器，但每个生成器都针对每个码本设有一个学习得到的嵌入表（每个表有 2048 行）。如果这些编码是 Mimi 的，那么每个表都应能根据 Mimi 对应层的码本向量预测出来。我们用 Mimi 的码本向量进行岭回归 k to 预测每个 Qwen 表，并报告 5 折交叉验证结果R^2（表2）。每个 Qwen 表都能根据对应的 Mimi 层预测出来，而无法根据其他层预测；打乱行顺序后，拟合效果便不复存在。作为负对照，Qwen3-TTS-Tokenizer-12Hz（Hu 等人，2026年）是一种架构完全相同的编解码器（225 个张量的名称和形状完全一致，其中 223 个张量的值不同），却无法预测出任何结果。这排除了使用不同分词器的可能性（iv）。但它无法区分编码器相同（i）与编码器经过修改（iii）这两种情况：即使是经过微调但量化器被冻结的编码器，也会通过检验。

3.2 基于解码器的测试无法提供有效信息

Code2Wav 是用配对数据训练的 $(E_Q(x)\ x)$，且没有文本或说话人条件，因此它是一个天然的检验对象。我们将 fp32 参考值中的单个编码替换为次近码字，并根据各编码器自身的量化边界能否预测解码器会抵抗哪些替换，为每个编码器评分。Code2Wav 将 fp32 参考值排在所有候选项之前，这看起来支持同编码器假设。但校准结果却显示并非如此：三个从头训练的 Code2Wav 架构解码器（其中一个只使用随机 5% 的标签训练）也都将参考值排在第一位；而随机种子的影响约为标签影响的四倍（附录 C）。该测试无法检测到规模相当于随机 5% 或更小的编码器变化，因此我们不将其作为证据：解码器几乎不包含关于编码器量化边界的信息。

3.3 校准后的 Talker 似然

Talker 和 MTP 使用 E_Q 的编码进行交叉熵训练，因此标签差异会直接体现在它们的似然中。对于每个话语，我们使用每个候选编码器C 对音频进行编码，并在教师强制下，将编码输入 Talker（码本 0）和 MTP（码本 1–15），同时使用普通的朗读提示（“请朗读以下句子：”或其中文对应版本）和预设语音 Ethan；使用声音 Chelsie 得出的结论相同。我们报告

$$\Lambda(C) = \frac{1}{T}\sum_{t}\sum_{k=0}^{15}\Big[\log p\big(c^{\mathrm{ref}}_{t,k}\big) - \log p\big(c^{C}_{t,k}\big)\Big] \tag{1}$$

每帧纳特，其中 c^{ref} 是 fp32 参考编码； Λ（C）< 0 表示模型偏好 C的编码，而不是参考编码（附录A中的实现检查）。

候选编码器		一致率	Λ基于以下标签训练的校准模型 fp32 随机种子 1 / 随机种子 2	随机 5%	transformers bf16	带限 带限	带限, bf16	Talker Λ
随机扰动	2%	0.979	+0.051 / 0.052	**−0.096**	+0.038	+0.114	+0.070	+0.026
	5%	0.948	+0.096 / 0.095		+0.074	+0.091	+0.099	+0.023
	10%	0.905	+0.239 / 0.241	−0.007	+0.128	+0.280	+0.166	+0.077
微调点	中文	0.917	+0.318 / 0。291	+0.233	+0.287	+0.518	+0.420	+0.245
	英语	0.917	+0.226 / 0.211	+0.146	+0.130	+0.218	+0.175	+0.143
精度	transformers bf16	0.878	+0.145 / 0.156	+0.080	**−0.309**	−0.148	**−0.397**	**−0.509**
输入	带限	0.601	+1.145 / 1。140	+1.127	+1.029	**−5.154**	**−4.130**	−0.743
	响度归一化	0.327	+3.65 / 3.66	+3.57	+3.63	+4.63	+4.43	+2.77
交叉熵，参考代码		—	83.5 / 83.4	83.7	83.7	87.8	87.1	58.2

表 1：候选编码器。一致率：在 512 条测试语句中，码本 1–4 里的编码与 fp32 参考值相同的比例。 Λ （纳特/帧，16 个码本），使用基于一种标签变体训练的校准模型以及 Talker 得出的结果。单元格颜色：模型更偏好候选编码器而非 fp32 参考值（$\Lambda < 0$）；模型会对其进行惩罚（$\Lambda > 0$）；颜色越深表示 $|\Lambda|$ 越大（对数刻度）。fp32模型训练了两次；第二个随机种子（灰色）显示噪声。加框：模型对自身标签的评分。粗体：正文中讨论的偏好。

嵌入表 codebooks		0 0.5 1	R^2
Qwen3-Omni	Code2Wav 0–15		0.25–0.65
	Talker 0		0.45
	MTP 1–15		0.77–0.89
null	行打乱		≤ -0.03
	其他层		≤ -0.03
对照	Qwen3-TTS 编解码器		≤ 0

表 2：白盒检查：从 Mimi 的码本向量预测每个 Qwen 码本嵌入表的交叉验证 R^2 （条形表示各码本的取值范围）。空值是根据打乱顺序的行或另一个 Mimi 层进行预测（取各层的最大值）；对照组来自 Qwen3-TTS-Tokenizer-12Hz，它采用与 Mimi 相同的架构，但权重不同。

某些编码可能只是更容易预测，这会使 Λ 无论标签如何都产生偏差。因此，我们使用 Talker 的因子分解训练小型代码语言模型（针对码本 0 使用时间 Transformer，针对码本 1–15 使用深度 Transformer；不输入文本），每个模型都基于由同一个编码器编码的 107.6 小时训练集进行训练，并以相同方式评分。在 C的标签上训练的模型应偏好 C；在相同标签上使用不同随机种子训练的两个模型则用于衡量噪声。

表1 表明，这项测试在解码器测试失效的情况下仍然有效：随机种子的影响约为 0.01，而使用随机 5% 或 transformers bf16 标签训练的模型对这些标签的偏好分别高出 0.27 和 0.31，约为前者的 40 倍。对照 Talker 列与校准列来看：

•**半精度**。Talker 偏好 transformers bf16 编码（-0.509，置信区间$[-0.654, -0.384]$）。所有使用 fp32 标签训练的模型都会惩罚这些编码，或对它们的偏好弱得多（置信区间下界 ≥ -0.238；只有使用完全限带输入训练的模型达到该下界）；使用 bf16 标签训练的模型则表现得与 Talker 一致。

• **不进行响度归一化：** 包括 Talker 在内的所有模型，都会对响度归一化后的编码施加很重的惩罚。

•**不沿着我们构建的微调方向：** Talker 对中文微调点 (+0.245) 的惩罚程度与 fp32 模型相当，而在其标签上训练的模型则更偏好这些标签（-0.234，置信区间 $[-0.31, -0.16]$）。

• **随机扰动不存在剂量反应：** Talker 对三种随机扰动的惩罚都很小（+0.026、 +0.023、 +0.077，最多占其交叉熵的 0.13%）。只有当扰动编码器的标签恰好沿着该扰动方向时，模型才会偏好它，因此随机探测无法检测未知变化。

•**带限基本可以排除**。Talker 在测试语音上更偏好带限编码（-0.743），但检查 Talker 自行生成的语音后发现，其标签中包含的带限材料不足 20%（附录 D）。

4 哪种实现?

“半精度”并非只有一种形式：不同库在内核选择、类型转换点以及量化器中的距离计算方面存在差异；而在残差量化中，一个编码发生翻转就会改变后续的所有码本。我们使用 22 种实现变体对测试集进行编码（附录 B），并基于这些标签进一步训练了校准模型。

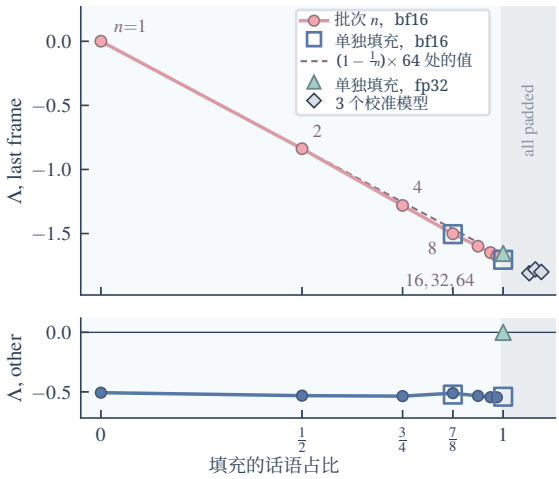

图 3：最后一帧承载了批量大小效应（Talker Λ，声音 Ethan，纳特/帧；两部分均按所有帧取平均）。上图：在大小为 n 的批次中，比例为 $1-1/n$ 的话语会被填充，而末帧部分随之变化（$1-1/n$ 倍于其在 $n=64$ 处的值，虚线）。以 bf16（方块：如同在批次大小为 8 时填充，以及填充至其长度的两倍）或 fp32（三角形）填充的单条话语落在同一条线上；在不含部分帧的 10 秒片段上训练的三个校准模型，对填充后的 fp32 编码给出的评分也相同（菱形）。下图：其余帧在 bf16 下保持在 -0.51 到 -0.55，在 fp32 下则与参考值相同。

最后一帧会造成较大的非诊断性影响。Mimi 编码器是因果的：追加零值不会改变任何内容，只有最后一帧会变（中央处理器 fp32，其他位置的最大差异为 4×10^{-7}）。transformers 会在每个卷积中填充最后一个不完整帧；当输入经过填充后，深层看到的则是网络对静音的实际响应。按批次编码时，除最长话语外，其余话语都会经过填充；这正是我们在初次分析中发现，批处理 bf16 看起来最像 Qwen 的原因。图 3 将 Λ 拆分为两部分：对于批大小 n，末帧部分等于填充最后一帧的影响乘以 $1-1/n$（预测值：-0.84、-1.26、-1.47，对应 $n=2$、4、8；测量值：-0.84、-1.28、-1.50），而对于任何批大小，其他帧都保持在 -0.51 至 -0.55。每个校准模型，包括仅用未填充的 fp32 编码训练的模型，对填充后的最后一帧的偏好都与 Talker 相差不多（-1.81、-1.78、-1.80 对比 -1.66），因为它们的 10 秒训练片段从不以部分帧结束。因此，最后一帧无法说明 Qwen 是如何编码的；从这里开始，我们会将它从 Λ 中排除。

批大小时，两次运行中都发生翻转的编码各自受到 0.74 纳特的偏好，而仅在一次运行中发生翻转的编码受到的偏好约为 0。在训练集上，moshi bf16 编码与 fp32 参考值在 82% 的编码上相同，与 transformers bf16 编码在码本 1–4 中 85% 的编码上相同。

Talker 对实现的排序，就像一个在以下标签上训练的模型moshi bf16 标签。 图 4 显示，Talker 和一个在以下标签上训练的校准模型moshi bf16 标签给出了相同的排序：moshi 采用 bf16 权重（无论是否进行流式处理）时遥遥领先，其次是 transformers bf16 变体和 moshi采用自动混合精度时的结果，再其次是 fp16、fp32 和 TF32。Talker 的数值约为校准模型的 0.6 倍；由于两个模型的尺度不同，我们只比较排序和符号。其他校准模型都未呈现这种模式：对于在 transformers bf16、fp16 或 fp32 标签上训练的模型，moshibf16 编码比 fp32 参考编码更难预测。关键在于将权重转换为 bf16：moshi在自动混合精度下使用 fp32 权重时，表现与 transformers 变体相似。编码级分解表明，Talker 的偏好源于 bf16 舍入的系统性部分：在同一 bf16 编码器的两次运行中，使用不同

稳健性检查。 该结果不取决于 GPU，尽管编码会有所不同：在另一块 GPU 上进行编码会使码本 1–4 中 4–7% 的 bf16 编码发生变化，而 Talker 对两组编码的偏好相同；在全部 16 个码本中，约四分之一的编码会发生变化（附录 D）。该指纹约束了实现，但不约束权重：使用经过扰动的 moshi 权重生成的 bf16 标签训练校准模型后，仍能保留 Talker 的偏好特征，因此无法据此限定 E_Q的权重与公开权重相差多远。作为一项不涉及解码或重新编码的独立检查，Talker 自身采样的编码与使用 moshi bf16 标签训练的校准模型拟合得最好（相较于使用 fp32 标签训练的模型高出 0.683 纳特/帧，置信区间 $[0.60, 0.75]$）。在 FLEURS（标签_检查（第5.2节）中，使用 Conneau 等人的2023年年）数据进行检查时，该配方在 Talker 的全部 10 种语音语言中都优于 transformers 标签，因此这一识

(a)

校准模型，标签来自		transformers			moshi	
实现		fp32	fp16	bf16	bf16	**Talker**
moshi bf16	离线	+0.559	+0.504	+0.104	**−1.955**	**−1.212**
	流式处理	+0.543	+0.489	+0.098	−1.949	−1.209
bf16 / 自动混合精度	transformers，即时	+0.169	+0.115	−0.287	−0.869	−0.511
	transformers，缩放点积注意力	+0.147	+0.111	−0.314	−0.887	−0.507
	transformers，自动混合精		+0.129	−0.186	−0.723	−0.483
	moshi ，自动混合精度	+0.181	+0.112	−0.249	−0.815	−0.478
fp16	transformers	+0.027	−0.046	−0.028	−0.081	−0.040
TF32	transformers	+0.013	+0.001	−0.003	+0.398	+0.146

(b)

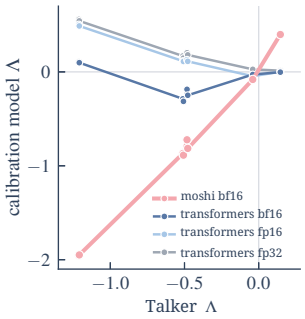

图 4：实现指纹（Λ 不包括最后一帧，纳特/帧）。(a) 按 Talker 的 Λ 顺序排列的编码器实现（表 4）（声音为 Ethan；Chelsie 的差异最多为 0.02），由四个校准模型和 Talker 评分；单元格颜色和帧与表 1相同。Talker 列的模式仅与使用 `moshi` bf16 标签训练的模型一致。(b) 将相同数值与 Talker 的 Λ进行比较：只有该模型与 Talker 的排序一致。

别不仅适用于中文和英语。附录 D 给出了详细信息。

总结。 白盒检查排除了分词器不同这一可能性 (iv)。在假设 (i) 下（该假设允许存在数值实现差异），E_Q的表现与公开的 Mimi 编码器（前 16 个码本）一致，即使用 kyutai 的 moshi 软件包运行，并将权重转换为 bf16。固定预处理（响度归一化；对 20% 或更多数据进行带限）以及我们采用的两个微调方向均已排除。无法排除其他方向上的权重变化，无论其幅度多大：似然比较只能说明两个候选中哪一个更接近这些标签，而随机探测方向几乎与未知变化正交。我们识别的是行为，而非位精确权重。

5 微调 Mimi 标签

5.1 末帧陷阱

transformers 与 kyutai。 kyutai 的参考编码器会在网络处理前，通过调用`pad_for_conv1d` `inmoshi/models/compression.py`，将输入填充到帧大小（1920 个采样点）的整数倍；而 transformers 的`MimiModel` 会让每个步幅卷积自行填充边缘，和 EnCodec（Défossez 等人，2023年）一样。两者都会生成 $\lceil n/1920 \rceil$帧。对每个步幅卷积的输入进行零填充，并不等同于用静音填充音频：经过前几层后，静音不再是零，因此最后一帧是根据不同的隐藏状态编码得到的。在 512 条测试语句上（中央处理器 fp32，32 个码本），两者在较早帧中的词元一致率为 99.99%，但在最后一帧中仅为 5.5%；结果完全相同的 6 条话语，其长度都是 1920 的整数倍。同样的机制也会导致右侧填充输入的批量编码与单独编码不一致（harshaljanjani，2026年），因为较短的项在结束后会遇到真实的零值：使用 transformers、每批 8 条话语时，较早帧完全一致，而最后一帧的词元一致率为 15.6%（64 条话语）。在编码前将输入填充到 1920 个采样点的整数倍后，transformers 在所有帧上都与`moshi` 一致（512 条话语中有 508 条完全相同），批量编码和单独编码也完全一致（100%）。我们已向 transformers 报告此问题，并建议通过这项改动来修复。[2]

对 Qwen3-Omni 微调的影响。我们使用 LoRA（Hu 等人，2022年）对 Talker 进行适配，在一次运行中适配两种新声音：AISHELL-3 说话人 SSB0033（中文，23.4 分钟），占用预设语音 Chelsie 的位置；以及 VCTK 说话人 p239（英语，23.5 分钟），占用 Aiden 的位置，每个标签集使用 3 个训练随机种子。各次运行之间只有标签不同：不带填充的 transformers fp32（库的默认设置）、不带填充的 transformers bf16、不带填充的 `moshi` bf16，以及我们的配方（`moshi` bf16，采用尾部填充）；另一组运行则加入了采用尾部填充的 transformers fp32 和 bf16。不带填充的 transformers bf16 是社区训练代码实际采用的设置（Hert4，2026年；wjm9765，2026年）。我们针对每种声音的 40 个留出句子测量可懂度（字符错误率/词错误率）、

[2] https://github.com/huggingface/transformers/ 问题/50396

		不进行填充			尾部填充			
标签通过以下方式编码		transformers fp32	transformers bf16	moshi	**moshi bf16 (配方)**	transformers fp32	transformers bf16	real 录音
Qwen3-Omni	响亮末帧（%）	49/55	44/53	0/0.8	0/0.3	0/0	0.3/0.3	0/2.5
	末帧跳变（分贝）	+2				−4.9/−5.7	−5.6/−5.8	−1.3/+0.7
	字符错误率 / 词错误率（%）	3.08/1.36	3.19/1.24	3.22/1.24	3.11/1.16	3.45/1.44	3.45/1.24	2.35/1.44
	相似度（sim.）	.969/.958	.968/.958	.970/.959	.969/.958	.969/.958	.968/.956	—
CSM-1B	响亮末帧（%）	46	50	0.6	—	0	—	—
	末帧跳变（分贝）	+6.6	+6.5	−7.2	—	−6.9	—	—
	词错误率（%）/ 相似度	2.9/.957	2.7/.944	2.0/.957	—	1.7/.953	—	1.4/—

表3：标签对 LoRA 微调的影响。Qwen3-Omni：两种声音、3 个训练随机种子、使用 bf16 的 moshi、中文/英文；CSM-1B：Unsloth 笔记本、VCTK p239、2 个训练随机种子、使用 fp32 的 moshi、英语。响亮末帧：生成句子中最后 80 毫秒相对于该句最响的 80 毫秒高于 −20 分贝的比例（柱状图范围为 0 至 60%；上方柱对应中文）；末帧跳变：最后 80 毫秒相对于前一帧的响度水平（珊瑚色：上升；蓝色：正常衰减）。灰色行：可懂度（字符错误率/词错误率）和说话人相似度。所有生成均成功停止。

说话人相似度，以及最后 80 毫秒的能量相对于同一句子中最响的 80 毫秒的比值；80 毫秒窗口从音频末尾开始计算。若最后一帧高于 −20分贝，则视为响亮。附录E 介绍训练和评估设置，以及各项比率如何随此阈值变化。

首轮运行的四组标签集在可懂度、说话人相似度、时长和停止方面没有差异（表3）：与配方相比，所有差异的句子级自助法置信区间都包含 0（字符错误率差异 -0.23 至 $+0.27$ 个百分点，相似度差异 ≤ 0.001）。对于两组经过填充的 transformers 标签集，字符错误率、词错误率和相似度与配方之间的差异也都具有包含 0 的置信区间。然而，在未填充的 transformers 标签中，约一半的生成句子以异常帧结尾，即一段短暂的噪声突发：对所有句子取平均后，最后 80 毫秒在中文中上升了 22 分贝（从接近静音的 −47 分贝上升至 −25 分贝），在英语中上升了 7 分贝（从 −27 上升至 −20 分贝；图 5）。这种突发噪声已经存在于标签中：使用 Code2Wav 解码后，约一半未填充的 transformers fp32 训练标签的结尾高于 −20 分贝（中文 54%，英语 50%）；相比之下，moshi 或填充标签以及真实训练录音的比例最高仅为 1.5%；经过微调的 Talker 以大致相同的频率出现结尾突发噪声。使用 `moshi` 标签时，最后一帧会正常衰减；使用 fp32 或 bf16 的填充 transformers 标签时也是如此：仅填充操作就能消除噪声突发。

CSM-1B。 CSM-1B（Sesame，2025年）同样使用 Mimi 编码，其 transformers 训练路径在不填充的情况下对编码进行编码。它显示出相同的问题：未填充的 transformers 标签的最后一帧需要付出已发布模型多出 119 纳特，而`moshi` 标签在 LoRA 微调后，约一半的生成结果会以响亮末帧结尾；填充会消除这两个问题（表3；详情见附录 E）。

5.2 工具包与合并检查点

标签_检查 并Talker_微调。 `label_check`会对 Talker 中的一组标签进行评分，并给出 Λ 与参考编码的对比结果，包括有无最后一帧的情况；这样可以立即看出标签是否经过填充，以及它们与哪种实现相似。`talker_微调` 使用配方准备标签，在 Talker 和 MTP 上训练 LoRA（或稠密）适配器，合成语音，将适配器合并到采用原始布局的完整检查点中，并评估字符错误率/词错误率、说话人相似度、话语结束时的噪声以及停止情况。在我们的测试中（附录 E），每位说话人只需 2.5分钟的语音，就能呈现出其声音；声音由 Talker LoRA 保留，而非 MTP。要试听或评估 16码本Mimi编码，Code2Wav 比 Mimi 自带的解码器效果更好（附录 G ）。

聊天中的合并检查点。每种语言提问 20 个问题后，由 23 分钟 LoRA 合并得到的完整模型，能够以新的声音回答文本问题和中文口语问题（相似度为 0.98），且没有嘈杂的结尾。出现了两个副作用。第一，LoRA 会改变 Talker 的共享权重，因此预设 ID Chelsie 和 Aiden 也会使用新的声音。第二，对于英语口语问题，英语语音会逐渐偏向中文目标说话人（p239 的声音与其自身回答文本问题时的相似度从 0.995 降至 0.81）：

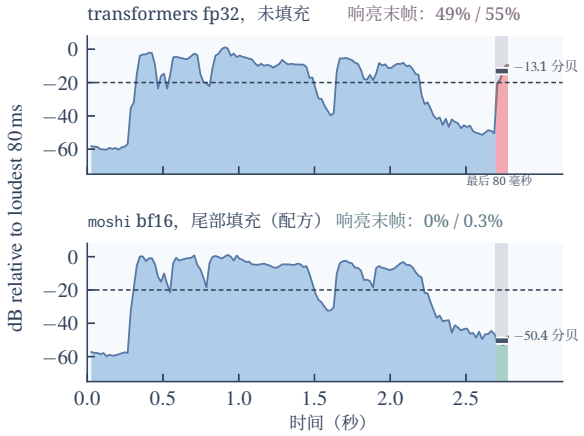

图5：LoRA 语音适配后生成的一条留出中文句子每 20 毫秒的能量。使用相同文本和采样种子，分别采用未填充的 transformers fp32 标签（上）和该配方（下）；相对于句子中最响的 80 毫秒，以分贝表示。由于模型不同，两段话略有差异。柱形表示阴影标出的最后 80 毫秒的电平，虚线 −20 分贝。使用 transformers 标签时，最后 80 毫秒的能量跃升 36 分贝（从 −49 升至 −13分贝；中文平均值为+22 分贝）；使用该配方时，则降至 −50分贝。百分比：最后 80 毫秒高于 −20分贝的生成句子占比（中文 / 英语，3 个训练随机种子的平均值；见表3）。

因此，在每种语言只有一名说话者的情况下，LoRA 似乎更倾向于将声音与提示词的语言绑定，而非与说话人 ID 绑定。对更多说话者的实验（附录F）让我们对多语言语音提出以下建议：每种语言至少训练两名不同性别的说话者，并让每名说话者都完成全部步数；如果无法做到，则添加同一语言的口语发言轮次。该工具包支持这两种做法。

6 相关工作

Mimi（Défossez 等，2024）沿袭了 SoundStream/EnCodec 残差量化编解码器的路线（Zeghidour 等，2022；Défossez 等，2023），并采用语义优先的码本；Qwen3-Omni（Xu 等，2025b）以 12.5 Hz 使用前 16 个码本，而 Qwen2.5-Omni 使用的是另一种分词器（Xu 等，2025a）。撰写本文时，Qwen3.5-Omni（Qwen 团队，2026）尚未开放权重。从 2025 年 10 月到 2026 年 6 月，Qwen3-Omni GitHub 讨论中多次有人询问 Talker 的标签由哪个编码器生成（wcy1122, 2025; Joshon-Smith, 2025; aryan-rathore10, 2026; evelynyhc, 2026），但始终没有官方答复。社区提出了编解码器是 Mimi 这一猜测：在 Qwen2.5-Omni 代码仓库中，abrar360 提出了这一猜测，SeungyounShin 则通过使用 Mimi 解码器解码 Qwen3-Omni 编码，以及考察 Talker/MTP 损失对其进行了验证（abrar360，2025；Seungy-ounShin，2025）。Hert4 和 SCA 项目发布的 Talker 训练代码已使用 Mimi 生成标签（Hert4，2026；wjm9765，2026），Unsloth 于 2026 年添加了 Talker 和 MTP 微调内核（Unsloth，2026），但没有提供生成标签的配方。近期基于 Qwen3-Omni 构建的全双工系统，要么保持原始 Talker 冻结（Chang 等人，2026），要么使用“the Mimi encoder”生成的标签重新训练它（Huang 等人，2026）。我们的贡献是检验这一判断究竟有多准确、哪种实现与之匹配、为何直观的基于解码器的测试会失败，以及使用默认 transformers 编码器会导致下游出现什么问题。

即使编解码器已知，同一段音频的编码也并非唯一。Liu 等人(2025年指出，编码会随上下文和不可感知的扰动而变化，而且提高其一致性有助于改进编解码器语言模型；我们研究其如何随数值精度、编码器实现和最后一帧的填充而变化。在 transformers 中，据报道，批次编码和单条 Mimi 编码会产生不同的编码（avishaiElmakies，2024年；harshal-janjani，2026年），但没有与 kyutai 的编码器进行比较，也未关联到最后一帧。以往也有人根据已发布模型推断未公开的细节，例如从合并列表推断 BPE 分词器的训练数据混合比例，并在使用已知混合比例训练的分词器上进行验证（Hayase 等人，2024年）；我们则根据 Talker 赋予的似然识别未发布的标签编码器，并在使用已知标签训练的模型上进行验证。

7 结论

Qwen3-Omni 的 Talker 表现得仿佛其标签来自使用 moshi 软件包和 bf16 权重运行的 kyutai Mimi 编码器（前 16 个码本）；我们识别的是这种行为，而非位精确权重。基于解码器的测试看似令人信服，但校准失败，因此应先在使用已知标签训练的模型上检验此类测试。transformers 的 MimiModel 与 kyutai 的编码器在最后一帧之前的编码有 99.99% 一致；然而，使用未填充标签对其进行 LoRA 微调后，Qwen3-Omni 的句子中有 44–55% 会以噪声突发结尾，CSM-1B 也出现了同样的问题；在编码前填充输入即可消除这些问题。

局限性

这是一种行为识别。它无法证明对于每个输入， $E_Q(x) = \text{Mimi}(x)$，也无法证明权重位级完全一致；按照设计，我们无法检测未构造的方向上的权重变化。在不同 GPU、PyTorch 版本和张量形状下，码本 1–4 中的 bf16 编码有 3–7% 不同；在不同 GPU 和批大小下，全部 16 个码本中的编码约有 24% 不同。由于我们不知道 Qwen 使用的硬件、软件和批处理方式，因此无法逐个编码复现其标签；在我们的测试中，这并不影响结果：Talker 对任何一块 GPU 生成的编码都没有偏好，bf16 和 fp32 标签也产生了相同的 LoRA 结果。下游研究涵盖了使用两名说话者和约 47 分钟语音进行 LoRA 语音适配（漂移研究使用八名说话者）；全量微调、新语言和表现风格尚未测试。使用朗读提示词训练的 LoRA 也会改变预设语音；除非每种语言都使用多名同一性别的说话者，并训练足够步数，否则在以疑问语气说话时，英语语音会退回为中文语音。带限结果仅能排除占比达到或超过 20% 的情况。

伦理声明

我们依照各自的许可使用 AISHELL-3（Apache-2.0）、VCTK 0.92（CC BY 4.0）和 FLEURS（CC BY4.0）、Mimi 权重（CC BY 4.0）、Qwen3-Omni（Apache-2.0）和 CSM-1B（Apache-2.0）；评估模型（Paraformer、Parakeet-TDT、Whisper、WavLM-SV）和 Qwen3-TTS-Tokenizer-12Hz 对照则遵循各自的许可。我们发布的工具包能够通过几分钟的录音，让语音模型适配新的声音，因此可能被用于在未经他人同意的情况下模仿其声音。我们发布的八种示例声音来自 AISHELL-3 和 VCTK 的说话人，并遵循这些语料库的许可；我们要求用户仅在获得对方同意后，才将模型适配为模仿他人的声音。

参考值

abrar360. 2025年。 使用 Mimi 识别 Qwen3-Omni Talker 编解码器的提案。GitHub issue QwenLM/Qwen2.5-Omni#219 上的评论。

arun-elr。2026年。 Qwen3-Omni 流式处理音频在每个 Code2Wav 分块边界丢失 555 个采样点。GitHub issue mstar-project/mstar#428。

aryan-rathore10。2026年。 为新的声音和语言微调 Qwen3-Omni Talker。GitHub讨论 QwenLM/Qwen3-Omni#154。

avishaiElmakies. 2024年。 Mimi 模型在使用批量编码与单次编码时产生不同的输出。GitHub issue huggingface/transformers#35166。

Cheng-Kuang Chang、Alexander H. Liu、Brian Kingsbury、Hilde Kuehne、Rogerio Feris、Samuel Thomas、James Glass 和 Kai-Wei Chang。2026年。D2：基于半双工主干网络构建高能力全双工口语语言模型。 *arXiv* 预印本 *arXiv:2610.17124*。

Sanyuan Chen、Chengyi Wang、Zhengyang Chen、Yu Wu、Shujie Liu、Zhuo Chen 等。2022年。WavLM：面向全栈语音处理的大规模自监督预训练。*IEEE*《信号处理精选主题期刊》，16(6)：1505–1518。Alexis Conneau、Min Ma、Simran Khanuja、Yu Zhang、Vera Axelrod、Siddharth Dalmia 等。2023年。FLEURS：通用语音表征的少样本学习评估。载于*2022*年 *IEEE*口语技术研讨会（*SLT*），第798–805页。Alexandre Défossez、Jade Copet、Gabriel Synnaeve 和 Yossi Adi。2023年。高保真神经音频压缩。 《机器学习研究汇刊》。Alexandre Défossez、Laurent Mazaré、Manu Orsini、Amélie Royer、Patrick Pérez、Hervé Jégou、Edouard Grave 和 Neil Zeghidour。2024年。Moshi：用于实时对话的语音文本基础模型。 *arXiv* 预印本 *arXiv:2410.00037*。edwingao28。2026年。 Streamed Code2Wav 在每个分块边界丢弃555个采样点，导致可听见的咔嗒声。GitHub issue sgl-project/sglang-omni#1354。evelynyhc。2026年。 [wav2codec 是否已经开源? 我想微调 Talker]。GitHub讨论 QwenLM/Qwen3-Omni#176。中文。Zhifu Gao、Shiliang Zhang、Ian McLoughlin 和 Zhijie Yan。2022年。 Paraformer：用于非自回归端到端语音识别的快速准确并行Transformer。载于*Interspeech 2022*年论文集，第2063–2067页。Daniel Han、Michael Han 和 Unsloth团队。2023年。Unsloth。GitHub代码仓库 unslothai/unsloth。harshaljanjani。2026年。 MIMI 编码器由于缺少对填充掩码的支持，在批次输入与单个输入下产生不同的输出。GitHub issue hug- gingface/transformers#43377。Jonathan Hayase、Alisa Liu、Yejin Choi、Sewoong Oh 和 Noah A. Smith。2024年。 数据混合推断攻击：BPE分词器揭示训练数据

组合。载于 神经信息处理系统进展，第37卷，第8956–8983页。

Hert4. 2026年。 Qwen-Omni-Training-Talker。GitHub代码仓库 Hert4/Qwen-Omni-Training-Talker。

Edward J. Hu、Yelong Shen、Phillip Wallis、Zeyuan Allen-Zhu、Yuanzhi Li、Shean Wang、Lu Wang 和 Weizhu Chen。2022年。 LoRA：大语言模型的低秩适配。发表于国际学习表征会议。

Hangrui Hu、Xinfa Zhu、Ting He、DakeGuo、Bin Zhang、Xiong Wang 等。2026年。 Qwen3-TTS 技术报告。 *arXiv* 预印本 *arXiv:2601.15621*。

Muye Huang、Lingling Zhang、Xingyu Yu、Lei Shi、Zhanyu Ma、Jun Xu 等人。2026年。 DuplexOmni：面向全双工交互的实时聆听、视觉感知、思考与说话。*arXiv* 预印本 *arXiv:2606.09186*。

JoshonSmith。2025年。 [如何微调 Talker 部分]。GitHub讨论 QwenLM/Qwen3-Omni#149。中文。

Wenrui Liu、Zhifang Guo、Jin Xu、Yuanjun Lv、Yunfei Chu、Zemin Liu 和 Junyang Lin。2025年。分析并缓解神经编解码器语言模型中离散语音令牌的不一致性。发表于第*63*届计算语言学协会年会论文集（第一卷：长篇论文），第31035–31046页，奥地利维也纳。计算语言学协会。Qwen 团队。2026年。Qwen3.5-Omni 技术报告。*arXiv* 预印本 *arXiv:2604.15804*。Alec Radford、Jong Wook Kim、Tao Xu、Greg Brockman、Christine McLeavey 和 Ilya Sutskever。2023年。通过大规模弱监督实现稳健的语音识别。发表于第*40*届国际机器学习大会论文集，第28492–28518页。Sesame。2025年。跨越对话声音的恐怖谷。Sesame 研究博客；Hugging Face 上的模型：sesame/csm-1b。SeungyounShin。2025年。Qwen3-Omni Talker 和基于 Mimi 编码的 MTP 损失（前向_test.py）。GitHub 问题 QwenLM/Qwen2.5-Omni#219 和 unslothai/unsloth#3636 上的评论。Yao Shi、Hui Bu、Xin Xu、Shaoji Zhang 和 Ming Li。2021年。AISHELL-3：多说话人普通话文本转语音语料库。发表于*Interspeech 2021* 论文集，第2756–2760页。Cees H. Taal、Richard C. Hendriks、Richard Heusdens 和 Jesper Jensen。2011年。一种用于预测时频加权噪声语音可懂度的算法。*IEEE* 音频、语音与语言处理汇刊，19(7):2125–2136。

Unsloth。2026年。新增对 Qwen3-Omni 的支持，并优化 MTP 微调。GitHub 拉取请求 unslothai/unsloth#4235。

wcy1122。2025年。 面向 Talker 的 Qwen3-Omni 编解码器。GitHub讨论 QwenLM/Qwen3-Omni#70。

wjm9765。2026年。 SCA：Qwen3-Omni Talker 微调代码。GitHub代码仓库 wjm9765/SCA。

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond,Clement Delangue,Anthony Moi, 等. 2020年。Transformers：最先进的自然语言处理。载于*2020*年自然语言处理经验方法会议论文集：系统演示，第38–45页。

Hainan Xu, Fei Jia, Somshubra Majumdar, He Huang, Shinji Watanabe 和 Boris Ginsburg。2023年。通过联合预测词元和时长实现高效序列转导。载于第*40*届国际机器学习会议论文集，第38462–38484页。Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai 等。2025a。Qwen2.5-Omni 技术报告。*arXiv* 预印本 *arXiv:2503.20215*。Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He 等。2025b。Qwen3-Omni 技术报告。*arXiv* 预印本 *arXiv:2509.17765*。Junichi Yamagishi, Christophe Veaux 和 Kirsten MacDonald。2019年。CSTR VCTK 语料库：用于 CSTR 语音克隆工具包的英语多说话人语料库（版本 0.92）。爱丁堡大学语音技术研究中心。Peiqi Yin, Jiangyun Zhu, Han Gao, Chenguang Zheng, Yongxiang Huang, Taichang Zhou 等。2026年。vLLM-Omni：面向任意到任意多模态模型的完全解耦式服务。*arXiv* 预印本 *arXiv:2602.02204*。Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund 和 Marco Tagliasacchi。2022年。SoundStream：端到端神经音频编解码器。*IEEE/ACM* 音频、语音与语言处理汇刊，30：495–507。

候选编码器与评分详情

候选编码器。 随机扰动会向权重添加高斯噪声，并按每个张量的范数缩放；通过二分法确定缩放幅度，使码本 1–4 中有 2%、5% 和 10% 的编码发生变化。微调点朝真实的微调方向插值（在冻结量化器的情况下，使用中文或英语语音执行 25 步 Adam），直至不一致率达到 8%。带限处理会将输入从 24 千赫重采样至 16 千赫，再重采样回 24 千赫；响度归一化会将其缩放至 −23LUFS。

留出说话人。 170 名 AISHELL-3 测试集说话人也出现在训练集中，因此我们按说话人而非文件进行留出。

Talker 评分。 我们的实现遵循 SeungyounShin 的 `forward_test.py`(Se-ungyounShin，2025年) 以及 transformers 类 `Qwen3OmniMoeForConditionalGeneration`的代码路径；贪心生成后再通过教师强制重新评分，可在 97–100% 的码本 0 帧上复现 argmax。

B 编码器变体及实现结果

表 4 描述了本报告中的每个标签变体及其结果所在位置； Λ 实现变体的值列于图 4中。所有变体均使用 PyTorch 2.9.1 和 transformers 4.57.6 进行编码。在 H100 上使用相同的 PyTorch 时，`moshi` 0.2.13 逐位复现了我们的 H100 配方编码（512 个话语中的 512 个）。

C 解码器检验详情

对于 fp32 参考编码中的每个编码，我们替换为第二近的码字，并判断解码器输出是否偏离原始音频（计算接下来 3 帧上的对数梅尔 L1 距离）。在某个编码器的标签上训练的解码器，应当在选择接近量化边界时最偏好该编码器的选择；因此，我们针对每个参考编码器 R，使用 R 自身的量化裕度预测解码器偏好的曲线下面积（码本 1–4；在功率试验中，码本 0 和码本 ≥ 9不携带信号）。Code2Wav 为 fp32 参考值提供了最高的曲线下面积（0.701），高于所有候选项（例如 $\Delta A = -0.0104$，随机 5% 的置信区间为 $[-0.0129, -0.0080]$ ），并且沿每个扰动方向进行扫描时，峰值都接近参考值。

不过，参考值正好位于候选集的中心，因此无论标签如何，解码器都可能将其排在第一位。因此，我们从头开始训练了三个 Code2Wav 架构解码器，训练 30,000 步：两个使用 fp32 参考值标签，随机种子不同；另一个使用随机 5% 标签，随机种子与第一个相同。三个解码器都将参考值排在第一位，包括从未见过参考值标签的那个。随机种子的影响约为标签影响的四倍：对于 ΔA 随机 5% 而言，数值为 $+0.0077$ ，而对比之下为 $+0.0019$；沿随机 5% 方向的峰值位置则是 $+1.13$ ，而对比之下为 $+0.32$。在一个 Code2Wav 副本上使用随机 5% 标签进行微调训练 4,000 步也没有改变其峰值。避免以参考值探针为锚点的对称统计量，在这个尺度下没有检验能力。

D 稳健性检查

带限。 在测试语音上，Talker 的 $\Lambda = -0.743$ 带限输入表现得像是部分带限训练数据。在 Talker 自行生成的语音上（由 Code2Wav 解码的 190 个句子，即处于其自身的训练分布中），Talker 对带限输入的惩罚（$+0.923$）高于 fp32模型（$+0.393$）；而仅使用 20% 带限数据块训练的模型已经偏好带限输入（-0.651，置信区间 $[-0.83, -0.47]$）。因此，Talker 的标签中带限材料少于 20%；更小的比例尚未进行测试，而测试语音上的 -0.743 仍未得到解释。

GPU 和 PyTorch 版本。 在 H100 而非 RTX PRO 6000 上对相同变体进行编码，会使码本 1–4 中 4–7% 的 bf16 编码发生变化（fp32 编码几乎完全相同），这与在同一 GPU 上改变张量形状的影响相近。Talker 对 H100 编码的偏好程度相同（Λ 的 moshi bf16 代码： -1.232 与 -1.212），而使用 PRO 6000 编码训练的校准模型对两者的偏好也相同（-1.978 与 -1.955）。GPU 之间有所不同的那部分 bf16 舍入并不是 Talker 学到的内容，因此可以在任意 GPU 上复现标签，但不会逐编码完全一致：在 H100 而不是 PRO 6000 上重新编码 38,741 个训练数据块（moshi 和 PyTorch 版本相同，批大小为 16 而非 64）后，码本 0 的编码有 99.0% 保持一致，码本 1–4 中有 92.8% 保持一致，全部 16 个码本中有 76.0% 保持一致；只有 38% 的帧在全部 16 个码本中都一致。本比较未将 GPU 与批大小的影响分离。将 H100 上的 PyTorch 从 2.9.1 更改为 2.15，会使相近比例的配方编码发生变化（码本 1–4 中为 6.5%，码本 0 中为 0.9%）；我们未使用 Talker 对这些编码进行评分。

扰动权重。 moshi bf16 的偏好边界能反映 E_Q的权重与公开权重相差多远？我们将 `moshi`的编码器权重扰动至 2% 和 10% 的 fp32 编码不一致程度，使用 bf16 进行编码，并分别在每种情况下训练一个校准模型。在运行之前，我们预先设定了解读方式：如果带扰动标签的模型不再偏好 `moshi` bf16 编码，那么该指纹便特定于权重，并可提供一个界限。表5展示了另一种结果：对 `moshi` bf16 编码的偏好变化了 $+7\%/-6\%$ ，且

标签变体		其生成方式	结果
图 4	moshi bf16	加载器.get_mimi，权重 .to(torch.bfloat16)	另见表1
	moshi bf16，流式处理	相同，每步一帧	
	transformers bf16，即时	bf16 模式下的 MimiModel，使用即时注意力	
	transformers bf16，缩放点积注意力	bf16 模式下的 MimiModel，使用缩放点积注意力	
	transformers +自动混合精度	MimiModel，bf16 自动混合精度下的 fp32 权重	
	moshi + 自动混合精度	loaders.get_mimi，bf16 自动混合精度下的 fp32 权重	
	transformers fp16	fp16 精度下的 MimiModel	
	transformers TF32	fp32 精度下的 MimiModel，使用 TF32 矩阵乘法	
其他位置	fp32 参考值	fp32 精度下的 MimiModel；所有 Λ 均相对于其编码	第 2
	moshi fp32	loaders.get_mimi 使用 fp32；除最后一帧外，均作为参考值	第 5.1
	transformers bf16，批次 n	n 条话语/批次，右侧填充（$n = 2$、.....、64）	图 3
	moshi bf16，受扰动的	权重加噪声（2% 或 10% 的编码发生变化），然后转换为 bf16	表 5
	transformers，已填充	fp32 或 bf16，输入尾部补零	表 3
	配方	moshi bf16，输入尾部补零	图 2，表3
	候选编码器	随机、微调、带限、响度归一化	第2节，表1

表 4：标签变体及其生成方式，按结果所属类别分组。所有变体都对 24 kHz 音频进行编码，保留前 16 个码本，除非另有说明，否则每条话语都单独编码，不进行填充。

比例分布最多变化 0.05。该分布由实现决定，并且对这一幅度的权重变化并不敏感，因此无法提供界限。成对比较同样没有帮助：Talker 偏好未经扰动的 moshi bf16 编码，但包括使用扰动标签训练的模型在内，每个模型都具有相同偏好，因为扰动标签通常更难建模。

Talker 自身样本。我们使用每种校准模型对 Talker 的原始采样编码（190 个句子）进行评分。所有模型对测试语音中各自标签的拟合程度几乎相同（对数似然为 -82.09 至 -82.27纳特/帧，带限版本除外），因此差异反映了样本与哪种标签更相似。相较于使用 fp32 标签训练的模型，使用 moshi bf16 标签训练的模型高出 0.683（置信区间 $[0.60, 0.75]$），使用 transformers bf16 标签训练的模型高出 0.235，使用另一个 fp32 随机种子训练的模型高出 0.205；而使用中文微调点和带限标签训练的模型则更低（$-0.618, -6.20$）。在真实 moshi bf16 代码上，相应差距为 0.715 和 0.443，与样本上观察到的 0.683 和 0.448 几乎相同。

其他语言。使用 标签_检查 FLEURS 上的（Conneau 等人，2023年）（Talker 的 10 种语音语言中各有 40 条测试语句，16kHz 音频重采样，基础模型，预设语音 Ethan），该配方在除最后一帧外的所有帧上，均优于 transformers fp32 标签 0.8–4.9 纳特/帧，优于 Transformer bf16 标签 0.4–4.0 纳特/帧（所有置信区间均不包含 0），且与未填充的 moshi bf16 标签（$|\Delta| \leq 0.09$，所有置信区间均包含 0）无法区分。填充会使 transformers 标签在每种语言中的末帧负对数似然降低 15–50 纳特。

E 微调设置与工具包消融实验

Qwen3-Omni 语音适配。SSB0033 有 458 条训练话语，p239 有 462 条，每位说话人各有 40 条留出话语。Thinker 保持冻结（提示词仅使用其令牌嵌入）。对 Talker 的注意力和共享专家，以及 MTP 的注意力和多层感知机应用秩 16 LoRA （ 630 万个参数），在 16 码本交叉熵和语音结束目标上以批次大小 8、学习率 10^{-4}（余弦退火）训练 800步。使用尾部填充的 transformers 标签进行的训练采用相同设置。Hert4 的脚本（Hert4，2026年）和 SCA 训练代码（wjm9765，2026年）都使用 Talker 的数据类型（bf16），通过 transformers 的 MimiModel 进行编码，且不进行填充。我们用 3 个随机种子为每条留出句子采样（top-k 50，温度 0.9），用 Code2Wav 解码，并使用 Paraformer（Gao 等人，2022年）（中文）测量字符错误率，用 Parakeet-TDT（Xu 等人，2023年）（英语）测量词错误率，用 WavLM-SV（Chen 等人，2022年）测量与留出录音的说话人相似度，并测量句子最后 80 毫秒内的能量（分贝），以该句子最响的 80 毫秒窗口为基准（窗口从音频末尾开始计数；高于 -20分贝即计为响亮末帧）。绝对比率取决于这一判定方式。我们还在 -30、-25、-20 和 -15分贝阈值，以及 40、80 和 160 毫秒窗口下重复了计数。在中文的所有设置下，moshi 标签（无论是否填充）产生的响亮末帧至多为 2%；在英语中，阈值为 -20 和 -15分贝时也至多为 2%。然而，p239 的留出录音以较高的相对能量结束（最后 80 毫秒平均为 10^{-4} 分贝；SSB0033 为 k 50分贝），因此较低的阈值会将正常的英语句尾计为响亮：使用该配方时，80 毫秒窗口下，阈值为 -20分贝时占 11%，阈值为 -30分贝时占 49%；160 毫秒窗口下，阈值为 -25分贝时占 90%。因此，英语中 moshi 标签产生较少响亮末帧这一结论仅在 -20 和 -15分贝阈值下成立。未填充的 transformers fp32 标签与该配方之间的差异，在阈值、窗口和语言的全部 24 种组合中均为正（对随机种子和句子进行自助法抽样；每个置信区间均不包含 0）。作为参考，使用预设语音的基础模型字符错误率为 12.8%，词错误率为 8.5%。

评分者		Λ 个 moshi bf16 代码	95% 置信区间	transformers bf16	moshi 自动混合精度	transformers fp16	transformers TF32
校准模型	基于 moshi bfloat16 标签训练	−1.955	[−2.35, −1.61]	0.45	0.42	0.04	−0.20
	相同，权重扰动 2%	−2.088	[−2.53, −1.73]	0.45	0.42	0.03	−0.21
	相同，权重扰动 10%	−1.832	[−2.28, −1.48]	0.40	0.37	0.01	−0.25
Talker	声音 Ethan	−1.213	[−1.42, −1.01]	0.42	0.39	0.03	−0.12

表5：指纹衰减：在 moshi 的 bf16 标签上训练的校准模型，无论是否使用扰动后的编码器权重，都保留了 Talker 的偏好特征（Λ 单位为纳特/帧，不包括最后一帧）。右侧四列给出 $\Lambda(c)/\Lambda$（moshi bf16）编码 c 来自指定实现：符号相同时为珊瑚色，不同时为蓝色；其他颜色同表1。

然而，p239 的留出录音以较高的相对能量结束（最后 80 毫秒平均为 −26分贝；SSB0033 为 −44 分贝），因此较低的阈值会将正常的英语句尾计为响亮：使用该配方时，80 毫秒窗口下，阈值为 −25 分贝时占 11%，阈值为 −30分贝时占 49%；160 毫秒窗口下，阈值为 −30分贝时占 90%。因此，英语中 moshi 标签产生较少响亮末帧这一结论仅在 −20 和 −15分贝阈值下成立。未填充的 transformers fp32 标签与该配方之间的差异，在阈值、窗口和语言的全部 24 种组合中均为正（对随机种子和句子进行自助法抽样；每个置信区间均不包含 0）。作为参考，使用预设语音的基础模型字符错误率为 12.8%，词错误率为 8.5%。

CSM-1B。 CSM 使用全部 32 个 Mimi 码本，而 transformers 的 CSM 训练流程会使用 编解码器_model.encode 对每个音频片段进行编码且不使用填充，这正是 Unsloth（Hanet al.，2023年）微调笔记本所采用的方式。在已发布模型（160 个 VCTK 话语）中，transformers 标签的最后一帧在 32 个码本上的总成本为 199.4 纳特，而使用 moshi fp32 标签时为 80.1（+119，置信区间 $[+109, +130]$）；对输入进行填充后，该数值变为 80.2（差值+0.08，置信区间 $[-0.23, +0.42]$）。其他帧不受影响（−0.003 纳特）。与 Qwen Talker 不同，CSM 更适合使用 fp32 标签：在其他帧上，transformers bf16 标签的成本为 +0.39 纳特/帧（置信区间 $[+0.21, +0.56]$）。因此，应根据似然对每个模型分别检查精度，而不能想当然地假定。微调表现出与 Talker 相同的症状（主干网络和深度解码器使用 LoRA 秩 32，400步，批次 8，学习率 2×10^{-4}；40 个留出句子各采样两次）：使用未填充的 transformers 标签时，约有一半生成结果会以响亮末帧结束，而使用填充后的 moshi 标签时几乎没有这种情况（fp32 未填充减去moshi：+46点，置信区间 $[+38, +53]$）。在本次采样中，词错误率和说话人相似度无法区分这两组标签

run		CER zh	WER en	相似度 zh / en	最后一帧响度 帧 中文 / 英文
声音	新标识符	3.3	3.7	.969/.960	0/0
	预设时段	3.8	3.0	.970/.963	0/0
data	10分钟	4.8	5.8	.965/.959	0/0
	5分钟	5.8	5.9	.966/.957	0/2.5
	2.5分钟	5.7	5.9	.968/.955	0/5
LoRA	仅 Talker	3.0	4.3	.966/.958	0/0
	仅 MTP	11.3	3.7	.962/.880	18/10
真实录音		2.5	3.4	—	0/2.5

表6： Talker_微调 运行结果（%；除非另有说明，每位说话人有 23 分钟语音；800步，一个随机种子，每位说话人的 40 个留出句子各采样一次）。单元格颜色：比最佳运行结果差（珊瑚色，颜色越深表示越差）；条形图与表3相同。真实录音使用相同的自动语音识别进行评分。约一个点的差异属于噪声范围。

尺寸。

Talker_调优。 Thinker 加载时未包含其解码器层，因为朗读训练只需要其嵌入：仅 Talker 就占用 7.2 GB，在 transformers 5.20 上进行 LoRA训练时，分配内存峰值为 14 GB，并且可在每进程 22 GB 的上限下运行（每步 0.45 秒），因此 24 GB 的显卡即可满足需求。表6 使用工具包重复了语音适配实验（一个随机种子，每个留出句子使用一个采样点；其评估器对中文使用 Paraformer，对英语使用 Whisper-large-v3-turbo（Radfordet al.，2023年））。新的说话人 ID 与替换后的预设语音效果同样良好；每位说话人只需 2.5分钟的语音即可获得相应声音，更多数据主要用于降低错误率；Talker LoRA 携带了该声音，而仅使用 MTP 的 LoRA 会使英语相似度保持在 0.88，并重新出现嘈杂的结尾。

F 聊天中的声音漂移

聊天中的合并检查点。 我们让由 23 分钟 LoRA 合并而成的完整模型，使用 transformers 针对每种语言回答 20 个单轮问题；Thinker 负责回答，Talker 负责朗读（每个回答约 50 秒）。对于两种语言的文本问题以及中文语音问题，回答与目标说话人的相似度均为 0.98，且没有嘈杂的结尾。预设 ID Chelsie 和 Aiden 也能用新的声音说话（与目标说话人的相似度分别为 0.98 和 0.97）。对于英语语音问题（由基础模型使用预设语音合成），声音会发生漂移：以每种声音自身文本问题回答的质心为基准，相似度从 p239 的 0.995 降至 0.81，从 Aiden 的 0.995 降至 0.74。基础模型回答相同的语音问题时，会保持 Aiden 的声音（0.994，而文本问题为 0.993），因此漂移来自 LoRA。发生漂移的英语回答使用的是中文目标说话人的声音（与 SSB0033 的相似度为 0.97；WavLM-SV 几乎无法将其与 Chelsie 区分开来，而新的说话人行正是从 Chelsie 这一声音开始的）。将跨语言的语音用户轮次随机配对并不能解决这一问题：加入 200 个此类轮次后（训练句子由基础模型使用两种预设语音朗读，并预先计算 Thinker 的隐藏状态），在一半采样点中，英语回答在所有条件下都会在两种声音之间切换，包括朗读条件（相似度为 0.88，对比 0.96）；而中文不受影响。vLLM-Omni（Yin 等人，2026年）0.30 为合并检查点提供服务且没有任何变化，在聊天请求中接收 说话人 参数（每种语言 10 个文本问题：新声音的相似度分别为 0.985 和 0.988；预设 ID 同样会使用新的声音）。

每种性别和语言不止一名说话者。为了检验单一说话者的解释，我们使用每种语言的四名说话者重新训练（AISHELL-3：女性 SSB0033 和 SSB0435，男性 SSB0603 和 SSB0590；VCTK：女性 p239 和 p236，男性 p259 和 p286；每人至多 23.5 分钟；女性 id 从 Chelsie 开始，男性 id 从 Ethan 开始），每次运行使用三个随机种子（图6 的第一行重复了 23 分钟的 LoRA）。每名说话者都以文本和语音形式回答其语言中的 20 个问题，并朗读 20 个留出句子。由于 LoRA 只作用于 Talker，Thinker 的回答和口语问题的隐藏状态只计算一次，供所有运行共享；我们合成了前 64 个回答词元（约 20 秒），每批次处理 120 个回答，并使用左侧填充的提示词

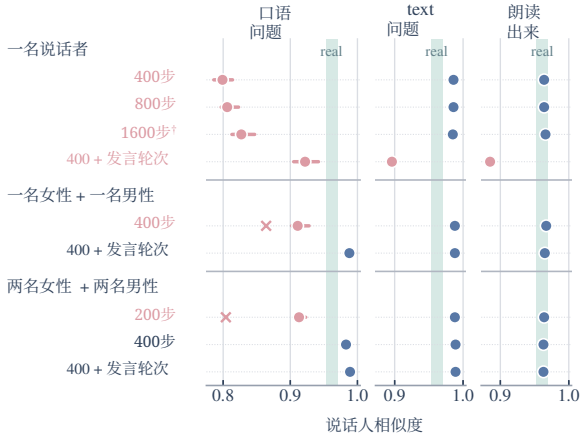

图 6：使用每种语言中不同说话人集合进行 LoRA训练后的英语声音（每位说话人的训练步数；发言轮次：一半采样点中包含同语言语音问题），每种提示词对应一个面板。圆点：与说话人自身留出录音的说话人相似度，取三个随机种子的平均值（柱状条：随机种子范围）；珊瑚色：低于真实录音的范围（绿色带），即声音发生了漂移。叉号：女性声音 p239。

†过拟合（字符错误率 22%，词错误率 15%）。中文声音在每次运行中都达到 0.95–0.99。

（在 fp32 下，批次和单条贪心解码得到的编码相同）。相似度是与说话者自身留出录音质心的相似度（真实录音，留一法：0.953–0.970）；英语词错误率来自 Whisper-large-v3-turbo。图6 展示了结果。每种语言只有一名说话者时，每个随机种子下都会发生声音漂移（60 条英语口语回答中有 59 条最接近 SSB0033），增加到两倍或四倍的步数也无济于事；步数增加到四倍还会导致过拟合（中文字符错误率为 19–22%，英语词错误率为 15–21%）。每种语言有两名女性和两名男性说话者、每名说话者训练 400步时，仅使用朗读提示词进行训练即可消除漂移（文本问题为 0.989，相比之下为 0.983），且错误率没有变化。同一批说话者每人训练 200步则不行：两种英语女性声音都漂移成中文女性声音（p239：0.804），男性声音则没有。每种语言只有一名女性和一名男性说话者时，只有女性声音发生漂移（p239：0.864）；在一半的采样点中加入同一语言的口语发言轮次可以解决这一问题（0.988），但若每种语言只有一名说话者，这些发言轮次也会导致英语文本回答和朗读发生漂移（分别为 0.896 和 0.885）。女性和男性 ID 使用不同的预设，因此这里无法区分性别与初始化的影响。这些结果构成第5.2节中的建议；该工具包支持这两种选项（每名说话者使用初始声音、同一语言的口语

解码器	对数梅尔 L1（分贝）相似度		字符错误率/词错误率 (%)	STOI
Code2Wav，16 cb.	3.2/3.0	.981/.982	3.3/3.0	.94/.86
Mimi，16 cb.	4.5/4.0	.965/.966	3.9/3.9	.92/.83
Mimi，32 cb.	3.9/3.6	.977/.980	3.3/2.9	.95/.86
原始音频	—	—	3.1/2.5	—

表 7：解码 512 条测试话语的配方编码（中文/英文；Code2Wav 的输出向后移位 555 个采样点）。相似度是与原始音频的相似度；短时客观可懂度（Taal 等人，2011年）。单元格颜色：比最佳解码器差（珊瑚色，颜色越深表示效果越差）。“cb.”：码本。

发言轮次）。同一名说话者使用两种语言录制的情况尚未测试。

G Code2Wav 作为 Mimi 编码的解码器

Code2Wav（trans- formers 实现）中的每个转置卷积都会在两端裁掉 $k - s$ 采样点，因此每帧输出比 1920 个采样点短 555 个（23 毫秒）。此前已有报告指出，流式解码也存在这一缺陷（e dwingao28，2026年；arun-elr，2026年）；左侧裁剪还会使输出比输入提前同样的 555 个采样点。与参考音频比较之前，应先将输出向后平移；否则，频谱和可懂度指标都会偏低。对齐后，Code2Wav 对 512 条测试语句的配方编码（16 个码本）的解码效果，与 Mimi 解码器使用全部 32 个码本时大致相当，且优于 Mimi 解码器使用相同 16 个码本时的效果（表 7）。对于从基础 Talker 采样的编码，两个解码器生成的声音相同（两种解码结果之间的相似度为 0.97/0.98）。因此，要试听或评估 16码本Mimi编码，我们使用 Code2Wav（仅评估客观指标；不进行听力测试）。这说明的是解码器，而非编码器（第 3.2 节）。