Day3

在day2的调研中，我对Omni or MLLM的架构方法进行了调研。同时，通过两篇综述对于模型的训练进行了强调和介绍。总的来说，MLLM需要pretrain and post training，其中后者包含指令微调和对齐调优（主要是RL为主）。MLLM 的训练核心是把非文本输入对齐到 LLM，然后让 LLM 按指令输出文本；Omni 的训练核心是在此基础上进一步加入多模态输出对齐、任意模态组合指令微调、跨模态生成和模态平衡。

在今天的调研中，我打算通过调研一些**经典的Omni家族的模型**，并且**从中看它们分别是如何组织模型训练的**，并且从中洞察Omni的训练目标、克服问题等。

NExT-GPT

> NExT-GPT:Any-to-AnyMultimodal LLM
>
> 2023年9月

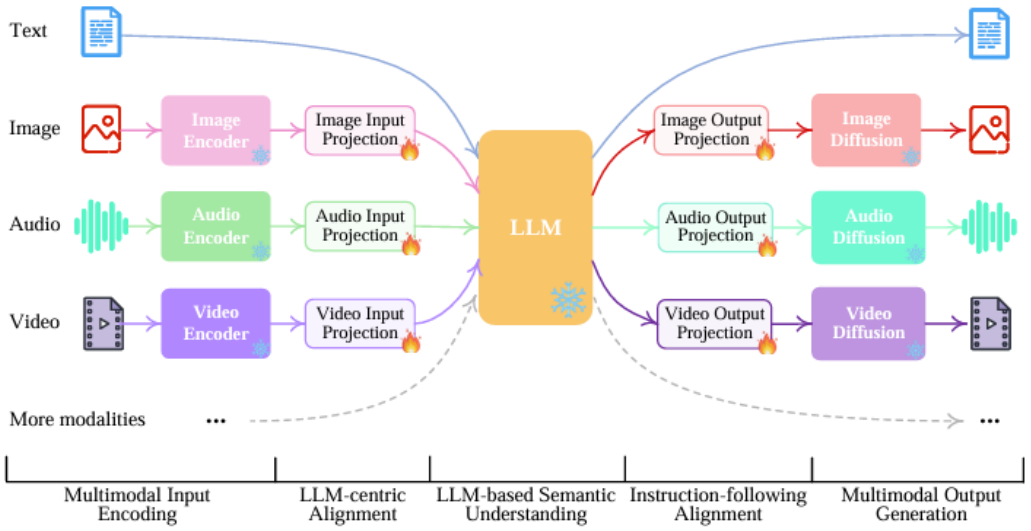


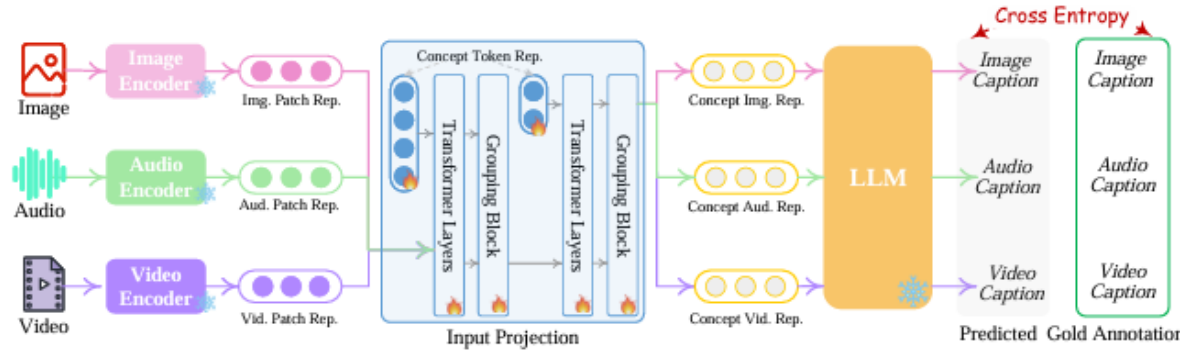
Figure 1. By connecting LLM with multimodal adaptors and diffusion decoders, NExT-GPT achieves universal multimodal understanding and any-to-any modality input and output. ❄and 🔥represent the frozen and trainable modules, respectively.

NExT-GPT被认为是十分经典的 representation-based generation training。前面的encoder and projection就是encode and connect(or called align)，然后LLM内部react之后， 输出特殊的 modality signal tokens，通过output之后把这些signal token的representation映射到 diffusion decoder 可理解的空间，最后由 Stable Diffusion、AudioLDM、ZeroScope 等外部生成器生成图像、音频、视频。

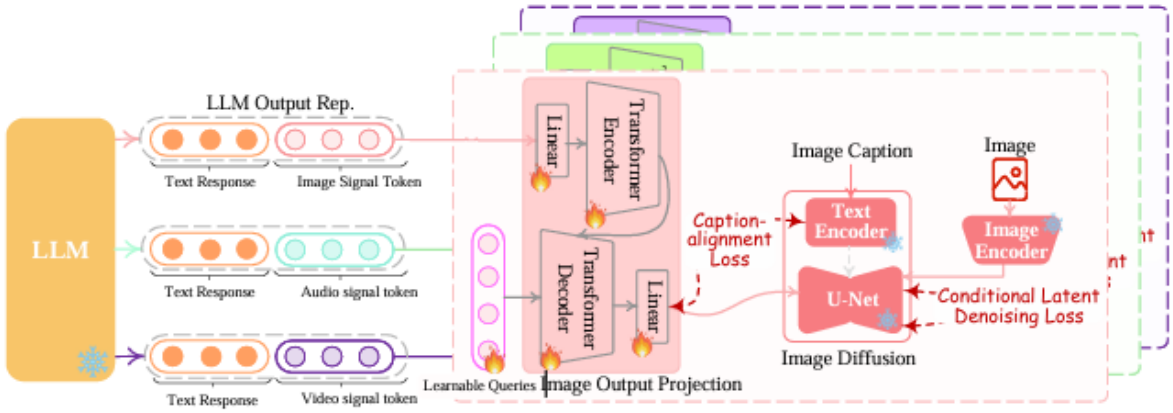
这貌似和AnyGPT所干的事情很像，但是又隐隐上感觉有些不同的地方：NExT-GPT 和 AnyGPT 都在LLM 输出之后接一个非文本生成模块。但是思考下来，区别在于：

- AnyGPT 让 LLM 生成目标模态 tokenizer 的离散内容 token
- NExT-GPT 让 LLM 生成触发某个模态生成的 signal token，并利用这个 token 的 hidden representation 作为生成条件

token 的性质完全不同。AnyGPT中生成的其他模态的token，是直接代表着对应的内容的，只不过是需要detokenize；但是NExT-GPT生成的signal token，更像是控制生成某种模态的控制信号，然后使用这些控制信号对应的hidden state作为diffusion decoder条件，所以它们不是其他模态目标object本体，而是充其量是它们的表征或是depiction。



(a) Encoding-side LLM-centric Alignment



(b) Decoding-side Instruction-following Alignment

NExT-GPT 采用 three-stage learning process: Stage-1 Encoding-side Alignment Learning、Stage-2 Decoding-side Alignment Learning、Stage-3 End-to-end Instruction Tuning。

第一阶段训练的是输入侧对齐, 解决的问题是如何让让非文本模态特征变成 Vicuna 能理解的 language-like representation，其实也就是align。NExT-GPT 在 encoder 后面加了一个 input projection layer。这个 projection 不是简单 linear layer，而是带有 learnable concept tokens 和 grouping mechanism 的模块。它把 patch-level multimodal features 聚合成 concept-level tokens，再送入 LLM。而训练的时候，选择的任务是：

`image / audio / video → input projection → frozen Vicuna → caption text`

论文明确说：给定一个 X 的表示，prompt frozen LLM 生成对应文本描述。这个阶段使用交叉熵损失，并且只有 input projection layer 是可训练的，NExT-GPT 其他部分全部 frozen。作者还说claim，它理解为训练一个 compatible multimodal tokenizer for the frozen LLM。

第二阶段训练的是输出侧对齐。作者引导LLM扔出一些Image等模态的special signal token，然后Stage-2 要训练 output projection layer，把 LLM 输出的 signal token representation 映射到 diffusion decoder 能理解的条件空间。这一阶段 loss 有三个部分：

1. **negative log-likelihood / cross-entropy**：让 LLM 产生正确的 signal tokens；
2. **caption alignment loss**：signal token hidden states 经过 output projection 后，与 diffusion text encoder 得到的 caption representation 做 L_2 对齐；
3. **conditional latent denoising loss**：沿用 diffusion model 的 latent denoising objective。

论文在 implementation details 里说，Stage-2 只有 output projection layers 是 learnable，其他部分 frozen。

最后是整体 instruction tuning，让模型真正能做 any-to-any instruction following。Stage-3使用 instruction-following datasets 训练整个 NExT-GPT，并且用 LoRA 微调 LLM 权重；同时，input projection 和 output projection 都是 trainable 的。训练目标包括两个部分：生成结果和 gold response 之间的 cross-entropy，以及 multimodal generation loss。

AnyGPT

> AnyGPT: Unified Multimodal LLM with Discrete Sequence Modeling
>
> 2024年2月

AnyGPT中，每一个模态使用对应的tokenizer，并且转化为token的时候，使用标识符来标记这一段为另一个模态的tokens。处理好之后，扩充LLM词表(使用可学习的embedding)，然后统一处理，训练的时候也是统一的next token prediction，并且对于生成内容进行detokenize。其中关于token的统一和处理，在day2调研中已涉及。

关于Multimodal Generation，文章中有一段更为细致的介绍：AnyGPT并不让 LLM 直接生成全部低层细节，而采用"语义 token 生成 + 感知细节解码"的两阶段方案。文章claim到：如果直接让 LLM 生成最终图像或音频，会有两个问题，一个是序列太长，一个是LLM 不擅长生成低层感知细节(LLM 更擅长建模语义、结构、逻辑和跨模态关系，不一定适合直接生成所有低层细节)。第一阶段由 language model，也就是LLaMA-2 7B来生成融合和对齐之后的 semantic-level multimodal tokens。第二阶段是non-autoregressive models 完成，把 LLM 生成的 semantic tokens 转换成high fidelity的最终模态内容。其中，visual是SEED tokens送到diffusion model里面（叫做seed token是因为tokenizer叫做SEED tokenizer），audio是使用的训过的Soundstorm的变体，音乐使用的是Encodec tokens送进Encodec decoder。

AnyGPT 的 training 核心思想可以一句话概括：把 image / speech / music 全部先离散化成 token，然后像训练普通 LLM 一样，用统一的自回归 next-token prediction 去训练扩展词表后的 LLaMA-2。论文claim这些来自不同模态的 token 会在语言模型中被训练到一个共享表示空间里。新增的 modality token embedding row 是在 LLM embedding matrix 里随机初始化，然后在后续 multimodal pretraining / instruction tuning 中学习出来的。

AnyGPT 没有设计新的 contrastive loss、matching loss、diffusion loss 或 multimodal alignment loss。它的核心训练目标就是标准自回归语言建模：

> Then the sequences are trained by the language model using the next token prediction training objective.

Training 分成两个主要阶段：pretraining + instruction tuning。预训练阶段的目标是先让各个模态和文本对齐。因为真实的 image-speech-music-text 全模态配对数据很少，所以作者采用 text-centric 策略：让 image-text、speech-text、music-text 分别和 text 对齐，然后通过 text 作为桥梁，间接实现多模态之间的互相对齐。

> To enable the generation from any modality to any other, it is crucial to have data that is well-aligned across these modalities. Unfortunately, such data is notably scarce. To address this challenge, we build a text-centric bi-modal alignment dataset. Here, text is employed as a vital intermediary to bridge the gap between various modalities.

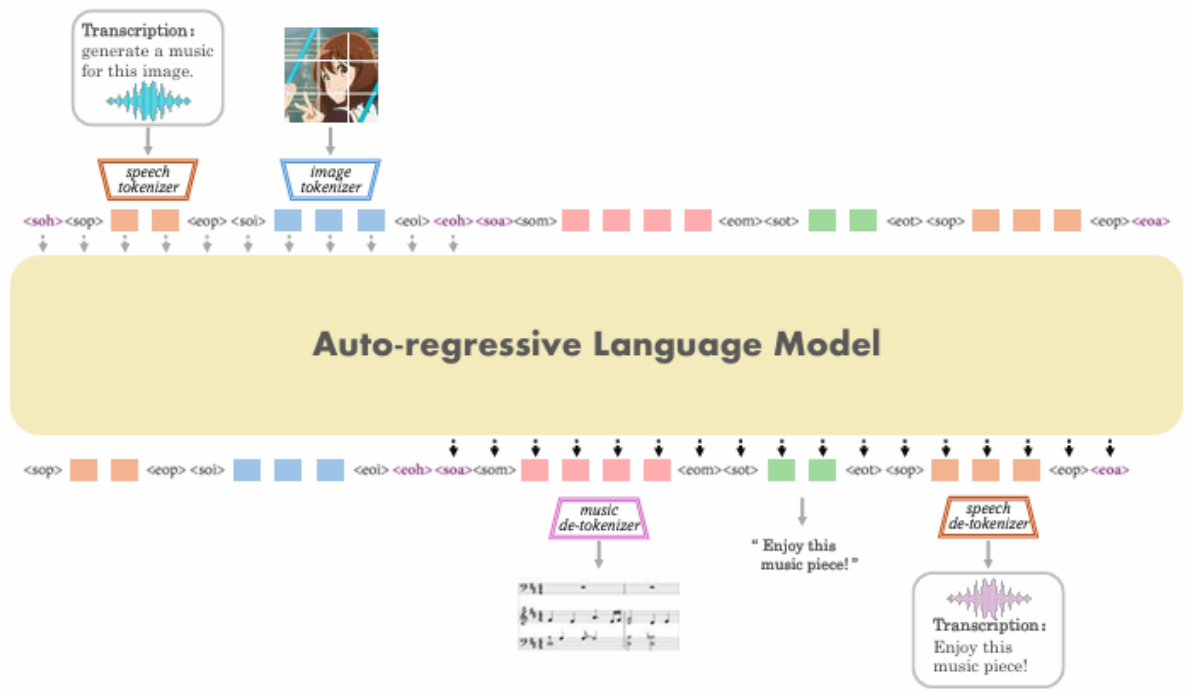
文章扩充了vocab以及LLaMA中的prediction layer，那么新增模态 token 会在 language model 中训练到共享表示空间，并且除 reshaping embedding matrix 和 prediction layer 之外，语言模型其余结构保持不变。

> We expand the vocabulary with new modality-specific tokens, and consequently extend the correspond ing embeddings and prediction layer, the newly incorporated parameters are initialized randomly. The tokens from all modalities combine to form a new vocabulary, where each modality is trained within the language model to align in a shared representational space. Apart from reshaping the embedding matrix and prediction layer, the rest of the language model remains unaltered.

关于LLaMA2是否是freeze的，从repo来看，预训练中是没有任何的freeze的，load pretrained ckpt后，并没有`requires_grad=False`之类的操作，或者是显式的LoRA之类的finetune：

```
trainer = Trainer(
    model=model,
    tokenizer=tokenizer,
    args=training_args,
    train_dataset=train_data if training_args.do_train else None,
    eval_dataset=val_data if training_args.do_eval else None,
    data_collator=data_collator
)
```

那么这个pretrain以text-centric的方式对齐了modalities，作者认为原来的pretrain数据有一定的噪声，因此继续使用更高质量的数据进行训练。第二阶段的instruction tuning同样也是对LLaMA2所有参数进行finetune。并且文章claim，在进行了指令微调之后，AnyGPT真正有了交互和按照输入指令来进行生成的能力。

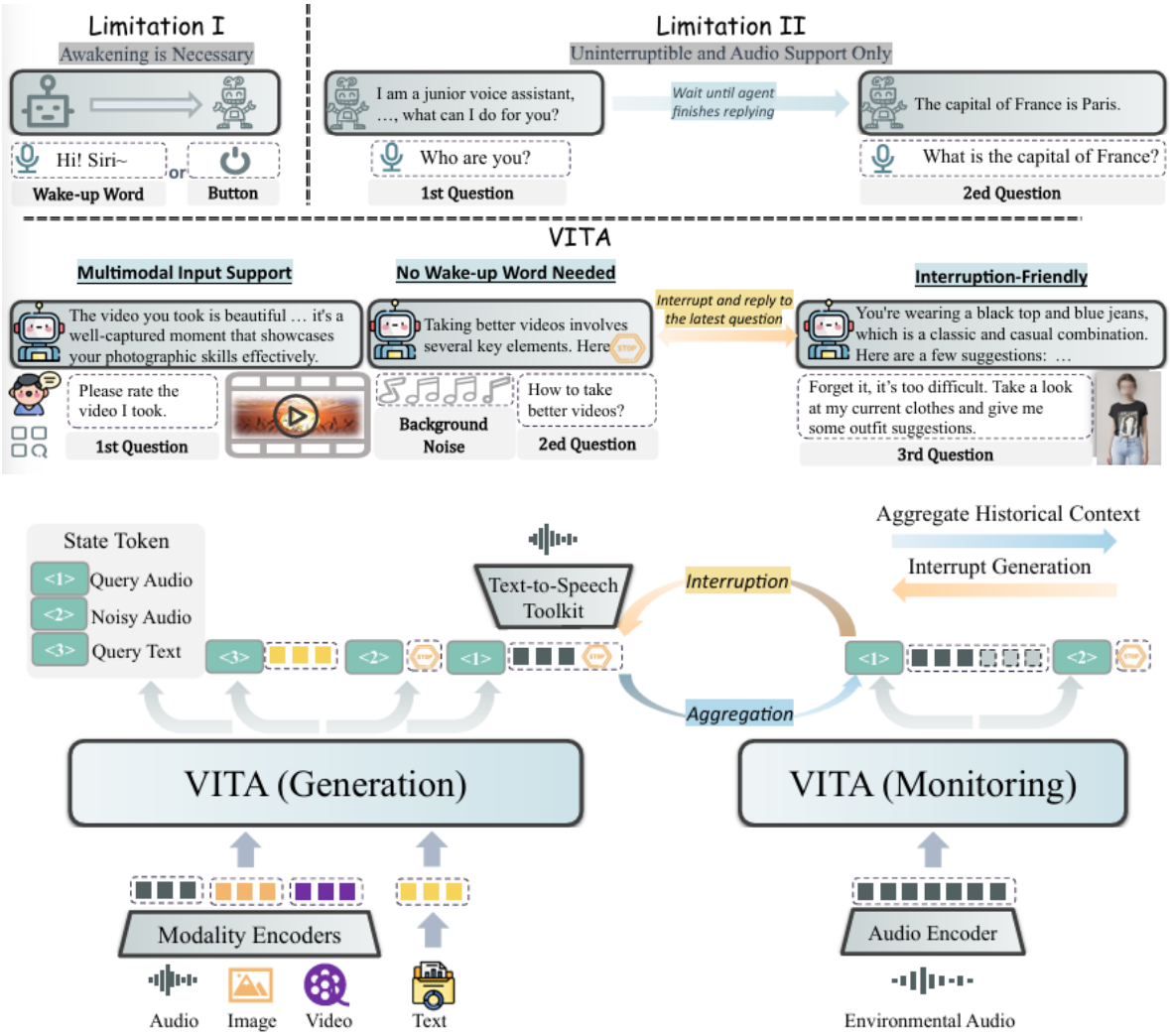


VITA

> VITA: Towards Open-Source Interactive Omni Multimodal LLM

这篇文章的position在摘要中说道：the first-ever open-source Multimodal Large Language Model (MLLM) adept at simultaneous processing and analysis of Video, Image, Text, and Audio modalities, and meanwhile has an advanced multimodal interactive experience. 文章position VITA是多模态交互式助手，想要解决对应产生的痛点：

- 传统系统通常需要 wake-up word 或按钮触发
- 当模型正在回答时，用户如果插入新的问题，系统往往不能自然中断并转向最新问题

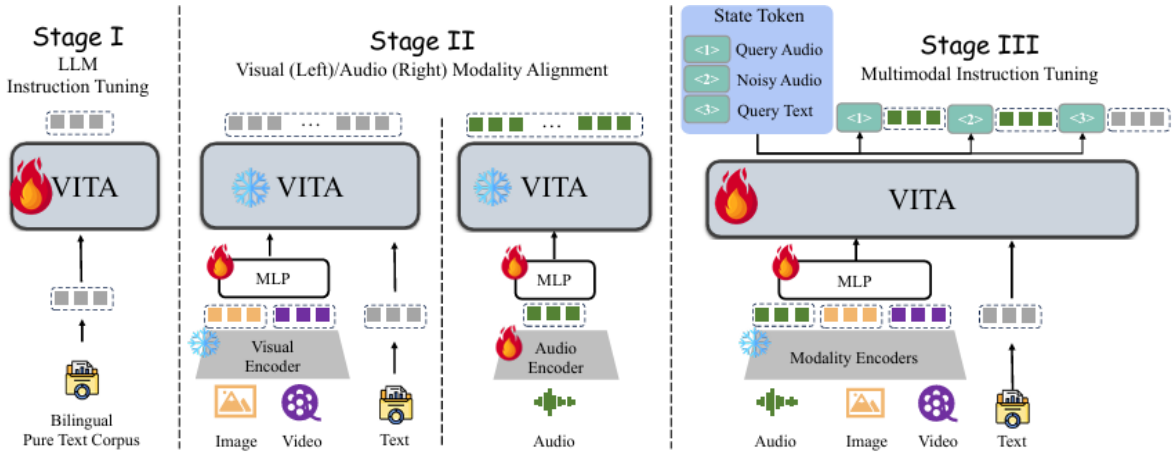


VITA选择Mixtral作为LLM，并且加入中文词表并且使用bilingual corpus记性instruction tuning。上图为VITA的架构，其中最有意思的，也是直接针对'语音交互助手'的，是VITA Monitoring端；并且架构中是有两个 VITA 实例。Generation model 负责处理用户当前 query；Monitoring model 在 Generation model 工作时持续监听环境声音。如果用户在模型回答过程中提出新的有效语音问题，Monitoring model 会聚合历史上下文并回答最新问题，同时当前 Generation model 被暂停，两个模型交换身份。换而言之，系统一边生成回答，一边实时 tracking/filtering 外部 query；当新问题出现时，需要停止当前生成、整合历史上下文并回答当前 query。

VITA 设计了三个 state tokens：

```
<1>: corresponds to the effective query audio,such as “what is the biggest
animal in
the world?”
<2>: corresponds to the noisy audio
<3>: corresponds to the query text, i.e.,the question given by the user in
text form.
```

训练时，VITA 把对应 state token 插在答案开头，让模型学习先判断当前输入属于哪种交互状态。论文claim这些 state token 被放在答案开头，用来让模型灵活处理不同交互行为。论文中还说明，推理时 `<2>` 会被当成一种特殊 EOS token；如果模型判断输入是 noisy audio，就直接终止生成。



第一阶段的目标是先把 Mixtral 变成一个更适合中英双语场景的 LLM。第二阶段是把视觉和音频模态接到 LLM 上。本质是alignment，也就是让 image/video/audio encoder 的输出变成 LLM 能理解的 token-like representation。只训练 visual connector。训练视觉connector的包含一些经典的任务，如VQA、captioning and description之类的；而训练audio使用的是automatic speech recognition and audio captioning任务以及对应的数据集。

VITA 还做了一个训练效率上的设计：对纯文本和图像数据，把上下文拼接到大约 6K tokens。图像会先拆成 local patches，再把多个 image-text pair 拼接起来；视频数据则按帧输入，不做拼接。作者认为这样可以统一不同 batch 中的数据长度，提高训练效率，也能从单图单问扩展到多图多问形式。和AnyGPT中sequence packing的操作(同一模态的样本拼在一起训)很相似。

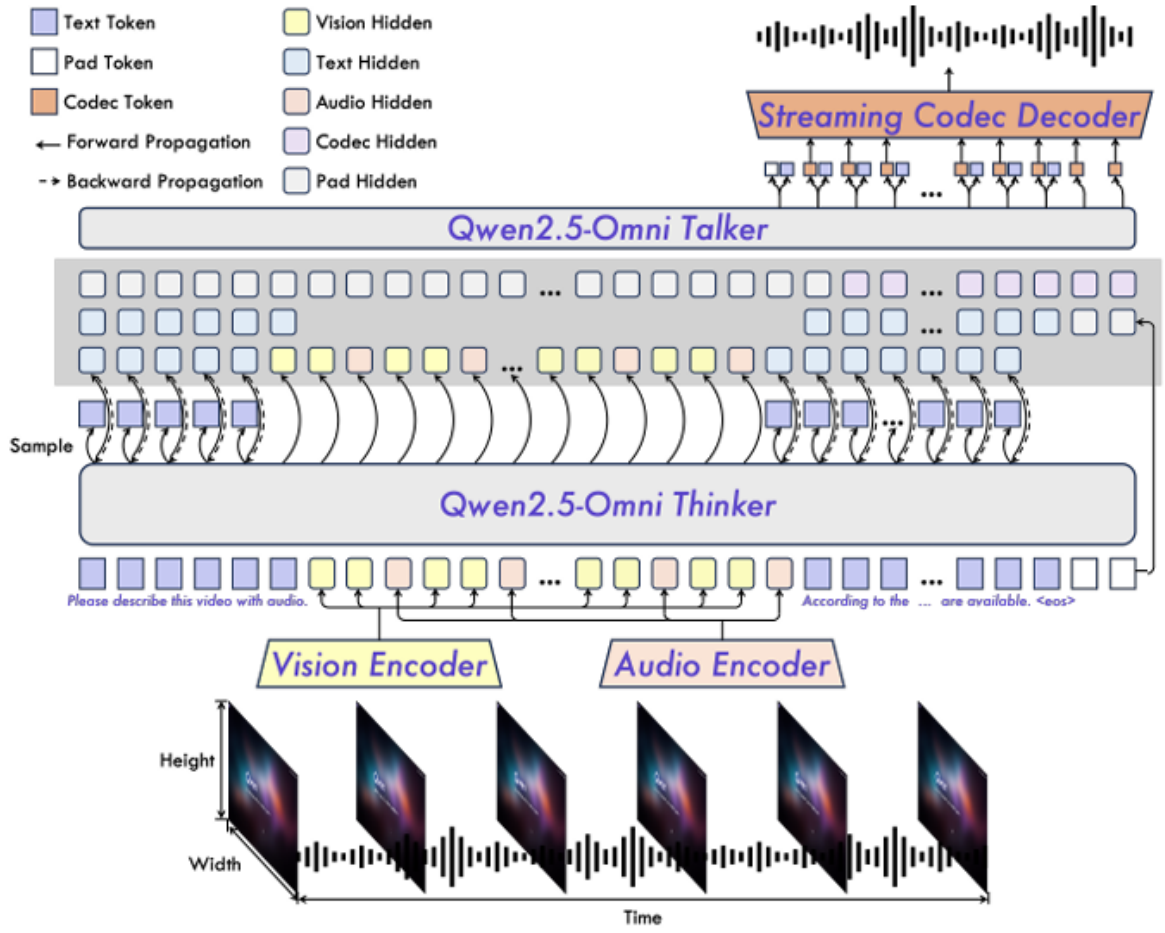
第三阶段让模型真正具备多模态指令跟随能力，并且能处理文本指令和语音指令。论文说这一阶段使用和 alignment 阶段类似的数据源，但做了几个关键改造。第一，大约一半问题被替换成音频版本。作者使用 GPT-SoVITS 等 TTS 工具，把文本问题转成语音问题，从而训练模型理解 audio query，并能在视觉/视频上下文中根据语音指令回答。第二，不同数据类型使用不同 system prompt。例如 image 输入时，system prompt 会要求模型严格根据图像内容回答；video 输入时则要求严格根据视频内容回答；纯文本输入时则按一般文本助手处理。这样做是为了避免模型混淆“基于视觉内容回答”和“凭自身知识回答”，也避免 image patches 与 video frames 的形式相似导致混淆。**第三，构造 noisy audio 数据**，从已有多模态和单模态 QA 数据的**答案**中随机抽取 474K 句子，把这些非问题文本通过 TTS 转成音频，作为**不需要回答**的 noisy audio negative samples。这样模型就可以学习并不是所有听到的内容都应该触发回答。

第三阶段训练时，论文明确说最终 instruction tuning 阶段不是只训 connector，而是让 connector 和 LLM 一起适配多模态指令分布。

Qwen2.5-Omni

> Qwen2.5-Omni Technical Report

摘要开篇定位工作为：an end-to-end multimodal model designed to perceive diverse modalities, including text, images, audio, and video, while simultane ously generating text and natural speech responses in a streaming manner. 报告提出 Thinker-Talker architecture：Thinker 是负责理解和文本生成的 LLM，Talker 是双轨自回归模型，使用 Thinker 的 hidden representations 生成 audio tokens；最终通过 streaming decoder 生成语音。



Thinker负责处理多模态输入，生成高层语义表示并且直接自回归生成文本；Talker是接受Thinker的hidden reepresentation和生成的文本embedding，自回归生成语音的codec token。论文强调 Talker 在训练和推理时都直接接收 Thinker 的高维表示，并共享 Thinker 的历史上下文

文章很重要的一个设计是 TMRoPE：Time-aligned Multimodal RoPE。视频有视觉帧，音频有连续语音，两者本来时间粒度不同。如果只是简单把 video token 和 audio token 拼起来，模型不知道“这一段声音”和“这一帧画面”在时间上是否对应。TMRoPE 的做法是把 RoPE 拆成三部分：temporal, height and width position. 对于 text，三个 position ID 相同，因此等价于普通 1D-RoPE。对于 audio，也使用相同 position ID，并引入绝对时间位置，其中 1 个 temporal ID 对应 40ms。对于 image，temporal ID 保持不变，height / width ID 根据图像空间位置变化。对于 video，视觉帧的 temporal ID 按真实时间递增，height / width 仍按照图像空间位置赋值。还使用 time-interleaving method：把带音频的视频按真实时间切成每 2 秒一个 chunk，在每个 2 秒 chunk 里，先放视觉表示，再放音频表示，从而让模型按时间顺序处理音视频信息.

Talker 不是简单地把最终文本丢给一个外部 TTS。它会接收high-level hidden representations 和已经 sampled 出来的 discrete text token embeddings两种信息。报告解释说，流式语音生成需要在完整文本还没生成完之前，就提前判断语气、情绪、态度等信息；Thinker 的高维 representation 能提供这些隐式语义信息。但 Thinker representation 主要表达语义相似，而不是发音相似，所以还需要 sampled discrete text tokens 来消除发音层面的不确定性。

论文把训练分成两大部分。Pre-training有三个阶段。第一阶段中，LLM参数不动，主要训练 vision encoder 和 audio encoder，使用大规模 audio-text 和 image-text pairs 来增强 LLM 的多模态语义理解。初始化方面，LLM 来自 Qwen2.5，vision encoder 使用 Qwen2.5-VL，audio encoder 初始化自 Whisper-large-v3。两个 encoder 会在固定 LLM 上分别训练，并且一开始先训练各自 adapter，再训练 encoder。第二阶段，LLM参数解冻，使用更广泛的多模态数据进行训练。最后第三阶段继续训练长序列能力，进行32K 长上下文训练。

之后是后训练。Thinker 使用 ChatML 格式做 instruction fine-tuning。而Talker 的 post-training 更特殊，有三个阶段。第一阶段训练 Talker 学习 context continuation(ICL)。除了类似 Thinker 的文本监督，还做 speech continuation task，也就是通过 next-token prediction 学习在多模态上下文和 spoken responses 下继续生成语音。论文说这一步让 Talker 学会从语义表示到语音的单调映射，同时学习 prosody、emotion、accent 等上下文相关语音属性。第二阶段使用 DPO 来增强语音生成稳定性，有人类偏好三元组来用于训练。第三阶段做 multi-speaker instruction fine-tuning，目的是让 Talker 能采用特定声音，并提高语音自然度和可控性。很多操作是为了更好的TTS，而不是出于MLLM角度出发。

MiniCPM-o 4.5

> MiniCPM-o 4.5: Towards Real-Time Full-Duplex Omni-Modal Interaction

We present MiniCPM-o 4.5, our latest effort towards **human-like** multimodal interaction, which mitigates these gaps by **real-time full-duplex** omni-modal interaction. 这个文章想要做到的是：如何让Omni模型真的能像人一样，边听边看边说。现有 omni / MLLM 的问题交互方式和人类的方式还是有点区别，perception 和 response 是分离的；并且很多时候模型大多是被动相应的：它通常只有在用户显式提问后才回答，而不是像人一样根据环境变化主动提醒、评论或描述。论文说MiniCPM-o 4.5 的目标是让模型在实时场景中持续感知并生成响应，从而支持 proactive behaviors。

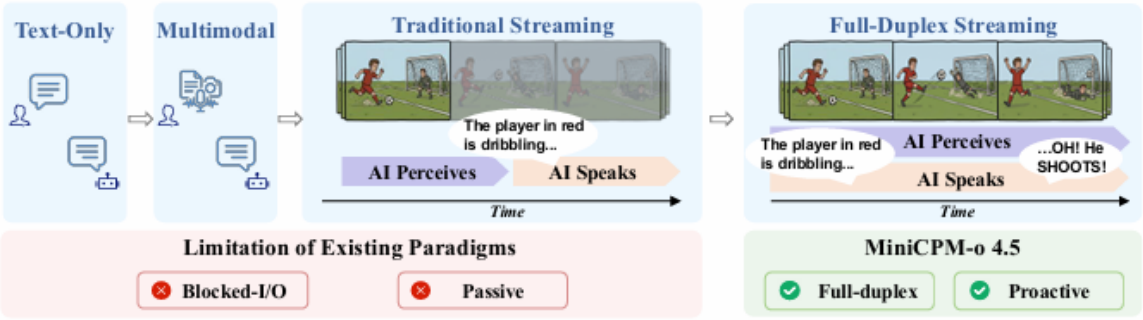
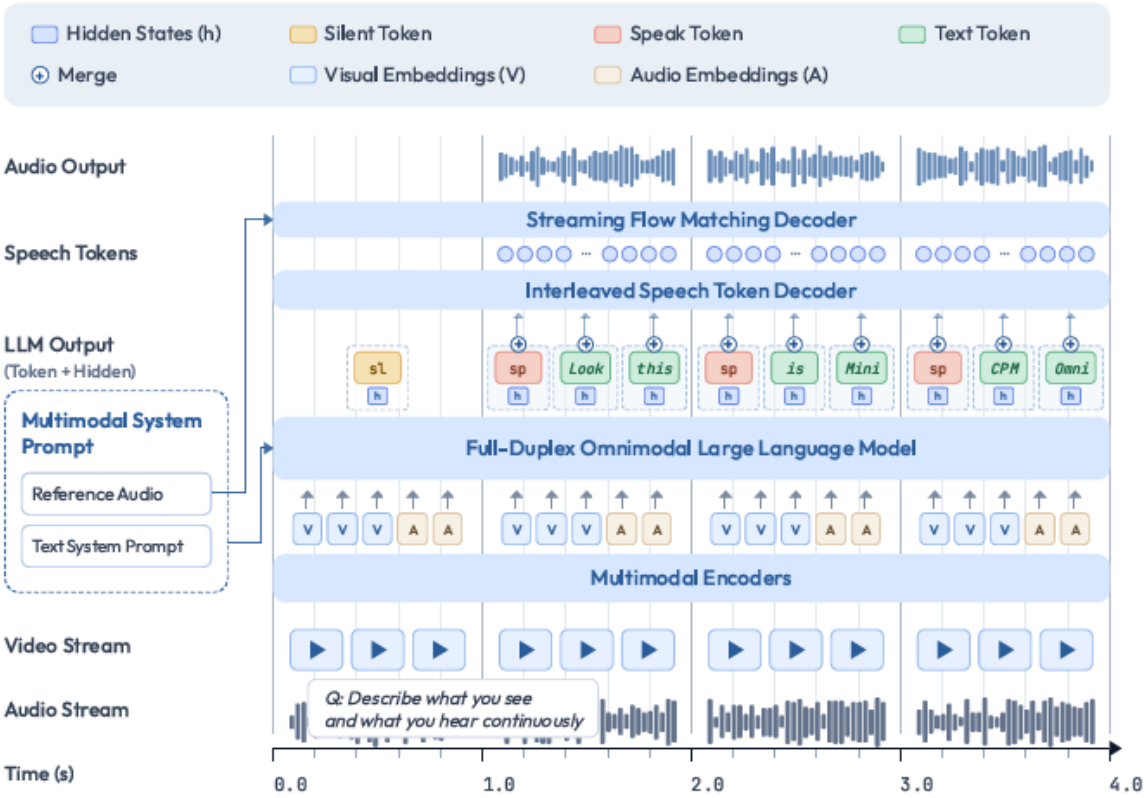


Figure 3: **From turn-based interaction to full-duplex streaming**. Existing interaction paradigms separate perception and response as alternating phases, leading to blocked information flow and passive behavior. In contrast, MiniCPM-o 4.5 continuously perceives incoming multimodal streams while speaking, allowing the model to update its response in real time and act proactively.

概括一下，task formulation可以是：

```
输入侧：continuous video stream + continuous audio stream + optional text/system prompt/reference audio
输出侧：text tokens + speech tokens + final waveform
关键点：输入与输出沿同一时间轴持续耦合，而不是一问一答式串行处理
```



MiniCPM-o 4.5 是一个约 **9B 参数**的端到端 omni-modal 架构，包含多模态encoder, LLM backbone and speech decoder。GPT总结的模型的pipeline如下：

```
video stream / image
    → SigLIP ViT visual encoder
    → visual resampler
    → visual tokens

audio stream
    → Whisper Medium encoder
    → audio projector
    → compressed audio tokens

visual tokens + audio tokens + text prompt
    → Qwen3-8B LLM backbone
    → text tokens + LLM hidden states

text tokens + projected hidden states
    → lightweight speech token decoder
    → S3 speech tokens

S3 speech tokens + reference audio
    → streaming flow-matching decoder
    → audio waveform
```

full-duplex 要求低延迟，因此在encoder的选择和sampling strategy上面有所设计。LLM backbone是 Qwen3-8B，负责统一处理视觉、音频、文本上下文，并生成文本 token 和 hidden states。论文强调：LLM backbone 只在文本域生成 token，因此实时交互时大约只需要 3–4 decoding steps per second，接近人类说话速度。而speech则是使用轻量化的token decoder，负责根据 text token + hidden states 生成 speech tokens（这里token and embedding都接受的操作很像Qwen2.5-Omni中的操作）。

这篇文章最关键的方法是 Omni-Flow，以尽可能做到real-time。它把交互过程拆成很多细粒度时间窗口，每个窗口长度为 t。在每个时间窗口内，模型先接收新到来的视觉和音频信号，然后生成当前时刻的输出。这样传统的 turn-based 交互就变成了连续的 time-local updates。交互时候的stream有：

```
env-visual：环境视觉流
env-audio：环境音频流，包括用户语音
out-stream：助手输出流，包括 text 和 speech
```

如果当前窗口不应该输出内容，o_k 就只包含一个特殊的 [listen] token。在每个 chunk 内，模型先处理新到来的 perceptual tokens，再生成 output tokens，因此每次输出都可以条件于最新观察。所以Omni-Flow总结下来，其实就是每一个时间窗口，接受最新的video/audio，然后决定是Listen 还是 speak（binary decision），如果speak，那么就会生成当前窗口的输出。

既然是边看边说，那么就会有说话和环境的时间轴对不上的情况。或者说，文本生成速度和语音播放时间不一致。如果模型在某个时间窗口里生成了太多文本，这些文本的语音播放可能持续很久，导致用户听到的是"已经过时"的内容。为此，文章提出 TAIL：Time-Aligned Interleaving。它不是固定生成若干 text tokens 再生成 speech，也不是固定 text:speech ratio，而是根据累计播放进度动态决定每个 chunk 生成多少文本。第 k 个 chunk 的目标是：新生成内容被说完后，speech stream 尽量接近当前时间边界 kt。如果前面已经有播放延迟，当前 chunk 就少生成一些文本，让语音追上来。

训练时，作者从 full-duplex streaming training data 中收集每个 text token 的 start/end time，把 start time 落在 $[(k-1)t, kt)$ 的 text tokens 及其对应 speech tokens 分配给第 k 个 Omni-Flow chunk。这样模型学到的不是固定比例，而是 history-dependent interleaving pattern。

TAIL 还用了 bounded look-ahead：最后几个 text tokens 对应的 speech tokens 会延迟到下一个 chunk 生成，以便获得一点未来上下文，帮助发音和韵律，防止说话句子间的发音有些奇怪。

文章的训练 pipeline 是分阶段的，目标是在保留视觉和语言能力的同时，逐步把语音理解、语音生成、full-duplex 交互能力接入系统。它基于 MiniCPM-V 4.5 的预训练 checkpoint 开始，然后依次做 speech pretraining、joint pretraining、joint SFT 和 RL。

第一阶段是为了保护已有的视觉和语言能力，冻结 pretrained components，只更新新加入的模块。训练目标有两个：第一，把 Whisper features 对齐到 LLM hidden space；第二，训练 speech decoder，把 LLM backbone hidden states 转成语义和韵律都合理的 speech tokens。第二阶段 unfreeze all parameters，在 vision-language、speech、omni-modal data 的 balanced mixture 上联合预训练。为了稳定优化，文章把不同 modality combinations 分配到不同 data-parallel ranks，从而保证每一步训练时数据比例固定。这一阶段的数据不只有常规 turn-based samples，还包括 proactive 和 full-duplex interaction data，其中 text tokens 与 speech / visual signals 在共享时间轴上对齐。训练目标是 unified next-token prediction

objective。这一阶段是为了模型获得 real-time omni-modal interaction 能力，同时保持基础视觉理解能力。

第三阶段是 joint SFT，用来激活 omni-modal capabilities 并加强 instruction following，有两个phase。第一阶段做广覆盖能力适配，第二阶段用高质量人工标注数据做更细粒度的行为修正，从而学会按照用户意图自然交互、在不同帧率下工作等。最后，作者用 RL 进一步提升 reasoning、instruction following，并降低 hallucination。具体包含两个部分。GRPO、Kimi-K1.5 的 smooth length reward、RLAIF-V 分别被用于提高正确响应的召回率、提高 token efficiency和降低视觉场景中的 hallucination。

总结和思考

从几个代表模型可以看到不同的架构路线。NExT-GPT 体现的是一种 representation-based generation思路，其中 LLM 更像是一个跨模态语义规划器和生成调度器，真正的图像、音频、视频细节仍然依赖外部 diffusion generator。AnyGPT 则走向另一条any to any的路线：它把图像、语音、音乐等模态全部离散 token 化，扩展 LLM vocab 和 prediction head，然后用统一的 next-token prediction 训练。相比 NExT-GPT，AnyGPT 更接近“把所有模态都当成语言”的思想，因为 LLM 生成的是目标模态 tokenizer 的内容 token，而不仅是生成信号。二者看似相似，都是 LLM 后面接 generator，但本质区别在于：NExT-GPT 的 token 是控制信号，AnyGPT 的 token 是内容表示。

而从 VITA、Qwen2.5-Omni 到 MiniCPM-o 4.5，可以看到 Omni 任务场景的进一步演变。Omni 的目标不再只是完成静态的 any-to-any generation，例如 text-to-image、image-to-text、speech-to-text，而是逐渐走向 real-time, interactive, streaming, full-duplex 的人机交互场景。VITA 开始关注语音助手中的实际交互问题，例如是否需要 wake-up word、环境噪声是否应该触发回答、用户能否在模型回答时插入新问题。它通过 state token、noisy audio negative samples 和 dual-model monitoring 机制，把 Omni 从“多模态问答模型”推进到“交互式助手”。Qwen2.5-Omni 进一步提出 Thinker-Talker 架构，让 Thinker 负责多模态理解和文本生成，Talker 根据 Thinker hidden representations 和文本 token embeddings 生成 speech codec tokens，从而不是简单地把文本丢给外部 TTS，而是把语音生成纳入模型内部的表示链条。MiniCPM-o 4.5 则把任务场景推进到 full-duplex：模型要持续接收视频流和音频流，同时持续生成文本和语音输出，还要处理 listen / speak 决策、语音播放延迟、用户打断和 proactive response。

训练设计上，我的一个核心体会是：Omni 的训练难度远高于 MLLM，因为它不只需要 input alignment，还需要 output alignment、interaction tuning 和 time-aligned behavior learning（if needed）。 在 MLLM 中，训练主线通常是 pretraining → instruction tuning → alignment tuning。pretraining 主要把非文本输入对齐到 LLM，instruction tuning 让模型学会根据指令输出文本，alignment tuning 则通过强化学习等方式提升偏好。而 Omni 在此基础上增加了至少三类新的训练目标。

第一类是 输出侧生成对齐。例如 NExT-GPT 需要训练 output projection layer，使 LLM 的 signal token representation 能够被 diffusion decoder 理解；AnyGPT 需要让 LLM 直接学习 image / speech / music tokens 的自回归分布；Qwen2.5-Omni 和 MiniCPM-o 则需要训练 speech decoder，把 LLM hidden states 转换成可播放的 speech tokens。第二类是 任意模态组合的 instruction tuning。Omni 模型不能只学 VQA，而要学 text-to-speech、text-to-image、video-audio-to-text、interleaved dialogue 等复杂任务。第三类是 交互行为训练。VITA 的 noisy audio negative samples、state token，MiniCPM-o 的 [listen] token、shared

timeline data、full-duplex training，本质上都在训练模型什么时候该回答、什么时候该沉默、什么时候该打断、什么时候该根据新环境主动响应。

从这些模型的训练 pipeline 也可以看到一个共同趋势：越成熟的 Omni 模型越倾向于分阶段训练。通常先冻结 LLM 和 encoder，只训练 projector / adapter，完成输入侧 alignment；再训练 output projector、speech decoder 或 modality token head，完成输出侧 generation alignment；之后进行 joint multimodal pretraining，让不同模态在统一模型中共同优化；最后再通过 instruction tuning、RL 等方式激活交互能力和偏好行为。总而言之，Omni 模型要非常小心地控制先学会看懂/听懂，再学会生成，再学会交互，再学会偏好的顺序。