Day7

在前6天的调研中，我从架构、训练、数据集、benchmark、痛点和未来方向几条主线上把Omni的整体面貌走过了一遍，并且对AnyGPT做了一次代码精读。今天我打算补两个层次的内容：第一补充一些横切技术，包括位置编码与时序对齐、Streaming/Full-duplex工程、训练效率(MoE/参数共享)、对齐调优。这四块在前面散落在不同day里被提到过。第二是应用与边界，包括Omni-Agent、Action作为新模态(VLA)以及闭源旗舰的横向对比。

一些横切技术

位置编码

Qwen2.5-Omni中有TMRoPE作为时间戳。回顾到：RoPE实际上是在query/key上面按照dimension位置m上面做旋转：

$$\mathrm{RoPE}(x_m)_{2i:2i+2} = \begin{pmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{pmatrix} \begin{pmatrix} x_{2i} \\ x_{2i+1} \end{pmatrix}, \quad \theta_i = 10000^{-2i/d}$$

这是一种相对位置编码。这些运用在text上面姑且没有问题，但是如果直接把图像patch flatten成1D序列再用1D RoPE，会有几个本质缺陷：模型很难学到真正的2D邻接、时间维度无处安放以及跨模态时间无法对齐（如音频对应的那那一帧图像）。因此Qwen2-VL把RoPE的position id从1D的scalar变成三元组(t, h, w)，做法是把RoPE的旋转角度切成三段，分别由temporal/height/width三个独立id驱动。规则非常直接：

- Text：`t=h=w=pos`，退化为标准1D RoPE，保证与纯文本LLM兼容
- Image：所有patch共享一个常量`t`，`h`和`w`取patch行列号
- Video：每帧`t`递增1，同一帧内`h, w`按二维网格
- 混合输入：下一段的起始id = 上一段max id + 1

但M-RoPE还遗留两个问题：(1)视频的`t`是帧序号、与真实时间无关，30 fps视频和1 fps视频在LLM看来完全一样；(2)没有处理音频。

于是TMRoPE在M-RoPE基础上做两件事，把"绝对时间"和"音视频对齐"塞进位置编码体系：

(1) temporal id 绑定绝对时间，1 id = 40 ms。在配置里`position_id_per_seconds = 25`，所以每1个temporal id正好对应40 ms。这个数字是因为Whisper-style audio encoder每40 ms输出一帧token的下采样率精确匹配，从而：

- 音频帧 `t = 0, 1, 2, ...` 直接对应40ms时间网格
- 视频帧按 `t = frame_idx × seconds_per_frame × 25` 量化到同一网格
- 音频和视频的`t`数值可比 → "同一时刻"在attention里旋转相位相近

(2) Time-interleaving：按 2 秒 chunk 把视觉和音频拼接。如果video token全部排前面、audio全部排后面，causal attention下早期音频看不到对应视频。Qwen2.5-Omni的解法是每2秒切一个chunk，chunk内先放该2秒的视觉token，再放该2秒的音频token。

TMRoPE还是有遗留问题：长视频意味着temporal id跨度超级大，远超训练阶段见过的最大id，RoPE在这种稀疏外推区间数值不稳定。Qwen3-Omni先把粒度从40ms放大到80ms/id缓解，同时做了一个更激进的范式转移：直接在视觉/音频token之前插入timestamp字符串token。如：

```
... <vision_start> [0.0s] <patch tokens of frame@0.0s>
                  [1.2s] <patch tokens of frame@1.2s>
                  [2.4s] <patch tokens of frame@2.4s> <vision_end> ...
```

这样一个核心好处是LLM能够利用这个时间信息帮助推理，而且推理能够关于时间更为的精细。因此，训练输出需要配备好时间戳，并且这些时间戳也会占用token数量。

Streaming / Full-duplex Omni 工程

chunked attention：对于视频音频token大量涌入，不可能每一帧都full attention，因此主流的做法是chunked attention。Qwen2.5-Omni的DiT decoder把音频token分块，attention mask是lookback 2 blocks + lookahead 1 block，把无限上下文压成常数感受野。

输出端流水线：text token → speech codec token → mel → waveform，每一段都要chunk-friendly，从而使得talker均匀输出内容，避免mismatched-rate。

Streaming / Full-duplex Omni 的核心思想是把传统"等用户说完→ASR→LLM→TTS→播放"的串行级联，重构成沿同一条时间轴并行展开的流水线——感知、推理、生成在每个时间窗口内同步推进，从而把端到端延迟从几秒压到几百毫秒，并支持边说边听、被打断、主动响应。工程上要解决三类问题：算力/显存，主流解法是 chunked sliding-window attention；输出端流水线；何时说 / 何时听的时机决策（持续听到的音频中只有一部分是真问题，模型还得知道自己什么时候该沉默），如MiniCPM-o 的[listen]、VITA 的 state token。

Omni 训练效率：MoE 与参数共享

Omni模型同时挂载vision encoder、audio encoder、speech decoder和backbone，单次query实际只走其中一条链路。这种激活路径的稀疏性和MoE的sparse activation天然吻合，因此最近的Omni模型几乎都在往MoE方向走。总结下来，有两个原因支持Omni需要MoE:

(1) 模态特征分布差异大，dense FFN会modality interference。视觉patch token偏空间grounding、语音token偏时序韵律、文本token偏高阶语义，三者塞进同一个dense FFN里更新，某模态的梯度会反向恶化其他模态——文献里叫"see-saw"效应或modality conflict。这是LLaVA-MoLE、Uni-MoE的实验出发点。

(2) Omni 参数总量爆炸但单次推理路径稀疏。一个full-duplex Omni可能挂vision encoder(几亿) + audio encoder(几亿) + speech decoder(几亿) + 7-30B backbone。dense推理FLOPs巨大，但一次对话往往只走"视觉理解+文本输出"或"语音输入+语音输出"中的一条。MoE的top-k routing让总参数与激活参数解耦，模型容量做大而单次成本可控。

代表工作为Qwen3-Omni，它把Thinker和Talker都MoE化，并把不同类型attention(local/global或线性/全局)和MoE FFN混合排列。

关于参数共享的业界公式，总结如下：

层 / 模块	倾向 modality-specific	倾向共享
各模态encoder (ViT, Whisper)	✓✓	–
Projector / Q-Former / connector	✓✓	–
底层FFN (接近embedding)	✓ (近期工作)	无，早期工作都没管
中高层FFN	MoE路由	无，早期工作都没管
跨模态Attention	–	✓
高层reasoning / decoder head	–	✓✓ (modality-agnostic)

Omni 对齐调优

Omni中的modality bias、跨模态幻觉、语音质量、Full-Duplex等，都可以使用偏好数据来进行对齐微调。那么就说明：i偏好对至少要分别构造：(1)视觉幻觉偏好对(RLHF-V segment correction / RLAIF-V atomic claim)；(2)跨模态一致性偏好对(同prompt正确图vs错图)；(3)语音偏好对(WER+prosody+speaker similarity自动判定)；(4)全双工交互偏好对(该沉默/该接话/该打断)。专门的多模态reward model仍稀缺，大多数工作走LLM-as-judge + rule-based reward组合。

目前绝大部分工作都是各个部分、各个模态的负责组建分别进行RL，是否未来可能有end-to-end的RL呢?

Omni应用与边界

Omni-Agent

Omni让Agent的边界从LLM API操作扩展到真实世界感知决策。Qwen3.5-Omni的技术报告里明确把native omnimodal agentic behavior作为关键能力，包含三件事：

- **Autonomous WebSearch**：模型直接看screenshot做web交互
- **Complex FunctionCall**：从音视频上下文里推断该调什么工具、传什么参数
- **Audio-Visual Vibe Coding**：从音视频指令直接生成可执行代码——例如用户拿着手机录一段"我想要一个能算潮汐的网页"，模型直接出HTML/JS

但是多模态环境的perception远没有文本环境那么准确而鲁棒，并且多模态工具调用的schema在多模态环境可能会更为复杂。

Action as Modality

这是我个人觉得最有意思的一支。前几天调研AnyGPT时，最让我惊艳的是它"图像/语音/音乐都能tokenize进LLM词表统一训练"的思想。如果这个哲学是对的，那机器人的动作(action)——本质上也是一种时序连续信号——凭什么不能也被tokenize进LLM？这条思路就是Vision-Language-Action(VLA)模型。

值得注意的是，目前主要场景应该是没有action modality作为输入的，因此perception端不会有相关的输入，也不会参与到LLM内部的推理。

目前主流的VLA几乎都是直接拿Omni/VLM当backbone：RT-2基于PaLI-X/PaLM-E，OpenVLA基于Llama2+DINOv2+SigLIP，π0基于PaliGemma，GR00T基于NVIDIA Eagle 2.5，Gemini Robotics基于Gemini。即"VLM/Omni + 一个动作头"就是VLA的通用范式。

闭源旗舰横向对比

最后简单整理一下当前闭源旗舰Omni系统的一些特点：

GPT-4o (OpenAI, 2024.05)。官方自述：旧voice mode是ASR + GPT-4 + TTS级联，大约是5秒延迟；GPT-4o端到端单模型处理音频，平均320ms，被认为非常优秀。

Gemini 2.5 / 3 (Google DeepMind)。Gemini从1.0开始就主打"natively multimodal"——预训练阶段就联合学text/image/audio/video。Gemini 2.5的Live API支持双向音视频流，公开的latency benchmark略高于GPT-4o Realtime但能力更广(支持video streaming输入)。Gemini Robotics系列把这个能力延伸到embodied领域。

Claude (Anthropic)。Claude 4之前主要是文本+图像，没有原生音频。Claude 4加入了Computer-Use(看屏幕+用键鼠)，本质上是"图像理解 + 工具使用"（浏览器端使用claude），但还没有进入streaming语音交互领域。Anthropic的策略似乎更偏向Agent能力而不是音视频原生处理。