文生视频大模型：模型谱系、Wan/Bernini 与规模化训练系统（中册）

从开源底座的结构解剖，到数据、算力、微调、推理与论文级复现

面向具备深度学习基础的计算机博士生

2026-07-12

目录

中册前言：从“理解公式”走向“读懂系统并完成复现”

上册建立了文生视频的统一数学框架：条件分布、Video VAE、扩散与 Flow Matching、Video DiT、MLLM Planner 和评测体系。中册回答更工程化、也更接近真实科研的问题：

- 一篇技术报告中的模型图，如何映射到具体张量、模块和代码文件?
- “14B 模型”究竟意味着多少权重显存、多少训练状态、多少单步计算?
- 为什么有的 5B 模型可以生成更长、更高分辨率的视频，而某些 14B 模型反而更慢?
- Wan2.1 与 Wan2.2 的差异究竟来自模型容量、VAE 压缩、时间专家，还是数据课程?
- Bernini 中的 MLLM Planner 与普通 prompt rewrite 有何本质区别? 连续 ViT 语义计划如何进入视频 DiT?
- 公开论文没有披露训练 GPU 数时，怎样给出诚实、可复算、带假设的成本估计?
- 如何从官方仓库开始，完成形状审计、推理复现、LoRA 微调、系统测量和论文级消融?

本册不把模型当作一串产品名，而是将每个系统分解为同一套六维设计向量：表示、生成主干、训练目标、条件接口、数据课程和计算系统。这样，当未来出现新模型时，读者不需要重新记忆全部术语，只需判断它在六个坐标轴上改变了什么。

> **资料时点**：公开论文、技术报告和开源仓库核对至 2026 年 7 月。Wan 的公开技术结构以 Wan2.1 与 Wan2.2 为主；未公开的训练 GPU 数、总训练步数或总成本不会被伪装成已知事实。Bernini 的结构与实验按其 2026 年公开论文整理，其权重和完整训练资产是否开放应以项目当时状态为准。

本册学习目标

完成中册后，读者应当能够：

1. 使用统一坐标系比较 CogVideoX、HunyuanVideo、LTX-Video、Open-Sora、Step-Video、Wan 与 Bernini;
2. 从视频尺寸、帧数、VAE stride 和 patch size 精确计算 latent 形状与 DiT token 数;
3. 解读 Wan 官方配置中的 `dim`、`ffn_dim`、`num_heads`、`num_layers`、`sample_shift` 与 `boundary`;
4. 推导视频 DiT 主要计算项、权重显存、AdamW 训练状态和 GPU-hours 估计;
5. 设计正确的 LoRA/adapter 微调方案，并避免把 VAE、采样器或 prompt 差异误判为模型改进;
6. 完整解释 Bernini 的 latent semantic planning、ViT-flow decoder、SA-3D RoPE 与三阶段训练;
7. 建立包含数据治理、并行、内核、监控和评测的端到端训练系统;
8. 从“模型能运行”推进到“结论可重复、成本可比较、消融可发表”。

统一符号

沿用上册符号，并增加模型与系统相关记号。

符号	含义
$\mathbf{x} \in \mathbb{R}^{3\times T\times H\times W}$	像素视频
E_ϕ, D_ψ	Video VAE 编码器与解码器
$\mathbf{z}_\star = E_\phi(\mathbf{x})$	干净视频 latent
(s_t, s_h, s_w)	VAE 的时间、空间高、空间宽压缩 stride
(p_t, p_h, p_w)	DiT patch size
(T_z, H_z, W_z)	VAE latent 网格
(T_p, H_p, W_p)	patch 后 Transformer 网格
$N = T_p H_p W_p$	视频 token 数
$d, d_{\mathrm{ff}}, L, n_h$	隐藏维度、FFN 维度、层数、头数
$\tau \in [0,1]$	扩散/流时间，不与视频帧下标混用
$v_\theta(\mathbf{z}_\tau, \tau, \mathbf{c})$	速度场网络
K	采样 NFE 或离散求解步数；上下文会明确
P	可训练参数量
G	GPU 数量
U	每块 GPU 的有效计算吞吐（FLOP/s）
η	相对峰值利用率，或将其吸收到 U 中
$\mathcal{D}$	训练数据分布或数据池
$\pi(q)$	训练样本/任务 bucket 的采样概率

因果 Video VAE 常保留首帧，其时间 latent 长度不是简单的 T/s_t，而是

$$T_z = 1 + \left\lfloor \frac{T-1}{s_t} \right\rfloor .$$

若尺寸可整除，空间与 patch 后形状为

$$H_z = \frac{H}{s_h}, \quad W_z = \frac{W}{s_w}, \qquad T_p = \frac{T_z}{p_t}, \quad H_p = \frac{H_z}{p_h}, \quad W_p = \frac{W_z}{p_w}.$$

证据等级与成本披露规则

本册对动态事实使用以下等级：

等级	含义	可采用的表述
A	论文、技术报告或官方配置直接披露	“官方配置为”“论文报告”
B	可由 A 级量直接、唯一地计算	“由 stride 与分辨率可得”
C	依赖简化模型、吞吐假设或硬件效率的估计	“在假设……下，数量级约为”
ND	未披露，且无法可靠反推	明确写“未公开”，不填入猜测值

证据等级

参数量、VAE stride、默认帧数、模型层数等可由论文或官方配置核对；真实训练总 token、失败重启、数据重复使用次数、平均硬件利用率和内部数据成本通常没有公开。GPU-hours 估计必须列出公式、吞吐假设和不确定区间，不能把一个精确到个位数的结果包装成事实。

另一个重要规则：训练数据的“十亿级图像与视频”并不等价于十亿条独立高质量视频，也不等价于十亿个训练 sample。原始资产、切分 clip、配对样本、训练采样次数和去重后的独立来源必须区分。

版本规则：在线服务名、社区微调名和论文底座名可能不同。除非存在官方技术披露，本册不根据名称推断隐藏架构。

13. 用统一六维坐标系阅读视频生成模型

13.1 为什么“模型名称比较”不够

说“模型 A 比模型 B 更强”往往没有可解释性，因为二者可能同时改变：

- 数据质量和数量；
- VAE 压缩率；
- 生成主干大小；
- 采样步数和 CFG；
- 输出帧率、时长与分辨率；
- prompt rewrite；
- 后训练和偏好优化；
- 推理硬件及内核。

因此定义模型设计向量

$$\mathcal{M} = (\mathcal{R}, \mathcal{B}, \mathcal{O}, \mathcal{C}, \mathcal{K}, \mathcal{S}),$$

其中：

- $\mathcal{R}$（Representation）：像素、连续 VAE latent 或离散视觉 token；压缩几何、latent channel 与重建损失；
- $\mathcal{B}$（Backbone）：3D U-Net、Video DiT、自回归 Transformer；全局、分解、窗口或稀疏注意力；
- $\mathcal{O}$（Objective）：噪声预测、v-prediction、Flow Matching、Rectified Flow、masked modeling、DPO/偏好目标；
- $\mathcal{C}$（Conditioning）：文本、图像、视频、音频、姿态、轨迹、参考身份或语义计划；
- $\mathcal{K}$（Curriculum）：图像/视频混合、分辨率和时长课程、任务混合、质量重加权与后训练；
- $\mathcal{S}$（System）：数据加载、混合精度、并行、offload、通信、内核、蒸馏和服务部署。

任何实验结论都应尽量写成“改变哪个坐标、保持哪些坐标不变”。例如，比较 Wan2.1 与 Wan2.2 TI2V-5B 时，不能只看参数量；后者还改变 VAE 压缩和默认帧率。比较 Bernini 与 Wan2.2-A14B 时，也不能把收益全归因于 renderer，因为 Bernini 增加了 MLLM planner、编辑数据、位置编码和新的 guidance。

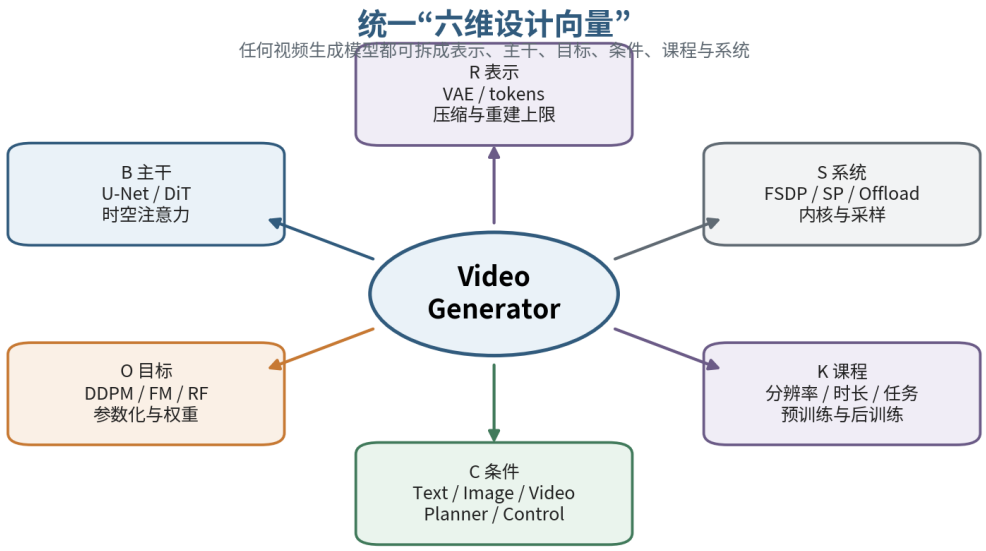

图 1: 统一六维设计向量

13.2 模型族的三种主要生成分解

自回归视觉 token

将视频离散为 token $u_{1:N}$，学习

$$p_\theta(u_{1:N} \mid \mathbf{c}) = \prod_{j=1}^{N} p_\theta(u_j \mid u_{<j}, \mathbf{c}).$$

优势是可自然扩展为长序列、与语言模型结构兼容；缺点是串行解码、误差累积与高质量 tokenizer 训练困难。Phenaki 是代表性长视频探索，后续也出现 masked parallel decoding。

扩散/流式连续 latent

在连续 Video VAE latent 中学习反向去噪或 ODE 速度场。现代大规模开源系统的主流是 Video DiT + Flow Matching:

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{\mathbf{z}_0,\mathbf{z}_1,\tau}\left[\|v_\theta(\mathbf{z}_\tau,\tau,\mathbf{c}) - u_\tau(\mathbf{z}_\tau \mid \mathbf{z}_0,\mathbf{z}_1)\|_2^2\right].$$

它允许并行预测全部视频 token，但推理需要多次网络评估。

分层 Planner-Renderer

先预测语义计划 $\mathbf{s}$，再渲染视频:

$$p(\mathbf{x} \mid \mathbf{c}) = \int p_\psi(\mathbf{x} \mid \mathbf{s}, \mathbf{c}) p_\omega(\mathbf{s} \mid \mathbf{c})\, d\mathbf{s}.$$

若 $\mathbf{s}$ 是自然语言 storyboard，它可解释、易编辑，但表达带宽低；若 $\mathbf{s}$ 是连续 ViT embedding，它包含更丰富视觉语义，但评测和调试更困难。Bernini 属于后一种。

13.3 评价架构时的四个边界

VAE 上限

生成结果不可能稳定超过 decoder 的表示上限。先计算 oracle reconstruction:

$$\hat{\mathbf{x}}_{\mathrm{oracle}} = D_\psi(E_\phi(\mathbf{x})).$$

若 oracle 已经丢失文字、脸部细节或高速运动，那么更强 DiT 只能更准确地生成一个受损 latent。

条件上限

文本编码器看不懂精细空间关系时，DiT 不会凭空恢复。MLLM planner 可以提升组合语义，但也可能产生计划幻觉。应分别评估 condition encoder、planner 与 renderer。

求解器上限

同一模型在不同 time shift、步数、solver 和 CFG 下差异很大。对模型结构做结论时，必须对采样配置进行公平调参，或报告质量-延迟 Pareto 曲线。

评测上限

VBench、FVD、CLIPScore 与人类偏好覆盖不同维度。单一总体分数不能识别：视频很美但动作错误、动作很大但主体漂移，或文本对齐好但物理规律错误。

13.4 版本与任务的三层命名

研究中应区分：

1. **基础权重版本**：例如 Wan2.1-T2V-14B、Wan2.2-TI2V-5B；
2. **任务适配版本**：I2V、FLF2V、VACE、Animate、S2V；
3. **运行配置**：分辨率、帧数、fps、NFE、CFG、prompt extension、offload 与量化。

只有第一层通常对应模型参数。把“720p”“5 秒”“高动态模式”当作全新模型，会导致实验记录不可复现。

13.5 一个标准模型卡应记录什么

至少记录：

```
model_id / commit / checkpoint hash
VAE: type, stride, latent channels, causal?, chunking?
DiT: patch, hidden dim, FFN dim, layers, heads, attention pattern
text/image encoder and maximum context
objective and prediction parameterization
training task mixture and known data disclosure
sampler, NFE, time shift, CFG, negative prompt
output: frames, fps, H×W, duration
precision, quantization, offload, device topology
peak VRAM, wall time, warm-up policy, seed
```

核心结论

模型比较的最小单位不是“checkpoint 名称”，而是“checkpoint + VAE + conditioner + sampler + 输出几何 + 系统设置”。缺少任何一项，都可能让同一模型看起来像两个不同系统。

14. 模型谱系：从级联扩散到 Video DiT 与语义规划

14.1 Video Diffusion Models：把图像扩散扩展到时间轴

早期 Video Diffusion Models 证明可以在 3D 时空张量上进行扩散生成，并通过联合图像-视频训练利用廉价图像数据。典型做法是在 2D 空间模块之外增加时间卷积或时间注意力：

$$\mathbf{h}' = \mathbf{h} + \mathrm{TempAttn}(\mathrm{Norm}(\mathbf{h})).$$

图像可以视为 $T = 1$ 的退化视频，从而共享大部分参数。该思想一直延续到现代系统：高质量图像数据负责外观与纹理，视频数据负责运动与时间一致性。

其限制是像素空间或低压缩空间成本过高，通常依赖级联超分模型；跨阶段误差和工程复杂度也随之增加。

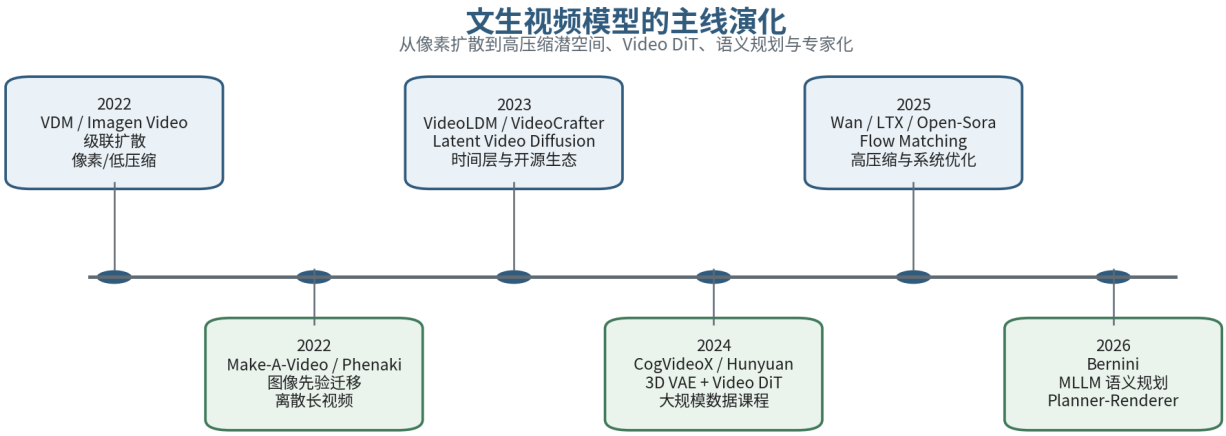

图 2: 文生视频模型的主线演化

14.2 Imagen Video：时空级联与高分辨率生成

Imagen Video 采用基础视频生成器和多个空间/时间超分阶段。级联分解可写为

$$p(\mathbf{x}^{(0:K)} \mid \mathbf{y}) = p(\mathbf{x}^{(0)} \mid \mathbf{y}) \prod_{k=1}^{K} p(\mathbf{x}^{(k)} \mid \mathbf{x}^{(k-1)}, \mathbf{y}).$$

低分辨率阶段学习全局语义和运动，高分辨率阶段恢复纹理。优点是每阶段序列较小；缺点是总训练和推理链很长，运动错误可能在超分后被放大。

14.3 Make-A-Video：从图像文本先验迁移到视频

Make-A-Video 的关键思想是：文本-外观对应关系可主要从图文数据获得，运动先验可从无文本视频获得。概念上将监督解耦为

$$\underbrace{p(\text{appearance} \mid \text{text})}_{\text{image-text data}} \quad + \quad \underbrace{p(\text{motion} \mid \text{frames})}_{\text{unlabeled video}}.$$

这种“图像先验 + 视频动力学”的迁移范式后来成为通用课程设计，而不再局限于某个模型结构。

14.4 Phenaki：离散 token 与长叙事视频

Phenaki 使用视频 tokenizer 和 masked token prediction，并允许用一串时间变化的文本条件生成较长叙事。它突出一个重要事实：长视频不仅是把 T 增大，还需要显式表示事件顺序和跨片段状态。

设第 m 个故事片段条件为 $\mathbf{y}_m$，长视频可写为

$$p(\mathbf{u}^{(1:M)} \mid \mathbf{y}_{1:M}) = \prod_{m=1}^{M} p(\mathbf{u}^{(m)} \mid \mathbf{u}^{(<m)}, \mathbf{y}_{\le m}).$$

离散 token 有利于长序列建模，但 tokenizer 的代码本和重建误差决定最终画质。

14.5 VideoLDM：潜空间视频扩散成为主流

VideoLDM 将扩散放入图像/视频 VAE latent，并强调时间层可以在预训练图像 LDM 上进行插入或微调。成本从像素网格

$$THW$$

降低为

$$T_z H_z W_z \ll THW.$$

该路线直接启发 VideoCrafter、ModelScopeT2V 以及许多开源模型。它的历史意义不是某个具体 benchmark，而是确立“VAE 负责压缩，生成主干负责 latent 分布”的模块化范式。

14.6 ModelScopeT2V 与 VideoCrafter：可复现开源生态

ModelScopeT2V 提供较早的开放 T2V pipeline；VideoCrafter 系列系统化支持 T2V 和 I2V，并探索图像/视频数据混合、运动模块和高分辨率生成。它们使研究者可以在有限 GPU 上研究：

- temporal attention；
- motion adapter；
- prompt-to-video 对齐；
- I2V 条件；
- LoRA 与 ControlNet 类方法。

现代 DiT 底座更强，但这类较小 U-Net 模型仍适合做低成本算法原型和可解释性实验。

14.7 AnimateDiff：把“运动”封装为可插拔模块

AnimateDiff 在图像扩散模型中插入 temporal motion modules，使已有图像个性化模型获得动画能力。抽象写为

$$F_{\text{video}}(\mathbf{h}) = F_{\text{image}}(\mathbf{h}) + \Delta_{\text{motion}}(\mathbf{h}_{1:T}).$$

它揭示了参数高效迁移的两点：

1. 外观与运动在一定程度上可分离；
2. 只训练时间模块时，模型容易继承图像底座的风格，但复杂运动和长期一致性受限。

该思想在今天对应 motion LoRA、temporal adapter 和冻结空间主干的低成本研究。

14.8 DiT 与 Flow Matching 成为大模型主线

当 VAE 把视频压缩为 token 序列，Transformer 具备更好的规模化、并行和多模态融合能力。现代 Video DiT 常包含：

- 3D patchification；
- 3D RoPE 或可学习时空位置编码；
- full 或分解时空 self-attention；
- 文本 cross-attention 或 joint attention；
- 时间条件 AdaLN/modulation；
- QK normalization；
- Flow Matching 或 Rectified Flow 目标。

DiT 的主要计算可粗略分解为

$$C_{\text{fwd}} \approx L\left(8Nd^2 + 4Ndd_{\text{ff}} + 4N^2 d\right),$$

这里按一次乘加约为 2 FLOPs 计。前三部分分别近似表示注意力 QKV/输出投影、FFN 与 attention score/value。序列变长时，N^2 项会迅速主导。

14.9 高压缩 VAE 与“把细节交给 decoder”

LTX-Video 将更激进的 patchification 移入 VAE，使每个 token 覆盖更大的时空块，并让 decoder 完成最后的像素去噪。该思路不是免费午餐：

- 优点：极大缩短 DiT 序列，允许 full attention 和实时推理；
- 风险：latent 瓶颈可能丢失文字、细线和高速局部运动；
- 补偿：增强 decoder、在像素端完成最终去噪、联合优化 VAE 与生成器接口。

视频生成系统的效率竞争，越来越像“把计算放在哪一层”的重新分配，而非单纯缩小 Transformer。

14.10 从 prompt rewrite 到 MLLM semantic planning

传统 prompt extension 生成更详细文本 $\tilde{\mathbf{y}}$:

$$\tilde{\mathbf{y}} = g_{\mathrm{LLM}}(\mathbf{y}), \qquad \mathbf{x} \sim p_\theta(\mathbf{x} \mid \tilde{\mathbf{y}}).$$

MLLM planner 则显式预测目标视觉语义表示:

$$\mathbf{s} \sim p_\omega(\mathbf{s} \mid \mathbf{c}), \qquad \mathbf{x} \sim p_\psi(\mathbf{x} \mid \mathbf{s}, \mathbf{c}).$$

两者差异在于: $\tilde{\mathbf{y}}$ 仍受语言带宽和文本 encoder 限制; $\mathbf{s}$ 可以是逐段、逐帧或连续视觉 embedding，直接携带空间、主体和状态变化信息。Bernini 是这种“理解模型作为生成规划器”的代表。

15. 现代代表模型的结构坐标与取舍

15.1 一张表先建立全局地图

下表只使用公开技术报告的代表设置；输出尺寸和推理速度不应跨硬件直接比较。

模型	代表规模	表示/压缩	主干与目标	关键设计	适合研究的问题
CogVideoX	2B/5B	3D causal VAE	expert Transformer, 扩散	expert AdaLN、progressive、multi-resolution frame packing	文本融合、长视频训练课程
HunyuanVideo	$>$13B	3D VAE	large Video DiT, flow	双流到单流、系统化数据/训练	大底座、文本对齐、并行
HunyuanVideo 1.5	8.3B	高效 VAE + VSR	DiT	SSTA、glyph-aware bilingual encoder、超分	消费级高质量、稀疏注意力
LTX-Video	数十亿级	1:192, $8 \times 32 \times 32$ 像素/token	full-attention DiT	VAE decoder 完成最终像素去噪	极致压缩、实时生成
Open-Sora 2.0	11B 级	Video VAE	Video DiT, flow	透明成本、数据/系统优化	成本可复算的大规模预训练
Step-Video-T2V	30B	时间 8、空间 16×16	full 3D DiT, flow	双语 encoder、Video-DPO、204 帧	超大模型、偏好后训练
Pyramid Flow	多规模	多尺度 latent pyramid	单一 DiT, pyramidal flow	跨分辨率连续流、时域金字塔	训练效率、长视频
Wan2.1/2.2	1.3B/5B/14B	$4 \times 8 \times 8$ 或 $4 \times 16 \times 16$	full-attention DiT, flow	因果 VAE、3D RoPE、时间专家	开源复现、微调、VAE/DiT 研究
Bernini	Planner + A14B renderer	ViT semantic plan + VAE latent	MLLM planning + DiT flow	continuous planning、SA-3D RoPE	统一生成/编辑、语义规划

15.2 CogVideoX: 3D VAE、expert Transformer 与 frame packing

CogVideoX 的公开报告强调三点。

第一，3D causal VAE 同时进行时空压缩，使 10 秒、16 fps 视频可在 latent 中建模。第二，expert Transformer 使用针对文本和视频 token 的不同归一化/调制参数，提高深层跨模态融合。其概念形式为

$$\mathrm{AdaLN}^{(m)}(\mathbf{h}) = \gamma^{(m)}(\mathbf{c}) \odot \mathrm{LN}(\mathbf{h}) + \beta^{(m)}(\mathbf{c}), \quad m \in \{\text{text}, \text{video}\}.$$

第三，multi-resolution frame packing 将不同尺寸、帧数的样本打包，以提高 token 利用率。若每个 sample 的有效 token 为 N_i，固定 padding 到 $N_{\max}$ 的效率为

$$\eta_{\text{pad}} = \frac{\sum_i N_i}{BN_{\max}}.$$

packing 通过让多个短样本共享一条长度预算，显著提高 η_{pad}，但需要 block-diagonal attention mask，避免样本间信息泄漏。

CogVideoX 是理解“模型结构创新与数据系统创新同等重要”的良好案例。它的 10 秒连续生成不仅来自更大模型，也来自渐进式分辨率/帧数训练和 caption pipeline。

15.3 HunyuanVideo：超过 13B 的系统化大模型路线

HunyuanVideo 将数据治理、架构、progressive scaling 和训练基础设施视为一个整体。大模型的文本条件通常不再只依赖 CLIP，而使用更强语言模型编码器与视觉语言对齐模块，以处理长 prompt、镜头语言和中文语义。

其结构思想可概括为“先让不同模态保持独立统计，再在深层统一建模”。双流阶段可写为

$$\mathbf{h}_v^{\ell+1} = F_v^{\ell}(\mathbf{h}_v^{\ell}, \mathbf{h}_t^{\ell}), \qquad \mathbf{h}_t^{\ell+1} = F_t^{\ell}(\mathbf{h}_t^{\ell}, \mathbf{h}_v^{\ell}),$$

之后拼接成单流序列进行联合注意力。相较简单 cross-attention，这允许文本 token 也在视觉上下文中更新，但带来更高计算与并行复杂度。

超过 13B 参数意味着：仅 BF16 权重约需 26 GB；标准 AdamW 的参数、梯度、FP32 master weights 与一二阶矩在未分片时可能达到每参数 14–18 bytes，即 182–234 GB 量级，还未包含 activation。实际训练必须使用 FSDP/ZeRO、activation checkpointing、sequence/context parallel 和高效注意力。

15.4 HunyuanVideo 1.5：用结构效率代替纯规模

HunyuanVideo 1.5 以 8.3B 参数面向消费级推理，核心包括：

- Selective and Sliding Tile Attention（SSTA）：在局部 tile 中计算大部分注意力，并选择性保留跨 tile/global 连接；
- glyph-aware bilingual text encoding：强化中文字符、排版和双语语义；
- progressive pre-training 与 post-training；
- Video Super-Resolution（VSR）：将基础生成与高分辨率细节恢复分离。

若 local window 含 w 个 token，注意力复杂度由 $O(N^2 d)$ 降至近似

$$O(Nwd) + O(Ngd),$$

其中 g 是少量 global/selected token。代价是跨窗口长程运动和全局几何可能受限，需要层间 window shift 或稀疏全局路径。

15.5 LTX-Video：1:192 压缩与实时生成

LTX-Video 报告的 Video VAE 压缩比为 1:192，每个 latent token 覆盖约 $8 \times 32 \times 32$ 像素体素。它将 patchification 移入 VAE，并让 decoder 同时承担 latent-to-pixel 与最后一步像素去噪。

若传统系统把最终误差写为

$$\|D(\hat{\mathbf{z}}) - \mathbf{x}\|,$$

LTX 的设计更接近让 decoder 接收仍含少量噪声的 latent，并直接预测干净像素：

$$\hat{\mathbf{x}} = D_\psi(\mathbf{z}_{\tau_{\text{last}}}, \tau_{\text{last}}, \mathbf{c}).$$

论文报告在 H100 上生成 5 秒、24 fps、768×512 视频可达到约 2 秒，但该结果依赖模型版本、内核、精度和完整 pipeline，不能外推到任意 GPU。研究价值在于：它把 VAE 与 denoiser 的边界重新设计，而非仅追求高压缩数字。

15.6 Open-Sora 2.0：公开训练成本的研究范式

Open-Sora 2.0 的重要贡献之一是披露商业级视频模型约 20 万美元的训练预算，并拆解数据、架构、课程和系统优化。其公开阶段大致包含：

阶段	代表目标	公开成本量级
256p T2V 预训练	大规模低分辨率语义/运动学习	约 70M 样本、85k steps、224 GPU
256p T/I2V	加入图像条件和任务统一	约 10M 样本、13k steps、192 GPU
768p T/I2V	高分辨率精调	约 5M 样本、13k steps、192 GPU

报告总量约 99,840 H200 GPU-hours。成本透明不意味着结果可被精确复制：数据获取、清洗模型、存储、工程人力和失败实验往往未纳入 GPU 账单。但它提供了极有价值的数量级锚点。

15.7 Step-Video-T2V：30B、深压缩与 Video-DPO

Step-Video-T2V 公开 30B 参数、最长 204 帧，Video VAE 时间压缩 8、空间压缩 16×16，使用双语文本编码器、full 3D attention DiT 和 Flow Matching。

高压缩使长视频 token 可控。对 $T = 204$，因果压缩若近似保留首帧：

$$T_z \approx 1 + \left\lfloor \frac{203}{8} \right\rfloor = 26.$$

但 30B full-attention DiT 的 d 和层数使线性投影/FFN 仍很昂贵。系统还引入 Video-DPO，以偏好对 $(\mathbf{x}^+, \mathbf{x}^-)$ 优化：

$$\mathcal{L}_{\mathrm{DPO}} = -\log \sigma \left(\beta \left[\log \frac{p_\theta(\mathbf{x}^+ \mid \mathbf{y})}{p_{\mathrm{ref}}(\mathbf{x}^+ \mid \mathbf{y})} - \log \frac{p_\theta(\mathbf{x}^- \mid \mathbf{y})}{p_{\mathrm{ref}}(\mathbf{x}^- \mid \mathbf{y})} \right] \right).$$

扩散模型没有直接可得的精确 log-likelihood，实际实现通常使用去噪损失差或变分代理，因此“Video-DPO”需要结合论文具体 surrogate 阅读。

15.8 Pyramid Flow：把分辨率金字塔写入流路径

Pyramid Flow 不训练多个独立级联模型，而把去噪轨迹分成不同空间金字塔阶段，仅后段在全分辨率运行。设第 k 个尺度算子为 P_k，可以定义分段路径

$$\mathbf{z}_\tau^{(k)} = (1 - \lambda_k(\tau)) P_k(\mathbf{z}_0) + \lambda_k(\tau) P_k(\mathbf{z}_1),$$

并保证阶段边界的状态连续。这样，全局结构在低分辨率完成，高成本细节阶段只占部分轨迹。

论文报告 5–10 秒、768p、24 fps 模型约使用 20.7k A100 GPU-hours。该数字与 H200、不同模型规模和数据量不能直接换算，但说明“改变流路径”可以产生数量级训练节省。

15.9 选择研究底座的决策表

研究目标	更合适的底座	理由
单卡入门、LoRA、形状审计	Wan2.1-1.3B	官方代码简洁，8–12GB 级可推理
24GB 单卡高质量 T2V/I2V	Wan2.2-TI2V-5B、Hunyuan1.5 的合适配置	高压缩和 offload 路线
大模型文本融合	CogVideoX/HunyuanVideo	expert/joint multimodal Transformer
极限 VAE 压缩与实时	LTX-Video	1:192 与 decoder final denoising
透明成本研究	Open-Sora 2.0、Pyramid Flow	公开 GPU-hours 与课程信息
超大 full-attention scaling	Step-Video、Wan2.2-A14B	大模型与长序列系统问题
MLLM 规划与统一编辑	Bernini	planner-renderer 与多源条件

练习与自检

选择两个模型，分别写出六维向量 $\mathcal{M}$。不要使用“更先进”“更强”这类形容词，而要明确：VAE stride、token 数、主干、训练目标、条件接口、数据课程与系统约束。然后指出一个公平比较必须冻结的变量。

16. Wan2.1 总览：开放 Video DiT 的标准解剖案例

Wan2.1 是适合博士生入门的底座，原因不是它“最好”，而是其公开材料同时具备：较完整技术报告、1.3B/14B 多规模权重、T2V/I2V 等任务、相对清晰的官方 Python 实现，以及大量社区训练与部署工具。

16.1 端到端概率结构

T2V pipeline 可以写成

$$\mathbf{c}_y = F_{\mathrm{T5}}(\mathbf{y}),$$

$$\mathbf{z}_0 \sim \mathcal{N}(0, I), \qquad \frac{d\mathbf{z}_\tau}{d\tau} = v_\theta(\mathbf{z}_\tau, \tau, \mathbf{c}_y),$$

$$\hat{\mathbf{x}} = D_\psi(\mathbf{z}_1).$$

训练时，真实视频先经 VAE:

$$\mathbf{z}_\star = E_\phi(\mathbf{x}),$$

然后构造噪声与数据之间的概率路径并训练速度场。推理时不需要 encoder，只需要文本 encoder、DiT、ODE/离散 flow scheduler 和 VAE decoder。

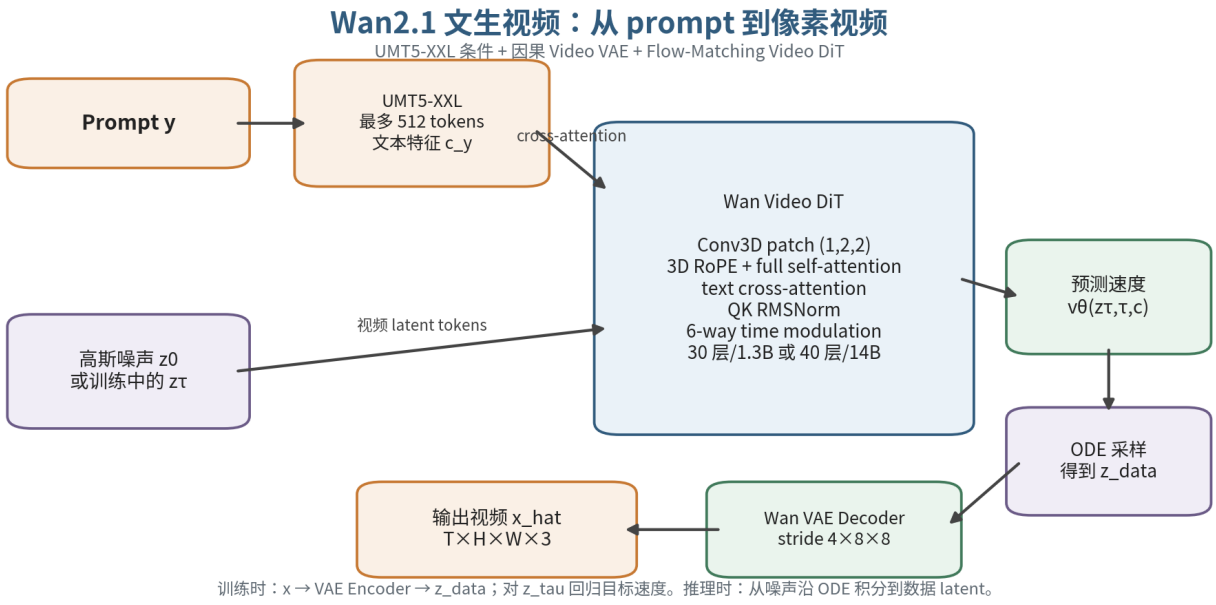

图 3: Wan2.1 文生视频架构

16.2 公开配置的精确结构

官方 T2V 配置显示:

配置	Wan2.1-T2V-1.3B	Wan2.1-T2V-14B
文本编码器	UMT5-XXL	UMT5-XXL
最大文本长度	512	512
VAE stride	$(4, 8, 8)$	$(4, 8, 8)$
VAE latent channel	16	16
DiT patch	$(1, 2, 2)$	$(1, 2, 2)$
hidden d	1536	5120
FFN d_{ff}	8960	13824

配置	Wan2.1-T2V-1.3B	Wan2.1-T2V-14B
heads n_h	12	40
head dimension	128	128
layers L	30	40
self-attention	全局	全局
QK norm	是	是
参数 dtype	BF16	BF16
训练时间离散点	1000	1000
代表输出 fps	16	16

二者保持 head dimension 128，放大方式主要是增加 hidden dimension 与层数。FFN 比例并非标准 $4d$：

$$\frac{8960}{1536} \approx 5.83, \qquad \frac{13824}{5120} = 2.70.$$

因此不能用“每层约 $12d^2$”这一固定经验式精确反推参数量；应读取具体配置。

16.3 为什么使用 UMT5-XXL

UMT5-XXL 是多语言文本 encoder。视频 prompt 往往包含主体、动作、镜头、光照、材质、速度和风格，512 token 上限允许较长描述。文本 embedding 先经过两层 MLP 投影至 DiT hidden dimension：

$$\tilde{\mathbf{c}}_y = W_2\,\mathrm{GELU}(W_1\mathbf{c}_y + b_1) + b_2.$$

使用大型文本 encoder 的代价包括：

- 权重显存和首次编码延迟；
- 训练时若不缓存 embedding，会增加总计算；
- prompt extension 的收益与 T5 表示能力耦合；
- 文本 encoder 冻结时，领域术语只能通过 DiT 学会“解释”固定 embedding。

实际微调可预计算文本 embedding，但 caption augmentation、随机 dropout 或 prompt rewrite 若在线变化，就不能简单缓存唯一 embedding。

16.4 T2V 与 I2V 的条件接口差异

T2V block 使用文本 cross-attention：

$$\mathrm{Attn}(Q_x, K_y, V_y).$$

I2V 版本还从视觉 encoder 取得图像 token，并使用独立的 image key/value 投影：

$$\mathbf{h}_{\mathrm{img}} = \mathrm{Attn}(Q_x, K_{\mathrm{img}}, V_{\mathrm{img}}),$$

$$\mathbf{h}_{\mathrm{text}} = \mathrm{Attn}(Q_x, K_y, V_y),$$

$$\mathbf{h}' = W_o(\mathbf{h}_{\mathrm{img}} + \mathbf{h}_{\mathrm{text}}).$$

独立 K/V 允许图像条件和文本条件具有不同统计。直接相加很简单，但二者强度未必自动平衡：图像条件过强会导致静态复制，文本过强会导致身份和构图漂移。

16.5 为什么 1.3B 是优秀的研究起点

官方材料报告 1.3B 版本推理显存约 8.19 GB，并给出消费级 GPU 运行示例。对研究而言，它有四个优势：

1. 单次实验成本低，可跑多个随机种子；
2. LoRA rank、target modules、数据量和 loss weighting 可做系统网格；

3. 形状与数值错误更容易调试；
4. 可以先验证机制，再迁移到 14B 检验 scaling。

限制也很明确：较小容量可能低估复杂语义、长 prompt 和高质量细节方法的收益。一个在 1.3B 上有效的模块，不一定在 14B 上仍有效；反之，14B 的 emergent ability 也不能用 1.3B 完全模拟。

16.6 输出几何必须满足的约束

官方默认尺寸通常选择能被 VAE stride 与 patch size 整除的值。对于 Wan2.1：

$$H \equiv 0 \pmod{16}, \qquad W \equiv 0 \pmod{16},$$

因为空间总 token stride 为 $8 \times 2 = 16$。帧数常取

$$T = 4k + 1,$$

从而因果 VAE 得到 $T_z = k + 1$。例如 $T = 81$：

$$T_z = 1 + \frac{80}{4} = 21.$$

不满足该形式时，代码可能 padding、floor 或产生边界帧行为。论文实验必须记录原始帧数与 padding 后帧数。

16.7 权重显存的第一性估计

仅模型权重：

$$M_{\text{weight}} = P \cdot b,$$

BF16 时 $b = 2$ bytes。于是：

- 1.3B：约 2.6 GB；
- 14B：约 28 GB。

这不含 T5、VAE、CUDA kernel workspace、activation、CFG 双分支或 offload buffer。因此 14B 不可能仅凭“28 GB 权重”就稳定在 32 GB GPU 完整推理；实践通常需要模型分片、CPU offload、量化或多 GPU。

16.8 模型报告中的“billions of images and videos”如何理解

Wan 技术报告公开其 14B 模型在十亿级图像与视频数据上训练，但没有公开足以重建完整训练语料的逐来源清单、最终去重样本数、总 token 或 GPU-hours。因此：

- 可以确认其训练规模是大规模、图像与视频混合；
- 不能从一句“billions”反推出视频条数、视频小时数或 epoch；
- 不能给出一个虚构的精确训练天数；
- 可以在明确吞吐假设下做 C 级区间估计，但必须与官方事实分开。

常见误区

网页宣传中的“数据量”可能指原始 URL、下载成功资产、shot、clip、图文 pair、caption 版本或训练采样次数。比较两个模型前，先统一单位；否则十亿 image-video pairs 与千万小时视频没有直接可比性。

17. Wan Video VAE：形状、因果性、压缩与重建上限

17.1 为什么 VAE 是系统的第一瓶颈

Video VAE 同时决定：

- DiT 序列长度；
- latent channel 和单 token 信息量；
- 可恢复的空间细节；

- 运动频率与时间边界；
- encoder/decoder 的峰值显存和吞吐；
- 长视频能否通过 chunk streaming 处理。

把 DiT 调得更大，却不测 VAE oracle，是视频生成研究中最常见的归因错误之一。

17.2 Wan2.1 VAE 的结构

官方实现使用 causal 3D convolution、residual blocks、空间 attention，以及分阶段时空 down/up-sampling。主要配置为：

项目	值
base channel	96
latent channel C_z	16
channel multipliers	$[1, 2, 4, 4]$
residual blocks/stage	2
temporal downsample	两次有效，合计 4 倍
spatial downsample	三次，合计 8 倍
总 stride	$(4, 8, 8)$
convolution	causal Conv3D
编解码	支持时间 chunk 与 feature cache

因果卷积对时间维只在过去侧 padding。对 kernel size $k_t = 3$，输出时刻 i 依赖

$$\{i-2, i-1, i\},$$

而不依赖未来帧。好处是可以 chunk streaming 并保持边界一致；代价是开头帧统计与中间帧不同。

17.3 首帧特殊处理

代码在时间上按

$$1, 4, 4, 4, ...$$

的方式输入 encoder：第一块只含首帧，后续每块 4 帧，并使用缓存连接历史。于是

$$T_z = 1 + \left\lfloor \frac{T-1}{4} \right\rfloor.$$

例如：

像素帧数 T	latent 帧数 T_z
1	1
17	5
49	13
81	21
121	31

首帧特殊性对 I2V 很有利：第一帧可以作为稳定锚点。但它也意味着随机裁剪的 clip 起点会被视为“序列开端”，训练数据切片策略会影响边界行为。

17.4 latent normalization

VAE 输出通道的均值与标准差不完全是 0 和 1。官方实现对每个 latent channel 使用

$$\tilde{z}_c = (z_c - \mu_c)/\sigma_c.$$

代码等价地存储 `mean` 与 `1/std`。若微调时遗漏该 scaling，DiT 看到的输入分布会严重偏移：

- loss 初始值异常；
- 某些 channel 梯度过大；
- scheduler 的噪声尺度不再匹配；
- 解码颜色与对比度异常。

加载第三方 VAE 时，必须同时迁移 scaling convention，而不是只替换权重文件。

17.5 压缩率的三种口径

几何压缩

$$\rho_{\text{site}} = s_t s_h s_w = 4 \times 8 \times 8 = 256.$$

每个 latent site 对应约 256 个像素位置。

标量元素压缩

输入每像素有 3 个通道，latent 每 site 有 16 个通道：

$$\rho_{\text{scalar}} = \frac{3 s_t s_h s_w}{C_z} = \frac{3 \times 256}{16} = 48.$$

即浮点元素数量约减少 48 倍。

DiT token 压缩

DiT 再以 patch $(1, 2, 2)$ 聚合 latent，因此每个 Transformer token 对应

$$4 \times (8 \times 2) \times (8 \times 2) = 1024$$

个像素位置的几何块。注意，它不是把 1024 个像素压成一个标量，而是映射成 d 维 token。

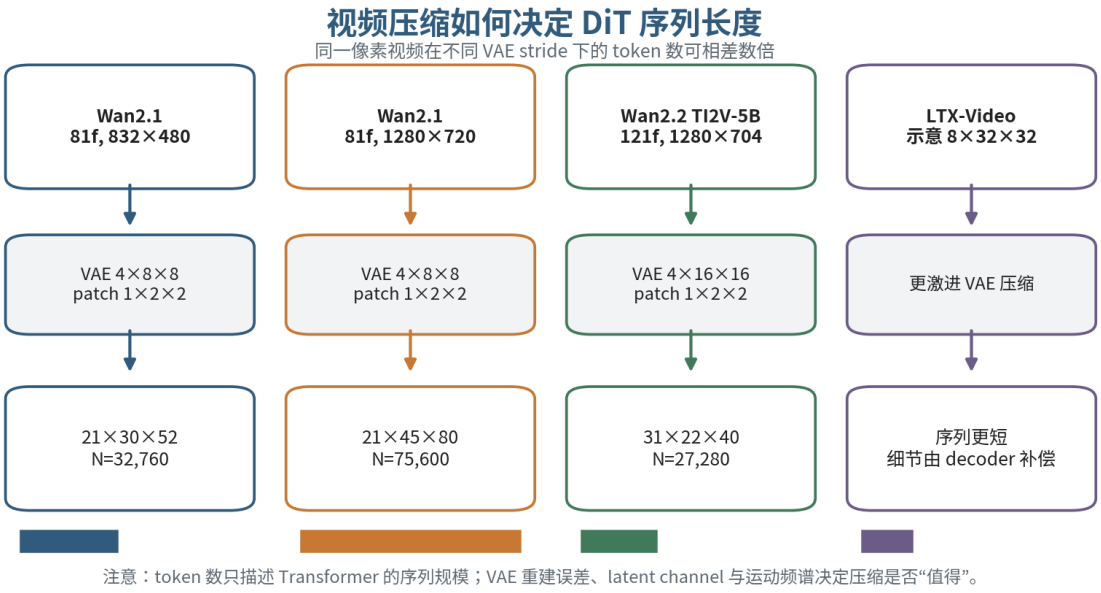

图 4: Wan 与高压缩模型的 token 几何

17.6 两个精确形状例子

81 帧、832×480

按张量顺序 $T \times H \times W$：

$$T_z = 21, \quad H_z = 480/8 = 60, \quad W_z = 832/8 = 104.$$

patch 后：

$$(T_p, H_p, W_p) = (21, 30, 52),$$

$$N = 21 \times 30 \times 52 = 32{,}760.$$

81 帧、1280×720

$$(T_z, H_z, W_z) = (21, 90, 160),$$

$$(T_p, H_p, W_p) = (21, 45, 80),$$

$$N = 75{,}600.$$

分辨率像素数从 832×480 增到 1280×720，约增 2.31 倍；token 数也从 32,760 增到 75,600，约 2.31 倍。但 full attention 的 N^2 项增约

$$(75{,}600/32{,}760)^2 \approx 5.33$$

倍。因此“720p 比 480p 多两倍像素”远低估了注意力成本增长。

17.7 chunked encode/decode 与 cache

长视频若一次性进入 3D VAE，activation 随 T 线性增长。Wan 使用 feature cache，使 encoder/decoder 分块处理。设每块包含 c 个 latent 帧，缓存最近 r 帧特征，则内存近似从

$$O(TCHW)$$

降为

$$O((c + r)CHW),$$

但总计算近似不变。正确实现必须满足：

1. causal padding 与整段推理一致；
2. 缓存 dtype/device 一致；
3. 首块和后续块采用不同边界逻辑；
4. 拼接后帧数精确恢复；
5. chunk size 改变不应显著改变重建结果。

建议测试

$$\Delta_{\text{chunk}} = \|D_{\text{full}}(\mathbf{z}) - D_{\text{chunk}}(\mathbf{z})\|_2.$$

17.8 VAE oracle 评测协议

至少报告：

- PSNR / SSIM：低频重建和整体结构；
- LPIPS：感知差异；
- temporal LPIPS：相邻帧感知变化误差；
- optical-flow endpoint error 或轨迹误差；
- OCR accuracy：画面文字；
- face identity similarity：身份保持；
- 高频纹理频谱：细线、毛发、水波；
- 分速度 bucket：静态、慢运动、快速运动。

对真实视频 $\mathbf{x}$、oracle $\hat{\mathbf{x}}$，相邻帧残差可写为

$$\mathcal{E}_{\text{temp}} = \frac{1}{T-1}\sum_{i=1}^{T-1} \left\| (\hat{\mathbf{x}}_{i+1} - \hat{\mathbf{x}}_i) - (\mathbf{x}_{i+1} - \mathbf{x}_i) \right\|_1 .$$

它比单帧 PSNR 更敏感于时间平滑和运动幅度损失。

17.9 是否应该微调 VAE

大多数领域微调先冻结 VAE，因为：

- 重新训练会改变 latent 分布，使 DiT 预训练失配；
- 需要高质量重建损失、感知网络和对抗训练；
- 编解码器与 DiT 联合训练成本高；
- 小数据容易过拟合纹理。

只有在以下情况才优先考虑 VAE：

- 医学、遥感、动画线稿等域与自然视频相差很大；
- oracle 明确是主要瓶颈；
- 需要更高时间压缩或低延迟；
- 目标是 codec/representation 研究，而非单纯内容适配。

一种安全路线是冻结 decoder 主体，仅训练 latent adapter：

$$\tilde{\mathbf{z}} = A_\omega(\mathbf{z}),$$

或在 encoder/decoder 的少数 residual block 上加低秩模块，并使用 latent distribution matching 防止漂移。

18. Wan Video DiT 与 Flow Matching：从代码块到公式

18.1 patch embedding 与序列化

输入 latent 形状为

$$\mathbf{z}_\tau \in \mathbb{R}^{B\times C_z\times T_z\times H_z\times W_z}.$$

`Conv3d(kernel=stride=patch_size)` 完成 patchification：

$$\mathbf{h}_0 = \text{Flatten}\left(\text{Conv3D}_{p_t,p_h,p_w}(\mathbf{z}_\tau)\right) \in \mathbb{R}^{B\times N\times d}.$$

对 Wan2.1，$C_z = 16$、patch $(1,2,2)$。每个 patch 原始输入维度为 $16\times 1\times 2\times 2 = 64$，再线性映射到 d。

18.2 3D RoPE

每个 token 对应坐标 (i_t, i_h, i_w)。Wan 将 head dimension 分成时间、高、宽三个频率子空间，并对 query/key 施加旋转：

$$\text{RoPE}(\mathbf{q}; i_t, i_h, i_w) = R_t(i_t)R_h(i_h)R_w(i_w)\mathbf{q}.$$

代码中的维度划分不是简单等分，因为 head dimension 必须保持偶数复数对。其核心性质是内积依赖相对位置：

$$\langle R(i)\mathbf{q}, R(j)\mathbf{k}\rangle = \langle \mathbf{q}, R(j-i)\mathbf{k}\rangle.$$

对视频而言，3D RoPE 同时提供时间位移和二维空间位移信息。它没有直接编码真实秒数；16 fps 与 24 fps 若使用相同帧坐标，物理速度语义会不同。因此跨 fps 训练需要 fps conditioning 或规范化采样。

18.3 QK RMSNorm

query/key 投影后先做 RMSNorm:

$$\hat{\mathbf{q}} = \frac{\mathbf{q}}{\sqrt{d_h^{-1}\|\mathbf{q}\|_2^2 + \epsilon}} \odot \gamma_q,$$

key 类似。它限制 attention logits 的尺度，减少大模型训练中 softmax 饱和:

$$A = \mathrm{softmax}\left(\frac{\widehat{Q}\widehat{K}^\top}{\sqrt{d_h}}\right).$$

QK norm 尤其适合长序列与混合分辨率，因为 token 统计变化大。

18.4 一个 Wan attention block

每层主要流程:

1. time embedding 产生 6 组调制向量;
2. AdaLN-modulated self-attention;
3. 文本/图像 cross-attention;
4. AdaLN-modulated FFN。

设时间调制为

$$(\beta_1, \gamma_1, \alpha_1, \beta_2, \gamma_2, \alpha_2) \in \mathbb{R}^{6\times d}.$$

则

$$\tilde{\mathbf{h}} = \mathrm{LN}(\mathbf{h}) \odot (1+\gamma_1) + \beta_1,$$

$$\mathbf{h}' = \mathbf{h} + \alpha_1 \odot \mathrm{SelfAttn}(\tilde{\mathbf{h}}),$$

$$\mathbf{h}'' = \mathbf{h}' + \mathrm{CrossAttn}(\mathrm{Norm}(\mathbf{h}'), \mathbf{c}),$$

$$\mathbf{h}_{\mathrm{out}} = \mathbf{h}'' + \alpha_2 \odot \mathrm{FFN}\left(\mathrm{LN}(\mathbf{h}'') \odot (1+\gamma_2) + \beta_2\right).$$

18.5 文本 cross-attention

文本 token 数最多 512。cross-attention 复杂度为

$$O(NN_t d),$$

其中 $N_t \le 512$。当视频 $N = 75{,}600$ 时，这仍低于 self-attention 的 $O(N^2 d)$，但每层都会执行，且文本 K/V 投影和缓存策略影响显存。

在 CFG 中，conditional 与 unconditional 分支通常具有不同文本 embedding。若模型主干不支持批处理共享，可产生接近 2 倍 DiT forward。

18.6 Flow Matching 目标

采用噪声端点 $\mathbf{z}_0 \sim \mathcal{N}(0, I)$、数据端点 $\mathbf{z}_1 = \mathbf{z}_\star$，最简单直线路径:

$$\mathbf{z}_\tau = (1-\tau)\mathbf{z}_0 + \tau \mathbf{z}_1.$$

目标速度为

$$\mathbf{u}_\tau = \frac{d\mathbf{z}_\tau}{d\tau} = \mathbf{z}_1 - \mathbf{z}_0.$$

训练损失：

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[w(\tau)\|v_\theta(\mathbf{z}_\tau, \tau, \mathbf{c}) - (\mathbf{z}_1 - \mathbf{z}_0)\|_2^2\right].$$

实际模型可能使用非均匀时间采样、time shift 或不同路径参数化。只看到 `flow matching` 名称不足以复现；必须核对：端点方向、τ 分布、目标速度、loss weighting 和 scheduler 映射。

18.7 time shift 的作用

推理 scheduler 常对均匀离散时间 t 做 shift，例如通用形式

$$\tilde{t} = \frac{\mu t}{1 + (\mu - 1)t},$$

其中 $\mu > 1$ 会把更多步数分配到某一噪声区域。不同实现的方向约定可能相反，因此不要只比较 shift 数字大小。

研究时绘制：

$$\{\tilde{t}_k\}_{k=0}^{K}, \qquad \Delta_k = |\tilde{t}_{k+1} - \tilde{t}_k|,$$

确认步数究竟集中在高噪声的布局阶段，还是低噪声的细节阶段。

18.8 CFG 在速度场中的形式

条件与无条件速度：

$$v_c = v_\theta(\mathbf{z}_\tau, \tau, \mathbf{c}), \qquad v_u = v_\theta(\mathbf{z}_\tau, \tau, \emptyset).$$

标准 guidance：

$$v_{\mathrm{cfg}} = v_u + s(v_c - v_u).$$

$s = 1$ 等价条件分支，$s > 1$ 强化条件方向。过大 guidance 可能导致：

- 运动僵硬；
- 饱和、过锐化；
- 人体结构崩坏；
- 多主体被压成一个高置信模式；
- temporal flicker。

视频需要同时评估文本忠实和动态，多数情况下不存在对所有 prompt 最优的单一 s。

18.9 输出头与零初始化

最终 head 也受时间 embedding 调制，并将 d 维 token 投影回

$$p_t p_h p_w C_z$$

维。输出层零初始化使初始网络接近零速度场，避免训练刚开始随机大输出破坏数值稳定。类似 AdaLN-Zero 的思想广泛用于 DiT。

18.10 主要单步 FLOPs 推导

每层近似：

- Q、K、V、O 四个 $d \to d$ 线性层：$8Nd^2$ FLOPs；
- 两个 FFN 线性层：$4Ndd_{\mathrm{ff}}$；

- $QK^\top$ 与 AV：$4N^2d$。

于是

$$C_{\text{fwd}} \approx L(8Nd^2 + 4Ndd_{\text{ff}} + 4N^2d).$$

对 Wan2.1-14B、81 帧 480p：

$$N = 32{,}760,\ d = 5120,\ d_{\text{ff}} = 13824,\ L = 40,$$

得到约 1.5 PFLOP/forward 的数量级。720p 时约 6.2 PFLOP。该式不含 cross-attention、norm、激活函数、VAE、文本 encoder、padding、通信和 kernel inefficiency，因此是 C 级下界式估计。

18.11 一段最小训练伪代码

```
for batch in loader:
    video = batch["video"].to(device)          # [B, 3, T, H, W]
    text = batch["caption"]

    with torch.no_grad():
        z_data = vae.encode(video)              # latent normalization included
        text_ctx = text_encoder(text)

    z_noise = torch.randn_like(z_data)
    tau = sample_training_time(z_data.shape[0], device=device)
    tau_view = tau.view(-1, 1, 1, 1, 1)

    z_tau = (1.0 - tau_view) * z_noise + tau_view * z_data
    target_v = z_data - z_noise

    # classifier-free conditioning dropout
    text_ctx = maybe_drop_condition(text_ctx, p=cfg_dropout)
    pred_v = dit(z_tau, tau, context=text_ctx)

    loss = weighted_mse(pred_v, target_v, tau)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad(set_to_none=True)
```

真实训练还必须处理：variable length/size packing、loss mask、FSDP、mixed precision、gradient clipping、EMA、checkpoint recovery、data corruption 与 all-reduce NaN。

19. Wan 的数据、预训练、推理成本与官方仓库

19.1 一个可扩展的预训练课程

尽管完整内部 recipe 未公开，结合技术报告可将大模型视频预训练理解为以下层次：

1. **图像基础**：构图、纹理、对象和语言对齐；
2. **短低分辨率视频**：基本运动与时间一致性；
3. **更长/更高分辨率视频**：长程状态与细节；
4. **图像条件任务**：I2V、首尾帧、参考主体；
5. **编辑与控制**：源视频、mask、姿态、轨迹；
6. **质量后训练**：审美、偏好、prompt following、失败样本修复。

课程采样可写为

$$q \sim \pi_k(q),$$

其中 q 是任务/尺寸/质量 bucket，k 是训练阶段。π_k 随训练进度改变，而非固定混合。

19.2 图像为何仍是视频模型的重要数据

图像没有真实运动，但具备：

- 数量大；
- 高分辨率比例高；
- caption 通常更准确；
- 复杂对象与美学风格丰富；
- 解码成本低。

可以将图像复制成静态视频，或直接设 $T_z = 1$。但若图像占比过高，模型会学到“低风险静态解”：文本对齐和画质高，motion magnitude 却低。需要通过 motion-aware sampling、视频 loss 权重或动态后训练抵消。

19.3 caption pipeline

网页 alt-text 和标题通常只描述主题，不描述动作。视频 caption 应尽量覆盖：

$$\mathbf{y} = \{\text{subject, attributes, action, scene, relation, camera, lighting, style, temporal order}\}.$$

一个实用 pipeline：

1. ASR 提取语音；
2. OCR 识别画面文字并标记水印；
3. 均匀帧、关键帧和短 clip 输入 VLM/MLLM；
4. 生成 dense caption 与结构化标签；
5. 与原始文本做一致性检查；
6. 过滤 hallucination、主体数量错误和时间顺序错误；
7. 保存短 caption、长 caption 与结构字段，而非只存一段文本。

训练时随机选择 caption 粒度可提高鲁棒性：

$$\mathbf{y} \sim \pi_s y_{\text{short}} + \pi_l y_{\text{long}} + \pi_o y_{\text{original}}.$$

19.4 数据去重与泄漏

视频近重复可能经过裁剪、加字幕、变速、镜像或重新编码。应组合：

- perceptual hash；
- 视频 embedding 近邻；
- 音频 fingerprint；
- OCR/ASR 文本；
- 时序局部匹配。

评测集泄漏会使 VBench prompt 或公开视频测试被训练样本覆盖。仅按 URL 去重远远不够。至少在 shot embedding 上做 ANN 检索，并人工复核近邻。

19.5 未公开训练成本的诚实估计

总训练 FLOPs 可写为

$$F_{\text{train}} = \sum_{k=1}^{S} B_k \left(C_{\text{fwd},k} + C_{\text{bwd},k} \right),$$

其中反向通常约为前向的 2 倍或更高，粗略可取

$$C_{\text{train,step}} \approx 3 C_{\text{fwd}} B.$$

GPU-hours：

$$\text{GPUh} = \frac{F_{\text{train}}}{3600\, U},$$

U 是每 GPU 的实际有效 FLOP/s，而非峰值。若使用 G 块 GPU，墙钟时间：

$$\text{hours}_{\text{wall}} = \frac{\text{GPUh}}{G}.$$

Wan 未披露足够的 S, B_k, N_k, U，所以不能得到唯一答案。合理做法是给场景表：

假设	低值	高值
平均每样本 token	16k	50k
有效吞吐/GPU	200 TFLOP/s	500 TFLOP/s
总优化步数	100k	500k
global batch	64	512

然后用脚本输出范围，并明确这是“假设敏感性分析”，不是官方训练账单。

19.6 推理成本分解

总推理时间近似

$$t_{\text{total}} = t_{\text{text}} + K\, n_{\text{branch}}\, t_{\text{DiT}} + t_{\text{VAE-decode}} + t_{\text{I/O}}.$$

标准 CFG 时 $n_{\text{branch}} = 2$；某些实现把两分支拼 batch，一次 kernel 完成但 FLOPs 仍近似翻倍。高分辨率下 DiT 占主导；高压缩/少步模型中 VAE decoder 与数据搬移比例会上升。

推理报告应包含：

- cold start 与 warm run；
- 是否包含 T5 加载/编码；
- 是否包含 VAE decode 和 mp4 写盘；
- NFE 与 CFG 分支数；
- offload/量化；
- GPU 型号、数量、互联；
- 峰值 VRAM 与 CPU RAM；
- 输出实际帧数、fps、尺寸。

19.7 官方仓库调用图

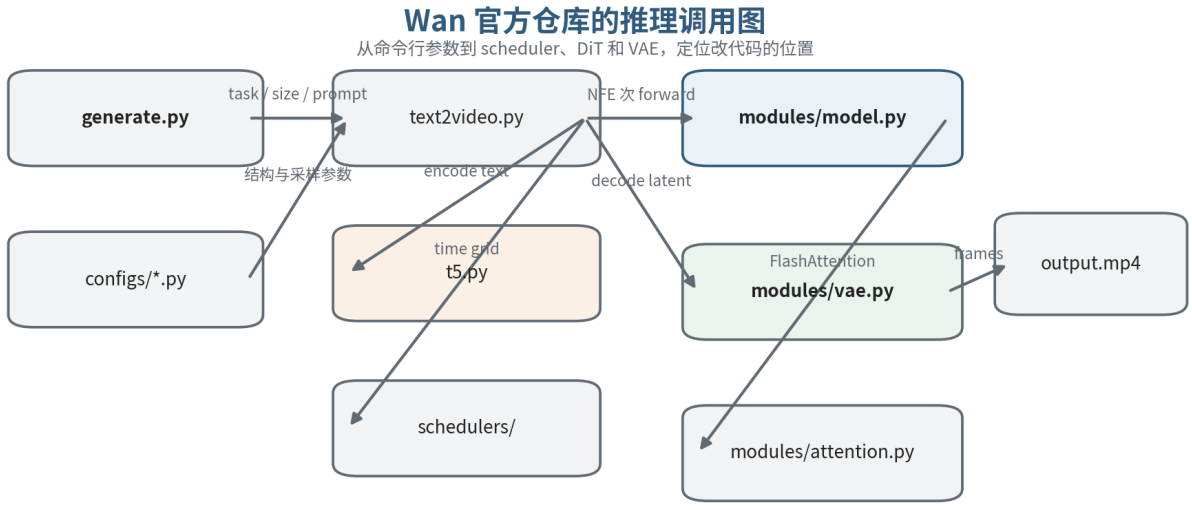

图 5: Wan 官方仓库的推理调用图

典型职责：

- `generate.py`：解析 task、size、prompt、seed、offload；
- `wan/configs/*.py`：模型维度、VAE stride、采样默认值；
- `wan/text2video.py`：加载组件、编码文本、构造 scheduler、循环采样；
- `wan/modules/model.py`：Video DiT block、RoPE、cross-attention、unpatchify；
- `wan/modules/vae.py`：causal VAE 与 chunk cache；
- `wan/modules/attention.py`：FlashAttention 包装；
- `wan/schedulers/*`：time grid 与 solver。

研究者不应一开始就修改 `generate.py` 的表层参数，而应先画出张量经过各模块的形状。

19.8 形状 hook

```python
from __future__ import annotations
from collections import defaultdict
import torch

shapes: dict[str, list[tuple[int, ...]]] = defaultdict(list)

def hook(name: str):
    def _fn(module, inputs, output):
        def record(x):
            if isinstance(x, torch.Tensor):
                shapes[name].append(tuple(x.shape))
        if isinstance(output, (tuple, list)):
            for item in output:
                record(item)
        else:
            record(output)
    return _fn

handles = []
for name, module in model.named_modules():
    if name.endswith(("patch_embedding", "self_attn", "cross_attn", "head")):
        handles.append(module.register_forward_hook(hook(name)))

# run one very small inference step here
# ...
for h in handles:
    h.remove()

for name, value in shapes.items():
    print(name, value[:3])
```

在 FSDP 或 compiled graph 下 hook 可能影响性能，只用于最小调试，不用于正式 benchmark。

19.9 复现的最小四项校验

1. 相同 seed、prompt、negative prompt、shape、scheduler；
2. checkpoint 与 VAE 哈希一致；
3. latent normalization 和 dtype 一致；
4. 输出视频写盘前的 raw frames 一致。

mp4 编码器可能产生像素差异，因此比对模型输出时先保存无损帧或 tensor。

20. Wan2.2：时间专家、高压缩 VAE 与统一 TI2V

20.1 两条不同的效率路线

Wan2.2 提供的核心方向可分为：

- **A14B 路线**：用高噪声/低噪声两个专家提升容量和阶段专门化；
- **TI2V-5B 路线**：用更高 VAE 压缩和较小 DiT 降低序列与参数成本，并统一 T2V/I2V。

二者不是同一种模型的简单大小版本，不能只按 5B 与 14B 比较。

20.2 A14B 的 timestep-specialized experts

A14B 包含两个独立专家：

- high-noise expert：更偏全局布局、对象关系、粗运动；
- low-noise expert：更偏边缘、纹理、面部和局部细节。

官方配置每个专家均为：

项目	值
hidden d	5120
FFN d_{ff}	13824
heads	40
layers	40
patch	$(1, 2, 2)$
high/low checkpoints	两套
默认 NFE	40
boundary	0.875
guidance	low-noise 3.0，high-noise 4.0

其门控是时间确定性的：

$$e(\tau) = \begin{cases} \text{high}, & \tau \in \mathcal{I}_{\text{high}}, \\ \text{low}, & \tau \in \mathcal{I}_{\text{low}}. \end{cases}$$

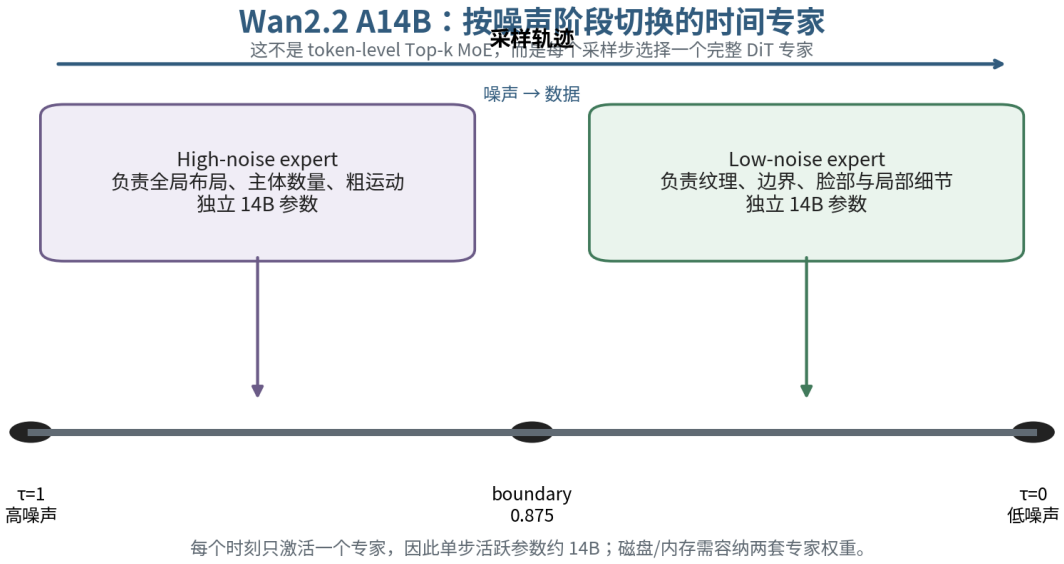

图 6: Wan2.2 时间专家

这不是 token-level Top-k MoE。每个采样步的所有视频 token 进入同一个专家；因此单步 active parameters 约 14B，但权重存储接近两套 14B。若内存不足，可在阶段切换时 offload 一个专家、加载另一个，代价是 PCIe/NVLink 传输。

20.3 为什么按时间分专家合理

高噪声状态的信息熵高，模型主要决定低频结构；低噪声状态接近数据流形，模型主要修复高频细节。可用频域直觉表示：

$$\mathbb{E}\|\hat{\mathbf{z}}_1^{\text{lowfreq}} - \mathbf{z}_1^{\text{lowfreq}}\|^2$$

更受高噪声阶段影响，而

$$\mathbb{E}\|\hat{\mathbf{z}}_1^{\text{highfreq}} - \mathbf{z}_1^{\text{highfreq}}\|^2$$

更受低噪声阶段影响。共享模型必须用同一组参数拟合两种统计，专家化可以减少梯度冲突。

但硬边界也有风险：

- 边界附近速度场不连续；
- 两专家预测尺度不同导致求解器误差；
- 训练样本在两区间分布不平衡；
- 磁盘和加载成本近似翻倍。

可研究 soft blending：

$$v(\tau) = \lambda(\tau) v_{\text{high}}(\tau) + [1 - \lambda(\tau)] v_{\text{low}}(\tau),$$

但会在过渡区域同时运行两专家，增加 FLOPs。

20.4 审美标签与数据扩展

官方说明相对 Wan2.1，Wan2.2 增加约 65.6% 图像和 83.2% 视频，并引入更精细审美标签。注意这些是相对增长，不能推导绝对条数。

审美条件可表示为

$$p_\theta(\mathbf{x} \mid \mathbf{y}, \mathbf{a}),$$

其中 $\mathbf{a}$ 可包括摄影、光照、色调、构图和质量等级。若 $\mathbf{a}$ 由自动模型产生，标签噪声可能让模型学会风格捷径；应做人评和反事实测试，例如只改变“low-key lighting”而保持内容不变。

20.5 TI2V-5B 的公开配置

配置项	Wan2.2-TI2V-5B
文本编码器	UMT5-XXL
VAE stride	$(4, 16, 16)$
VAE latent channel	48
DiT patch	$(1, 2, 2)$
hidden d	3072
FFN d_{ff}	14336
heads	24
layers	30
默认帧数	121
fps	24
sample shift	5.0
默认 NFE	50
guidance	5.0
代表尺寸	1280×704
官方最低显存示例	24 GB（offload 等开启）

统一 T2V/I2V 可将图像条件视为可选变量 $\mathbf{I}$：

$$p_\theta(\mathbf{x} \mid \mathbf{y}, \mathbf{I}), \qquad \mathbf{I} = \emptyset \text{ 时退化为 T2V.}$$

训练时应显式 condition dropout，防止模型只依赖图像：

$$\mathbf{I}' = \begin{cases} \emptyset, & r < p_{\text{drop}}, \\ \mathbf{I}, & \text{otherwise.} \end{cases}$$

20.6 Wan2.2 VAE 的 16×16 空间压缩

代码显示新 VAE 在 encoder 前做空间 `patchify=2`，网络内部再做 8 倍空间下采样，总 stride 为 16；decoder 反向 unpatchify。latent channel 增至 48。

几何压缩：

$$4 \times 16 \times 16 = 1024.$$

标量元素压缩：

$$\rho_{\text{scalar}} = \frac{3 \times 4 \times 16 \times 16}{48} = 64.$$

相比 Wan2.1 的 48，标量压缩进一步增加。更多 channel 用于补偿每个 site 覆盖更大的空间范围。

20.7 121 帧 1280×704 的 token 数

$$T_z = 1 + \frac{121 - 1}{4} = 31,$$

$$H_z = 704/16 = 44, \qquad W_z = 1280/16 = 80.$$

patch 后：

$$(T_p, H_p, W_p) = (31, 22, 40),$$

$$N = 31 \times 22 \times 40 = 27{,}280.$$

它生成 121 帧，token 数却低于 Wan2.1 81 帧 480p 的 32,760。加上较小 hidden dimension，单步 DiT 粗估约 0.48 PFLOP，而 Wan2.1-14B 720p 约 6.2 PFLOP。

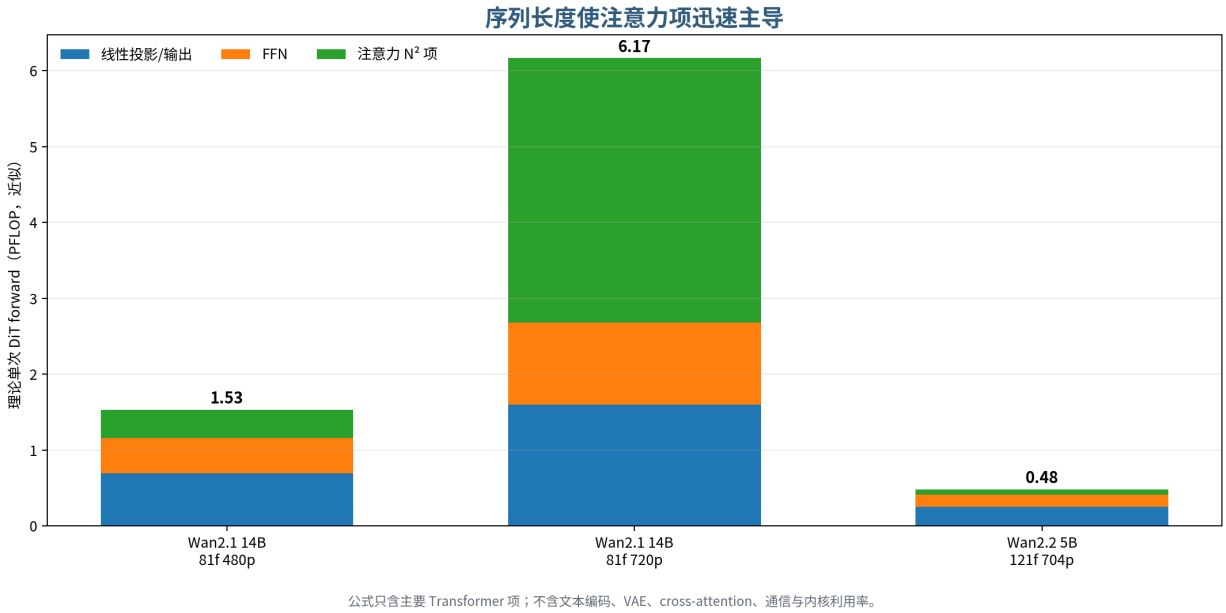

图 7: 三种 Wan 配置的理论单步计算

高压缩是 24GB 运行的关键之一，但需要特别测试 OCR、面部、线条、手指和高速小物体的 VAE oracle。

20.8 S2V 与 Animate：条件时间轴对齐

Speech-to-Video 需要将音频特征 $\mathbf{a}_{1:T_a}$ 对齐视频 latent $\mathbf{z}_{1:T_z}$。可以用时间插值或 cross-attention：

$$\tilde{\mathbf{a}}_j = \sum_{i=1}^{T_a} w_{ji}\mathbf{a}_i, \qquad w_{ji} \propto \exp\left(-\frac{|t_i^{(a)} - t_j^{(z)}|^2}{2\sigma^2}\right).$$

Animate 类任务还要输入参考身份与姿态序列。条件不只是“多一个 encoder”，而是多采样率、多空间坐标和多 CFG 分支的统一。

20.9 Wan2.1 与 Wan2.2 如何选择

目标	推荐起点	原因
课程学习、单卡 LoRA、改 block	Wan2.1-1.3B	结构简单、迭代快
大规模开源 T2V 研究	Wan2.1/2.2-14B	生态与质量较强
24GB 消费卡 720p T2V/I2V	Wan2.2-TI2V-5B	高压缩 VAE、统一任务
timestep specialization	Wan2.2-A14B	清晰的高/低噪声专家
VAE 压缩对照	Wan2.1 vs TI2V-5B	8×8 与 16×16 天然实验组
低层代码可解释性	Wan2.1	模块更直接、社区资料多

核心结论

Wan2.2 的效率不是一个技巧：A14B 用时间专家增加总容量而保持单步活跃容量；TI2V-5B 用更高压缩 VAE、较小 DiT 和 offload 降低运行成本。论文比较必须说明研究的是“容量专家化”还是“表示压缩”。

21. Wan 的微调、控制扩展与论文级复现

21.1 先确定微调目标

常见目标可分为：

1. **概念/风格适配**：人物、商品、艺术风格；
2. **领域适配**：医学、遥感、工业、游戏、动画；
3. **动作适配**：特定运动、镜头或物理过程；
4. **条件扩展**：姿态、深度、轨迹、音频、布局；
5. **编辑能力**：局部替换、主体迁移、时间编辑；
6. **效率研究**：蒸馏、稀疏、缓存、量化、VAE 压缩。

不同目标对应不同 trainable modules。用同一个 LoRA recipe 处理所有任务通常不是最优。

21.2 LoRA 数学与参数量

对线性层 $W \in \mathbb{R}^{d_{out}\times d_{in}}$，LoRA：

$$W' = W + \frac{\alpha}{r}BA,$$

其中

$$A \in \mathbb{R}^{r\times d_{in}}, \qquad B \in \mathbb{R}^{d_{out}\times r}.$$

新增参数：

$$P_{\text{LoRA}} = r(d_{in} + d_{out}).$$

若对方阵 $d \times d$ 的 Q/K/V/O 全部加 rank-r：

$$P_{\text{attn,layer}} = 4 \cdot 2rd = 8rd.$$

14B 配置 $d = 5120, r = 16, L = 40$，仅 self-attention 四投影约

$$8 \times 16 \times 5120 \times 40 \approx 26.2\text{M}$$

参数。若再加 FFN 和 cross-attention，会显著增加。

21.3 target modules 的选择

目标	优先模块	解释
风格/外观	cross-attn Q/K/V、FFN、部分 self-attn	改变文本到视觉映射和局部纹理
新主体身份	cross-attn、image-condition projection、早中层 self-attn	需要条件绑定和跨帧身份
动作/运动	self-attn Q/K/V/O、时间 modulation	运动依赖视频 token 交互
新控制条件	新 encoder + zero-init injection，必要时少量主干 LoRA	先保护底座，再学习条件接口
领域纹理	FFN、输出附近层，必要时 VAE adapter	高频外观与 latent 解码有关
采样蒸馏	主干广泛更新或 student	目标速度场全轨迹改变，局部 LoRA 可能不足

只对 cross-attention 做 LoRA 可能提升 prompt 对齐，却几乎不改变运动动力学。只对 self-attention 做 LoRA 可能学会动作，但领域术语绑定较弱。

21.4 显存预算

冻结基座、只训练 LoRA 时，基座权重仍需驻留或 offload，但不保存其梯度/optimizer state。大致：

$$M \approx M_{\text{base}} + M_{\text{act}} + M_{\text{LoRA-grad+opt}} + M_{\text{temp}}.$$

LoRA 参数的 AdamW 状态若使用 BF16 参数/梯度与 FP32 moments，大约每参数 12–16 bytes；但 activation 通常是主瓶颈。减少显存的优先级：

1. 降低 N：分辨率、帧数、VAE/patch；
2. activation checkpointing；
3. sequence/context parallel；
4. FlashAttention；
5. gradient accumulation；
6. CPU optimizer/offload；
7. 8-bit optimizer；
8. 量化冻结基座（QLoRA 类）。

量化视频 DiT 的风险比 LLM 更高：速度场回归对小数值误差敏感，误差会在多步 ODE 中累积。必须比较不同 NFE 和不同噪声阶段。

21.5 数据量与正则化

概念 LoRA 可从几十到几百段高质量 clip 起步；领域/动作适配通常需要数千到数十万 clip。真正重要的是有效覆盖：

- 视角、背景、光照；
- 动作速度与方向；
- 主体尺度；
- 镜头运动；
- prompt 表达变化；
- 负例与非目标属性。

小数据可使用 prior preservation：

$$\mathcal{L} = \mathcal{L}_{\text{target}} + \lambda_{\text{prior}} \mathcal{L}_{\text{base-domain}},$$

避免模型把所有 prompt 都生成目标主体/风格。

21.6 静态塌缩与运动重加权

若训练集包含大量近静态 clip，MSE 会偏好低运动。定义运动量

$$m(\mathbf{x}) = \frac{1}{T-1}\sum_i \|\mathbf{x}_{i+1} - \mathbf{x}_i\|_1,$$

可按 bucket 重采样或加权：

$$w_m = \mathrm{clip}\left(\frac{m(\mathbf{x})}{\mathrm{median}(m)}, w_{\min}, w_{\max}\right).$$

但像素差会把相机运动和闪烁都当作动态，更稳健的是 optical flow、轨迹或 VLM motion label。

21.7 条件控制的 zero-init 注入

对姿态/深度条件 encoder $G_\omega(\mathbf{r})$，在第 ℓ 层注入：

$$\mathbf{h}'_\ell = \mathbf{h}_\ell + Z_\ell(G_\omega(\mathbf{r})),$$

其中 Z_ℓ 为零初始化线性/卷积层。训练开始时系统等价基座，随后逐渐学习控制，降低灾难性破坏。

多条件可使用门控：

$$\mathbf{h}'_\ell = \mathbf{h}_\ell + \sum_{k=1}^{K} g_{\ell k}(\tau, \mathbf{c}) Z_{\ell k}(G_k(\mathbf{r}_k)).$$

高噪声阶段可加强布局/轨迹，低噪声阶段加强边缘/身份。

21.8 一个可执行的训练阶段

阶段 A：32 样本过拟合

目标不是泛化，而是验证：

- 数据和 caption 对齐；
- VAE latent 无 NaN；
- loss 可降；
- LoRA 真正插入目标层；
- checkpoint 可恢复；
- 采样器与训练方向一致。

阶段 B：小规模泛化

训练 1k–10k clips，固定 480p、49/81 帧；每 500–1000 steps 保存固定 prompt × seed 网格。监控：

- validation flow loss；
- CLIP/VLM alignment；
- motion magnitude；
- identity；
- temporal flicker；
- base prompt regression。

阶段 C：课程扩展

增加分辨率、时长、任务与 caption 粒度。每次只改变一个维度，避免无法归因。

21.9 训练 skeleton 中容易漏的细节

```
# pseudocode: distributed LoRA fine-tuning
model.requires_grad_(False)
insert_lora(model, targets=target_modules, rank=rank, alpha=alpha)
mark_only_lora_trainable(model)

vae.eval().requires_grad_(False)
text_encoder.eval().requires_grad_(False)

for batch in loader:
    with torch.no_grad(), torch.autocast("cuda", dtype=torch.bfloat16):
        z_data = encode_and_scale(vae, batch.video)
        cond = encode_text(text_encoder, batch.caption)

    tau = sample_tau(batch_size=z_data.size(0), scheme=time_scheme)
    noise = torch.randn_like(z_data)
    z_tau, target = construct_flow_pair(noise, z_data, tau)

    cond = cfg_dropout(cond, p=0.1)
    with torch.autocast("cuda", dtype=torch.bfloat16):
        pred = model(z_tau, tau, cond, attention_mask=batch.token_mask)
        loss = masked_weighted_mse(pred, target, batch.latent_mask, tau)
        loss = loss / grad_accum_steps

    scaler_or_plain_backward(loss)
    if is_update_step:
        clip_grad_norm_(trainable_params, 1.0)
        optimizer.step()
        scheduler.step()
        optimizer.zero_grad(set_to_none=True)
```

必须有 latent mask；variable-size padding 区域若进入 MSE，会训练模型拟合无意义的零 padding。

21.10 论文级消融矩阵

假设提出“噪声阶段自适应 motion LoRA”，应至少包含：

因子	取值
base	1.3B 与至少一个更大底座
LoRA target	cross-only / self-only / both
rank	4 / 16 / 64
time gate	none / hard / learned soft
data	同一训练 clips 与采样次数
inference	相同 NFE/CFG/time shift
seeds	至少 3，报告均值与 CI
metrics	文本、质量、动态、时间、身份、人评
cost	train GPUh、peak VRAM、inference latency

改进若只在单个 prompt 或单个 seed 可见，不构成可靠结论。

21.11 复现阶梯

每一级都应产生可审计产物：

- L0：环境锁文件和权重 hash；
- L1：VAE shape/oracle report；
- L2：固定推理样例和 tensor checksum；
- L3：小数据 loss 曲线和 overfit video；
- L4：LoRA checkpoint 与 config；
- L5：逐 prompt 结果、失败分类与置信区间；
- L6：完整消融、成本-质量 Pareto 和跨底座验证。

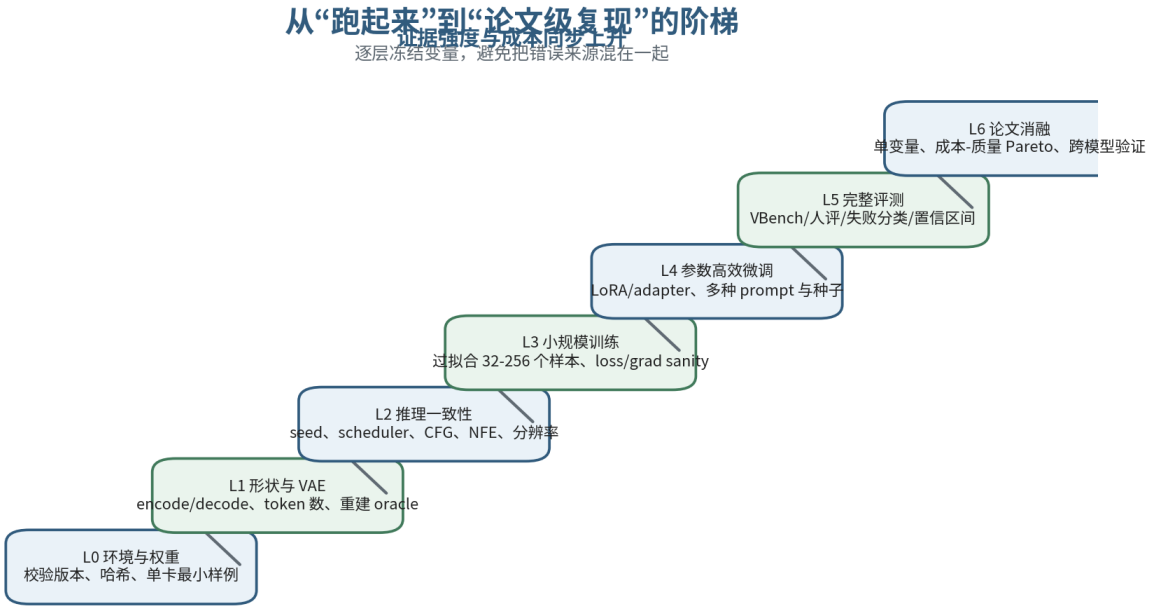

图 8: 从运行到论文级复现

21.12 Wan 研究的高价值问题

1. **VAE-生成器协同**：如何提高 16×16 压缩而不牺牲文字和高速运动?
2. **阶段专家**：硬边界、软门控、共享底层/分离高层哪个更优?
3. **动态自适应 NFE**：简单 prompt 和低运动样本是否需要更少步?
4. **条件冲突**：文本、首帧、姿态、身份条件如何分层 guidance?
5. **长视频**：如何缓存历史而不冻结运动或累积身份漂移?
6. **小模型蒸馏**：如何保持 motion diversity，而非只蒸馏单一 teacher trajectory?
7. **评测可诊断性**：如何把 VAE、planner、DiT 与 solver 的错误分离?

练习与自检

在附带的 `wan_shape_cost_calculator.py` 中输入 81 帧 480p、81 帧 720p 与 121 帧 704p 三组配置，核对 token 数和理论 FLOPs。然后将帧数翻倍、空间尺寸缩小到原来的 $1/\sqrt{2}$，观察 N 与 N^2 项是否保持不变。

22. Bernini 架构：连续视觉语义规划如何连接视频扩散

Bernini 的核心主张是将成熟的 MLLM 与成熟的视频扩散模型按能力分工:

- MLLM 擅长理解多模态输入、关系、编辑指令与高层推理;
- DiT 擅长将条件渲染为连续、高保真视频像素;
- 二者通过**目标视觉语义表示**连接，而不是只把 MLLM 输出改写成长 prompt。

22.1 生成与编辑的统一条件分布

设输入条件为

$$\mathbf{c} = \{\mathbf{y}, \mathbf{I}_{1:m}, \mathbf{x}_{\text{src}}, \mathbf{r}\},$$

可包含文本、参考图像、源视频和编辑指令。Bernini 引入 latent semantic plan $\mathbf{s}$:

$$p(\mathbf{x}_{\text{tgt}} \mid \mathbf{c}) = \int p_\psi(\mathbf{x}_{\text{tgt}} \mid \mathbf{s}, \mathbf{c}_{\text{low}}) p_\omega(\mathbf{s} \mid \mathbf{c})\, d\mathbf{s}.$$

$\mathbf{c}_{\text{low}}$ 表示编辑任务中用于保留纹理、背景和运动的源 VAE 特征。Planner 负责 p_ω，Renderer 负责 p_ψ。

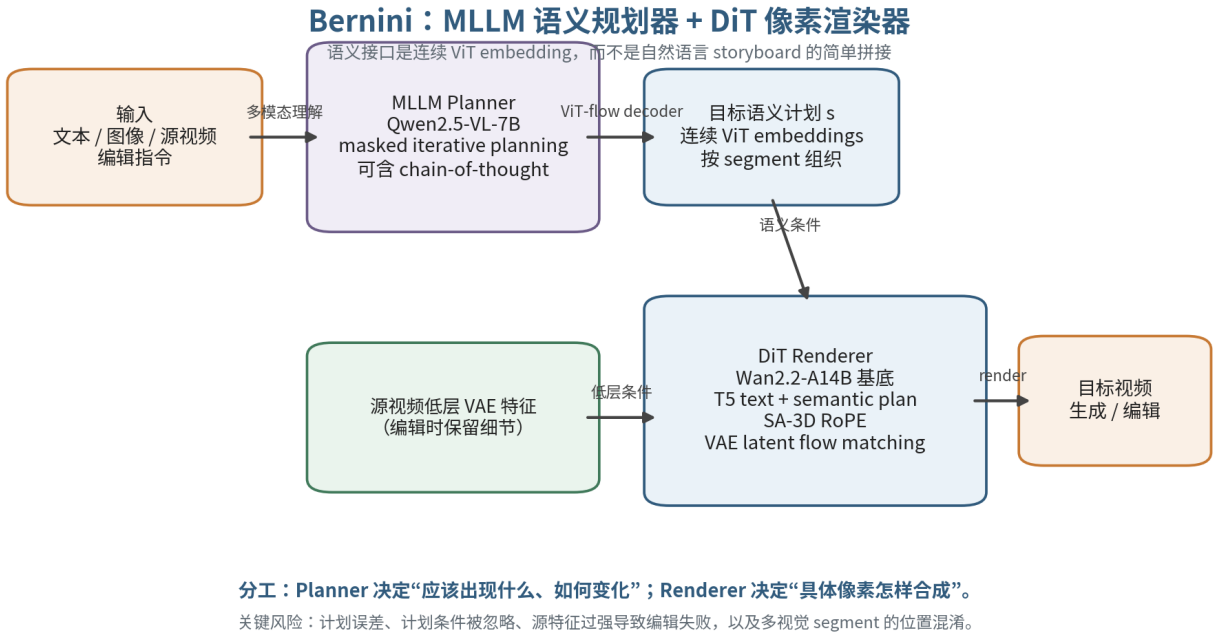

图 9: Bernini Planner-Renderer 架构

22.2 为什么普通 prompt rewrite 不够

Prompt rewrite 的接口是自然语言：

$$\tilde{\mathbf{y}} = g(\mathbf{c}), \qquad \mathbf{x} \sim p_\psi(\mathbf{x} \mid F_{\text{text}}(\tilde{\mathbf{y}})).$$

它存在三个瓶颈：

1. 语言难以无歧义表达逐帧位置、姿态、遮挡和细粒度状态；
2. renderer 的文本 encoder 可能压缩或忽略长描述；
3. 编辑中的源视觉信息需要先“翻译成语言”再翻回视觉，造成信息损失。

Bernini 直接预测 ViT embedding 空间中的目标语义。设目标视频经视觉 encoder 得到

$$\mathbf{s}_\star = F_{\text{ViT}}(\mathbf{x}_{\text{tgt}}) \in \mathbb{R}^{N_s \times d_s}.$$

Planner 学习

$$p_\omega(\mathbf{s}_\star \mid \mathbf{c}),$$

避免把全部视觉语义压成离散词序列。

22.3 连续 ViT embedding 为什么适合作为接口

ViT representation 位于像素与语言之间：

- 比像素/VAE latent 更抽象，减少 renderer 前的低层细节负担；
- 比自然语言带宽高，可表示形状、空间布局、主体状态和视觉相似性；
- 与 MLLM 的视觉 token 原生兼容；
- 可以由真实目标视频监督，无需人工写 storyboard；
- 可用于生成、I2V、V2V 与参考编辑的统一目标。

但连续语义接口也带来问题：

- 维度高，planner 输出不是普通离散 next-token prediction；
- embedding 的可解释性较弱；
- ViT 表示可能包含低层纹理或数据集偏差；

- plan 的时间/segment 排列必须明确；
- renderer 可能忽略 plan，退化为原始 T5 条件生成。

22.4 MLLM Planner 的混合输出空间

公开实现以 Qwen2.5-VL-7B 为 planner 基底。其序列中可同时包含：

- 文本输入与文本推理 token；
- 源图像/视频视觉 token；
- 待预测的目标语义 segment；
- 任务控制 token 与 mask。

文本 token 使用标准 next-token prediction：

$$\mathcal{L}_{\mathrm{NTP}} = -\sum_{j} \log p_{\omega}(w_j \mid w_{<j}, \mathbf{c}).$$

连续 ViT token 不能用 softmax 词表预测，因此由额外的 flow decoder 从 MLLM hidden state 生成。

22.5 ViT-flow decoder

设 MLLM 在目标语义位置产生隐藏状态 $\mathbf{h}$，真实目标 ViT embedding 为 $\mathbf{s}_1$，噪声为 $\mathbf{s}_0 \sim \mathcal{N}(0, I)$。构造

$$\mathbf{s}_\tau = (1-\tau)\mathbf{s}_0 + \tau \mathbf{s}_1,$$

目标速度

$$\mathbf{u}_s = \mathbf{s}_1 - \mathbf{s}_0.$$

语义 flow decoder 学习

$$\mathcal{L}_{\mathrm{ViT-FM}} = \mathbb{E}\left[\left\| v_{\omega}^{(s)}(\mathbf{s}_\tau, \tau, \mathbf{h}) - \mathbf{u}_s \right\|_2^2\right].$$

这意味着 planner 本身是一个混合生成器：离散文字使用 autoregressive objective，连续视觉语义使用 Flow Matching。

22.6 masked iterative planning

Bernini 不一定一次生成全部目标语义，而使用 masked iterative modeling：初始目标位置部分或全部被 mask，模型重复选择位置并填充/更新语义。

设 mask $\mathbf{m}^{(k)} \in \{0,1\}^{N_s}$，第 k 轮已知语义为

$$\tilde{\mathbf{s}}^{(k)} = (1-\mathbf{m}^{(k)}) \odot \mathbf{s}^{(k)} + \mathbf{m}^{(k)} \odot \mathbf{e}_{\mathrm{mask}}.$$

Planner 预测被 mask 位置，并更新置信度；下一轮减少 mask。抽象算法：

```
initialize all target semantic positions as MASK
for k = 1,...,K_plan:
    run MLLM on input + current semantic plan
    select masked positions according to schedule/confidence
    sample continuous ViT embeddings with semantic flow decoder
    write predictions back; optionally remask low-confidence positions
return semantic plan s
```

相比严格 left-to-right，它可并行预测多个位置，并允许后续轮根据全局计划修正前面低置信 token。

22.7 mask ratio 的 Beta 分布

论文对不同任务使用不同 Beta 分布采样 mask ratio：

$$r \sim \mathrm{Beta}(a_q, b_q),$$

q 是任务类型。公开配置中代表参数包括：

任务	(a_q, b_q)	直觉
T2I	$(5, 1.1)$	高 mask，主要从语言生成目标语义
T2V	$(8, 1.05)$	更高 mask，开放式视频规划
I2I	$(8, 1.05)$	参考图像提供部分语义
I2V	$(10, 1)$	强调根据输入图像推演目标视频
V2V/IV2V	$(12, 0.9)$	大比例目标未知，同时依赖源视频与指令

因为 a 大、b 近 1，分布偏向高 mask ratio，训练模型在少量可见目标信息下完成规划。具体数值属于实现 recipe，不应脱离任务混合孤立解释。

22.8 chain-of-thought 的作用与边界

Planner 可先生成文本推理，再生成视觉计划：

$$p(\mathbf{r}, \mathbf{s} \mid \mathbf{c}) = p(\mathbf{r} \mid \mathbf{c}) p(\mathbf{s} \mid \mathbf{r}, \mathbf{c}).$$

$\mathbf{r}$ 可以描述：要保持哪些主体、改变哪些属性、动作如何发展、镜头如何移动。优势是把理解能力转移给生成；风险是：

- 文本 reasoning 可能幻觉；
- 更长序列增加推理延迟；
- rationale 可能与真正影响输出的隐藏状态不一致；
- 训练时 oracle reasoning 与推理时 generated reasoning 存在 exposure gap。

正确消融应比较：无 reasoning、人工/教师 reasoning、模型自生成 reasoning，并检查 plan 与最终视频，而不只看文本看似合理。

22.9 Segment-Aware 3D RoPE

多个视觉输入可能各自从局部坐标 $(0, 0, 0)$ 开始。普通 3D RoPE 对来自不同 segment、但坐标相同的 token 给相同位置相位，造成参考图、源视频和目标计划混淆。

设 segment id 为 g，空间时间坐标为 (i_t, i_h, i_w)。SA-3D RoPE 增加 segment phase：

$$R_{\mathrm{SA3D}}(g, i_t, i_h, i_w) = R_g(g) R_t(i_t) R_h(i_h) R_w(i_w).$$

于是两个 token 的注意力内积依赖

$$(\Delta g, \Delta i_t, \Delta i_h, \Delta i_w),$$

而不仅是 3D 相对坐标。可将 R_g 看作额外一维 RoPE，或在频率中加入 segment-specific offset。

SA-3D RoPE 的目标不是简单“加 segment embedding”，而是在 query/key 旋转相位层面显式区分视觉来源。论文消融显示它能降低 reference leakage，即参考内容错误复制到目标位置。

22.10 Renderer 的条件融合

公开实现以 Wan2.2-A14B 为 renderer 基底，并保留原始 T5 文本条件。MLLM 的倒数第二层 hidden state 经零初始化一层 MLP 投影后，与 T5 条件拼接/融合：

$$\mathbf{c}_{\mathrm{render}} = \mathrm{Concat}\left[\mathbf{c}_{\mathrm{T5}}, Z_\rho(\mathbf{h}_{\mathrm{MLLM}})\right],$$

其中 Z_ρ 初始输出接近 0。这样训练开始时 renderer 接近原 Wan 行为，随后逐渐利用 semantic plan。

编辑任务还输入源视频 VAE 低层特征：

$$\mathbf{c}_{\mathrm{low}} = G(E_\phi(\mathbf{x}_{\mathrm{src}})).$$

高层语义计划回答“改成什么”，低层源特征回答“哪些具体细节要保留”。

22.11 为什么语义计划与源低层特征必须分开

若只用 semantic plan：

- 背景纹理、人物身份和相机运动可能丢失；
- 编辑结果像重新生成，而非编辑。

若只用源低层特征：

- 模型容易复制源视频；
- 大幅语义编辑难以执行；
- 新对象和新运动被源特征压制。

可写成两个互相竞争的条件方向：

$$v = v_0 + s_s(v_s - v_0) + s_l(v_l - v_0),$$

其中 s_s 是语义 guidance，s_l 是低层保持 guidance。二者需要按任务调节。

22.12 Planner-Renderer 的误差分解

设 oracle plan 为 $\mathbf{s}_\star$，预测 plan 为 $\hat{\mathbf{s}}$。总误差可概念分解：

$$\mathcal{E}_{\text{total}} = \underbrace{\mathcal{E}_{\text{renderer}}(\mathbf{s}_\star)}_{\text{给正确计划仍渲染失败}} + \underbrace{\Delta_{\text{plan}}}_{\text{预测计划造成的额外误差}} + \underbrace{\Delta_{\text{interface}}}_{\text{renderer 忽略或误读计划}}.$$

必须做三组评测：

1. **No-plan**：仅 T5/原底座；
2. **Predicted-plan**：完整系统；
3. **Oracle-plan**：真实目标视频 ViT embedding 条件。

若 oracle-plan 很强、predicted-plan 弱，问题在 planner；若二者都弱，问题在 renderer/interface；若 predicted 与 no-plan 接近，可能是 condition collapse。

22.13 与传统 storyboard planner 的比较

接口	可解释性	信息带宽	监督来源	可编辑性	主要风险
扩写 prompt	高	低	文本/LLM	高	文本 encoder 忽略细节
结构化 JSON/ storyboard	很高	中	LLM/人工/规则	很高	离散结构表达有限
布局/轨迹	高	中高	检测/跟踪	高	难表达纹理与隐式语义
连续 ViT plan	中低	高	目标视频自动提取	中	难诊断、表示偏差
VAE latent/ keyframes	低	很高	像素视频	中	planner 负担接近生成本身

Bernini 选择连续 ViT plan，是在语义与像素之间取中间带宽。未来研究可使用多层计划：语言 storyboard + layout/trajectory + ViT embeddings。

核心结论

MLLM Planner 不是“用大语言模型润色 prompt”。Bernini 的关键在于：Planner 直接生成目标视觉语义空间中的连续表示；Renderer 以该表示和低层源特征为条件完成 VAE latent flow。其研究问题是模块接口与误差分工，而不仅是增加一个更大的文本模型。

23. Bernini 的数据、三阶段训练、系统优化与实验解读

23.1 数据组成：配对编辑数据比单纯 T2V 更关键

公开论文披露了多类数据。

视频 pair

约 20M 视频 pair，来自通用 T2V 视频池，代表性筛选包括：

- pair 相似度约 0.65–0.95；
- 时长约 2–10 秒；
- 人类/非人类主体约 1:1，由 Qwen3-VL-30B-A3B 类模型标注；
- 每个原始视频最多构造约 100 对；
- 编辑/关系 prompt 由更大 Qwen3-VL-235B-A22B 类模型生成。

中等相似度区间很重要：过低相似度不是可学习编辑，过高相似度接近复制。pair 采样目标是让源与目标共享部分内容，同时存在清晰可描述变化。

图像 pair

约 30M 图像 pair，部分来自超过 300k 教程视频，代表相似度约 0.75–0.95。教程视频天然包含同一对象在操作前后、不同阶段或视角的图像，适合学习编辑关系。

interleaved understanding data

约 10M 来自 OmniCorpus 等图文交错数据，另有约 2M 由视频构造的交错样本，用于维持 MLLM 理解、多图关系和序列推理能力。

self-text reasoning data

约 1M 自生成文本推理样本，用于让 planner 在生成视觉计划前形成显式任务分解。

这些数字是 A 级论文披露，但不代表训练时每类被完整遍历一次；采样权重、重复次数、去重后独立来源和许可结构仍需区分。

23.2 三阶段训练总览

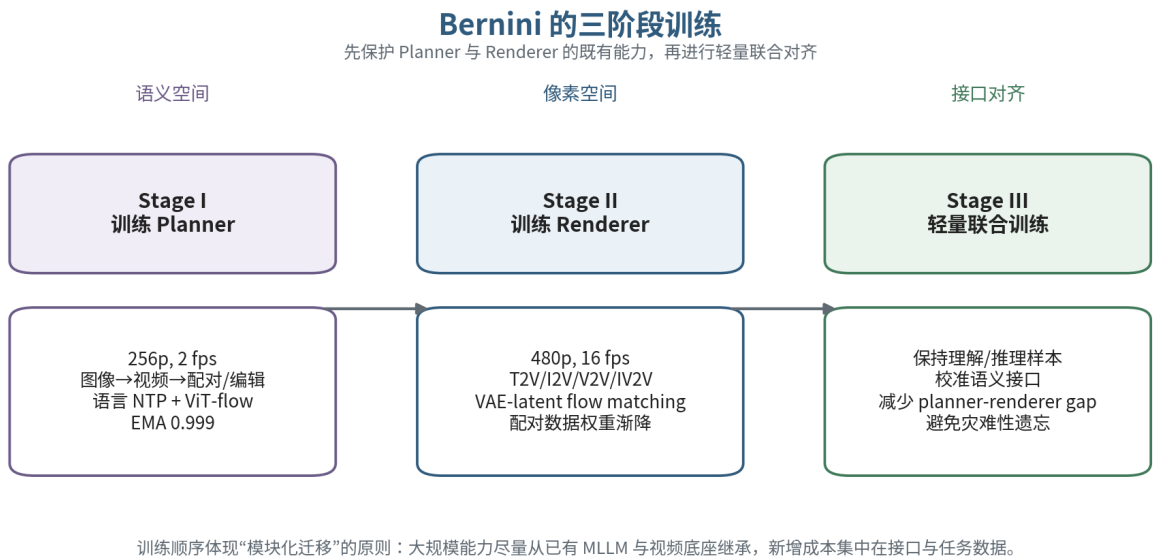

图 10: Bernini 三阶段训练

Stage I: Planner

代表配置：256p、2 fps，学习率 10^{-5}、EMA 0.999。课程从 T2I 开始，再到 T2V、图像/视频 pair 和理解任务。

目标:

$$\mathcal{L}_{\text{planner}} = \lambda_{\text{text}}\mathcal{L}_{\text{NTP}} + \lambda_{\text{vit}}\mathcal{L}_{\text{ViT-FM}} + \lambda_{\text{aux}}\mathcal{L}_{\text{understanding}}.$$

低分辨率、低 fps 降低目标 ViT 序列和数据解码成本。Planner 主要学语义，不需要从一开始处理 480p/16fps 全细节。

Stage II: Renderer

代表配置：480p、16 fps。使用 T2V、I2V、V2V、IV2V 等任务训练 VAE latent flow。pair data 的权重从较高值逐步线性衰减到 0，使模型早期学会条件接口，后期恢复高质量自然视频分布。

目标:

$$\mathcal{L}_{\text{renderer}} = \mathbb{E}_{q\sim\pi(q),\tau\sim p_q(\tau)}\left[w_q(\tau)\|v_\psi(\mathbf{z}_\tau,\tau,\mathbf{c}_{\text{T5}},\mathbf{s},\mathbf{c}_{\text{low}}) - \mathbf{u}_q\|_2^2\right].$$

Stage III: 轻量联合训练

联合更新 planner-interface-renderer 的部分或全部参数，以缩小 predicted-plan 与 renderer 训练条件之间的 gap，同时混入 understanding/reasoning 数据防止 MLLM 灾难性遗忘。

23.3 为什么先分开训练再联合

直接端到端从头训练会有:

- MLLM 和 14B renderer 同时占用巨大显存;
- 两侧预训练能力被随机接口梯度破坏;
- planner 输出快速漂移，renderer 的条件分布不稳定;
- 难以诊断性能来自哪一侧。

分阶段等价于先优化

$$\omega^* = \arg\min_\omega \mathcal{L}_{\text{planner}},$$

再优化

$$\psi^* = \arg\min_\psi \mathcal{L}_{\text{renderer}}(\omega^*),$$

最后在邻域内联合调整 (ω, ψ)。这是一种 block coordinate optimization，牺牲理论上的一次性全局最优，换取稳定和可复用。

23.4 任务相关的噪声时间分布

论文对图像与视频任务使用不同 time sampling；代表 time shift 包括:

任务	代表 shift
T2I	3
I2I	4
T2V	3
I2V/V2V/IV2V	5

编辑任务需要在保留源信息与产生变化之间平衡，较大 shift 可把训练密度放在更关键噪声段。由于 scheduler 的时间方向实现可能不同，复现时应画出实际 $p_q(\tau)$，而不是仅抄数字。

23.5 多源增量 guidance

Bernini 推理可能同时有：源视频、源图像、文本和 semantic plan。对条件集合

$$\mathcal{C}_0 \subset \mathcal{C}_1 \subset \cdots \subset \mathcal{C}_M,$$

可构造增量 guidance:

$$v_{\text{guided}} = v(\mathcal{C}_0) + \sum_{m=1}^{M} s_m \left[v(\mathcal{C}_m) - v(\mathcal{C}_{m-1}) \right].$$

例如依次加入 source video、reference image、text、semantic plan。每一差分近似该条件的边际方向。优点是可独立调节“保留”“文本编辑”“计划执行”；缺点是分支数可能达到 $M+1$，推理成本显著增加。

Adaptive Projected Guidance（APG）类方法可将过强 guidance 分解为与条件方向平行/正交部分，限制范数，缓解过饱和:

$$\Delta v = v_c - v_u, \qquad \Delta v_{\parallel} = \frac{\langle \Delta v, v_c \rangle}{\|v_c\|^2} v_c,$$

再对平行分量衰减或投影。具体实现应以代码为准。

23.6 训练系统：为什么长视频不是只靠 FSDP

论文报告的系统优化包括:

- FSDP 参数/梯度/optimizer 分片;
- direct index scatter 与预分配 buffer，将每 GPU 内存从约 72 GB 降到约 40 GB，并消除约 17 GB 中间张量;
- activation offload，使可处理序列从约 100K token 提升到约 440K token;
- FlashAttention 4、FlexAttention、异步 QKV、TND layout、定制 RMSNorm;
- Ulysses sequence parallel;
- sample packing 与负载均衡;
- 数据加载和通信重叠。

上述数字是论文在其环境中的报告，不是对任意硬件的保证。

23.7 direct index scatter 的内存直觉

variable-size packed sample 常先构造大中间张量再 copy 到最终 buffer:

$$M_{\text{peak}} = M_{\text{final}} + M_{\text{intermediate}} + M_{\text{workspace}}.$$

若预先计算每个 sample 的目标 offset，并直接 scatter 到预分配序列，可去掉 $M_{\text{intermediate}}$。这类“看似数据布局”的优化，对 400K token 序列可能比减少一个小模块更重要。

23.8 Ulysses sequence parallel

将 sequence 维切到 p 个 GPU：每卡持有约 N/p token。attention 前通过 all-to-all 重新分布 heads，使每卡计算部分 attention heads 的完整序列，再交换回来。

理想 activation 内存从

$$O(Nd)$$

降至

$$O(Nd/p),$$

但通信量近似 $O(Nd)$，因此依赖高速 NVLink/InfiniBand。长序列更易受网络带宽影响，必须报告 communication overlap 与有效 tokens/s。

23.9 packing 与负载均衡

若各 GPU 分到的 token 数差异大，同步训练由最慢 rank 决定。定义

$$\rho_{\text{imbalance}} = \frac{N_{\max}}{N_{\min}}.$$

论文报告优化后最大/最小 workload 比小于约 1.01，并带来约 15% 吞吐收益。这里的关键不是某个固定算法，而是：bucket 只按分辨率还不够，还要考虑帧数、文本长度、任务分支和条件 token。

23.10 端到端吞吐收益如何归因

论文报告整体训练吞吐可提升约 4.5 倍，多 GPU 推理超过 7.2 倍。不能把该倍数归因于单一 FlashAttention；它是算法、数据布局、并行、offload 和 kernel 的复合结果。

严谨系统消融应依次添加：

1. baseline FSDP；
2. packing；
3. sequence parallel；
4. memory layout/scatter；
5. fused kernels；
6. overlap/offload；

并报告每步 tokens/s、峰值显存、通信占比和可支持最大 token。

23.11 CFG distillation 与 ReFlow

Bernini 先蒸馏多分支 CFG，再做 ReFlow/少步蒸馏，使 student 约 4 NFE 接近 teacher 80 NFE（论文报告条件下）。

CFG distillation

teacher guided field：

$$v_T^{\text{cfg}} = v_T^u + s(v_T^c - v_T^u).$$

student 直接回归：

$$\mathcal{L}_{\text{CFG-distill}} = \|v_S(\mathbf{z}_\tau, \tau, \mathbf{c}) - v_T^{\text{cfg}}(\mathbf{z}_\tau, \tau, \mathbf{c})\|^2.$$

这样推理只需单分支。

ReFlow/trajectory straightening

用 teacher ODE 生成噪声-数据配对 $(\mathbf{z}_0, \tilde{\mathbf{z}}_1)$，再训练 student 走更直路径：

$$\mathbf{z}_\tau = (1-\tau)\mathbf{z}_0 + \tau\tilde{\mathbf{z}}_1,$$

$$\mathcal{L}_{\text{ReFlow}} = \|v_S(\mathbf{z}_\tau, \tau) - (\tilde{\mathbf{z}}_1 - \mathbf{z}_0)\|^2.$$

少步成功依赖 trajectory curvature 降低，而不只是换一个高阶 solver。

23.12 公开实验结果如何读

论文在 VBench 报告代表值：

模型	Total	Quality	Semantic	Dynamics
Bernini	84.64	85.18	82.49	81.11
Wan2.2-A14B	84.79	85.33	82.61	69.72

这些是论文报告值。可观察到 Bernini 总分与 Wan2.2 接近，但 dynamics 子项更高；这支持 planner/训练改善动态的可能性，却不能单独证明因果，因为二者的数据、prompt processing、采样和后训练也不同。

论文还在 OpenVE 报告 Bernini overall 约 4.04、对比模型 VINO 约 3.18。不同 benchmark 的 judge 和 prompt 分布不同，不应把绝对数跨表比较。

23.13 必要消融

Bernini 类系统至少应做：

消融	诊断问题
去掉 planner	收益是否只是 renderer/data
prompt rewrite 替代 continuous plan	连续视觉接口是否必要
oracle ViT plan	planner 还有多大上限差距
去掉 source low-level features	编辑保持是否来自低层条件
普通 3D RoPE 替代 SA-3D RoPE	segment leakage 是否改善
无 CoT / teacher CoT / self CoT	reasoning 是否真正有效
分阶段 vs 端到端	模块化训练是否保护能力
单 CFG vs incremental guidance	多条件控制收益与额外 NFE
teacher 80 NFE vs student 4 NFE	蒸馏的质量-成本曲线

23.14 与 Wan、CogVideoX 的本质比较

维度	Wan2.2-A14B	CogVideoX	Bernini
高层条件	T5 文本/图像等	文本-视频 expert fusion	T5 + MLLM continuous semantic plan
生成主干	时间专家 Video DiT	expert Transformer	Wan2.2-A14B renderer
编辑	需任务适配	非核心主线	生成/编辑统一目标
多视觉输入位置	标准 3D/条件接口	模型特定	SA-3D RoPE 强调 segment
训练分工	单一生成底座	单一生成底座	planner、renderer、联合三阶段
主要成本	大 DiT、多步	大 DiT、多步	MLLM + 多条件分支 + renderer
主要优势	强开放生成底座	深层文本融合与长视频	理解能力迁移到规划/编辑
主要风险	语义规划有限	编辑与多源统一有限	系统复杂、plan 难诊断、延迟高

23.15 局限与开放问题

1. 连续 plan 的可解释性和可控编辑不如结构化 storyboard；
2. planner 自身会幻觉，且错误会被 renderer 放大；
3. 多条件增量 CFG 计算昂贵；
4. MLLM 与 14B renderer 的组合提高训练/部署门槛；
5. complex editing 仍依赖 prompt rewrite 和数据覆盖；
6. 最强封闭系统的纯视觉质量可能仍更高；
7. 权重、数据与完整训练代码的开放程度决定独立复现上限。

练习与自检

设计一个 Bernini 误差诊断实验：选 200 条编辑 prompt，保存 no-plan、predicted-plan、oracle-plan 三组输出；分别用 VLM judge、身份相似度、源视频 optical flow 保持和人工偏好评测。根据三组差距，将失败归因给 planner、interface 或 renderer。

24. 视频生成数据工程：从公开数据集到可训练课程

模型规模容易被看到，数据分布往往被低估。对视频生成，数据工程至少占据以下作用：

- 决定模型能否学习真实运动，而不是静态图像插值；
- 决定 caption 是否包含动作和时间顺序；
- 决定人物、物体、镜头与风格的长尾覆盖；
- 决定训练吞吐是否被解码和 I/O 限制；
- 决定评测泄漏、版权、隐私和安全风险。

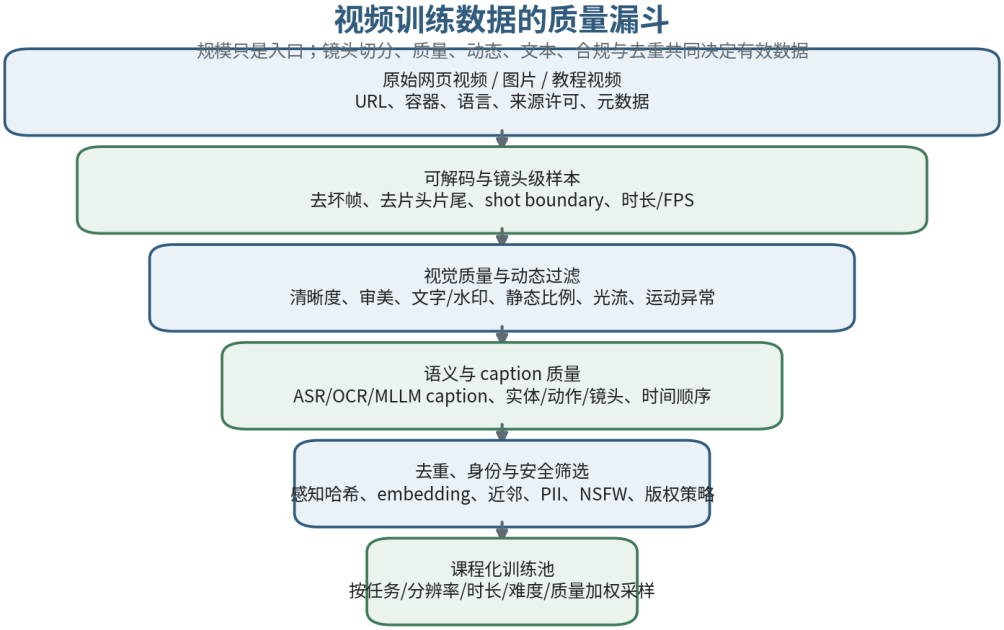

图 11: 视频训练数据的质量漏斗

24.1 数据规模的五种单位

同一个数据集可以用不同单位描述:

单位	记号	适合回答的问题
原始视频数	N_{asset}	来源多样性有多大
总时长	H_{video}	原始时间覆盖多少
shot/clip 数	N_{clip}	可训练样本有多少
总解码帧	N_{frame}	预处理和存储成本
VAE/DiT token	N_{token}	实际训练计算量

若平均 clip 时长为 d 秒、采样 fps 为 f，则帧数近似

$$N_{\text{frame}} = N_{\text{clip}} df.$$

若 VAE/patch 后每帧约 n_s 个空间 token，则

$$N_{\text{token}} \approx N_{\text{clip}} \left(1 + \frac{df - 1}{s_t}\right) n_s.$$

两个同为 10M clips 的数据集，若一个是 2 秒 8 fps，另一个是 10 秒 24 fps，计算规模完全不同。

24.2 代表性公开数据集

WebVid-2M / WebVid-10M

WebVid-2M 随 Frozen in Time 提出，包含超过 2M 个弱标注网页视频；后来生态中常见更大 WebVid-10M 变体。优点是容易使用、覆盖广；缺点是 caption 短、网页噪声大、动作描述弱，且下载可用率会随时间变化。

适合：小中规模 T2V 预训练、检索预训练和 pipeline 验证。不适合单独承担高质量复杂动作训练。

HowTo100M

HowTo100M 包含约 1.22M narrated instructional videos、136M clips、超过 23k 任务。文本主要来自 ASR narration，往往描述步骤但与画面存在时间偏移。

优势：人-物交互、操作过程和长时序丰富。风险：

- 语音可能描述过去/未来动作；
- 画面有手部、字幕、切镜和讲解者；
- 教程域偏差明显；
- ASR 文本不等价于视觉 caption。

适合从同一教程构造 Bernini 类前后状态 pair，但必须做视觉一致性筛选。

HD-VILA-100M

公开报告包含约 371.5k 小时 720p 视频，覆盖 15 个 YouTube 类别。其价值在高分辨率和多样性，但“100M”是规模命名，不应直接当作高质量生成 clips 数。

Panda-70M

Panda-70M 从约 3.8M 个高分辨率视频切分语义一致 clips，并使用多种跨模态教师生成/选择 caption，最终形成约 70M video-text pairs。

其方法论价值在于：caption 不是由一个 VLM 单次生成，而是组合字幕、帧、原始文本和多个 teacher，再用 retrieval 模型选择。对生成训练，caption selection 与 caption generation 同样重要。

InternVid

InternVid 报告超过 7M 原始视频、约 760k 小时、234M clips 和 4.1B words，使用多尺度描述。它适合理解与生成联合预训练，但规模巨大意味着下载、去重、许可和重新 caption 成本很高。

OpenVid-1M / OpenVidHD-0.4M

OpenVid-1M 提供超过 1M 高质量 text-video pairs，并从中构造约 433k 个 1080p 视频子集 OpenVidHD-0.4M。它适合资源有限实验室做高质量预训练或精调，但仍需核对具体可下载资产和使用条款。

MiraData

MiraData 强调更长时长、更高运动和结构化 caption。caption 从主体、场景、运动、镜头等多个视角生成，并配套 MiraBench。它适合研究长视频、motion strength 与 dense caption，但公开版本和实际可用资产应以项目状态为准。

HOIGen-1M

HOIGen-1M 聚焦 human-object interaction，约一百万高质量视频，并结合 MLLM 自动筛选与人工清洗。它体现“专项数据集”路线：不是追求全部开放域，而是填补手-物交互这一模型弱点。

24.3 数据集规模不可直接排序质量

一个实用质量函数：

$$Q_i = w_r q_i^{\text{resolution}} + w_a q_i^{\text{aesthetic}} + w_m q_i^{\text{motion}} + w_c q_i^{\text{caption}} + w_s q_i^{\text{safety}} - w_d q_i^{\text{duplicate}}.$$

训练价值还依赖覆盖度：

$$V(\mathcal{D}) \neq \sum_i Q_i,$$

因为 100 万条近重复高质量猫视频不如覆盖人物、动物、物理、镜头、场景和动作长尾的多样数据。可使用聚类覆盖或 facility-location objective：

$$F(S) = \sum_{i \in \mathcal{D}} \max_{j \in S} \text{sim}(e_i, e_j),$$

选择既高质量又覆盖语义空间的子集 S。

24.4 下载与解码审计

原始 manifest 至少包含：

```
{
  "sample_id": "source_video__shot_003__clip_001",
  "source_dataset": "...",
  "source_uri_hash": "...",
  "license_tag": "...",
  "start_sec": 12.4,
  "end_sec": 18.0,
  "container": "mp4",
  "codec": "h264",
  "width": 1280,
  "height": 720,
  "fps_num": 30000,
  "fps_den": 1001,
  "num_frames": 168,
  "decode_ok": true,
  "sha256": "...",
  "parent_sha256": "..."
}
```

不要只存浮点 fps。30000/1001 ≈ 29.97 若被写成 30，长视频时间戳会漂移。variable frame rate 视频还要保存逐帧 PTS 或在预处理阶段转为 constant frame rate。

24.5 shot boundary 与 clip 切分

一个训练 clip 应尽量包含单一连续镜头，否则模型可能把硬切学成物体瞬移。shot boundary 可结合：

- HSV/embedding 帧差；
- histogram distance；
- learned shot detector；
- fade/dissolve 识别；
- OCR/黑帧/片头规则。

若原 shot 长度为 D，训练窗口 d，stride s，窗口数：

$$N_{\text{win}} = 1 + \left\lfloor \frac{D-d}{s} \right\rfloor .$$

小 stride 会制造大量近重复 clip，导致数据量数字虚高。应保存 `parent_shot_id`，在 train/val split 时按 parent 分组，避免同一镜头泄漏。

24.6 帧率规范化

统一采样 fps f_t：目标帧时刻

$$t_i = t_0 + i/f_t.$$

从原视频选择最近 PTS 或插值。低 fps 降低成本，却可能 alias 快速运动；高 fps 增加相邻冗余。课程可从 2–8 fps 学高层语义，再升至 16–24 fps 学平滑运动。

建议保存：

- 原始 fps；
- 采样 fps；
- speed factor；
- 是否插帧；
- 真实时长。

否则模型把 8 fps 重复到 24 fps 的视频当作“慢运动”。

24.7 分辨率与 aspect-ratio buckets

固定 resize 到正方形会扭曲人物和相机运动。定义 bucket 集

$$\mathcal{B} = \{(H_b, W_b, T_b)\}_{b=1}^{B}.$$

对样本选择使裁剪损失最小的 bucket：

$$b^* = \arg\min_b \left| \log \frac{H/W}{H_b/W_b} \right| + \lambda \left| \log \frac{HW}{H_b W_b} \right|.$$

然后短边 resize + center/random crop。训练日志应按 bucket 报告吞吐和 loss，避免某些极端长宽比长期欠训练。

24.8 视觉质量过滤

常用信号：

- Laplacian/learned blur score;
- compression/block artifact;
- exposure、saturation;
- aesthetic score;
- 水印/字幕占比;
- face/hand/object completeness;
- 黑边、拼接、屏幕录制;
- 生成内容/真实内容分类;
- NSFW、暴力和隐私。

硬阈值容易删除长尾。更稳健的是分桶采样：高质量样本权重大，低质量但稀有语义保留少量。

$$\pi_i \propto \exp(\lambda Q_i)/n_{\mathrm{cluster}(i)}^{\gamma}.$$

24.9 动态质量与“伪运动”

运动分数可用 optical flow：

$$m_{\mathrm{flow}} = \mathrm{median}_{i,p} \|\mathbf{f}_i(p)\|_2.$$

但大 flow 可能来自：

- 镜头快速摇动;
- 转场;
- 画面闪烁;
- 字幕滚动;
- 屏幕录制。

应进一步估计全局相机变换 H_i，将 flow 分解为相机与残差对象运动：

$$\mathbf{f}_i(p) = \mathbf{f}_i^{\mathrm{camera}}(p) + \mathbf{f}_i^{\mathrm{object}}(p).$$

训练集最好同时覆盖：静态镜头中对象运动、相机运动、二者组合，以及真正静态画面。

24.10 dense caption 的结构

建议生成结构化 JSON：

```
{
  "short": "A cyclist turns left at an intersection.",
  "subjects": ["adult cyclist", "red bicycle"],
  "scene": "urban intersection in daylight",
  "actions": [
    {"time": "0-2s", "event": "approaches the intersection"},
    {"time": "2-5s", "event": "leans and turns left"}
```

```
  ],
  "camera": "stationary medium-wide shot",
  "lighting": "natural daylight",
  "style": "realistic documentary",
  "uncertainty": ["cyclist gender uncertain"]
}
```

保留不确定性比强行 hallucinate 更好。MLLM caption 应基于多帧/clip，而非只看首帧。

24.11 caption 验证

自动 caption 的常见错误：

- 主体数错；
- 左右/前后关系错；
- 把相机运动当对象运动；
- 把意图当已发生事件；
- 根据常识补充画面不存在属性；
- 忽略动作变化。

可用独立 verifier：

$$s_{\text{align}} = \frac{1}{K}\sum_{k=1}^{K} \text{VQA}(\mathbf{x}, q_k(\mathbf{y})).$$

24.12 去重的多阶段策略

文件级

SHA256 去完全重复。

帧级

对关键帧 perceptual hash，识别重新编码/缩放。

clip embedding

用 ViCLIP/VideoMAE/自监督 encoder 得到 e_i，ANN 检索：

$$\text{sim}(e_i, e_j) > \delta.$$

时间局部匹配

对长视频片段使用 sliding embedding sequence + dynamic time warping，识别裁剪、变速或插帧副本。

文本/音频

ASR n-gram、音频 fingerprint 和 OCR 可辅助判断新闻搬运、影视剪辑和教程重复。

去重不一定只保留一条；可保留最高质量版本，并保存 duplicate group 以进行 split。

24.13 数据安全与许可不是附录问题

训练 manifest 应包含来源、许可证/使用依据、年龄/身份风险、是否含可识别个人、删除请求映射。至少支持：

- 按 source 删除所有派生 clips；
- 按 person/asset hash 审计；
- 追踪 caption 与 embedding 的派生关系；
- 评测 memorization 和近复制；
- 记录自动/人工过滤版本。

数据治理会影响能否公开权重、能否商业部署和能否回答研究伦理审查。

24.14 数据缓存的三种层级

层级	存储	优点	缺点
原视频	压缩 mp4	灵活改变 fps/crop	解码慢、随机访问差
预采样帧/clip	JPEG/WebDataset/Arrow	训练读取快	存储大、压缩伪影
VAE latent + text embedding	tensor/shard	大幅减少在线计算	固定 VAE/crop/caption，难增强

若 VAE 冻结且课程固定，latent cache 很有价值。每条 latent 大小约

$$M_z = 2C_zT_zH_zW_z \quad \text{bytes (BF16)}.$$

例如 Wan2.1 81f 832×480:

$$M_z = 2 \times 16 \times 21 \times 60 \times 104 \approx 4.2 \text{ MB/sample}.$$

千万样本约 42 TB，仍需分层存储和压缩。若缓存 patch 后 token，则模型 patch embedding 无法修改。

24.15 多任务采样

设任务 $q \in \{\text{T2I}, \text{T2V}, \text{I2V}, \text{V2V}, ...\}$。简单按样本数采样会让最大数据源支配训练。常用 temperature sampling:

$$\pi(q) = \frac{n_q^\alpha}{\sum_j n_j^\alpha}, \qquad 0 \le \alpha \le 1.$$

$\alpha = 1$ 按数据量，$\alpha = 0$ 各任务均匀。还可按 token 成本修正:

$$\pi_{\text{cost}}(q) \propto \frac{n_q^\alpha}{\mathbb{E}[N_q]^\beta},$$

防止长视频任务吞噬全部 GPU 时间。

24.16 数据版本与可重复性

每次数据构建应输出:

- manifest hash;
- 原始来源版本;
- 下载成功率;
- shot/clip 数和总小时;
- 分辨率/fps/时长直方图;
- filter 前后数量;
- duplicate group 数;
- caption 模型与 prompt 版本;
- safety/许可统计;
- train/val/test parent-level split;
- VAE latent cache 版本。

模型 checkpoint 必须记录 dataset manifest hash，否则“同一配置复现”没有意义。

24.17 一份训练前数据审计表

检查	通过标准示例
解码	随机 10k 样本 100% 可读，无时间戳倒退
形状	所有 bucket 满足 VAE/patch 整除
边界	clip 内无硬切或黑帧突变

检查	通过标准示例
caption	人工抽检主体/动作/镜头准确率
动态	静态、对象运动、相机运动比例可见
去重	train-val 无高相似 parent/clip
安全	高风险样本有过滤与审计记录
吞吐	loader 能持续供给，不使 GPU idle
cache	latent 可由 VAE version/hash 追踪
删除	能从 source id 删除全部派生产物

核心结论

高质量视频数据不是“下载后跑一次 caption”。它是一个可版本化的编译流程：原始资产经过解码、镜头切分、几何规范、质量/动态筛选、caption、验证、去重、合规和课程采样，最终才成为训练分布。

25. 规模化训练系统：显存、并行、吞吐与成本模型

视频模型的系统设计不是论文实现细节。对 full-attention Video DiT，算法是否可训练往往由 token 几何、activation、通信和数据供给共同决定。

图 12: 规模化视频训练系统栈

25.1 显存的完整账本

训练峰值显存可拆为

$$M_{\text{peak}} = M_{\text{param}} + M_{\text{grad}} + M_{\text{opt}} + M_{\text{act}} + M_{\text{comm}} + M_{\text{workspace}} + M_{\text{fragment}}.$$

对于标准 mixed-precision AdamW，未分片时常见每参数字节：

项	bytes/param（代表值）
BF16 参数	2
BF16/FP16 梯度	2
FP32 master weight	4

项	bytes/param（代表值）
FP32 first moment	4
FP32 second moment	4
合计	16

某些实现没有 FP32 master 或使用 8-bit optimizer，可能降至约 10–12 bytes。14B 参数仅模型状态即可约

$$14 \times 10^9 \times 16 \approx 224 \text{ GB}.$$

activation 另计，因此必须分片。

25.2 activation 为什么更难

每层至少保存若干 $B \times N \times d$ 张量。粗略:

$$M_{\text{act}} \propto BLNdb_a,$$

b_a 是每元素字节和保存因子。对 $N = 75{,}600, d = 5120, L = 40$，即使 batch 1，单个 $N \times d$ BF16 张量也约

$$75{,}600 \times 5120 \times 2 \approx 774 \text{ MB}.$$

一个 block 中有多份 Q/K/V、attention output、FFN 中间和 residual。FlashAttention 不显式保存 $N \times N$ attention matrix，但线性 activation 仍巨大。

25.3 activation checkpointing

只保存每组 block 的输入，反向时重算内部 forward。若每 c 层一个 checkpoint，存储可近似降为

$$O\left(\frac{L}{c}Nd + cNd\right),$$

计算增加约 20%–40%，具体取决于重算范围。对视频大模型，额外 FLOPs 通常比 OOM 更可接受。

应避免把文本 encoder、冻结 VAE 也无差别 checkpoint；冻结模块可 `no_grad`，无需保存反向图。

25.4 ZeRO/FSDP 分片

ZeRO-1

分片 optimizer states:

$$M_{\text{opt}}/G.$$

ZeRO-2

再分片 gradients:

$$(M_{\text{opt}} + M_{\text{grad}})/G.$$

ZeRO-3 / FSDP full shard

参数也分片:

$$(M_{\text{param}} + M_{\text{grad}} + M_{\text{opt}})/G,$$

但每层 forward 前需 all-gather 参数，反向后 reduce-scatter 梯度。通信与预取策略决定吞吐。

对 A14B 两时间专家，可以只 all-gather 当前专家；若高/低专家跨阶段切换，预取下一专家并 overlap 传输可能降低停顿。

25.5 Data Parallel 不解决单样本 activation

若一个 720p sample 本身无法放入单卡，增加纯 DP GPU 没用，因为每卡仍复制完整模型并处理完整 sample。需要：

- FSDP 降模型状态；
- sequence/context parallel 切 token；
- tensor parallel 切 hidden/head；
- pipeline parallel 切 layers；
- activation offload/checkpoint。

25.6 Tensor Parallel

将线性层按 hidden dimension 分片。以 QKV 为例：

$$W_Q = [W_Q^{(1)}, \dots, W_Q^{(p)}],$$

每卡计算部分 heads。优点是每卡参数和 activation channel 减少；缺点是每层需要 all-reduce/all-gather，且长序列使通信张量大。

Video DiT 中 head dimension 通常固定 128，因此 tensor parallel 度最好整除 heads。Wan14B 有 40 heads，常见 $p \in \{2, 4, 5, 8, 10\}$；实际还受拓扑影响。

25.7 Sequence/Context Parallel

将 N token 分到 p_s 卡。对 MLP/LayerNorm 很直接；self-attention 需要跨卡交换 K/V 或 heads。

Ulysses

通过 all-to-all 在 sequence 分片与 head 分片之间转置。适合 head 数足够多、互联快的场景。

Ring Attention

每卡持有一段 query，K/V block 沿环传递，在线累积 softmax。通信可以与 attention 计算 overlap，适合极长序列。

Context Parallel in framework

现代框架可能提供统一 API，但必须验证 padding mask、3D RoPE 坐标和 packed sample boundaries 在分片后正确。

25.8 Pipeline Parallel

将 L 层分到 p_p 个 stage。若 microbatch 数为 m，朴素 pipeline bubble 比例近似

$$\beta_{\text{bubble}} \approx \frac{p_p - 1}{m + p_p - 1}.$$

视频训练 global batch 常受 token 成本限制，microbatch 很少，pipeline bubble 可能高。可与 sequence parallel/FSDP 组合，但系统复杂度显著上升。

25.9 2D/3D 并行网格

一种大模型配置：

$$G = p_d \times p_t \times p_s \times p_p,$$

分别为 data、tensor、sequence、pipeline parallel。选择原则：

- 先满足单 sample/model 可放下；
- 同节点 NVLink 内放高频通信维度；
- 跨节点尽量 data parallel 或低频 pipeline；
- 保证 heads、layers、sequence 可整除；
- 用实测而非理论带宽决定。

25.10 FlashAttention 与注意力内存

标准 attention 显式 materialize

$$A = \mathrm{softmax}(QK^\top) \in \mathbb{R}^{N\times N},$$

内存 $O(N^2)$。FlashAttention 分块在线计算 softmax，内存近似 $O(Nd)$，但 FLOPs 仍为 $O(N^2d)$。

因此它解决“放不下 attention matrix”，不解决 720p full attention 的计算爆炸。要降低 FLOPs，需要 window/sparse attention、更高压缩或减少 NFE。

25.11 Variable-size packing

将多个样本 token 拼成

$$\mathbf{H} = \mathrm{Concat}(\mathbf{H}_1, \dots, \mathbf{H}_B),$$

并使用 block-diagonal mask:

$$M_{ij} = 0 \quad \text{仅当 token } i, j \text{来自同一 sample.}$$

必须同步拼接:

- 3D RoPE 坐标;
- time/noise embedding;
- text context offset;
- latent loss mask;
- task id;
- source/target segment id。

一个 offset 错误可能让不同视频互相 attention，loss 仍会下降但模型学习污染。

25.12 Data loader 吞吐

GPU 空闲率:

$$r_{\mathrm{idle}} = 1 - \frac{t_{\mathrm{compute}}}{t_{\mathrm{step}}}.$$

视频 loader 可被以下瓶颈限制:

- 网络存储随机读;
- H.264/H.265 CPU decode;
- clip seek;
- resize/crop;
- caption JSON 解析;
- VAE 在线编码。

常用优化：WebDataset/tar shard、顺序读、GPU decode、prefetch、多级 cache、offline latent、固定 bucket batch 和异步错误跳过。但错误跳过必须全 rank 同步，否则 distributed deadlock。

25.13 混合精度

BF16 指数范围与 FP32 相同，通常比 FP16 稳定。常见策略:

- 参数/activation BF16;
- LayerNorm/RMSNorm、time embedding、loss accumulation FP32;
- optimizer moments FP32 或 8-bit;
- attention softmax 内部 FP32 accumulation;
- VAE decoder 某些插值/卷积临时 FP32。

FP8 可用于线性层，但速度场在不同 τ 的尺度差异大，需要 per-tensor/per-channel scaling 和噪声阶段校准。不能只在一个 prompt 上测最终视频。

25.14 数值稳定监控

每 step 至少记录：

- loss 与按 τ bucket loss；
- gradient norm；
- parameter/update norm；
- pred/target velocity RMS；
- latent mean/std；
- attention logit/entropy 抽样；
- skipped step/overflow；
- 每 bucket tokens/s；
- max GPU memory；
- all-reduce/all-to-all 时间。

按时间 bucket：

$$\mathcal{L}_b = \mathbb{E}[\mathcal{L} \mid \tau \in I_b].$$

总体 loss 正常但某噪声区间爆炸，可能预示最终采样某阶段失败。

25.15 梯度裁剪与学习率

全局 norm clipping：

$$\mathbf{g} \leftarrow \mathbf{g} \cdot \min\left(1, \frac{c}{\|\mathbf{g}\|_2 + \epsilon}\right).$$

在 FSDP 下必须计算全局分片梯度 norm，而非每卡独立裁剪。大模型学习率常随 global token batch 调整，但视频的 sample token 差异大，更合理记录每 update 的总有效 token：

$$B_{\text{token}} = \sum_i N_i.$$

按 sample 数线性 scaling 可能在高分辨率阶段突然把有效 batch 放大数倍。

25.16 EMA

EMA 参数：

$$\theta_{\text{EMA}} \leftarrow \beta\theta_{\text{EMA}} + (1-\beta)\theta.$$

$\beta = 0.999$ 在不同 update 频率下时间常数不同。有效窗口约

$$\frac{1}{1-\beta} = 1000 \text{ updates}.$$

若 gradient accumulation 或 global batch 改变，应按 seen tokens 而非 steps 理解 EMA。

25.17 Checkpoint 与容错

大型训练 checkpoint 应包含：

- model/EMA；
- optimizer/scheduler；
- scaler；
- global step 与 seen tokens；
- RNG states（Python/NumPy/PyTorch/CUDA）；
- dataloader shard cursor；
- dataset manifest hash；
- parallel topology；

- code commit/config;
- loss-sampling state。

仅恢复权重但重新开始数据顺序和 LR，会产生不可见的训练分叉。保存频率要在写盘开销与故障损失间权衡。

25.18 理论 FLOPs 与实测 MFU

模型 FLOPs 利用率：

$$\mathrm{MFU} = \frac{F_{\mathrm{model}}/t_{\mathrm{step}}}{G \cdot F_{\mathrm{peak}}}.$$

视频模型的 F_{model} 估计常漏掉 padding、recompute、CFG-like training branches 或 VAE。另一个更可操作指标是有效 token throughput：

$$\mathrm{throughput} = \frac{\sum_i N_i}{t_{\mathrm{step}} G}.$$

同时报告二者：MFU 便于硬件效率，tokens/s 便于工作量比较。

25.19 GPU-hours 与美元成本

$$\mathrm{GPUh} = G \times t_{\mathrm{wall,h}}.$$

若按云价 c_{GPUh}：

$$C_{\mathrm{compute}} = \mathrm{GPUh} \times c_{\mathrm{GPUh}}.$$

但总成本还包括：

$$C_{\mathrm{total}} = C_{\mathrm{compute}} + C_{\mathrm{storage}} + C_{\mathrm{egress}} + C_{\mathrm{caption}} + C_{\mathrm{labor}} + C_{\mathrm{failed}}.$$

论文常只报告成功主训练的 GPU-hours。实验室做预算时应加入 1.3–3 倍探索/失败系数。

25.20 一个透明成本示例

假设某 5B Video DiT：

- 单 sample 单 forward 0.48 PFLOP;
- batch/global 128;
- 训练 forward+backward 取 3 倍;
- 100k updates;
- 每 GPU 有效 300 TFLOP/s。

总 FLOPs：

$$F = 0.48 \times 10^{15} \times 128 \times 3 \times 10^{5} \approx 1.84 \times 10^{22}.$$

GPU-hours：

$$\frac{1.84 \times 10^{22}}{300 \times 10^{12} \times 3600} \approx 17{,}067 \text{ GPUh}.$$

若 128 GPU，理想墙钟约 5.6 天。现实还要乘 padding、通信、重算、数据 idle 和 checkpoint 等修正。该例只展示方法，不代表 Wan 官方训练。

25.21 优化优先级

通常按数量级收益排序：

1. 降低视频 token N；
2. 降低 NFE/训练不必要分支；
3. 合理 attention pattern；
4. packing 和负载均衡；
5. FSDP + sequence/context parallel；
6. activation checkpoint/offload；
7. FlashAttention/fused kernels；
8. 小的 Python 与 I/O 微优化。

常见误区

FlashAttention 让 N^2 attention 的内存可控，但不会把 N^2 FLOPs 变成线性。若 720p 序列的理论单步成本已经过高，继续换更快 kernel 只能获得常数倍；VAE 压缩、稀疏注意力或分阶段生成才可能改变数量级。

26. 推理与部署：采样器、CFG、蒸馏、量化和服务系统

训练得到的是一个速度场/去噪网络；真正产品性能由采样轨迹、条件分支、VAE 解码、offload、并行和视频编码共同决定。

26.1 一个标准 Flow 推理循环

给定时间网格

$$1 = \tau_0 > \tau_1 > \cdots > \tau_K = 0$$

（方向按实现约定），Euler 更新：

$$\mathbf{z}_{k+1} = \mathbf{z}_k + (\tau_{k+1} - \tau_k) v_\theta(\mathbf{z}_k, \tau_k, \mathbf{c}).$$

Heun 二阶方法先预测：

$$\tilde{\mathbf{z}}_{k+1} = \mathbf{z}_k + \Delta\tau_k v_k,$$

再校正：

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \frac{\Delta\tau_k}{2}\left[v_k + v_\theta(\tilde{\mathbf{z}}_{k+1}, \tau_{k+1}, \mathbf{c})\right].$$

二阶每步通常需要两次 NFE；比较“20 steps”时必须报告 NFE，而不只报告 solver steps。

26.2 时间网格比 solver 名称更重要

局部误差取决于轨迹曲率和步长。可按速度变化自适应：

$$e_k = \|v(\tilde{\mathbf{z}}_{k+1}, \tau_{k+1}) - v(\mathbf{z}_k, \tau_k)\|,$$

当 e_k 大时减小步长。视频大模型通常使用固定步数以便批处理，但 prompt-adaptive NFE 是有价值方向。

研究中至少绘制：

- τ_k；
- $\Delta\tau_k$；
- 每步 velocity RMS；
- 每步 conditional-unconditional 差；
- 中间 latent 解码缩略图。

这能定位布局在哪一步形成、细节何时出现、哪一段需要更多 NFE。

26.3 CFG 的计算与 batch 实现

标准 CFG 需要无条件与条件两次预测。可以拼 batch：

$$\mathbf{Z} = [\mathbf{z}; \mathbf{z}], \qquad \mathbf{C} = [\emptyset; \mathbf{c}],$$

一次模型调用得到 $[v_u; v_c]$。这样减少 Python/kernel launch，但 activation batch 翻倍，峰值显存上升。低显存模式可顺序执行两分支，显存低、时间更长。

CFG rescale 可缓解过饱和：

$$\tilde{v}_{\text{cfg}} = v_u + s(v_c - v_u),$$

$$v' = \lambda \frac{\text{std}(v_c)}{\text{std}(\tilde{v}_{\text{cfg}})} \tilde{v}_{\text{cfg}} + (1-\lambda)\tilde{v}_{\text{cfg}}.$$

$$s(\tau) = s_{\min} + (s_{\max} - s_{\min}) g(\tau).$$

高噪声强化文本可锁定语义，低噪声降低 guidance 可保留自然细节和运动。

26.4 多条件 guidance 的组合爆炸

有文本、图像、源视频、姿态和 planner 时，朴素枚举所有条件组合需要 2^M 分支。实际采用：

- 只保留无条件 + 全条件；
- incremental guidance；
- 将部分条件融合为同一 encoder；
- 训练单分支 guided student；
- 对强条件不用 CFG，只对文本做 CFG。

应报告实际 `branch-equivalent NFE`：

$$K_{\text{eq}} = K \times n_{\text{branch}}.$$

一个 20-step、4 分支系统的主干计算接近 80 单分支 NFE。

26.5 Prompt extension 与 negative prompt

Prompt extension 可以补充：动作阶段、镜头、光照、材质和风格。但它可能改变用户意图。最好保存：

- 原 prompt；
- rewrite prompt；
- rewrite 模型与版本；
- temperature/seed；
- negative prompt。

Wan 官方配置包含较长默认 negative prompt，涉及过曝、静态、模糊、字幕、低质量和人体畸形。实验若一组使用默认 negative prompt、另一组不用，会产生不公平比较。

26.6 VAE decode 的峰值与 tile/chunk

高分辨率 decoder activation 可能超过 DiT 单步。空间 tiling 将 latent 分成重叠 tile，解码后加权融合：

$$\hat{\mathbf{x}}(p) = \frac{\sum_k w_k(p)\hat{\mathbf{x}}^{(k)}(p)}{\sum_k w_k(p)}.$$

需要 overlap 以覆盖卷积 receptive field。时间 chunk 对 causal VAE 使用 cache；非因果 VAE 需双向 overlap。tile 太小会出现接缝、颜色差异和运动不连续。

26.7 CPU offload

offload 可分：

- model-level：T5、DiT、VAE 轮流驻 GPU；
- block-level：逐层搬运；
- expert-level：Wan2.2 高/低专家切换；
- activation offload：中间张量存 CPU；
- sequential offload：极省显存但 PCIe 瓶颈大。

传输时间下界：

$$t_{\text{transfer}} \geq \frac{M}{B_{\text{PCIe}}}.$$

28 GB BF16 权重在实际 25 GB/s PCIe 上单向至少约 1.1 秒，若每步搬运就不可接受。应让权重在多个 NFE 期间驻留，或按专家阶段只搬一次。

26.8 多 GPU 推理

Tensor parallel

每步低延迟，但每层通信频繁。

Sequence/context parallel

对高分辨率长序列有效；attention all-to-all 依赖高速互联。

Pipeline parallel

单视频 latency 受 stage 串行限制，适合多请求吞吐。

Request parallel

每 GPU 独立生成不同 seed/prompt，是最简单的吞吐扩展；不降低单请求显存。

服务应区分 latency target 与 throughput target。批量离线生成可用较大 batch；交互产品更重视首帧/首预览时间和取消请求能力。

26.9 KV cache 是否适用于扩散 DiT

LLM 自回归中历史 token 不变，因此 K/V 可缓存。视频扩散每个 NFE 的所有 latent token 都变化，self-attention K/V 无法跨步直接复用。可缓存的部分包括：

- 文本 cross-attention K/V；
- 静态图像/参考条件 K/V；
- 某些低变化层/低频特征的近似缓存；
- CFG 无条件文本特征；
- RoPE frequencies。

近似 feature cache 必须测跨噪声时间误差。高噪声步之间变化大，低噪声可能更适合缓存，或相反取决于层。

26.10 采样蒸馏

目标是减少 K 或分支数。

Teacher trajectory regression

$$\mathcal{L} = \| f_{\theta_S}(\mathbf{z}_{\tau_a}, \tau_a, \tau_b, \mathbf{c}) - \tilde{\mathbf{z}}^{(T)}_{\tau_b} \|^2.$$

student 学习跨越多个 teacher steps。

Consistency/trajectory consistency

不同时间的预测经 solver 映射到同一数据端点：

$$f_\theta(\mathbf{z}_{\tau_a}, \tau_a) \approx f_\theta(\mathbf{z}_{\tau_b}, \tau_b).$$

Adversarial/perceptual distillation

少步 MSE 容易过平滑，可加入 VAE decode 后的感知、判别器或视频特征损失。但成本和稳定性提高。

蒸馏评测必须包含 diversity。student 若只复现 teacher 的平均轨迹，单样本质量高但随机种子多样性下降。

26.11 量化

权重量化

INT8/FP8/INT4 减少权重显存：

$$\hat{W} = s \cdot \mathrm{clip}(\mathrm{round}(W/s), q_{\min}, q_{\max}).$$

per-channel scale 通常优于 per-tensor。敏感层包括：

- 输入/输出 projection；
- time modulation；
- Q/K projection；
- VAE decoder 最后层；
- planner-renderer interface。

Activation 量化

更困难，因为不同 τ、分辨率和 prompt 的范围变化。应按噪声 bucket 校准 scale：

$$s_b = \mathrm{Quantile}_p(|a| \mid \tau \in I_b)/q_{\max}.$$

混合精度量化

保留敏感层 BF16，其余 FP8/INT8。评测应按 noise stage 做误差曲线：

$$e_b = \mathbb{E}_{\tau \in I_b} \| v_{\mathrm{quant}} - v_{\mathrm{bf16}} \|_2.$$

26.12 编译与内核

`torch.compile`、CUDA graph 和 fused kernels 可减少 launch overhead。但视频形状变化多，会触发 graph recompile。服务可固定有限 bucket：

$$(T, H, W) \in \mathcal{B}_{\mathrm{serve}},$$

为每个 bucket 预编译。动态文本长度可 pad 到固定 512；packed batch 与多条件分支需要独立 graph。

CUDA graph 要求地址稳定，因此预分配 latent、text buffer 和 scheduler tensor。与 CPU offload 混用时更复杂。

26.13 服务级调度

请求成本预测：

$$\hat{C} = aN + bN^2 + cK_{\mathrm{eq}} + dM_{\mathrm{decode}}.$$

调度器按预计成本而非请求数 batch，避免一个 720p 长视频拖慢多个短请求。可采用：

- 相同 shape/NFE 分桶；
- deadline-aware batching；

- preview 低分辨率先返回；
- 动态 NFE；
- 早停/取消；
- VAE decode 独立队列；
- 编码写盘异步。

26.14 延迟报告模板

阶段	时间	峰值显存	备注
权重加载/cold start			是否 mmap/offload
prompt rewrite			模型/API
text/MLLM planner			plan iterations
DiT sampling			NFE、branches、并行
VAE decode			tile/chunk
postprocess/encode			fps、codec
end-to-end			warm/cold、batch

同时报告生成视频秒数：

$$\mathrm{RTF} = \frac{t_{\mathrm{wall}}}{T / f_{\mathrm{out}}}.$$

RTF < 1 表示 faster-than-real-time。不要把 5 秒视频的模型采样时间与包含加载/写盘的端到端时间混为一谈。

26.15 质量-成本 Pareto

每个运行点

$$\mathbf{p} = (Q_{\mathrm{semantic}}, Q_{\mathrm{visual}}, Q_{\mathrm{motion}}, t, M, E),$$

包含质量、延迟、显存和能耗。点 a 若在所有成本不高、质量不低且至少一项更优，则支配点 b。论文应报告 Pareto frontier，而不是只挑一个“最佳”设置。

26.16 推理复现检查

- 固定 checkpoint/commit/hash；
- 固定 VAE、text encoder、planner；
- 固定 prompt rewrite 与 negative prompt；
- 固定 seed 与 RNG device；
- 固定 scheduler、time shift、NFE、solver；
- 固定 CFG 与多条件分支顺序；
- 固定 latent shape、fps、decode tile；
- 明确 quantization/offload/compile；
- warm-up 后计时；
- 保存 raw frame tensor 与最终 mp4。

核心结论

视频推理的“步数”不是成本单位。真正可比较的是主干 NFE、条件分支数、token 数、VAE decode 和系统配置。20 步四分支可能比 50 步单分支更贵；4 步蒸馏模型也可能因大 planner 和 decoder 失去端到端优势。

27. 从零到研究：四个递进实验与论文阅读路线

本章给出一个可以实际执行的学习计划。目标不是一次训练 14B，而是在可控成本下建立正确的研究闭环。

27.1 实验 0：只做形状与成本审计

目标：不运行完整生成，先能从配置推导所有张量。

步骤：

1. 读取 Wan2.1 1.3B/14B 和 Wan2.2 5B config；
2. 输入 (T, H, W)；
3. 计算 (T_z, H_z, W_z)；
4. 计算 patch grid 与 N；
5. 估计 BF16 权重、latent cache、主要 FLOPs；
6. 绘制 N、N^2 随分辨率/帧数曲线；
7. 验证实际 hook 输出。

验收：理论与代码 shape 完全一致；误差必须解释 padding 或因果首帧。

27.2 实验 1：VAE oracle benchmark

研究问题：Wan2.1 8×8 与 Wan2.2 16×16 VAE 在哪些内容上损失最大?

数据：200–1000 段视频，分为：

- 人脸近景；
- 手-物交互；
- 屏幕文字/OCR；
- 水、火、烟、毛发；
- 快速小物体；
- 相机快速运动；
- 动画线稿。

协议：

$$\hat{\mathbf{x}}^{(m)} = D^{(m)}(E^{(m)}(\mathbf{x})).$$

报告 LPIPS、PSNR、temporal residual、OCR、identity、flow error，并做人评。再用相同生成 DiT 不可行，因为 VAE latent 空间不同；该实验只比较 representation upper bound。

可形成的研究：内容自适应压缩、motion-aware VAE loss、OCR-aware decoder、混合 stride。

27.3 实验 2：Wan1.3B 的 motion LoRA

问题：self-attention LoRA 是否比 cross-attention LoRA 更能学习新动作?

实验组：

1. cross-attention only；
2. self-attention only；
3. self + cross；
4. FFN only；
5. full LoRA target。

冻结：数据、rank、总 trainable parameters（尽量匹配）、steps、LR、NFE、CFG。

评测：

- action recognition/VLM action QA；
- optical-flow magnitude；
- identity consistency；
- prompt alignment；
- base prompt regression；
- 人评动作正确性。

核心不是证明某组总体最好，而是建立模块-能力因果映射。

27.4 实验 3：噪声阶段自适应专家/adapter

从 Wan2.2 时间专家获得灵感，但在 1.3B 上做低成本验证。设两个 LoRA：

$$\Delta W(\tau) = \lambda(\tau)\Delta W_{\text{high}} + [1 - \lambda(\tau)]\Delta W_{\text{low}}.$$

比较：共享 LoRA、硬边界、sigmoid soft gate、learned gate。分析：

- high-noise adapter 是否影响布局/动作；
- low-noise adapter 是否影响纹理/身份；
- boundary sensitivity；
- 两 adapter 的梯度 cosine similarity：

$$\cos(g_h, g_l) = \frac{g_h^\top g_l}{\|g_h\|\|g_l\|}.$$

负 cosine 支持阶段冲突假设。

27.5 实验 4：轻量 semantic planner

不直接复现完整 Bernini，可先构造结构化 planner：

```
{
  "entities": [...],
  "timeline": [
    {"t": 0.0, "state": ...},
    {"t": 0.5, "state": ...},
    {"t": 1.0, "state": ...}
  ],
  "camera": [...],
  "constraints": [...]
}
```

将其编码为额外 text/learned tokens，注入 Wan cross-attention。比较原 prompt、rewrite、structured plan、oracle human plan。

进一步可训练一个小 ViT semantic plan adapter：真实目标视频经冻结 ViT 得到 $\mathbf{s}_\star$，用小 Transformer/flow decoder 预测，再条件化 DiT。这样可验证 Bernini 的核心接口，而无需同时训练 7B MLLM + 14B renderer。

27.6 六周学习路线

第 1 周：表示与形状

- 复习上册 Video VAE、DiT、Flow Matching；
- 跑形状计算器；
- 完成 VAE oracle；
- 阅读 VideoLDM、CogVideoX、Wan VAE 章节。

第 2 周：官方推理复现

- 固定 Wan2.1-1.3B 环境；
- 复现 480p 小样例；
- 测 NFE/CFG/time shift；
- 保存 raw tensors 与 latency 分解。

第 3 周：小数据训练

- 32 样本 overfit；
- 实现 LoRA target ablation；
- 建立固定 prompt × seed 网格；
- 记录 shape、loss bucket、VRAM、tokens/s。

第 4 周：数据与评测

- 构造 1k–10k clips；
- 运行 shot/caption/dedup audit；
- 加入 VBench-like 分维指标和人评表；
- 做 bootstrap CI。

第 5 周：系统与效率

- activation checkpoint；
- FSDP/sequence parallel；
- packing；
- profiler 分解 attention/FFN/communication/VAE。

第 6 周：研究原型

- 选择阶段 adapter 或 semantic planner；
- 完成主实验、消融、成本-Pareto；
- 写 4 页 workshop-style report。

27.7 论文阅读模板

每篇论文用一页回答：

1. 任务与输出几何是什么？
2. $\mathcal{R}$：VAE/tokenizer stride、channel、oracle 如何？
3. $\mathcal{B}$：主干、attention、位置编码、参数量？
4. $\mathcal{O}$：路径、预测目标、时间采样、loss weighting？
5. $\mathcal{C}$：条件 encoder 与注入位置？
6. $\mathcal{K}$：数据、课程、任务混合、后训练？
7. $\mathcal{S}$：并行、内核、显存、NFE、延迟？
8. 最关键消融是否隔离了变量？
9. 哪些事实 A/B/C/ND？
10. 复现最小闭环是什么？

27.8 从想法到论文的证据链

一个可靠研究结论应形成：

$$\text{Hypothesis} \to \text{Mechanism} \to \text{Controlled Experiment} \to \text{Diagnostic Metric} \to \text{Ablation} \to \text{Cross-base Validation}.$$

例如“高噪声阶段更需要全局 attention”：

- 假设：布局/粗运动在高噪声建立；
- 机制：高噪声 full attention、低噪声 window attention；
- 控制：相同参数/数据/NFE；
- 诊断：布局关系、轨迹、细节、延迟；
- 消融：反向配置、全 full、全 window、boundary；
- 跨底座：1.3B 与 5B。

27.9 常见失败与快速定位

现象	优先检查
视频全黑/色偏	VAE scaling、像素范围、decoder dtype
loss 正常但采样噪声	端点方向、target velocity、scheduler
主体正确但几乎静止	图像比例、motion distribution、CFG 过强
运动大但身份漂移	VAE/attention、图像条件强度、长程一致性
prompt 被忽略	text embedding、condition dropout、cross-attn LoRA
I2V 只复制首帧	图像条件过强、训练 motion 数据不足

现象	优先检查
LoRA 过拟合所有 prompt	prior preservation、caption 多样性、rank/steps
多卡结果与单卡不同	RoPE/packing mask、collective 顺序、随机数
速度没有提升	token/NFE 未变、通信/数据成为瓶颈
指标升但人评降	judge bias、动态/质量 trade-off、评测泄漏

27.10 最小研究产物目录

```
project/
  configs/
    model.yaml
    train.yaml
    inference.yaml
  manifests/
    train.jsonl
    val.jsonl
    DATA_CARD.md
  scripts/
    audit_shapes.py
    train.py
    sample.py
    evaluate.py
  checkpoints/
  samples/
    fixed_prompt_seed_grid/
  metrics/
    per_prompt.csv
    bootstrap_ci.json
  profiles/
    memory_trace.json
    nsys_report/
  reports/
    ablation.md
    failure_taxonomy.md
    reproducibility_card.md
  environment.lock
  git_commit.txt
```

核心结论

真正的“上手”不是成功生成一段视频，而是能解释每个张量、每个成本项和每个指标；能让另一个研究者在固定配置下复现结果；能用消融把收益归因到明确机制。

附录 A：形状、显存与计算速查

A.1 因果 VAE 与 DiT token

输入 (T, H, W)，VAE stride (s_t, s_h, s_w)，patch (p_t, p_h, p_w)：

$$T_z = 1 + \left\lfloor \frac{T-1}{s_t} \right\rfloor, \quad H_z = \left\lfloor \frac{H}{s_h} \right\rfloor, \quad W_z = \left\lfloor \frac{W}{s_w} \right\rfloor,$$

$$T_p = \left\lceil \frac{T_z}{p_t} \right\rceil, \quad H_p = \left\lceil \frac{H_z}{p_h} \right\rceil, \quad W_p = \left\lceil \frac{W_z}{p_w} \right\rceil,$$

$$N = T_p H_p W_p.$$

若实现要求整除，应在进入模型前 assert，而非静默 floor。

A.2 Wan 代表形状

模型/输出	latent shape $C_z \times T_z \times H_z \times W_z$	patch grid	N
Wan2.1, 81f, 832×480	$16 \times 21 \times 60 \times 104$	$21 \times 30 \times 52$	32,760
Wan2.1, 81f, 1280×720	$16 \times 21 \times 90 \times 160$	$21 \times 45 \times 80$	75,600
Wan2.2-5B, 121f, 1280×704	$48 \times 31 \times 44 \times 80$	$31 \times 22 \times 40$	27,280

A.3 权重与训练状态

对 P 参数、参数字节 b_p:

$$M_{\text{weight}} = Pb_p.$$

代表估计:

参数量	BF16 weights	8-bit weights	4-bit weights（理想裸值）
1.3B	2.6 GB	1.3 GB	0.65 GB
5B	10 GB	5 GB	2.5 GB
14B	28 GB	14 GB	7 GB
30B	60 GB	30 GB	15 GB

实际量化还有 scale、zero point、padding、未量化层和 runtime buffer。

AdamW 未分片代表值:

$$M_{\text{state}} \approx P(2 + 2 + 4 + 4 + 4) = 16P \text{ bytes}.$$

A.4 Transformer 单步计算

$$C_{\text{fwd}} \approx L\left(8Nd^2 + 4Ndd_{\text{ff}} + 4N^2d\right).$$

若使用 window attention，每 token 只关注 w 个 token:

$$4N^2d \to 4Nwd.$$

若标准 CFG:

$$C_{\text{sample}} \approx K\, n_{\text{branch}} C_{\text{fwd}}, \qquad n_{\text{branch}} \approx 2.$$

A.5 latent cache

BF16 latent:

$$M_z = 2C_zT_zH_zW_z \text{ bytes/sample}.$$

总缓存:

$$M_{\text{cache}} = N_{\text{samples}} M_z (1 + r_{\text{index}}),$$

r_{index} 包含 metadata、shard 空洞和文件系统开销，常取 5%–20% 做预算。

A.6 GPU-hours

若单样本 forward 为 C_f，global batch B，训练倍率 r_{fb}，steps S，有效吞吐 U：

$$\mathrm{GPUh} = \frac{C_f B r_{fb} S}{3600 U}.$$

墙钟：

$$t_{\mathrm{wall}} = \frac{\mathrm{GPUh}}{G}.$$

必须明确 C_f 是否包含 padding、cross-attention、VAE 和 recompute。

附录 B：代表模型事实卡

B.1 Wan2.1

- 论文：*Wan: Open and Advanced Large-Scale Video Generative Models*，2025。
- 规模：1.3B、14B。
- 主干：Video DiT，full spatiotemporal self-attention，Flow Matching。
- VAE：causal 3D，stride $(4, 8, 8)$，16 latent channels。
- 条件：UMT5-XXL；I2V 还使用视觉条件。
- 位置编码：3D RoPE。
- 官方效率锚点：1.3B 约 8.19 GB VRAM；代表 480p 5 秒在 RTX 4090 约数分钟，具体依运行设置。
- 数据：十亿级图像与视频；完整来源和总 GPU-hours 未披露。
- 适合：微调、控制、VAE/DiT 分析、消费级实验。

B.2 Wan2.2

- A14B：高/低噪声两套时间专家，每步激活一套约 14B。
- TI2V-5B：统一 T2V/I2V；VAE stride $(4, 16, 16)$，48 channels；121 帧、24 fps 代表设置。
- 数据相对 Wan2.1 增长：图像约 65.6%、视频约 83.2%（相对值）。
- 适合：阶段专家、高压缩 VAE、24GB offload 推理。

B.3 CogVideoX

- 论文：2024。
- 输出锚点：10 秒、16 fps、768×1360（报告设置）。
- 结构：3D causal VAE、expert Transformer、expert adaptive LayerNorm。
- 训练：progressive、multi-resolution frame packing、专门 caption pipeline。
- 适合：文本-视频深融合、长视频与 packing。

B.4 HunyuanVideo

- 论文：2024。
- 参数：超过 13B。
- 结构：大规模 Video DiT、强文本 encoder、progressive training 和系统优化。
- 适合：大模型并行、双语/长 prompt、开放底座 scaling。

B.5 HunyuanVideo 1.5

- 论文：2025。
- 参数：8.3B。
- 结构：SSTA、glyph-aware bilingual encoding、progressive pre/post-training、VSR。
- 适合：稀疏注意力、消费级推理与文字生成。

B.6 LTX-Video

- 论文：2024/2025。
- VAE：1:192 标量压缩，代表几何 $8 \times 32 \times 32$ 像素/token。

- 结构：full-attention Transformer；decoder 承担最终像素去噪。
- 速度锚点：论文报告 H100 上 5 秒、24 fps、768×512 约 2 秒。
- 适合：极致压缩、VAE/denoiser 协同、实时生成。

B.7 Open-Sora 2.0

- 论文：2025。
- 规模：约 11B 级。
- 公开训练账单：约 99,840 H200 GPU-hours、约 20 万美元主训练预算。
- 适合：成本透明、课程与系统复现。

B.8 Step-Video-T2V

- 论文：2025。
- 参数：30B；最长 204 帧。
- VAE：时间 8、空间 16 × 16 压缩。
- 主干：full 3D attention DiT、Flow Matching；双语文本 encoder。
- 后训练：Video-DPO。
- 适合：超大 Video DiT、偏好优化与长视频。

B.9 Pyramid Flow

- 论文：2024。
- 方法：单 DiT、分辨率/时间金字塔上的统一 Flow Matching。
- 公开成本：约 20.7k A100 GPU-hours。
- 适合：改变生成路径以降低训练成本。

B.10 Bernini

- 论文：2026，*Latent Semantic Planning for Video Diffusion*。
- Planner：Qwen2.5-VL-7B 基底，masked iterative planning，连续 ViT-flow decoder，可含 CoT。
- Renderer：Wan2.2-A14B 基底，T5 + semantic plan + 源 VAE 低层条件。
- 位置编码：SA-3D RoPE。
- 数据：约 20M video pairs、近 30M image pairs、约 10M interleaved + 2M 视频交错样本、约 1M reasoning。
- 系统：FSDP、Ulysses、packing、offload、FlashAttention/FlexAttention、定制 kernel。
- 蒸馏：CFG distillation + ReFlow，论文报告 4 NFE student 接近 80 NFE teacher。
- 适合：MLLM planner、统一生成/编辑、多源条件和语义接口。

附录 C：代码与配置模板

C.1 模型配置 YAML

```yaml
model:
  family: wan2.1
  task: t2v
  checkpoint: /path/to/checkpoint
  checkpoint_sha256: null
  dtype: bfloat16
  vae:
    stride: [4, 8, 8]
    latent_channels: 16
    causal: true
    scaling: official
  dit:
    patch: [1, 2, 2]
    hidden_dim: 1536
    ffn_dim: 8960
    heads: 12
    layers: 30
    attention: full_3d
  text_encoder:
    name: umt5-xxl
```

```yaml
    max_length: 512

sample:
  frames: 81
  fps: 16
  height: 480
  width: 832
  solver: flow_euler
  steps: 50
  shift: 5.0
  cfg: 5.0
  seed: 42
  negative_prompt: null
  offload: false
```

C.2 训练日志 JSONL

```jsonl
{
  "step": 1000,
  "seen_samples": 128000,
  "seen_video_tokens": 4193280000,
  "loss": 0.123,
  "loss_tau_0_025": 0.104,
  "loss_tau_025_050": 0.118,
  "loss_tau_050_075": 0.132,
  "loss_tau_075_100": 0.141,
  "grad_norm": 0.87,
  "lr": 0.00001,
  "tokens_per_gpu_second": 18200,
  "step_time_sec": 4.7,
  "data_time_sec": 0.21,
  "peak_vram_gb": 77.4,
  "parallel": {"dp": 8, "sp": 8, "tp": 1, "pp": 1},
  "dataset_manifest": "sha256:..."
}
```

C.3 评测记录 CSV

```csv
model,commit,prompt_id,prompt,seed,frames,fps,height,width,
steps,branch_equiv_nfe,cfg,shift,latency_sec,peak_vram_gb,
clip_alignment,motion_score,temporal_score,aesthetic,identity,human_pref
```

每条视频保留逐样本指标，不能只保存总体平均。

附录 D：Reproducibility Card

D.1 模型

- ☐ 基础模型、任务版本、commit、checkpoint hash；
- ☐ VAE 类型、stride、channel、scaling、chunk/tile；
- ☐ 文本/图像/MLLM encoder 与最大长度；
- ☐ DiT patch、d、d_{ff}、heads、layers、attention；
- ☐ trainable/frozen modules；
- ☐ LoRA target/rank/alpha/dropout；
- ☐ 参数量和 trainable 参数量。

D.2 数据

- ☐ 来源与许可；
- ☐ manifest hash；
- ☐ 原始视频/小时/clip/token；
- ☐ shot/clip 切分；
- ☐ fps、分辨率、时长 bucket；
- ☐ caption 模型/prompt/version；

☐ quality/motion/safety filters;
☐ duplicate groups;
☐ parent-level train/val/test split;
☐ 删除与审计机制。

D.3 训练

☐ objective、端点方向、时间采样、loss weighting;
☐ optimizer、LR、warmup、weight decay、clip;
☐ batch samples 与 batch tokens;
☐ steps、seen tokens、EMA;
☐ precision 与 scaler;
☐ DP/TP/SP/PP/FSDP;
☐ checkpoint/restart;
☐ GPU 型号、数量、GPU-hours、失败实验口径;
☐ tokens/s、MFU、peak VRAM。

D.4 推理

☐ 原 prompt 与 rewrite;
☐ negative prompt;
☐ solver、time grid、shift、NFE;
☐ CFG 与 branch-equivalent NFE;
☐ seed/RNG;
☐ frames/fps/size;
☐ offload/quantization/compile;
☐ cold/warm latency;
☐ VAE decode 和编码是否计时。

D.5 评测

☐ prompt 集和版本;
☐ 每 prompt 多 seed;
☐ 语义、画质、动态、时间、身份与物理指标;
☐ 人评界面、随机化、盲法和评审数;
☐ bootstrap CI;
☐ 失败分类;
☐ VAE oracle;
☐ 数据泄漏检索;
☐ 质量-成本 Pareto。

附录 E：术语表

术语	中文解释
branch-equivalent NFE	将 CFG/多条件分支乘入后的等效主干前向次数
causal Video VAE	时间卷积只依赖当前和过去，可分块流式编码/解码
condition collapse	renderer 忽略新增条件，退化为原始底座行为
context/sequence parallel	沿 token 序列维分片计算
continuous semantic plan	连续视觉特征空间中的目标语义计划，如 ViT embeddings
data curriculum	随训练阶段改变分辨率、时长、任务和质量采样
expert AdaLN	不同模态/专家使用不同归一化调制参数
Flow Matching	回归将源分布运输到目标分布的速度场
full 3D attention	所有时空 token 互相注意，复杂度 $O(N^2d)$
guidance distillation	将多分支 CFG 教师蒸馏为单分支 student
latent cache	预先存储冻结 VAE 输出，减少在线编码
latent semantic planning	先在高层视觉语义空间规划，再由扩散模型渲染
mask ratio schedule	masked planning 中每轮/每任务隐藏目标 token 的比例
NFE	Number of Function Evaluations，主干网络评估次数

术语	中文解释
oracle plan	从真实目标视频提取的语义计划，用于测 renderer 上限
oracle reconstruction	$D(E(x))$，用于测 VAE 表示上限
packing	将不同长度样本拼入共享 token buffer，并用隔离 mask
prompt extension	用 LLM/规则将短 prompt 改写为更详细文本
Rectified Flow / ReFlow	通过重配对和再训练使运输轨迹更直、便于少步采样
reference leakage	参考图/视频内容错误复制到目标，而非按指令变换
SA-3D RoPE	同时编码 segment 与时空坐标的旋转位置编码
sample shift / time shift	非线性重映射采样/训练时间，改变噪声区间密度
scalar compression ratio	考虑输入/latent channel 后的元素数量压缩
SSTA	selective and sliding tile attention，局部 tile 加选择性全局连接
timestep expert	按噪声时间区间选择的完整专家网络
token geometry	由帧数、分辨率、VAE stride 和 patch 决定的 token 网格
Ulysses	通过 all-to-all 在 sequence 与 head 分片间转换的序列并行
VAE scaling	对 latent 按通道均值/方差归一化，使生成器输入匹配训练分布
Video-DPO	用视频偏好对和扩散代理目标进行偏好优化

参考文献与官方资料

以下以论文/技术报告为主；开源仓库用于核对配置与实现。年份按公开版本列出。

1. Ho et al. *Video Diffusion Models*. NeurIPS, 2022.
2. Ho et al. *Imagen Video: High Definition Video Generation with Diffusion Models*. 2022.
3. Singer et al. *Make-A-Video: Text-to-Video Generation without Text-Video Data*. ICLR, 2023.
4. Villegas et al. *Phenaki: Variable Length Video Generation from Open Domain Textual Descriptions*. ICLR, 2023.
5. Blattmann et al. *Align Your Latents: High-Resolution Video Synthesis with Latent Diffusion Models*. CVPR, 2023.
6. Wang et al. *ModelScope Text-to-Video Technical Report*. 2023.
7. Chen et al. *VideoCrafter1: Open Diffusion Models for High-Quality Video Generation*. 2023.
8. Chen et al. *VideoCrafter2: Overcoming Data Limitations for High-Quality Video Diffusion Models*. CVPR, 2024.
9. Guo et al. *AnimateDiff: Animate Your Personalized Text-to-Image Diffusion Models without Specific Tuning*. ICLR, 2024.
10. Peebles and Xie. *Scalable Diffusion Models with Transformers*. ICCV, 2023.
11. Lipman et al. *Flow Matching for Generative Modeling*. ICLR, 2023.
12. Liu et al. *Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow*. ICLR, 2023.
13. Yang et al. *CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer*. 2024. arXiv:2408.06072.
14. Kong et al. *HunyuanVideo: A Systematic Framework for Large Video Generative Models*. 2024. arXiv:2412.03603.
15. Wu et al. *HunyuanVideo 1.5 Technical Report*. 2025. arXiv:2511.18870.
16. HaCohen et al. *LTX-Video: Realtime Video Latent Diffusion*. 2024/2025. arXiv:2501.00103.
17. Peng et al. *Open-Sora 2.0: Training a Commercial-Level Video Generation Model in $200k*. 2025. arXiv:2503.09642.
18. Ma et al. *Step-Video-T2V Technical Report: The Practice, Challenges, and Future of Video Foundation Model*. 2025. arXiv:2502.10248.
19. Huang et al. *Step-Video-TI2V Technical Report*. 2025. arXiv:2503.11251.
20. Jin et al. *Pyramidal Flow Matching for Efficient Video Generative Modeling*. 2024. arXiv:2410.05954.
21. Lin et al. *STIV: Scalable Text and Image Conditioned Video Generation*. 2024. arXiv:2412.07730.
22. Wan Team. *Wan: Open and Advanced Large-Scale Video Generative Models*. 2025. arXiv:2503.20314.
23. Wan-Video. *Wan2.1 Official Repository and Configuration Files*. GitHub, accessed 2026-07.
24. Wan-Video. *Wan2.2 Official Repository and Configuration Files*. GitHub, accessed 2026-07.
25. Liu et al. *Bernini: Latent Semantic Planning for Video Diffusion*. 2026. arXiv:2605.22344.
26. Bain et al. *Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval*. ICCV, 2021.
27. Miech et al. *HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips*. ICCV, 2019.
28. Xue et al. *Advancing High-Resolution Video-Language Representation with Large-Scale Video Transcriptions*.

CVPR, 2022.
29. Wang et al. *InternVid: A Large-Scale Video-Text Dataset for Multimodal Understanding and Generation*. ICLR, 2024.
30. Chen et al. *Panda-70M: Captioning 70M Videos with Multiple Cross-Modality Teachers*. CVPR, 2024.
31. Nan et al. *OpenVid-1M: A Large-Scale High-Quality Dataset for Text-to-Video Generation*. 2024. arXiv:2407.02371.
32. Ju et al. *MiraData: A Large-Scale Video Dataset with Long Durations and Structured Captions*. 2024. arXiv:2407.06358.
33. Liu et al. *HOIGen-1M: A Large-scale Dataset for Human-Object Interaction Video Generation*. 2025. arXiv:2503.23715.
34. Unterthiner et al. *Towards Accurate Generative Models of Video: A New Metric and Challenges*. 2018.（FVD）
35. Salimans et al. *Improved Techniques for Training GANs*. NeurIPS, 2016.（Inception Score）
36. Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. NeurIPS, 2017.（FID）
37. Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. ICML, 2021.（CLIP）
38. Huang et al. *VBench: Comprehensive Benchmark Suite for Video Generative Models*. CVPR, 2024.
39. Huang et al. *VBench++: Comprehensive and Versatile Benchmark Suite for Video Generative Models*. 2024.
40. Liu et al. *EvalCrafter: Benchmarking and Evaluating Large Video Generation Models*. CVPR, 2024.
41. Lin et al. *Video-LLaVA / Video Understanding Models for Evaluation*, related technical literature.
42. Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. NeurIPS, 2022.
43. Dao. *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning*. ICLR, 2024.
44. Zhao et al. *Ring Attention with Blockwise Transformers for Near-Infinite Context*. 2023.
45. Jacobs et al. / DeepSpeed Team. ZeRO and memory-optimization literature.
46. Zhao et al. *PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel*. VLDB, 2023.
47. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. ICLR, 2022.
48. Dettmers et al. *QLoRA: Efficient Finetuning of Quantized LLMs*. NeurIPS, 2023.
49. Song et al. *Consistency Models*. ICML, 2023.
50. Sauer et al. *Adversarial Diffusion Distillation*. ECCV, 2024.
51. Wallace et al. *Diffusion Model Alignment Using Direct Preference Optimization*. CVPR, 2024.
52. Esser et al. *Scaling Rectified Flow Transformers for High-Resolution Image Synthesis*. ICML, 2024.
53. Qwen Team. *Qwen2.5-VL Technical Report*. 2025.
54. Qwen Team. *Qwen3 Technical Report and Model Cards*. 2025–2026.
55. Li et al. *Multimodal Foundation Models: From Specialists to General-Purpose Assistants*. 2023.

结语：把视频生成模型看成可测量的复合系统

现代文生视频模型不是一个单独的神经网络，而是

Data Compiler+Video Representation+Conditional Generator+Planner/Controller+Numerical Solver+Distributed System+

Wan 展示了开放 Video DiT 如何从因果 VAE、3D RoPE、Flow Matching 和多任务权重发展到时间专家与高压缩 TI2V；Bernini 展示了 MLLM 理解能力如何通过连续视觉语义计划进入视频生成与编辑。二者共同指向一个趋势：下一代系统的进步不会只来自参数量，而来自**表示、规划、动力学、数据和系统之间更合理的接口**。

读完中册，最重要的能力不是背诵哪个模型有多少参数，而是面对任何新模型都能依次问：

1. 它压缩了什么，丢失了什么？
2. 它在什么空间、沿什么路径学习生成？
3. 条件真正进入了哪些层，能否被忽略？
4. 训练数据单位、课程和质量如何定义？
5. 主要成本是 token、参数、NFE、分支，还是 VAE？
6. 论文的收益是否通过单变量消融得到？
7. 哪些是公开事实，哪些只是合理推算？

这套问题比任何固定排行榜更耐久，也构成从复现走向原创研究的起点。