arXiv: 2601.05175

VideoAuto-R1

Video Auto Reasoning via Thinking Once, Answering Twice

Meta AI　KAUST　Princeton

论文基本信息

标题 VideoAuto-R1: Video Auto Reasoning via Thinking Once, Answering Twice

资源链接 PDF: arXiv:2601.05175
GitHub: IVUL-KAUST/VideoAuto-R1

作者团队 Shuming Liu, Mingchen Zhuge 等（主要来自 Meta AI 与 KAUST，以及 Princeton）

发表去向 2601.05175，分类为 cs.CV (计算机视觉)

> **技术坐标系：**多模态大模型（MLLM） → 视频理解（Video Understanding） → **自适应推理（Adaptive Reasoning / Dynamic Compute）**

1. 核心贡献

核心痛点

现有的视频推理模型（如 Video-R1）陷入了**"强制思考"的陷阱**。对于简单的感知任务（如"那个人穿什么颜色的衣服？"），强行生成长 CoT 不仅增加了推理延迟（Latency）和显存开销，甚至会因为"过度解读"导致幻觉，降低准确率。

核心方案

提出 **"Thinking Once, Answering Twice" (TOAT)** 范式。模型在一次生成过程中，先输出一个**直觉答案**，如果信心不足，再进入**思维链（Thinking）**模式并输出**修正答案**。

核心价值

- **动态计算图：**在推理时实现了 **System 1（直觉快思考）与 System 2（逻辑慢思考）的动态切换**，平均 Token 数从 149 降至 44（~3.3x 加速）。
- **端到端 RL 训练策略：**无需单独训练分类器/Router，通过**双答案奖励机制（Dual-Answer Reward）**和 **GRPO** 算法，让模型内化了"何时该思考"的策略。
- **SOTA 性能：**在推理密集型榜单（VideoMMMU）上超越 Video-R1，同时在感知型榜单（MVBench）上保持高效。

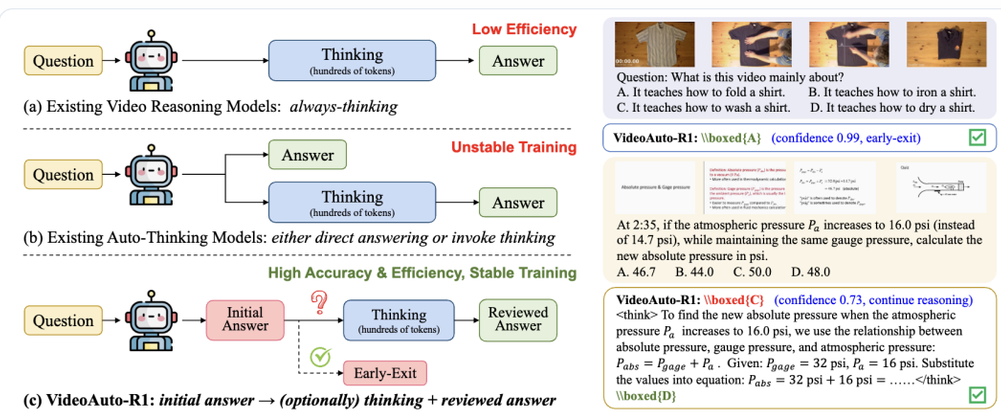

Figure 1 VideoAuto-R1 follows a thinking once, answering twice paradigm. In training, both the initial answer and the reviewed answer are supervised with verifiable rewards. During inference, an early-exit mechanism is adopted to dynamically determine whether to proceed with CoT reasoning. Robot icon from Flaticon (2025).

Figure 1: VideoAuto-R1 与其他方法的对比示意图

2. 核心思想

这篇论文在架构设计上最精妙的地方在于：**它没有引入任何额外的"物理"模块（如独立的 Router 网络或 Critic 模型），而是通过重构"逻辑"数据流，实现了一种"零参数成本"的动态计算架构。**

2.1 总体架构：内嵌式路由的单一模型

在传统的自适应推理系统中，工程师通常会设计一个独立的轻量级 Router（分类器），先判断问题难易，再决定分发给哪个模型或是否开启 CoT。这种做法被称为 **Pipeline System**。

VideoAuto-R1 摒弃了 Pipeline，采用了 End-to-End 的设计：

物理视图

系统只有一个巨大的 Transformer（基于 Qwen2.5-VL/Qwen3-VL）。它由一个冻结的视觉编码器（Visual Encoder）和一个经过 RL 微调的大语言模型（LLM）组成。

逻辑视图

通过训练数据的结构化（Prompt Structuring）和强化学习（RL），模型内部形成了一个隐式的**状态机**。它将"决策是否需要思考"这一元认知能力（Meta-cognition），内化到了 LLM 的预测概率分布中。

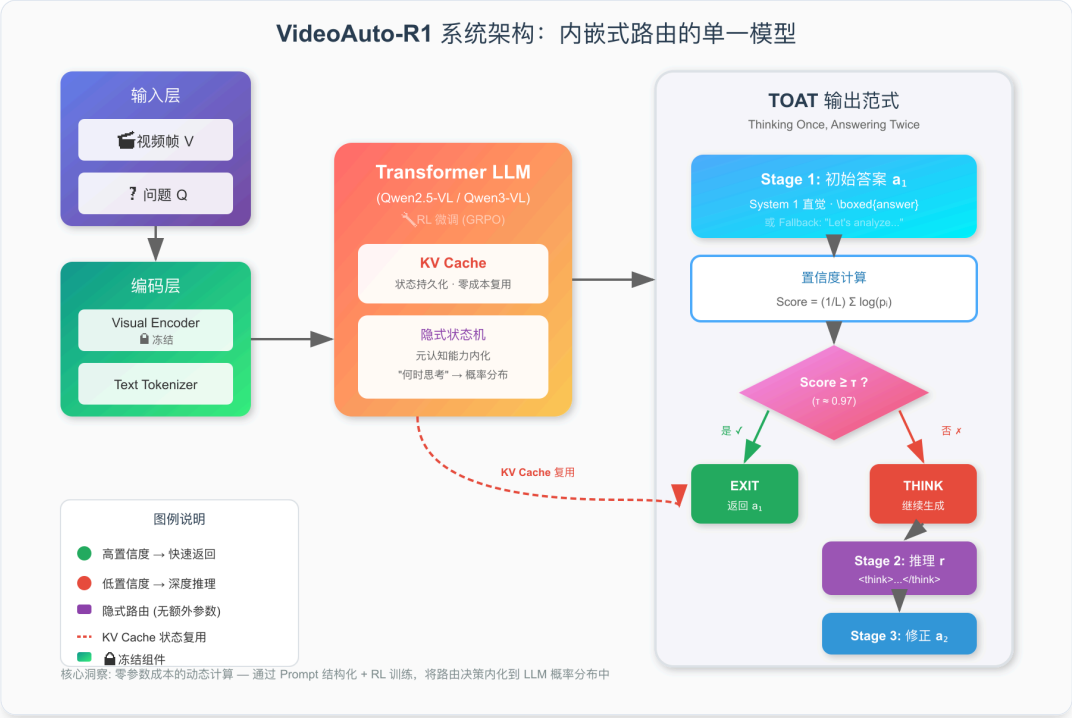

Figure: VideoAuto-R1 总体架构示意图

2.2 模块协同与解耦

在 VideoAuto-R1 中，各个模块的协同方式体现了非常成熟的工程思维：

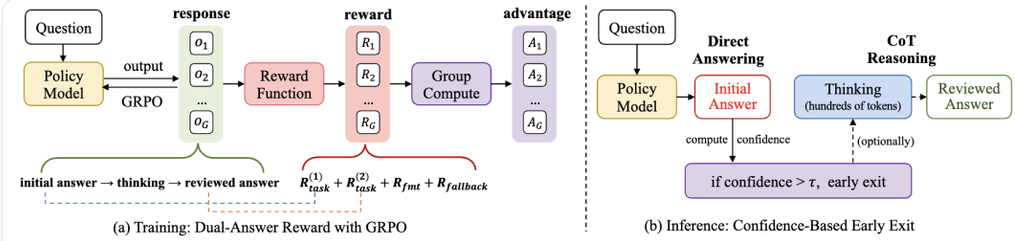

Figure 2 Overview of VideoAuto-R1. (a) **Training:** The response follows the *answer* → *think* → *answer* template, jointly optimizing both the initial and reviewed answers. Specifically, a fallback reward is introduced to avoid a spurious initial guess. (b) **Inference:** The model first produces an initial answer. If its length-normalized confidence exceeds a threshold τ, decoding terminates as direct answering; otherwise, the model continues with CoT reasoning and outputs a reviewed answer, enabling adaptive, confidence-based early exit.

Figure 2: VideoAuto-R1 模块协同架构图

1 训练与推理的解耦 (Training-Inference Decoupling)

训练时（Rigid）

模型**被迫**总是走完 $a_1 \to r \to a_2$ 的全过程。这确保了 a_1 受到直接监督（必须准），同时也确保了 r 能推导出正确的 a_2。

推理时（Flexible）

模型**自由**选择是否截断。

> **System Insight:** 这种解耦避免了在训练阶段引入复杂的 Router 损失函数，防止了 RL 训练中的不稳定（如 Mode Collapse）。

2 KV Cache 的复用 (State Persistence)

当系统决定从 a_1 进入 r 阶段时，不需要重新计算前缀的 Attention Key/Value 矩阵。系统直接复用 a_1 阶段积累的 KV Cache。

> 这意味着：**试探性生成 a_1 的计算成本不是沉没成本（Sunk Cost），而是后续推理的有效前缀（Valid Prefix）。**

3 Fallback 机制作为"软着陆" (Soft Landing)

系统设计了一个特殊的 Fallback String（"Let's analyze..."）。这不仅是一个文本输出，更是一个**语义锚点（Semantic Anchor）**。

当模型输出这个字符串时，它在语义空间中将状态重置为"准备推理"，从而规避了 a_1 瞎猜导致的错误上下文对后续 a_2 的干扰。这是一个非常聪明的**上下文净化（Context Sanitization）**手段。

这就是 VideoAuto-R1 的系统本质：
一个通过强化学习自组织的、具备置信度感知能力的弹性推理计算系统。

3. 关键创新

本节详细介绍 VideoAuto-R1 的三个核心创新点。

创新点 1：Thinking Once, Answering Twice (TOAT) 训练范式

1. 工程挑战：避免"二元对立"的训练坍塌

在构建自适应推理系统时，最直接的思路通常是：训练一个模型，让它在面对问题时，要么输出答案（路径 A），要么输出思考过程（路径 B）。

然而，在实际的大规模训练中，这种**二元互斥（Mutually Exclusive）**的策略往往会导致严重的**训练坍塌（Mode Collapse）**：

> ⚠ **懒惰坍塌：**模型发现直接输出答案更容易（Loss 下降快，惩罚小），于是所有样本都退化为直接回答，彻底放弃思考。

> ⚠ **过度思考坍塌：**或者模型发现只要写很长的废话就能骗过 Reward Model，于是所有样本都生成冗长的 CoT，导致推理效率极低。

> ✓ **TOAT 的核心洞察：**不要让模型做"非此即彼"的选择题，而是让它做"既要又要"的填空题。

2. 算法机制：结构化序列生成

VideoAuto-R1 将训练目标重新定义为一个固定的、三段式的序列生成任务。

```
Prompt Template（系统提示词）：
"First, output your initial answer... Then, think through the reasoning... At last, output the final answer..."
```

Output Sequence Structure（输出序列结构）：

$$Y = [\ \underbrace{\mathcal{B}(a_1)}_{\text{Stage 1: Intuition}} \rightarrow \underbrace{\mathcal{T}(r)}_{\text{Stage 2: Reasoning}} \rightarrow \underbrace{\mathcal{B}(a_2)}_{\text{Stage 3: Reflection}}\ ]$$

注：B(·) 表示 Boxed Answer 格式，T(·) 表示 Thinking Tags 格式。

训练目标函数（Loss Function）：

模型通过 GRPO（一种基于 RL 的优化算法）最大化以下奖励：

$$J(\theta) = \mathbb{E}_{y \sim \pi_\theta}[R(y)]$$

其中 y 必须包含完整的 a_1, r, a_2 结构。这不仅是 Prompt 的要求，也是 Reward Function 的强制约束（如果不符合格式，Reward 直接归零）。

3. 为什么这种范式有效？

我们可以从两个深层视角来理解 TOAT 的有效性：

A. 认知心理学视角：System 1 与 System 2 的解耦与对抗

a_1 **(Initial Answer)** 迫使模型动用 **System 1（直觉系统）**。由于 a_1 出现在序列的最前端，此时没有任何中间计算步骤（Token），模型只能依赖 Visual Encoder 提取的特征和参数里的"直觉"知识。这训练了模型的**感知敏锐度**。

a_2 **(Reviewed Answer)** 迫使模型动用 **System 2（逻辑系统）**。此时模型已经看到了自己生成的 r（推理过程）。这训练了模型的**纠错能力**。

> 💡 **妙处在于"对抗"：**如果 a_1 对了，a_2 必须保持对；如果 a_1 错了，a_2 必须修正它。模型在同一个 Context 窗口内，同时扮演了"鲁莽的猜测者"和"严谨的审核员"。

B. 优化视角：梯度的平滑流动

如果采用传统的"二选一"策略，梯度更新是非常稀疏且方差极大。模型很难搞清楚是因为"选择了思考模式"对了，还是因为"思考的内容"对了。

在 TOAT 范式中，梯度流是稠密且连贯的：

1. **Shared Representation：**a_1 和 a_2 共享同一个 Transformer 主干和 Vision Encoder。
2. **Sequential Dependency：**a_2 的生成条件概率是 $P(a_2 \mid V, Q, a_1, r)$。这意味着，a_2 的梯度会**反向传播（Back-propagate）**穿过 r，一直传导到 a_1。

为了让 a_2 准确，模型必须生成高质量的 r；为了生成高质量的 r，模型必须基于一个合理的 a_1（或者合理的 Fallback 声明）。这实际上建立了一条**从结果到原因的监督链条**。

4. Fallback 机制的数学意义

论文中提到，当 a_1 为特定字符串 $S_{fallback}$ ("Let's analyze...") 时，会触发特殊奖励。我们可以将其形式化为：

$$a_1 \in \{\text{Specific Answers}\} \cup \{S_{fallback}\}$$

这在数学上等同于给输出空间引入了一个**"拒绝选项（Rejection Option）"**。

不带 Fallback

模型被迫在 a_1 阶段就在 A, B, C, D 中选一个。对于数学难题，这纯粹是瞎猜，导致 a_1 的分布充满了噪声。

带 Fallback

模型学会了将 $S_{fallback}$ 映射到那些 "不确定性高" 的输入空间区域。

这不仅提高了训练稳定性，还为推理时的 Early Exit 提供了天然的**校准锚点（Calibration Anchor）**。当模型输出 $S_{fallback}$ 时，置信度通常极高（因为它确信自己不知道），这比在错误答案上犹豫不决要好处理得多。

5. 优势：相比 Baseline 的具体收益

相比于传统的 "Train-to-Choose"（训练分类器）或 "Always-Think"（总是思考）策略，TOAT 带来了以下收益：

1. **数据效率提升：**不需要专门标注"这条数据该不该思考"。每一条 QA 数据，无论是简单还是复杂，都能用来训练 TOAT 模型。简单数据训练了 a_1 的准确性，复杂数据训练了 $r \to a_2$ 的推理能力。
2. **避免模式坍塌：**因为训练目标始终是生成完整序列，模型永远不会退化成"只输出答案"或"只输出思考"。结构是刚性的，但内容是弹性的。
3. **推理灵活性：**虽然训练是刚性的，但因为 a_1 被独立监督且质量很高，推理时我们可以随时在 a_1 处截断。你可以根据当下的延迟预算，决定是拿了 a_1 就跑，还是等 a_2 出来。

总结：TOAT 范式通过**结构化的 Prompt 设计**和**多阶段的奖励监督**，将"何时思考"这个离散的决策问题，转化为了一个连续的序列生成问题。它不仅解决了 RL 训练中的稳定性难题，还完美契合了 Transformer 自回归生成的特性。这是一种极具**工程美感**的解决方案。

创新点 2：双重答案奖励机制 (with GRPO)

1. 工程挑战：多目标优化的权衡

在 TOAT 架构下，模型一次生成有两个输出点：a_1（初始答案）和 a_2（修正答案）。这就引入了一个典型的**多目标优化问题**。

只奖励 a_1：模型会把所有算力集中在第一步，导致 a_2 和推理过程 r 变成无意义的乱码。

只奖励 a_2：模型可能会把 a_1 当作草稿纸甚至随意乱填，导致 Early Exit 策略失效（因为 a_1 不可信了）。

核心难点：如何让模型既重视 a_1 的准确性（为了推理效率），又更重视 a_2 的最终正确性（为了推理上限），同时还要鼓励它在不懂的时候诚实地说"不知道"？

2. 算法机制：精细化的奖励函数

论文设计了一个组合式的奖励函数 R_{total}，并通过 **GRPO** 进行优化。

总奖励公式：

$$R(y) = \underbrace{w_1 \cdot \ (a_1 = y_{gt})}_{\text{System 1 Reward}} + \underbrace{w_2 \cdot \ (a_2 = y_{gt})}_{\text{System 2 Reward}} + \underbrace{\lambda \cdot R_{fmt}(y)}_{\text{Format Reward}} + \underbrace{\alpha \cdot R_{fallback}(a_1, a_2)}_{\text{Honesty Reward}}$$

逐项深度解析：

$w_1 \cdot \mathbb{I}(a_1 = y_{gt})$

含义： 如果初始答案 a_1 与 Ground Truth (y_{gt}) 一致，得 w_1 分。

作用： 强迫模型在第一步就尽力做对。这是保证 Early Exit 有效性的基石。

$w_2 \cdot \mathbb{I}(a_2 = y_{gt})$

含义： 如果修正答案 a_2 与 Ground Truth 一致，得 w_2 分。

关键设定： 论文中设定 $w_2 > w_1$（例如 $w_1 = 0.9, w_2 = 1.1$）。

> 这意味着 (a_1=Wrong, a_2=Right) 的总分（1.1）高于 (a_1=Right, a_2=Wrong) 的总分（0.9）。这向模型传递了一个明确的信号：**"知错能改，善莫大焉"**。最终结果的正确性是第一优先级的。

$\alpha \cdot R_{fallback}(a_1, a_2)$

定义： $R_{fallback} = 1 \Leftrightarrow (a_1 = S_{fallback}) \wedge (a_2 = y_{gt})$，否则为 0。

含义： 如果 a_1 输出了"Let's analyze..." **并且** a_2 最终答对了，额外奖励 α 分（例如 0.3）。

作用： 这是一个非常关键的**激励对齐**。它告诉模型："如果你没把握，不要猜。只要你诚实地申请进入思考模式，并且最后解决了问题，我会给你额外的奖励。"这有效抑制了 a_1 的幻觉。

$\lambda \cdot R_{fmt}(y)$

含义： 格式奖励。必须严格符合 `\boxed<redacted_reasoning>\boxed` 格式。

作用： 这是 RL 训练的边界约束，防止模型为了刷分而输出乱码。

3. 优势总结

端到端的可控性

通过调整 w_1, w_2, α 三个超参数，我们可以精确控制模型的行为偏好（激进 vs 保守，速度优先 vs 精度优先）。

训练稳定性

GRPO 的组内归一化机制，结合明确的奖励信号，使得 RL 训练在无需 Critic 的情况下依然收敛平稳。

无需标注的自我进化

这套机制不需要人类标注"这一步是对是错"。只要有最终答案的 GT，模型就能通过 a_1 和 a_2 的对比，自动反向传播学会中间的推理逻辑。

创新点 3：基于置信度的早退策略

1. 工程挑战：推理延迟与算力的"不可能三角"

在工业界落地大模型时，我们面临一个经典的"不可能三角"：**高精度、低延迟、低成本**。

Always CoT（如 Video-R1）

精度高，但延迟和成本爆炸。每个 Query 都要生成几百个 Token，对于实时视频流处理（如自动驾驶、安防监控）是不可接受的。

Direct Answer

快且便宜，但在复杂任务上精度崩塌。

> **核心痛点：** 我们需要一个机制，能在毫秒级的时间内判断 "这个问题值不值得花这几百个 Token 的算力去思考？" 而且这个判断本身的开销必须趋近于零。

2. 算法机制：内省式概率门控

VideoAuto-R1 没有训练额外的分类器，而是利用了生成模型自带的**不确定性度量**。

1. **解码启动**

   模型开始生成 a_1。

2 **Logits 捕获**

设 a_1 包含 L 个 Token：$t_1, t_2, \ldots, t_L$。

对于序列中的每一个 Token t_ℓ，模型在 Softmax 层都会输出一个概率分布 $P(v \mid \text{context})$。

我们只关注被选中的那个 Token 的概率：$p_\ell = P_\theta(t_\ell \mid t_{<\ell}, x)$。

3 **分数计算**

计算 a_1 的**长度归一化对数概率**作为置信度分数 $\text{Score}(a_1)$：

$$Score(a_1) = \begin{cases} -\infty & \text{if } a_1 = S_{fallback} \\ \frac{1}{L}\sum_{\ell=1}^{L} \log p_\ell & \text{otherwise} \end{cases}$$

注：这里取 Log 是为了数值稳定性（避免连乘下溢），除以 L 是为了消除答案长度的影响（避免长答案天然分数低）。

4 **门控决策**

设定阈值 τ（例如 $\tau = 0.97 \to \log \tau \approx -0.03$）。

$$\text{Decision} = \begin{cases} \text{EXIT} & \text{if } Score(a_1) \geq \log \tau \\ \text{THINK} & \text{if } Score(a_1) < \log \tau \end{cases}$$

场景示例

🎬 场景 1：简单感知任务——"这是什么水果？"

视频内容

一段 5 秒的视频，镜头拉近一个红色的苹果，光线充足，细节清晰。

问题

`What fruit is this?`（这是什么水果？）

Step 1-2: 解码启动 & Logits 捕获

模型在 Softmax 层输出概率分布：

```
P(apple)  = 0.99  ← 模型非常确信是"apple"
P(banana) = 0.005
P(orange) = 0.004
P(dog)    = 0.0001
P(car)    = 0.00001
```

模型选择：$t_1 = \text{apple}$，概率 $p_1 = 0.99$

Step 3: 完整序列概率

输出 `\boxed{apple}`，4 个 Token 的概率：

$p_1 = 0.999, p_2 = 0.998, p_3 = 0.99, p_4 = 0.999$

Step 4: 分数计算

$$\text{Score} = \frac{1}{4}[\log(0.999) + \log(0.998) + \log(0.99) + \log(0.999)] = -0.0035$$

Step 5: 门控决策

阈值：log(0.97) ≈ -0.03

比较：**-0.0035 > -0.03**（在负数中，-0.0035 更接近 0，更大）

Decision = EXIT（秒回，高效！）

🎬 场景 2：复杂推理任务——"计算小球落地速度"

B站/公众号/YouTube：唐国梁Tommy

视频内容

一段物理实验教学视频，讲解自由落体运动，黑板上写着公式 v = √(2gh)。视频中小球从 5 米高处落下。

问题

Calculate the velocity when the ball hits the ground (g=10 m/s²).

Step 1-2: 解码 & Logits 捕获

模型在纠结具体数值，概率分布不确定：

```
P(10) = 0.40  ← 最高，但并不是压倒性优势
P(5)  = 0.35
P(20) = 0.20
P(15) = 0.05
```

Step 3: 分数计算

4 个 Token 的概率：$p_1 = 0.40, p_2 = 0.80, p_3 = 0.40, p_4 = 0.90$

$$\text{Score} = \frac{1}{4}[\log(0.40) + \log(0.80) + \log(0.40) + \log(0.90)] = -0.54$$

Step 4: 门控决策

比较：**-0.54 < -0.03**（置信度很低）

Decision = THINK（进入深度思考模式）

🎬 场景 3：Fallback 触发——"我主动认怂"

输入

一道超级难的微积分题，黑板上密密麻麻全是公式。问题：Solve for the limit as x approaches infinity...

Step 1-2: 模型生成 Fallback

模型经过 RL 训练，学会了一种"求生技能"：遇到不会的题，就坦白承认。

```
\boxed{Let's analyze the problem step by step.}
```

这是预设的 Fallback String ($S_{fallback}$)。

Step 3: 特殊处理

```
if a_1 == "Let's analyze the problem step by step.":
    Score = -∞  # 强制设为负无穷
```

Step 4: 门控决策

无论模型生成这句话时多么自信，Score 都被强制改写为 -∞。

Decision = THINK（强制进入思考模式）

总结：VideoAuto-R1 的置信度门控，就是给模型装了一个"心虚检测器"。

- 检测原理：看你说每个字时的概率（Softmax 输出）
- 平均概率高（接近 1）→ 你很自信 → 放行（Exit）
- 平均概率低（远小于 1）→ 你很心虚 → 扣留（Think）
- 主动说"不会"（Fallback）→ 无条件扣留（Think）

3. 理由：校准性（Calibration）的胜利

这个策略能工作的前提是：**模型的置信度（Confidence）与准确率（Accuracy）是正相关的。**

论文在 Table 8 中提供了强有力的证据：

Table 8 Initial Answer's Confidence Separates Think-Needed Samples. VideoMMMU shows markedly lower probability than MVBench and MMVU, indicating greater uncertainty. Accordingly, our confidence-based early exit triggers thinking more often on Video-MMMU, yielding a +4% accuracy gain.

Setting	MVBench	MMVU	VideoMMMU
Probability of Initial Answer	0.948	0.933	0.874
Think Ratio	25%	39%	51%
Performance Gains	+0.1	+0.4	+4.0
Recall of Think-Needed Samples	100%	100%	94%

Table 8: 不同任务类型的平均置信度对比

Perception Tasks (MVBench)

模型生成的 a_1 平均置信度高达 **0.948**。这意味着绝大多数时候模型都确信自己看清了。这也解释了为什么这类任务 Early Exit 率高。

Reasoning Tasks (VideoMMMU)

模型生成的 a_1 平均置信度降至 **0.874**。模型自己"感觉到"了困难。

这表明，尽管现代 LLM 普遍存在 Overconfidence 问题，但在**相对排序（Relative Ranking）**上，VideoAuto-R1 依然保持了良好的校准性。简单的阈值切割就能有效地将"简单样本"和"困难样本"在统计上分离开来。

4. 优势：工程上的极致性价比

Zero Overhead（零开销）

- 不需要跑额外的 BERT 做分类
- 不需要多生成任何 Token 来做自我反思
- 计算 Log-Prob 是解码过程的副产物，计算量几乎为零

KV Cache 友好

如果决定 Continue，之前的 a_1 的 KV Cache 是可以直接复用的，作为 r 的前缀。没有浪费任何算力。

4. 实验透视

通过系统性的实验验证 VideoAuto-R1 的有效性。

4.1 配置全景：覆盖"感知-推理"全谱系

论文在选择数据集时，非常有策略地覆盖了视频理解的两个极端维度：**感知（Perception）**和**推理（Reasoning）**。

Perception-Heavy (感知型)

MVBench (20 tasks, 4K videos)
测试细粒度动作、物体属性。特点：需要看清，不需要深想。

Charades-STA (Temporal Grounding)
时序定位任务。特点：找准时间点，输出必须精确。

Reasoning-Heavy (推理型)

VideoMMMU (Multi-discipline)
涵盖物理、化学、数学等多学科视频题。特点：核心战场，需要多步推导。

VideoMME (Long Context)
长视频理解。特点：考察记忆力和长程依赖。

基线选择 (Baseline Selection)

Base Model
Qwen2.5-VL-7B (直接生成)。作为"下限"，验证 RL 是否有效。

Video-R1
Thinking-Only，总是强制思考。作为"上限"参考，也是主要的 PK 对象。

VideoChat-R1
早期的视频推理尝试。

4.2 结果解读：精度与效率的双赢

1. 推理任务的绝对碾压 (Dominance in Reasoning)

Table 3 Evaluation Results on Video QA Benchmarks. We compare VideoAuto-R1 with thinking-only video reasoning models on both perception-oriented and reasoning-heavy benchmarks, and also report the average response length (in tokens). † means reproduced results. We also report the think ratio, defined as the proportion of samples on which the model triggers CoT reasoning. Relative to the Qwen baseline, VideoAuto-R1 yields consistent and more pronounced gains on the reasoning benchmarks. We further observe that the thinking ratio is low on perception-oriented benchmarks but substantially higher on reasoning-heavy ones.

Model	Reasoning Mode	Response Length	Video Perception Benchmark				Video Reasoning Benchmark	
			VideoMME	MVBench	LongVideoBench	MMVU	VideoMMMU	MVP
Qwen2.5-VL-7B†	✗	3.0	66.0	67.1	60.9	66.2	54.7	36.5
Qwen3-VL-8B†	✗	2.2	72.5	69.4	67.6	69.9	61.0	40.5
Temporal-RLT	Think-Only	-	57.6	68.1	-	65.0	-	-
Video-RFT	Think-Only	-	59.8	62.1	-	68.5	51.1	-
Video-R1	Think-Only	386	61.8	65.5	-	65.0	51.4	33.0
Video-RTS	Think-Only	-	63.0	-	56.6	66.4	52.7	-
VITAL	Think-Only	-	64.1	-	-	68.7	54.2	-
LongVILA-R1	Think-Only	-	65.1	67.6	58.0	-	-	-
LOVE-R1	Think-Only	-	66.2	66.6	60.1	-	-	-
VideoChat-R1.5	Think-Only	133	65.2	70.6	61.4	-	49.6	38.6
VideoAuto-R1 (Qwen2.5-VL-7B)	**AutoThink**	**44**	**67.3**	**71.0**	**60.5**	**69.7**	**58.6**	**39.4**
(think ratio)			(40%)	(25%)	(39%)	(28%)	(51%)	(44%)
VideoAuto-R1 (Qwen3-VL-8B)	**AutoThink**	**52**	**71.7**	**72.0**	**67.4**	**71.1**	**65.0**	**43.0**
(think ratio)			(11%)	(31%)	(20%)	(38%)	(53%)	(56%)

Table 3: 各模型在不同 Benchmark 上的表现对比

指标 在 **VideoMMMU** 上，VideoAuto-R1 (58.6%) 相比 Video-R1 (51.4%) 提升了 **+7.2%**。

深度归因：

- Video-R1 是"盲目思考"。对于简单的感知题，强行 CoT 往往会导致模型产生幻觉（过度解读背景像素），从而选错答案。
- VideoAuto-R1 的 Dual-Answer Reward 强迫模型在 a_2 阶段修正 a_1。这种"先猜后改"的机制比"直接写长文"更符合人类解题逻辑（Hypothesis Testing）。

2. 感知任务的"无损加速" (Lossless Acceleration in Perception)

指标 在 **MVBench** 上，VideoAuto-R1 (71.0%) 远超 Video-R1 (65.5%)。

这非常反直觉！通常认为 CoT 至少不会降分。

Auto-Thinking 的胜利：MVBench 上 VideoAuto-R1 的 Think Ratio 只有 25%。这意味着 75% 的时候它都在用 a_1（System 1）。

System Insight: 这证明了在感知任务上，**System 1 > System 2**。视觉直觉往往是准确的，而过多的语言描述（Verbalization）会造成信息有损压缩和语义漂移（Semantic Drift）。

3. 效率的质变 (Efficiency Breakthrough)

数据 平均 Token 数从 Video-R1 的 **386 tokens** 降至 VideoAuto-R1 的 **44 tokens** (Qwen2.5-7B)。

3.3x - 8.7x

吞吐量提升

物理意义

同样的 H100 集群，可以服务 3-8 倍的用户。在工业界，这直接等于 **Cost Reduction (降本)**。

4.3 消融分析

A. SFT vs. RL (Table 6)

Table 6 Comparison between Different Training Strategies. VideoAuto-R1 delivers stronger performance than SFT on reasoning and grounding benchmarks, and surpasses standard RL with CoT while reducing response length, achieving efficiency and efficacy.

Training	Inference	Response Length	VideoMME	MMVU	VideoMMMU	Charades-STA
Qwen2.5-VL-7B	Direct	3.0	66.0	66.2	54.7	52.9
SFT	Direct	2.3	67.0	65.9	56.5	56.3
RL without Thinking	Direct	2.5	66.0	66.4	54.4	58.8
RL with Thinking	CoT	149	66.1	67.5	56.4	59.8
VideoAuto-R1	**Direct/CoT**	**44**	**67.3**(+1.3)	**69.7**(+3.5)	**58.6**(+3.9)	**60.0**(+7.1)

Table 6: SFT 与 RL 训练方式对比

实验：对比 SFT (Supervised Fine-Tuning) 和 RL (VideoAuto-R1)。

结果：SFT 在 Reasoning 任务上提升微弱 (54.7 → 56.5)，而 RL 直接干到了 58.6。

> 洞察：SFT 只是在模仿 CoT 的**格式**，而 RL（特别是 GRPO）真正学会了 CoT 的**逻辑**。因为 RL 是基于结果（Answer Correctness）反馈的，它不在乎中间过程写得像不像人，只在乎能不能推导出正确答案。

– B. 权重分配的玄学 (Table 9)

Table 9 Ablations on Reward Design. Emphasizing the reviewed answer by setting $w_2 > w_1$ outperforms equal weighting. Adding a small fallback reward α further improves accuracy.

$w_1 : w_2$	α	VideoMME	VideoMMMU	MVP	Charades-STA
1:1	✗	66.1	56.1	38.3	58.3
0.9:1.1	✗	66.0	56.4	37.2	59.1
0.9:1.1	✓	**67.3**	**58.6**	**39.4**	**60.0**
0.8:1.2	✗	65.8	56.9	38.1	58.7
0.8:1.2	✓	66.3	57.9	38.8	59.3

Table 9: 不同 w_1:w_2 比例的效果对比

实验：对比 $w_1 : w_2$ 的不同比例（1:1 vs 0.9:1.1）。

结果：$w_1 = w_2$ 时，VideoMMMU 只有 56.1；$w_1 < w_2$ 时，涨到 58.6。

> 洞察：这验证了必须给 a_2 更高的权重，才能形成**梯度势能差**，驱动模型将 r 视为修正 a_1 的工具，而不是简单的续写。

– C. 阈值 τ 的敏感性 (Figure 3)

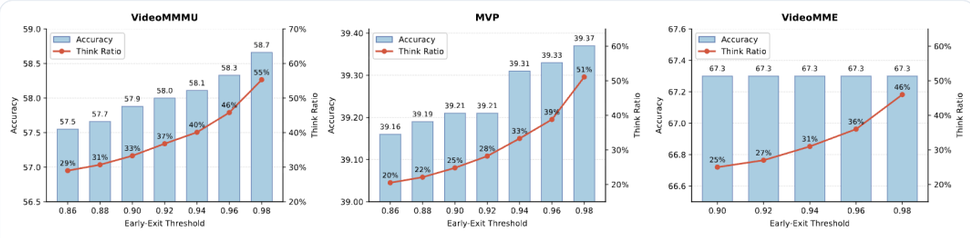

Figure 3 Effect of the Early-Exit Threshold on Accuracy and Think Ratio. In practice, we set $\tau = 0.97$ for all datasets.

Figure 3: 不同阈值 τ 对准确率的影响

现象：随着 τ 从 0.86 增加到 0.98，VideoMMMU（推理题）的准确率稳步上升，而 VideoMME（感知题）几乎不变。

> 洞察：
> - 这证明了模型的 **Confidence Calibration** 是极其鲁棒的。
> - 它揭示了一个重要性质：**感知任务的收益边际递减极快**。多思考并不能让你看得更清；但推理任务的收益边际递减较慢，多思考一步就多一分胜算。