/TemplateVersion (2026.1)

用于序列决策的强化语言模型

Jim Dilkes, Vahid Yazdanpanah, Sebastian Stein

University of Southampton
j.dilkes@soton.ac.uk

Abstract

大型语言模型（LLMs）在作为顺序决策代理方面显示出潜力，但由于依赖大规模且计算成本高的模型，其应用往往受到限制。这就需要改进较小的模型，然而现有的后训练方法是为单轮交互设计的，无法处理多步分析任务中的信用分配。为了解决这个问题，我们引入了多步组相对策略优化（MS-GRPO），这是一种用于后训练 LLM 代理的新算法，其基础是正式的文本调节随机游戏（TSMG）和语言代理策略（LAP）框架。对于信用分配，MS-GRPO 将整个累积的情节奖励归因于每个独立的情节步骤。我们为这一算法补充了一种新的绝对优势加权情节采样策略，我们表明这一策略改进了训练性能。我们通过在《贪吃蛇》和《冰冻湖》上对一个 30 亿参数模型进行后训练来评估我们的方法。我们的实验表明，该方法在提高决策性能方面是有效的：我们后训练的 30B 参数模型在《冰冻湖》任务中比 72B 参数基线超出 50% 。这项工作表明，针对性的后训练是依靠模型规模来创建使用 LLMs 的顺序决策代理的一个切实可行且高效的替代方案。

介绍

序列决策，指的是一个代理选择连续动作以最大化长期目标的问题，代表了人工智能中的一个基本且普遍存在的挑战。对此问题的计算方法已推动了在航天器控制 (Bernard et al. 1998) 、医疗治疗 (Murphy 2003; Bani-Harouni et al. 2025) 、机器人操作 (Levine et al. 2016) 、数据中心冷却效率 (Evans and Gao 2016) 和车辆路线规划 (Kool, Van Hoof, and Welling 2019) 等多种应用中取得重要成就。最近，大型语言模型（LLMs）强大的推理和自然语言理解能力开启了一种新的范式：能够遵循人类指令在通过文本传达的动态环境中操作的代理，无论是数字 (Zheng et al. 2025b) 还是物理环境 (Mower et al. 2024; Li et al. 2025) 。它们在利用 LLMs 固有的广泛世界知识和推理能力来灵活应对序列决策问题方面具有巨大潜力。

尽管有这种前景，但有效利用 LLMs 进行顺序决策仍然是一个未解的挑战。具体而言，有证据表明 LLMs 在低级别动作选择上存在困难 (Zhang et al. 2024) ，内在地不是好的规划者 (Kambhampati et al. 2024) ，并且有效的决策通常需要使用计算开销大的推理链的巨大模型 (Tanahashi et al. 2023; Yao et al. 2023; Shinn et al. 2023; Zhou, Du, and Li 2024) 。例如，Trivedi et al. (2024) 发现他们最有能力的智能体，在真实的数字任务上使用 GPT-4o (OpenAI et al. 2024) ，每项任务的成本为 $0.70 ，同时成功率低于 50% 。这些缺陷限制了 LLMs 的实用性和可扩展性，突显出需要新的训练方法来提高更高效模型的能力。

然而，现有的 LLM 后训练方法，即那些调整和适应预训练模型以满足特定应用需求的方法，并不适合这一领域。这些方法通常基于增强学习（RL）(Sutton and Barto 2018) ，旨在优化模型用于单回合任务，并从验证者处获得直接反馈，就像在具有可验证奖励的增强学习（RLVR）(DeepSeek-AI et al. 2025; Zheng et al. 2025a; Yu et al. 2025; Wang et al. 2025a; Hou et al. 2025; Park et al. 2025) 中一样，或来自人类偏好模型的反馈，如在从人类反馈中学习的增强学习（RLHF）(Ziegler et al. 2020; Ouyang et al. 2022; Rafailov et al. 2023; Zhong et al. 2025) 中。然而，这样的方法与需要结果与行动间信用分配的顺序决策任务不兼容。解决这一问题是一个新兴的研究领域。例如，RAGEN 系统 (Wang et al. 2025b) 根据完整环境的剧集来条件控制代理的语言生成，并将整个剧集的信用分配给代理的完整行动序列。

此外，将 LLM 作为决策代理时会出现概念上的限制：优化过程发生在符号序列上，这些符号是源于自然语言的交流单位，而有效的规划需要在问题领域内选择具体行动（例如在空间环境中的导航动作）。这种差异反映了交流行为（例如对话系统中的言语行为 (Traum 1999) ）与序列决策所需的操作行动 (Georgeff 1988) 之间的区别。弥合这一差距需要新的方法，这些方法可以正式将 LLM 语言中心的输出与代理规划和控制所需的结构化、领域特定行动对齐。

在这种背景下，我们首次：

1. 定义一个将基于语言的智能体与序列决策环境连接的形式框架，包括文本媒介随机游戏（TMSG），它通过显式的文本接口对环境进行建模，以及语言智能体策略（LAP），其定义了基于大语言模型（LLM）的智能体策略。
2. 引入多步分组相对策略优化（MS-GRPO），这是一种通过将整个累积的回合奖励分配给每一个独立步骤来调整 GRPO 方法用于序列决策任务的算法。为了提高效率，每个步骤的优化仅使用当前状态作为上下文。
3. 提出一种新颖的绝对优势加权（AAW）集采样策略，我们证明这种策略可以提升训练性能。
4. 证明我们的后训练 3B 参数模型在冻结湖任务中以 50% 超过了一个更大的 72B 参数基线 LLM，展示了领域特定训练相对于模型规模的价值。
5. 对 MS-GRPO 算法的功能进行批判性分析，重点突

出其训练方差高以及在 LLM 基础代理中引发泛化的结果参差不齐。

框架

本节定义了我们的框架，该框架用于在序列决策环境中基于语言模型的代理。该框架由两个核心贡献组成：(1) 一个文本中介随机游戏（TMSG），它形式化了一个所有交互完全通过文本进行中介的环境；以及 (2) 一个语言代理策略（LAP），它通过语言模型和接口组件来参数化代理的行为。我们的框架明确分离代理和环境，这对于两个原因很重要。首先，它允许我们使用部分可观测随机游戏（POSG）(Hansen, Bernstein, and Zilberstein 2004) 的形式化来建模 TMSG。其次，LAP 形式化明确指出我们可以控制哪些组件来影响代理的决策过程。

文本介导的随机博弈

为了正式建模具有基于文本接口的顺序决策环境，我们将一个文本介导的随机游戏（TMSG）定义为一个元组 $G = (\mathcal{P}, \mathcal{S}, \mathcal{A}, \Omega, O, P, R)$ 。这一形式化建立在随机游戏 (Shapley 1953) 的基础上，并且与部分可观察随机游戏有相似之处，其关键约束是每个代理的观察空间是所有文本字符串的集合。虽然 POSGs 提供了一种熟悉的结构，但 TMSG 使得基于 LLM 的代理与环境之间的文本接口变得明确。G 的组件定义如下：

- $\mathcal{P} = \{1, \ldots, p\}$ 是一个有限的 p 玩家的集合。[1]
- $\mathcal{S}$ 是有限的博弈状态集。终端状态集表示为 $\mathcal{S}_f \subset \mathcal{S}$ 。
- $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_p$ 是联合动作空间，其中 $\mathcal{A}_i$ 是代理 i 所有可能动作的集合。函数 $\mathcal{A}(s) \subseteq \mathcal{A}$ 返回状态 s 中合法联合动作的集合。
- $\Omega = \Omega_1 \times \cdots \times \Omega_p$ 是联合观测空间，其中每个特定代理的观测空间 $\Omega_i = \Sigma^*$ 是所有文本字符串 Σ^* 的集合，Σ 是标记的词汇表。
- 一个观测函数 $O : \mathcal{S} \times \mathcal{A} \to \Delta(\Omega)$ ，其中 $O(o|s', a)$ 是在采取联合动作 a 并转移到状态 s' 后进行联合观测 o 的概率。
- $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ 是一个状态转移函数，其中 $P(s'|s, a)$ 是在状态 s 执行联合动作 a 时转移到状态 s' 的概率。
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^p$ 是奖励函数，其中 $R(s, a)$ 返回一个 p 数值列表，表示在状态 s 下采取联合行动 a 后每个玩家的奖励。

语言代理策略

在我们考虑的设置中，玩家 $i \in P$ 由一个计算代理控制，其目标是最大化预期的累积奖励。为了实现这一目标，代理学习一种策略，该策略根据其观察制定选择行为的方案。在这里，我们概述我们为基于语言模型的代理提出的架构，我们称之为语言代理策略（LAP）。

先前关于 RLVR 的研究将策略视为由 LLM (Zheng et al. 2025a) 描述的 token 的概率分布。虽然这种框架是有用的，但我们对序列任务的关注需要优化环境动作，而不仅仅是产生这些动作的 token。

一个代理的策略指定了在给定观测 o 时采取行动 a 的概率。在我们的基于 LLM 的框架中，策略由一组组件 $\Pi_i = (\mathcal{L}_{\theta_i}, \mathcal{G}_i, \mathcal{T}_i, \psi_i)$ 参数化，包括：

- $\mathcal{L}_{\theta_i}$ ：具有参数 θ_i 的生成语言模型。
- $\mathcal{G}_i$ ：控制 $\mathcal{L}_{\theta_i}$ 的 token 采样行为的生成配置（例如，温度，top-k）。
- $\mathcal{T}_i$ ：一个提示模板，是一个带有占位符的文本字符串，可以填充一个观测字符串来创建完整的输入提示。
- $\psi_i : \Sigma^* \to \mathcal{A}_i \cup \{\perp\}$ ：一个动作提取函数，该函数解析 $\mathcal{L}_{\theta_i}$ 的文本输出，并将其映射到一个有效的游戏动作，或者如果输出无法解释为有效动作，则映射到 $\perp$ 。

。该参数化的策略可以表达为 $\pi_{\Pi_i}(a|o) \in [0, 1]$ 。我们将操作在特定生成配置 $\mathcal{G}_i$ 下的语言模型 $\mathcal{L}_{\theta_i}$ 表示为 $\mathcal{L}_{\theta_i|\mathcal{G}_i}$ 。该符号表示生成随机文本的函数。

在时间 t ，通过以下过程从语言代理策略中采样动作 a_t^i ：

1. 提示构建：根据观察 o_t^i 并使用代理的模板 $q_t^i = \mathcal{T}_i(o_t^i)$ 构建提示 $q_t^i \in \Sigma^*$ 。
2. 随机文本生成：一个文本补全，$c_t^i \in \Sigma^*$ ，是在给定输入提示 q_t^i 的情况下从语言模型中采样得到的：$c_t^i \sim \mathcal{L}_{\theta_i|\mathcal{G}_i}(\cdot|q_t^i)$ 。
3. 动作解析：代理 a_t^i 采取的动作由解析函数 $a_t^i = \psi_i(c_t^i)$ 从完成字符串中提取。

代理-环境交互

LAP 智能体与 TMSG 环境之间的交互以离散时间步骤进行。这个智能体-环境循环适用于任何一组智能体，每个智能体都实现一个策略 π_i 。在这项工作中，我们主要感兴趣的情况是策略是一个 LAP，π_{Π_i} 。

从初始状态 s_0 开始，每个时间步 t 的事件顺序如下：

1. 每个玩家 $i \in P$ 同时选择一个动作：$a_t^i \sim \pi_i(\cdot|o_t^i)$ 。如果 $a_t^i = \perp$ ，应用预定义的恢复策略（例如，不采取行动或随机行动）。所有代理的动作集合形成联合动作 $a_t = (a_t^1, \ldots, a_t^p)$ 。
2. 环境接收到联合动作 a_t ，并通过从状态转移函数中采样从状态 s_t 转移到状态 s_{t+1} ：$s_{t+1} \sim P(\cdot|s_t, a_t)$ 。
3. 环境生成一个奖励向量 $r_{t+1} = (r_{t+1}^1, \ldots, r_{t+1}^p)$ ，该向量由奖励函数 $R(s_t, a_t)$ 计算得出。
4. 环境从观察函数中采样一个联合观测 $o_{t+1} = (o_{t+1}^1, \ldots, o_{t+1}^p)$ ：$o_{t+1} \sim O(\cdot|s_{t+1}, a_t)$ 。
5. 如果 $s_{t+1} \in \mathcal{S}_f$ ，事件终止。否则，增加 T 并重复。

结合起来，TMSG 和 LAP 形式化方法为分析序列决策环境中的 LLM 代理提供了一个完整的框架。

创建用于顺序决策的有效 LLM 代理需要新的方法，以克服单回合优化算法的局限性，特别是来自稀疏、延迟奖励的信用分配问题。本节介绍了我们通过在 TMSG 框架内训练一个目标导向的 LAP 来解决这一问题的方法，该方法包括两个技术贡献：

为了优化 LAP 代理在 TMSG 环境中最大化预期累积奖励的行为，我们提出了多步群体相对政策优化

[1] 虽然这项工作侧重于单代理后训练，但我们采用更通用的多代理情况来提供一个稳健的框架。因此，非学习代理，例如在贪吃蛇环境中的对手，被视为环境的组成部分，而不是这项工作的主题。

(MS-GRPO)。我们的算法是 GRPO (Shao et al. 2024) 的一种变体，一种策略梯度方法 (Williams 1992) 。

虽然 GRPO 比较单步响应与相同提示的奖励，但 MS-GRPO 为多步任务调整了这一方法。它通过计算总累计奖励来确定优势值，并将该值分配给该情节中生成的每个标记。这种将完整的情节奖励归因于每个动作的技术是一种蒙特卡罗信用分配 (Sutton and Barto 2018) 。我们使用 GRPO 而不是诸如 PPO (Schulman et al. 2017) 这样的演员-评论家方法，是因为它占用的内存较少，允许训练更大的模型或使用更长的上下文。

虽然 LAP 代理的行为由完整策略 π_Π 定义，但我们仅优化决定其 LLM 在标记上的分布 $\mathcal{L}_{\theta|\mathcal{G}}(\cdot|q_t)$ 的参数 θ 。尽管 TMSG 框架支持多个玩家，但这项工作集中于优化单个代理，因此我们在以下定义中省略玩家索引 i 。

MS-GRPO 目标函数定义为：

$$\mathcal{J}_{\text{MS-GRPO}}(\theta) = \mathbb{E}_{o\sim\mathcal{D},\{y_{j,t}\}_{j=1}^{G}\sim p_{\theta_{\text{old}}}(\cdot|o_t)} \frac{1}{G}\sum_{j=1}^{G}\frac{1}{|y_j|}\sum_{t=0}^{T_j-1}\left[\mathcal{L}_{\text{CLIP}}(\theta,j,t)\right] - \beta\mathbb{D}_{\text{KL}}\left(p_\theta\|p_{\text{ref}}\right) \quad (1)$$

，其中 $\mathcal{L}_{\text{CLIP}}$ 是情节 j 中时间步 t 的令牌级目标：

$$\mathcal{L}_{\text{CLIP}}(\theta,j,t) = \sum_{k=1}^{|y_{j,t}|}\min\left(w_{j,t,k}A_j,\ \ \text{clip}\left(w_{j,t,k},1-\epsilon_{\text{low}},1+\epsilon_{\text{up}}\right)A_j\right) \quad (2)$$

，重要性比率 $w_{j,t,k}$ 为：

$$w_{j,t,k} = \frac{p_\theta(y_{j,t,k}\mid o_{j,t},y_{j,t,<k})}{p_{\theta_{\text{old}}}(y_{j,t,k}\mid o_{j,t},y_{j,t,<k})} \quad (3)$$

这里，G 是组大小，T_j 是在第 j 回合中的时间步数，$|y_j|$ 是生成的标记的总数，$y_{i,t,k}$ 是在时间步 t 的完成中第 k 个标记。$\mathcal{D}$ 表示由 TMSG 动力学决定的观察分布。回合优势 A_j 是通过标准化合成奖励 $C_j = \sum_{T_j}(r_{j,t}+\Phi_{j,t})$ 计算的，合成奖励结合了累计环境奖励和任务特定的塑形奖励（Φ_j ）：

$$A_j = \frac{C_j - \text{mean}(\{C_1,\ldots,C_G\})}{\text{std}(\{C_1,\ldots,C_G\})} \quad (4)$$

。最后，$\mathbb{D}_{\text{KL}}\left(p_\theta\|p_{\text{ref}}\right)$ 是相对于参考模型（即微调前的原始 LLM）的 KL 惩罚，ϵ_{low} 、ϵ_{up} 和 β 是超参数。MS-GRPO 算法详见算法 1 。

为了提高训练效率，我们提出了绝对优势加权（AAW）样本选择策略。这个策略优先选择具有较高优势值的样本，其灵感来源于优先经验回放。直观上，这些样本代表了最显著的成功或失败，对于学习是最具信息价值的。

我们计算所有 G 生成的情节的群体相对优势（方程 4 ），然后无放回地抽样 $G' < G$ 个情节。选择情节 j 的概率由缩放的绝对优势的 Softmax 给出：

$$p_j = \frac{\exp(|A_j|/T_{\text{ep}})}{\sum_{i=1}^{G}\exp(|A_i|/T_{\text{ep}})} \quad (5)$$

，其中温度 $T_{\text{ep}} \in (0,\infty)$ 控制加权的强度。较小的 T_{ep} 使抽样集中在极端优势的情节上，而较大的 T_{ep} 则趋向于均匀分布。

Algorithm 1: 多步骤群体相对策略优化用于语言代理策略

Require: Initial model parameters θ_{ref} ; initial state distribution $\mathcal{D}_0$; Group size G ; learning rate η ; Hyperparameters ϵ , β ; Sampled group size G' ' Sampling temperature T_{ep}

Initialize policy parameters $\theta \leftarrow \theta_{\text{ref}}$
for training iteration $= 1,\ldots,M$ do
  Set $\theta_{\text{old}} \leftarrow \theta$
  Sample initial state $s_0 \sim \mathcal{D}_0$
  Generate initial observation o_0 from s_0
  for episode $j = 1$ to G do
    Set $o_{j,0} \leftarrow o_0$
    for episode step $t = 0$ until termination do
      $q_{j,t} = \mathcal{T}(o_{j,t})$ {Construct prompt}
      $y_{j,t} \sim p_{\theta_{\text{old}}}(\cdot|q_{j,t})$ {Generate completion}
      $a_{j,t} = \psi(y_{j,t})$ {Parse action}
      Take action $a_{j,t}$, observe $o_{j,t+1}$ and $r_{j,t+1}$
      if terminal state then
        break inner loop
      end if
    end for
    Compute reward $C_j = \sum_t(r_{j,t+1}+\Phi_j,t)$
  end for
  Compute advantages $\{A_i\}_{j=1}^{G}$ as normalized rewards
  if $T_{\text{ep}} > 0.0$ then
    Sample G' episodes using AAW Sampling
    Recompute $\{A_i\}_{j=1}^{G'}$ using only sampled episodes
  end if
  Update policy parameters using gradient ascent: $\theta \to \theta + \eta\nabla_\theta\mathcal{J}_{\text{MS-GRPO}}(\theta)$
end for
return π_θ

实验装置

我们通过一系列实验评估我们所提出的方法。我们的实验旨在确定 MS-GRPO 是否能够提高小型 LLM 的决策能力，并评估这些改进是否能够推广到未见过的环境或训练环境的变体。为了实现这一目标，我们使用两个二维网格世界环境，Snake 和 Frozen Lake，并通过每一集的总累积奖励来评估代理的性能。

我们选择了 Snake 和 Frozen Lake 因为它们的动态简单却对小型语言模型具有挑战性，使它们成为评估学习算法有效性的理想选择。其简单的结构允许创建变体以测试泛化能力。

两个环境具有相同的动作空间 $\mathcal{A} = \{Up, Down, Left, Right\}$ 和相似的目标，每个都要求代理在导航二维网格时避开危险并到达目标。对于一个 LAP 代理，解决这些任务要求从文本观察中识别目标和危险，规划策略，并忠实地将该计划转化为行动。无效操作的恢复策略是不采取任何行动。

蛇 代理控制一条通过吃果实而变长的蛇。一个非-LAP 代理控制第二条蛇，该蛇采取随机有效的动作（避免墙壁和自己的尾巴）。在与蛇自身的尾巴、另一条蛇或网格

边界碰撞时，回合终止。果实被消耗后，会在一个随机的空棋盘格子上被重新放置。该环境改编自 Kamradt (2025) 。

冰湖 智能体在冰块的网格上导航以到达目标。一些瓷砖上有洞，移到一个洞上会终止此回合。确保有一条安全路径到达目标，碰到墙壁没有效果。环境来自 Gymnasium 库 (Towers et al. 2024) 。

我们为每个环境创建变体以测试泛化的不同方面：

- Snake-Standard（训练/评估）：一个 10x10 的网格，其中有一条蛇和 5 个苹果，给予 +1 奖励。碰撞会导致 −3 奖励并终止此回合。
- Snake-Poison（评估）：与 Snake-Standard 类似，但苹果提供 −1 奖励，测试代理覆盖其训练目标的能力。
- FrozenLake-NotSlippery（训练/评估）：一个 4x4 的网格，其中每个方格有 0.2 的概率是一个洞，到达目标会获得 +1 奖励。
- FrozenLake-Slippery（评估）：类似于 FrozenLake-NotSlippery，但运动是随机的。智能体以概率 1/3 朝选定方向移动，以 1/3 的概率向每个垂直方向移动。这个变体测试在不确定性条件下的规划能力。

代理-环境接口

每个环境的状态通过 TMSG 的观测函数 O 转换为文本。我们同时以两种方式提供观测：(1) 作为实体坐标列表和 (2) 作为二维字符网格。这些都随着描述其含义的静态文本一起补充提供。静态的环境规则和目标描述被加在动态状态表示之前，形成观测 o 。每个环境变体的静态和动态文本在技术附录中给出。完整的观测被插入到 LAP 代理的模板 $\mathcal{T}$ 中，该模板提供与环境无关的推理结构和输出格式说明。

奖励设计

奖励信号引导智能体实现两个目标：最大化其环境奖励和生成格式良好的文本。虽然仅靠环境奖励可能会隐式地鼓励良好的格式化，我们添加了一个显式的格式惩罚，Φ ，这种方法由 DeepSeek-AI et al. (2025) 采纳。智能体的总奖励是由两个组成部分组合而成：

- 环境奖励（R ）：来自环境的本地奖励，加上每次无效动作的 −0.5 罚分。
- 格式惩罚（Φ ）：一组用于不良文本模式的惩罚：
  - 长度惩罚，一种用于过长文本生成的线性惩罚，在 180 到 200 个标记之间的回复，其惩罚系数从 0 缩放到 −0.5
  - 结构惩罚，即对于每个缺失、不必要或嵌套不正确的 XML 标签的 −0.5 惩罚
  - 额外文本惩罚，如果在最终的 < /action > 标签后生成任何文本，将会有一个 −0.5 的惩罚

实验协议和模型

我们使用 Qwen2.5-3B-Instruct (Qwen et al. 2025) 模型进行后训练，因为它的规模在能力与计算足迹之间提供了良好的平衡。我们将其后训练性能与两个较大模型 Qwen2.5-32B-Instruct 和 Qwen2.5-72B-Instruct 进行比较。我们还在 Snake-Standard 上训练一个深度 Q 网络 (DQN) (Mnih et al. 2013) ，以提供一个非 LLM 基线。智能体分别在 Snake-Standard 和 FrozenLake-NotSlippery 环境中进行训练，并在所有四个变体上进行评估。我们进行了一项消融研究，比较 MS-GRPO 在有无 AAW 采样情况下的有效性和时间效率。完整的训练参数、评估细节以及 LAP 定义可以在技术附录中查看。

结果

我们的实验表明，MS-GRPO 能够成功提升 LLM 的连续决策能力。本节突出了几个关键发现：后训练增强了在训练环境中的表现，但在 Snake 上存在较高的方差；我们的 3B 后训练模型在冰冻湖上优于 72B 基线；我们的 AAW 策略显示出在不牺牲时间效率的情况下提升表现的迹象。另外，我们发现由 DQN 训练的基线远远优于我们的代理，并观察到泛化能力的混合证据。

使用 MS-GRPO 进行后训练改进顺序决策

使用 MS-GRPO 进行训练后，可以显著提升代理在各自训练环境中的性能，这在图 1 的学习曲线的上升趋势中得到体现。如表 1 所示，两种代理在各自的任务上都有所提升，但在 Snake 训练的代理的最终性能上存在很高的变异性。例如，Snake 训练的代理的最佳运行在 Snake-Standard 评估中获得了 0.45 的奖励，远高于平均值 −1.49 ，这表明训练结果存在很大的差异。这种差异表明训练过程对初始条件或早期探索较为敏感，有些代理收敛于有效策略，而其他代理则停滞不前。

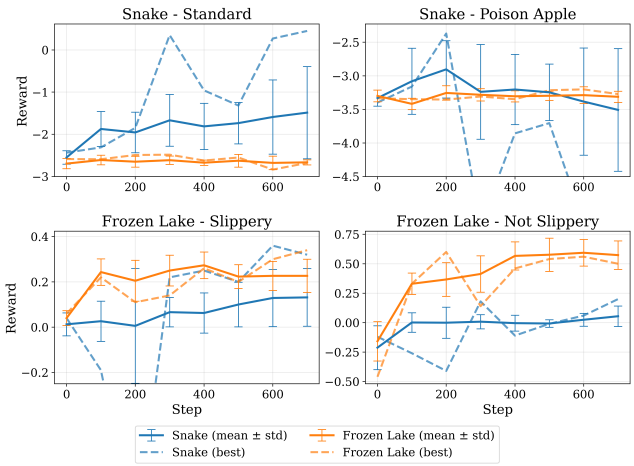

Figure 1: 在四种评估场景中的 700 步训练过程进展。每个面板中的实线表示 8 次独立训练运行的平均奖励，误差线表示标准偏差。虚线表示单次表现最佳的运行性能。在每一步时，性能是相同 50 个评估阶段的平均奖励。

后训练模型优于更大的基线模型

直接比较表明，我们的后训练方法使 3B 参数模型能够超越其更大的对应模型。正如图 2 所示，我们的 Frozen Lake 后训练代理在其训练环境 FrozenLake-NotSlippery 中达到了 0.57 ± 0.12 ，超过了 72B 参数模型所达到的 0.38 ± 0.48 ，尽管其操作限制为 200 个标记，而基线限制为 4096 个标记。

Table 1: 在初始和最终训练步骤中跨环境的性能比较，并提供每次运行的差异统计。数值显示为评估环境奖励的平均值（标准差）。

Training	Evaluation	Initial - 0	Final - 700	Δ
Snake	Snake - Standard	-2.607 (0.162)	-1.487 (1.093)	+1.120 (1.001)
	Snake - Poison Apple	-3.298 (0.126)	-3.508 (0.913)	-0.210 (0.827)
	Frozen Lake - Slippery	0.020 (0.035)	0.131 (0.127)	+0.111 (0.114)
	Frozen Lake - Not Slippery	-0.207 (0.213)	0.054 (0.087)	+0.261 (0.203)
Frozen Lake	Snake - Standard	-2.696 (0.120)	-2.665 (0.061)	+0.030 (0.143)
	Snake - Poison Apple	-3.299 (0.086)	-3.312 (0.083)	-0.013 (0.049)
	Frozen Lake - Slippery	0.040 (0.033)	0.227 (0.073)	+0.187 (0.059)
	Frozen Lake - Not Slippery	-0.158 (0.167)	0.573 (0.121)	+0.732 (0.201)

另一方面，我们后训练的 Snake 代理的平均奖励未表现出比较大的 LLM 明显的改善。然而，单个表现最佳的 Snake-Standard 代理在 Snake-Standard 上实现了 0.45 的最终奖励，相比于 72B 模型的 -1.26 ± 1.80 ，在 FrozenLake-Slippery 上则实现了 0.32 的最终奖励，相比于 0.094 ± 0.29 。

这些发现表明，对于顺序决策任务，任务特定的后期训练可能比扩大模型规模更加实用和高效。

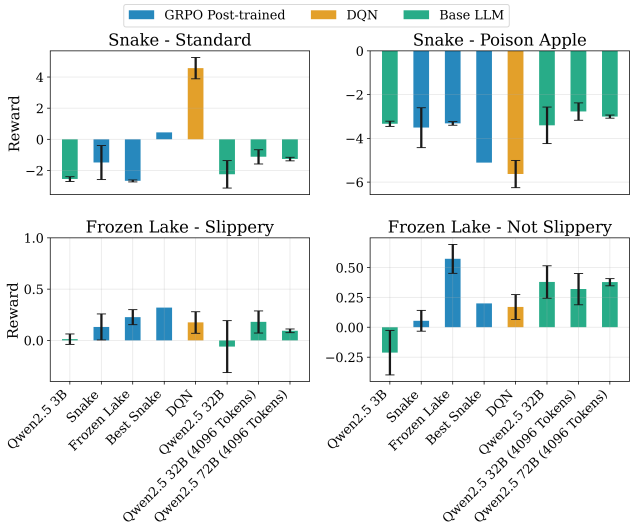

Figure 2: MS-GRPO 后训练代理和基线在四个评估环境中的每集平均奖励。MS-GRPO 结果是基于 8 次训练运行的平均值，每次评估 50 集。而基线是基于 1,000 集进行评估。误差条表示 MS-GRPO 代理在运行之间的标准差和基线的 95% 置信区间。

DQN 在域内任务上表现优于 MS-GRPO

DQN 代理在 Snake-Standard 上显著优于甚至最佳的单一 MS-GRPO，如图 2 所示。它实现了 4.58 ± 2.47 ，而最佳的 MS-GRPO 蛇代理实现了 0.45 ，并且所有蛇代理的平均值是 -1.49 ± 1.09 。这个性能差距突出显示了在特定任务中使用通用语言模型的挑战，这些任务在预训练数据中表现较差。要最大化特定任务的表现，一个专门的模型是更优的。

经后训练的 MS-GRPO 蛇形代理可以推广到未见过的冰湖环境

表现最好的 Snake 训练代理在未见过的任务上展示了有前途的零样本泛化能力，在 FrozenLake-Slippery 任务中获得了比 DQN 代理（0.17 ± 0.38 ）更高的平均奖励（0.32 ）。这表明尽管 MS-GRPO 代理在 Snake 环境中的表现明显较差，但它善于适应新颖的动态。

然而，同样的 Snake 智能体在 Snake-PoisonApple 任务上的泛化性能在后期训练后下降，变得比其来源的基础模型差得多（图 2 ）。这表明，该智能体学习到的寻找苹果的行为无法通过提示中说明它们有毒的指令来抵消。

AAW 采样提高了性能且不影响训练时间

我们发现，我们的 AAW 采样策略在保持或提高性能的同时减少了训练时间。如图 3 所示，与未采样的基线（$G = 100$ ，$G' = 100$ ）相比，在 700 步中使用 $G = 100$ 和 $G' = 25$ 进行训练节省了 3.5 倍的时间，同时实现了可比的最终奖励（-0.72 与 -0.86 ）。同时，通过在固定数量的训练片段中生成更多的片段，我们看到了更高的奖励：-0.72 与 $G = 100$ 和 $G' = 25$ 相比，-1.00 与 $G = 25$ 和 $G' = 25$ 。这表明，在 Snake 环境中进行训练时，我们的采样策略成功地选择了更高质量的片段，而没有显著增加计算负担。

这些结果表明，对于我们特定的环境和模型，使用 AAW 可以提高效率和性能。

讨论

我们的实验结果表明，MS-GRPO 能够成功地在顺序决策任务上对语言模型进行后训练。然而，结果也揭示了训练一致性上的相当大限制以及与专门设计的 DQN 智能体相比显著的性能差距。最佳的 Snake 智能体与平均水平之间的巨大性能差异既突显了这种训练方法的潜力，也说明了提高其一致性的重要性。

我们假设，这种不一致性源于训练期间探索不足，在这种情况下，找到有效策略过多依赖于初始条件和运气。不像传统 RL 代理直接通过状态-动作空间探索，并借助于如 ϵ -贪婪采样等方法，LAP 代理的探索是通过在标记空间的探索间接得到的结果。我们的 LAP 框架通过定义可以用来控制代理行为的不同组成部分，将这个问题明确化。除了 LLM 参数之外，代理的生成配置 $\mathcal{G}_i$ 和提示模板 $\mathcal{T}_i$ 提供了影响 LAP 采取行动的方式。例如，动态地调整 $\mathcal{G}_i$ 中的文本采样温度，当响应或奖

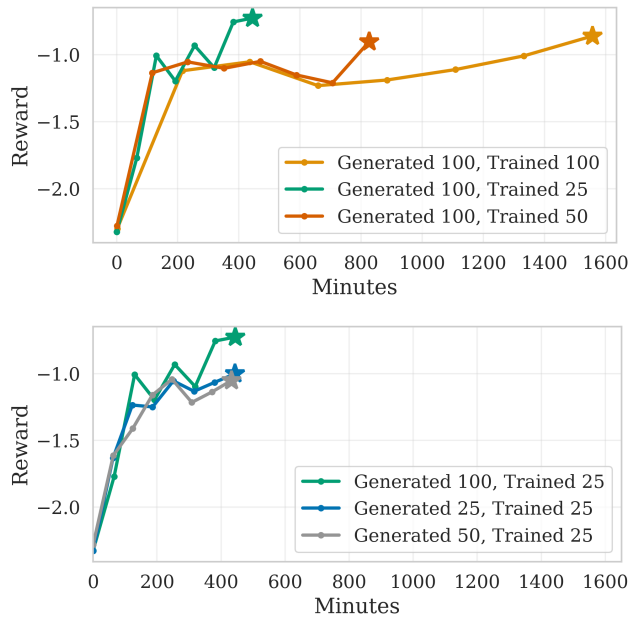

Figure 3: 关于收敛性和训练效率的消融研究，采用不同程度的 AAW 采样，展示了在 Snake-Standard 评估任务中不同采样配置下平均奖励与墙时间的关系。(顶部) 在固定生成次数 $G = 100$ 的情况下，改变采样的剧集数量 G' 。(底部) 在固定采样剧集数量 $G' = 25$ 的情况下，改变生成的剧集数量。

.

励停滞不前时增加温度，可能使学习算法适应，从而在训练期间文本生成不会变得过于一致，这是环境探索的前提条件。或者，用多样化的提示模板集（$\mathcal{T}_i$ ）对代理进行训练，以引出多种行为，也可能促进对环境的更加彻底的探索。

不一致训练的另一个可能的原因是使用不精确的蒙特卡洛信用分配方案，这可能会稀释来自代理探索的真正有效行为的学习信号。

此外，表现最佳的贪吃蛇代理在 Snake-PoisonApple 任务中的失败突显了后训练过程中的一个风险：强化特定技能可能会阻碍模型处理与该技能相关的重要语义细节。虽然训练成功提升了代理寻找苹果的能力，但它无法正确调整其行为以适应描述为有毒苹果的场景，导致性能相较于基础模型有所下降。

我们的 AAW 采样方法在不牺牲时间效率的情况下显示了有希望的性能提升（使用 $G = 100$ 的所有三个实验都优于那些使用较小 G 的实验）。然而，这些提升是温和的，仍需在更多环境中进一步验证。

最后，较大型 LLM 的性能提升证明了针对任务进行小型 LLM 后训练的价值。使用较小的模型生成更少的 tokens 可以降低计算需求并改善响应时间，使模型在实际应用中更具实用性。然而，与专门的 DQN 代理相比的性能差距突显了这种方法的基本局限性。即使在基础模型上明显有所改进并表现出泛化迹象，代理在一个狭窄、明确的任务上的绝对性能仍不及简单的专门替代方案。这表明，基于 LLM 的代理的价值可能不在于其超越专门代理的能力，而在于其灵活性，能够处理代理在现实世界场景中可能遇到的各种情况。

结论

在这项工作中，我们研究了小型 LLM 的决策能力是否可以在不依赖于广泛推理链的情况下为连续决策任务而改进。为此，我们引入了多步组相对策略优化（MS-GRPO）后训练算法。我们的实验展示了这种方法的有效性：一个经过后训练的 3B 参数模型优于 72B 参数的基线，表明有针对性的训练可以成为比单纯放大模型规模更有效的能力提升途径。此外，我们测试了一种选择性回合采样策略，发现其有迹象表明可以在不影响训练时间效率的情况下改善任务表现。这项工作确立了一种为创造更加高效和实用的基于 LLM 的决策代理的方法论。

我们的研究结果指出了未来工作的两个关键方向。首先，我们使用简单的蒙特卡洛信用分配机制可能导致观察到的训练不一致性。探索更精细的方法有助于提供更准确的学习信号以提高性能。其次，尽管我们的代理展示了对新环境动态的零样本泛化能力，但在语义上简单的毒苹果情境中同时失败，突显了一个关键挑战：确保训练后不覆盖模型的核心语义推理能力。解决这些挑战对于实现不仅高效而且具有鲁棒性和适应性的实用 LLM-代理至关重要。

References

Bani-Harouni, D.; Pellegrini, C.; Özsoy, E.; Keicher, M.; and Navab, N. 2025. Language Agents for Hypothesis-driven Clinical Decision Making with Reinforcement Learning. ArXiv:2506.13474 [cs].

Bernard, D.; Dorais, G.; Fry, C.; Gamble, E.; Kanefsky, B.; Kurien, J.; Millar, W.; et al. 1998. Design of the Remote Agent experiment for spacecraft autonomy. In 1998 IEEE Aerospace Conference Proceedings (Cat. No.98TH8339), volume 2, 259–281 vol.2. ISSN: 1095-323X.

DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. ArXiv:2501.12948 [cs].

Evans, R.; and Gao, J. 2016. DeepMind AI Reduces Google Data Centre Cooling Bill by 40%. Accessed: 2025-08-02. https://deepmind.google/discover/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-by-40/.

Georgeff, M. 1988. A Theory of Action for MultiAgent Planning. In Bond, A. H.; and Gasser, L., eds., Readings in Distributed Artificial Intelligence, 205–209. Morgan Kaufmann. ISBN 978-0-934613-63-7.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In Proceedings of the 19th national conference on Artifical intelligence, AAAI'04, 709–715. San Jose, California: AAAI Press. ISBN 978-0-262-51183-4.

Hou, Z.; Lv, X.; Lu, R.; Zhang, J.; Li, Y.; Yao, Z.; Li, J.; et al. 2025. T1: Advancing Language Model Reasoning through Reinforcement Learning and Inference Scaling. ArXiv:2501.11651 [cs].

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L.; and Murthy, A. 2024. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In Proceedings of the 41st International Conference on Machine Learning, volume 235 of ICML'24, 22895–22907. Vienna, Austria: JMLR.org.

Kamradt, G. 2025. Snake Bench: Competitive Snake Game Simulation with LLMs. Accessed: 2025-08-02. https://github.com/gkamradt/SnakeBench.

Kool, W.; Van Hoof, H.; and Welling, M. 2019. Attention, learn to solve routing problems! In 7th International Conference on Learning Representations, ICLR 2019, May 6, 2019 - May 9, 2019, 7th International Conference on Learning Representations, ICLR 2019. New Orleans, LA, United states: International Conference on Learning Representations, ICLR. Compendex.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. Journal of Machine Learning Research, 17(Compendex). Publisher: Microtome Publishing.

Li, Z.; Wu, W.; Wang, Y.; Xu, Y.; Hunt, W.; and Stein, S. 2025. HMCF: A Human-in-the-loop Multi-Robot Collaboration Framework Based on Large Language Models. ArXiv:2505.00820 [cs].

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. ArXiv:1312.5602 [cs].

Mower, C. E.; Wan, Y.; Yu, H.; Grosnit, A.; Gonzalez-Billandon, J.; Zimmer, M.; Wang, J.; et al. 2024. ROS-LLM: A ROS framework for embodied AI with task feedback and structured reasoning. ArXiv:2406.19741 [cs].

Murphy, S. A. 2003. Optimal dynamic treatment regimes. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 65(2): 331–355.

OpenAI; Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; et al. 2024. GPT-4o System Card. ArXiv:2410.21276 [cs].

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; et al. 2022. Training language models to follow instructions with human feedback. In Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, 27730–27744. Red Hook, NY, USA: Curran Associates Inc. ISBN 978-1-7138-7108-8.

Park, C.; Han, S.; Guo, X.; Ozdaglar, A. E.; Zhang, K.; and Kim, J.-K. 2025. MAPoRL: Multi-Agent Post-Co-Training for Collaborative Large Language Models with Reinforcement Learning. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 30215–30248. Vienna, Austria: Association for Computational Linguistics. ISBN 979-8-89176-251-0.

Qwen; Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; et al. 2025. Qwen2.5 Technical Report. ArXiv:2412.15115 [cs].

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Advances in Neural Information Processing Systems, 36: 53728–53741.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. ArXiv:1707.06347 [cs].

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; et al. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. ArXiv:2402.03300 [cs].

Shapley, L. S. 1953. Stochastic Games*. Proceedings of the National Academy of Sciences, 39(10): 1095–1100. Publisher: Proceedings of the National Academy of Sciences.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems, 36: 8634–8652.

Sutton, R. S.; and Barto, A. G. 2018. Reinforcement Learning: An Introduction. Cambridge, MA, USA: A Bradford Book. ISBN 978-0-262-03924-6.

Tanahashi, K.; Inoue, Y.; Yamaguchi, Y.; Yaginuma, H.; Shiotsuka, D.; Shimatani, H.; Iwamasa, K.; et al. 2023. Evaluation of Large Language Models for Decision Making in Autonomous Driving. ArXiv:2312.06351 [cs].

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; Cola, G. D.; Deleu, T.; Goulão, M.; et al. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. ArXiv:2407.17032 [cs].

Traum, D. R. 1999. Speech Acts for Dialogue Agents. In Wooldridge, M.; and Rao, A., eds., Foundations of Rational Agency, 169–201. Dordrecht: Springer Netherlands. ISBN 978-94-015-9204-8.

Trivedi, H.; Khot, T.; Hartmann, M.; Manku, R.; Dong, V.; Li, E.; Gupta, S.; et al. 2024. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 16022–16076. Bangkok, Thailand: Association for Computational Linguistics.

Wang, Y.; Yang, Q.; Zeng, Z.; Ren, L.; Liu, L.; Peng, B.; Cheng, H.; et al. 2025a. Reinforcement Learning for Reasoning in Large Language Models with One Training Example. ArXiv:2504.20571 [cs].

Wang, Z.; Wang, K.; Wang, Q.; Zhang, P.; Li, L.; Yang, Z.; Yu, K.; et al. 2025b. RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning. ArXiv:2504.20073 [cs].

Williams, R. J. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Mach. Learn., 8(3-4): 229–256.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. ArXiv:2210.03629 [cs].

Yu, Q.; Zhang, Z.; Zhu, R.; Yuan, Y.; Zuo, X.; Yue, Y.; Dai, W.; et al. 2025. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. ArXiv:2503.14476 [cs].

Zhang, H.; Du, W.; Shan, J.; Zhou, Q.; Du, Y.; Tenenbaum, J. B.; Shu, T.; and Gan, C. 2024. Building Cooperative Embodied Agents Modularly with Large Language Models. ArXiv:2307.02485 [cs].

Zheng, C.; Liu, S.; Li, M.; Chen, X.-H.; Yu, B.; Gao, C.; Dang, K.; et al. 2025a. Group Sequence Policy Optimization. ArXiv:2507.18071 [cs].

Zheng, Y.; Fu, D.; Hu, X.; Cai, X.; Ye, L.; Lu, P.; and Liu, P. 2025b. DeepResearcher: Scaling Deep Research via Reinforcement Learning in Real-world Environments. ArXiv:2504.03160 [cs].

Zhong, H.; Shan, Z.; Feng, G.; Xiong, W.; Cheng, X.; Zhao, L.; He, D.; et al. 2025. DPO Meets PPO: Reinforced Token Optimization for RLHF. ArXiv:2404.18922 [cs].

Zhou, R.; Du, S.; and Li, B. 2024. Reflect-RL: Two-Player Online RL Fine-Tuning for LMs. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 995–1015. Bangkok, Thailand: Association for Computational Linguistics.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G. 2020. Fine-Tuning Language Models from Human Preferences. ArXiv:1909.08593 [cs].

Technical Appendix

Agents

LAP Agents

We used *Qwen2.5-3B-Instruct* as the base model for post-training in all experiments, using Low-Rank Adaptation (LoRA) targeting parameter updates on all linear layers (Hu et al., 2021). LAP definitions for this and the comparative base LLMs are provided in Table 1. All agents have the same template $\mathcal{T}$ and action parser ψ.

Table 1: LAP definitions for the agents used in the experiments. The prompt template $\mathcal{T}$ and action parser ψ are the same for all agents.

Agent	LLM ($\mathcal{L}_\theta$)	Generation Configuration ($\mathcal{G}$)
$\Pi_{\text{MS-GRPO}}$	*Qwen2.5-3B-Instruct* + MS-GRPO LoRA updates	**Training:** Temp=1.5, top-k=3, tokens=200 **Evaluation:** Greedy (Temp=0), tokens=200
Π_{3B}	*Qwen2.5-3B-Instruct*	Greedy (Temp=0), tokens=200
Π_{32B}	*Qwen2.5-32B-Instruct*	Greedy (Temp=0), tokens=200
$\Pi_{\text{72B-200}}$	*Qwen2.5-72B-Instruct*	Greedy (Temp=0), tokens=200
$\Pi_{\text{72B-4096}}$	*Qwen2.5-72B-Instruct*	Greedy (Temp=0), tokens=4096

Action Parser ψ An action string is extracted from the LLM's response at each timestep. The action parsing function extracts the text inside the first set of < action > $\cdots$ < /action > tags. Each environment implementation must define a mapping from text string to action index. The action index corresponding to the extracted string is used as the agent's action in the next timestep. If there is not a valid pair of action tags, or the extracted string is not in the mapping, no action is taken.

Template $\mathcal{T}$ The LAP agent template, presented below, is shared for all LAP agents used in this study. There are two slots into which the observation is inserted. The static part of the observation, containing environment specific rules, is inserted into `{environment_prompt}`. The dynamic part of the observation, containing a combination of dynamic text describing the environment state and static template explaining the representation of the state, is inserted into `{game_state}`.

LAP Agent Template $\mathcal{T}$

```
<|im_start|>system
You're a helpful assistant. You always respond by wrapping your thoughts in the
    ↪ correct XML tags. Max response length: 200 words (tokens).
<|im_end|>
<|im_start|>user
{environment_prompt}
[Game State]
{game_state}
<|im_end|>
<|im_start|>assistant
Respond using ONLY valid XML with <observe>...</observe>, <think>...</think>, <
    ↪ plan>...</plan>, and <action>...</action> tags. Stop responding after the
    ↪  </action> tag.
[Response Template]
<observe>{Describe the situation concisely}</observe>
<think>{Think about the situation - what you should aim to do and what you
    ↪ should avoid doing.}</think>
<plan>{Describe the immediate plan you will follow to achieve your goal and
    ↪ avoid bad outcomes. Be explicit about the actions you will take: name the
    ↪  actions.}</plan>
<action>{Up/Down/Left/Right || Up/Down/Left/Right...}</action>
<|im_end|>
<|im_start|>assistant
```

DQN Agent

We trained a Deep Q-Network (Mnih et al., 2013) to act as a comparison in our experiments. We used a convolutional neural network with architecture detailed in Table 2, using a Rectified Linear Unit (ReLU) activation function after each convolutional and hidden fully-connected layer. We also performed a parameter sweep to determine the training hyperparameters. The search space is as follows:

Learning Rate (α) over $\{10^{-5}, 10^{-4}, 10^{-3}\}$

Discount Factor (γ) over $\{0.9, 0.95, 0.99\}$

ϵ-greedy Decay Steps over $\{5 \times 10^3, 2 \times 10^4, 10^5, 5 \times 10^5, 10^6\}$

The model presented for comparison in the Results, as determined by the greatest evaluation environment reward, used $\alpha = 10^{-5}$, $\gamma = 0.9$, $\epsilon_{\text{decay}} = 10^5$. The following hyperparameters were used for all DQN training experiments: initial $\epsilon = 1.0$, final $\epsilon = 0.1$, replay buffer size of 10^4, batch size of 128, and a target network update frequency of 1,000 steps. All experiments used 6 million training episodes.

Experiment Configuration

Training Protocol

We trained two types of agent: one exclusively on the *Snake-Standard* environment and another on *FrozenLake-Standard*. When training, we limited the number of episode steps to 10. We repeated training 8 times with different random seeds. All agents were then evaluated on all four environment variants.

Training was conducted for 700 steps of Algorithm 1 from the Methodology. We generated a group of $G = 100$ episodes and sampled $G' = 25$ from those, with an episode sampling temperature of

Table 2: Overview of the DQN architecture structure. Input shape is (B, C, H, W), where B is the batch size, C is the number of input channels, H is the height of the environment grid and W is its width. A is the number of discrete actions.

Layer Block	Layer Type	Parameters / Details	Output Shape
Input	-	-	(B, C, H, W)
Conv 1	Conv2d	32 filters, kernel 3x3, stride 1, pad 1	$(B, 32, H, W)$
	ReLU	-	$(B, 32, H, W)$
Conv 2	Conv2d	64 filters, kernel 3x3, stride 1, pad 1	$(B, 64, H, W)$
	ReLU	-	$(B, 64, H, W)$
Conv 3	Conv2d	64 filters, kernel 3x3, stride 1, pad 1	$(B, 64, H, W)$
	ReLU	-	$(B, 64, H, W)$
Flatten	-	-	$(B, 64 \times H \times W)$
FC 1	Linear	512 output units	$(B, 512)$
	ReLU	-	$(B, 512)$
FC 2 (Output)	Linear	A output units	(B, A)

$T_{\text{ep}} = 0.1$. In each episode, a maximum of 5 LAP actions (each of which can specify multiple sequential environment actions) and 10 environment steps was allowed. The LLM sampling temperature was 1.5 and used top-k sampling with $k = 3$. Other hyperparameters for MS-GRPO were a learning rate of 1×10^{-4}, clipping values of $\epsilon_{\text{low}} = \epsilon_{\text{up}} = 0.1$ and a KL-penalty weight of $\beta = 0.1$.

FrozenLake-NotSlippery | **Static** Observation Text | {environment_prompt}

```
You are navigating the surface of a frozen lake. You must reach the goal.
Rules:
If you step on a hole, you will fall through and die.
Your available actions are: Up, Down, Left, Right. You can make between 1 and 3
    ↪ actions, separated by the action separator " || "
```

FrozenLake-NotSlippery | **Dynamic** Observation Text Example | {game_state}

```
The board size is 4x4. Normal (X, Y) coordinates are used ranging from.
LEFT decreases X, RIGHT increases X, UP increases Y, and DOWN decreases Y.
Coordinates range from (0, 0) at bottom left to (3, 3) at top right.
Player position: (0, 3)
Holes: (1, 3), (2, 3), (3, 3), (3, 2)
Goal: (3, 0)
The meaning of each symbol in the state is:
- P: Player
- O: Hole
- G: Goal
- _: Empty space
State:
P O O O
_ _ _ O
_ _ _ _
_ _ _ G
```

Snake-Standard | **Static** Observation Text | {environment_prompt}

```
You are controlling a snake in a multi-player Snake game
Rules:
- You can move your head one space up, down, left, or right
- If you move onto an apple, you get 1 point and you gain a body segment
- You die if you move into a wall, another snake, or yourself
Your available actions are: Up, Down, Left, Right. You can make between 1 and 3
    ↪ actions, separated by the action separator " || "
```

Snake-Standard | **Dynamic** Observation Text Example | {game_state}

```
The board size is 7x7. Normal (X, Y) coordinates are used to denote positions.
LEFT decreases X, RIGHT increases X, UP increases Y, and DOWN decreases Y.
Coordinates range from (0, 0) at bottom left to (6, 6) at top right.
Apples at: (6, 2), (5, 3), (2, 6), (4, 5), (2, 3) (worth 1 points each)
Enemy snakes positions:
* Snake ID 2 has head at position (0, 0) and body segments at []
Your snake head (ID 1) is positioned at (6, 5) and body segments at []
You are controlling the snake at (6, 5)
The meaning of each symbol in the state is:
- 1: Your snake head
- 2: Enemy snake head
- T: Snake body
- A: Apple
- _: Empty space
State:
_ _ A _ _ _ _
_ _ _ _ A _ 1
_ _ _ _ _ _ _
_ _ A _ _ A _
_ _ _ _ _ _ A
_ _ _ _ _ _ _
2 _ _ _ _ _ _
```

Evaluation Protocol

To provide a comparison for the performance of the post-trained models, we used two larger models without MS-GRPO post-training, *Qwen2.5-32B-Instruct* and *Qwen2.5-72B-Instruct*. For the 72B parameter model, we evaluated once with a maximum number of generated tokens equal to that of the post-trained models (200) and once with a much greater limit (4096), providing both a like-for-like comparison as well as a measure of the model's full capability. Environment variant specific settings are detailed in the Experimental Setup section of the paper. The dynamic observation texts are identical to those in the Training Protocol above. The static observation texts for *Snake-PoisonApple* and *FrozenLake-Slippery* are detailed below. During evaluation, a longer episode of 20 LAP actions and environment steps was allowed to better assess long term performance. We used greedy decoding (text generation sampling temperature= 0). For consistency, the evaluation configuration file seed is set to 0 for all experiments, ensuring that we always use the same set of randomly generated initial conditions for evaluation.

Snake-PoisonApple | **Static** Observation Text | {environment_prompt}

```
You are controlling a snake in a multi-player Snake game
Rules:
- You can move your head one space up, down, left, or right
- If you move onto an apple, you *lose* 1 point. You must avoid the apples for
    ↪ as long as possible.
- You die if you move into a wall, another snake, or yourself
Your available actions are: Up, Down, Left, Right. You can make between 1 and 3
    ↪ actions, separated by the action separator " || "
```

FrozenLake-Slippery | **Static** Observation Text | {environment_prompt}

```
You are navigating the surface of a frozen lake. You must reach the goal. If you
    ↪  step on a hole, you will fall through and die. You may move in an
    ↪ unintended direction due to the slippery ice, including into a hole.
Your available actions are: Up, Down, Left, Right. You can make between 1 and 3
    ↪ actions, separated by the action separator " || "
```

Absolute-Advantage-Weighted (AAW) Episode Sampling Ablation Study

To evaluate the effectiveness of our AAW sampling strategy, we conducted two sets of experiments. First, we generated $G = 100$ episodes per training step and compared training on different subset sizes ($G' = 25$, 50 or 100 episodes). Additionally, we varied the total number of generated episodes ($G = 25$, 50, 100) and kept the training subset fixed at $G' = 25$ episodes. All subsets were sampled using $T_{\text{ep}} = 0.1$. All experiments used identical hardware, detailed in the Hardware section below. Other training settings were identical to those used in Training Protocol.

Hyperparameter Selection

When analyzing the MS-GRPO algorithm, we performed parameter sweeps for LLM generation temperature in the range 0.2 to 2.0 and top-k in the range $k = 1$ to $k = 10$ and without a limit. We determined the selected values for our experiments based on the evaluation reward on *Snake-Standard* for agents trained on Snake. We found the combination Temperature= 1.5 and $k = 3$ to give the best mean evaluation reward over 3 runs. Similarly, we used the results from the episode sampling study to determine which values of G and G' to use, finding that $G = 100$ with $G' = 25$ gave the best combination of training time efficiency and evaluation reward on *Snake-Standard*.

Hardware

All training and evaluation was performed on one of two types of hardware:

- *NVIDIA A100-SXM4-80GB* graphics card with AMD EPYC 7413 24-Core rocessor
- *NVIDIA H100 80GB HBM3* graphics card with Intel Xeon Platinum 8468 48 Core processor

Both setups use *Red Hat Enterprise Linux 8.9 (Ootpa)*. MS-GRPO training used a single GPU of either configuration. Evaluation of Π_{32B} used 2 H100 GPUs, linked by NVLink, and Π_{72B} and $\Pi_{72B\text{-}4096}$ used 4 H100 GPUs, linked by 2x NVLink and combined with 1x NVSwitch.

All training and evaluation for the AAW sampling experiments used the H100 configuration.

References

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. ArXiv:2106.09685 [cs].

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. ArXiv:1312.5602 [cs].