真正的知识来自实践：通过强化学习使大模型对齐具体环境

摘要

近年来，大型语言模型（LLMs）在自然语言生成与复杂推理方面取得了显著进展，展现出强大的语义理解和决策能力。然而，在动态环境中，LLMs 在生成有效策略、对齐环境规则以及提升决策效率方面仍面临诸多挑战。为解决这些问题，本文提出了 TWOSOME 框架，将 LLMs 的语义推理能力与强化学习（RL）的探索能力相结合，并引入 LoRA 参数高效微调和动作提示归一化（Token Normalization 和 Word Normalization）策略。该方法无需预先准备数据集，即可通过在线交互实现模型与环境的对齐，同时大幅降低计算成本。此外，本文引入 MC Dropout 技术，进一步量化模型的不确定性，优化策略的收敛性能，提高模型的稳定性，增强模型对目标的探索能力。

关键词：大型语言模型；强化学习；参数高效微调；动作提示归一化；MC Dropout

1 引言

近年来，大型语言模型（LLMs）在自然语言生成与理解方面取得了显著进展，并被成功应用于多模态任务以及复杂的环境交互中，具有强大的推理能力。然而，尽管这些模型在推理和计划能力上表现出色，但是它们在基于环境动态的决策任务中却面临显著的对齐问题，在决策任务中经常有失败的风险。而另一方面，强化学习（RL）方法是从随机策略开始的，根据不断地试错找到当前任务最优的策略，确保与环境的对齐，然而该过程的耗时较长，学习效率低，缺乏利用先验知识的能力，从而导致样本效率低下，在实时问题上并不是一个很好的选择。因此，如何结合 LLMs 的先验知识与 RL 的环境交互能力，成为了一个重要的研究方向，而通过结合这两者，既可以提升 RL 的学习效率，又能优化 LLMs 在决策环境中的对齐问题。

TWOSOME 框架通过结合 LLMs 生成行为策略与强化学习的 PPO（近端策略优化）进行参数高效微调，实现了无需预先准备数据集的环境对齐，显著提升了决策精度和样本效率，同时降低了计算成本。实验结果表明，TWOSOME 在多个任务环境中的任务成功率、泛化能力和样本效率均显著优于传统 RL 方法（如 PPO）和其他 Baseline 方法（如 SayCan），验证了其理论与实践的可行性。这一框架为结合 LLMs 与强化学习提供了通用在线解决方案，推动了 LLMs 在真实环境中的应用边界拓展，并通过参数高效微调克服了传统微调的高成本问题，展示了其在多任务环境中的强泛化能力，为实现更通用的自治代理提供了理论依据和实践支持。

2 相关工作

本部分对选题内容相关的工作进行简要分类概括与描述，重点从大语言模型（LLMs）与环境对齐、强化学习与环境交互、以及参数高效微调等方面展开论述。

2.1 基于 LLMs 的决策与推理

大语言模型（LLMs）近年来在自然语言生成与复杂推理方面取得了显著进展，并被应用于多模态任务和决策场景。然而，其在动态环境中的应用仍存在对齐问题，主要表现为：

- 动作无效性: 生成的动作可能与环境规则不符，如在环境未提供的物品上操作。
- 环境动态误解：对特定规则或动态转移缺乏准确理解，导致任务失败。

为了解决这些问题，研究者们提出了多种方案，例如基于提示设计的任务对齐方法，框架 SayCan [1] 通过提示引导 LLMs 生成合理的动作并评估可行性，或者基于环境反馈的对齐策略，框架 ReAct [2] 通过结合推理与动作优化提升任务完成率。这些方法在特定环境下表现出色，但过度依赖 LLMs 的本身的能力，对小型模型或语义复杂场景的适应性较差。相比之下，TWOSOME 能够在多任务环境中表现出更强的适应性和鲁棒性，并且通过参数高效微调（如 LoRA），显著降低了对 LLMs 的资源需求，适用于小型和中型模型。

2.2 参数高效微调技术

在使用 LLMs 进行任务优化时，参数高效微调（Parameter-Efficient Fine-tuning, PEFT）成为重要研究方向，相较于传统的全参数微调，PEFT 能够在保持模型核心能力的同时，以更低计算开销实现定制化任务优化。Lester 等人 [3] 提出了 Prompt Tuning，通过在输入或隐藏层前添加可调参数的 token 来优化模型，适合特定任务的调整，Houslby 等人 [4] 提出了 Adapter Tuning，通过插入适配器层来微调部分参数，减少了计算成本。然而，这两种方法的局限在于对复杂任务（例如环境交互或长时间规划）效果不佳，缺乏模型在动态任务中的适应性。Edward 等人 [5] 提出的 LoRA 方法通过低秩矩阵近似实现参数更新，进一步降低了微调成本。然而，LoRA 的主要缺陷是对任务环境的强假设依赖，尤其是在多模态任务或复杂环境中的泛化能力不足。不同于 Prompt Tuning 或 Adapter Tuning 仅适用于静态任务，TWOSOME 结合 LLM 的推理能力和 PPO 算法，能够适应多任务动态交互环境。并且，TWOSOME 展现了对复杂任务和多样化场景的强泛化能力，而 LoRA 等方法在这些方面仍存在明显不足。

3 本文方法

3.1 本文方法概述

TWOSOME 是一个结合大语言模型(LLMs)和强化学习(RL)的通用在线学习框架，旨在通过将 LLMs 部署为环境交互智能体生成有效动作，并提出两种正则化方法（Token Normalization 和 Word Normalization）以缓解动作提示长度对策略分布的影响。此外，TWOSOME 在 PPO 框架下设计了参数高效的训练方法，并总结了优化提示设计的四项原则，从而实现了 LLMs 与动态环境的高效对齐。其方法框架如图 1所示。

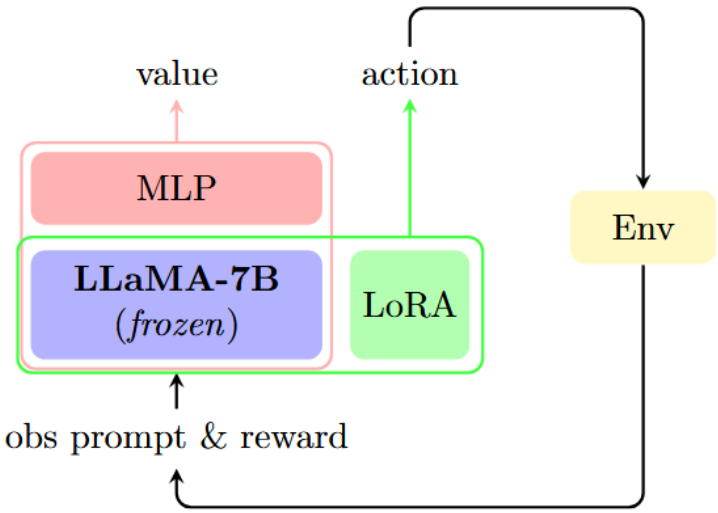

图 1. Twosome 框架

3.2 有效策略生成

TWOSOME 提出了一种有效策略生成方法，避免了 LLMs 直接生成无效动作的风险。通过对所有可能动作的分数进行查询，该方法确定了动作执行的概率，并确保策略生成的合法性。

具体而言，TWOSOME 将观察提示 (observation prompt) 和动作提示 (action prompt) 结合，形成输入序列供 LLMs 使用。动作提示 a_k 是由一系列 token 组成的语义化描述，而观察提示词 s 则代表当前环境状态。LLMs 利用此输入序列计算动作提示的联合概率 $P_{token}(a_k|s)$，公式如下：

$$P_{token}(a_k|s) = \prod_{i=1}^{N_k} P(w_i^k|s, w_1^k, ..., w_{i-1}^k)$$

其中 w_i^k 是动作提示词 a_k 的第 i 个 token,N_k 表示动作提示词的 token 数。接下来，通过 softmax 函数对所有概率进行归一化，以获得最终的策略分布：

$$P(a_k \mid s) = \frac{\exp(\log P_{\text{token}}(a_k \mid s))}{\sum_{a \in \mathcal{A}} \exp(\log P_{\text{token}}(a \mid s))}$$

这种方法有两个显著优点，一是生成的策略在环境中始终是有效的，符合动作执行的约束，二是充分利用 LLMs 的语义组合能力，将其适配于扩展的动作语义空间，提高了策略生成的灵活性和适用性。

3.3 动作提示归一化

在动作提示生成方法中，由于每个 token 的概率 $P(w_i^k|\cdot)$ 总是小于 1，较长的动作提示会倾向于具有较低的 token-level 概率。这种现象可能导致较短的动作提示更容易被选择，即使较长的动作提示在当前环境下更为合理。例如，动作提示 “pick up the tomato” 比 “serve the dish” 包含更多的 token，但在某些场景下是更优的动作。为了解决这一问题，TWOSOME 提出了两种归一化技术：

- Token Normalization
  通过动作提示的 token 数 N_k 对联合概率 $P_{token}(a_k|s)$ 进行归一化，公式如下：

$$\log P^{\text{tn}}_{\text{token}}(a_k \mid s) = \frac{\log P_{\text{token}}(a_k \mid s)}{N_k}$$

  这种方法确保了较长动作提示的 token-level 概率能够被拉回到与较短提示相似的量级。
- Word Normalization
  通过动作提示的单词数 W_k 对联合概率 $P_{token}(a_k|s)$ 进行归一化，公式如下：

$$\log P^{\text{wn}}_{\text{token}}(a_k \mid s) = \frac{\log P_{\text{token}}(a_k \mid s)}{W_k}$$

  相比 Token Normalization，Word Normalization 更关注语义完整性，避免将每个单词分割成多个 token 造成的额外影响。例如，动作提示 “pick up the tomato” 中 $N_k = 5$，而 $W_k = 4$。

Token Normalization 能够有效消除动作提示长度对概率分布的影响，但有时过于极端。例如，如果一个单词被分割成多个 token，首个 token 的概率可能相对较低，而后续 token 的概率接近 100%，从而影响模型的决策合理性。Word Normalization 则将每个单词视为独立的语义单元，使归一化过程更贴合实际情况。实验表明，Word Normalization 在捕捉动作提示的语义完整性方面表现更优，适合在复杂语义环境下使用。

3.4 参数高效的 PPO 微调

3.4.1 框架

如图 1所示，TWOSOME 的参数高效微调方法基于 LLaMA-7B 模型，结合了多层感知机（MLP）和 LoRA（低秩适配器）模块，用于高效对齐 LLMs 与环境。具体设计如下：

- Critic：在 LLaMA-7B 模型的最后一个 Transformer Block 上附加额外的 MLP 层。该 MLP 以观察提示（observation prompt）的最后一个 token 作为输入，输出估计的状态价值（value）。
- Actor：采用参数冻结的 LLaMA-7B 模型，并通过 LoRA 进行参数微调。Actor 的 LoRA 参数初始化为零，确保初始输出与原始 LLaMA-7B 模型一致。

为了避免训练不稳定性，LoRA 模块中未使用 dropout，因为随机性可能违反 PPO 的 KL 散度约束。在整个训练过程中，仅更新 Critic 的 MLP 和 Actor 的 LoRA 参数，这种设计大幅提升了训练效率。

3.4.2 训练过程

训练阶段遵循 PPO 框架的标准流程，主要包括以下关键点：

- 样本数据使用：每次训练后，采样数据立即丢弃，以避免因多次更新同一数据导致的不稳定性。
- 学习率设置：Critic 部分的 MLP 参数随机初始化，采用较大的学习率以加速收敛。Actor 的 LoRA 参数初始化为零，采用较小的学习率以保证训练稳定性。
- 参数高效性：LoRA 参数量仅为原始模型参数的 1/20，极大降低了训练开销。

3.4.3 推理过程

在推理阶段，仅使用 Actor 模块，Critic 被完全丢弃。LoRA 参数编码了 LLMs 与环境对齐的信息，支持跨环境的泛化应用。这些参数具备即插即用的特性，使其在不同任务和场景中具有出色的迁移能力。

4 复现细节

4.1 与已有开源代码对比

所复现论文的作者并未将代码完全开源，仅给出了 PPO 参数微调训练部分的 Actor-Critic（AC）框架的搭建的代码，然而缺少了重要的进行任务部署推理部分的代码。本人根据文章中提到的推理模型框架，与所使用到的开源游戏环境 [6, 7]，补充了推理部分的代码，实现了模型在 Overcooked 与 Virtualhome 中 2 个游戏环境上的部署，并根据论文中 prompts 的设计与游戏实现细节，完成了 Actor 单独推理的部分，实现了获取 LLMs 作为智能体在游戏行为预测上的决策的功能。

此外，由于在 Overcooked 与 Virtualhome 这两个动态环境中，TWOSOME 方法并未对策略的不确定性进行评估，而仅仅是对策略的价值进行判断，这可能导致处于分布外状态下的决策被忽视，降低 Critic 的有效性。所以本人在现有的 Critic 模型中，加入了 MC Dropout 模块来引入不确定性量化，这可以帮助模型更好地处理分布外样本（out-of-distribution, OOD）以及提升策略的鲁棒性，提升 AC 框架的有效性。并且在一定程度上，可以减少由不确定性带来的训练过程抖动。

4.2 实验环境搭建

本实验在单个 NVIDIA Tesla A100 40GB GPU 上进行，使用 python3.9，pytorch1.13.1 与 cuda11.7 来配置实验环境。

4.3 推理部分的代码实现

这一部分代码实现流程如下：游戏环境创建–> 加载 LLaMa-7B-Lora 代理–> 开始游戏–> 获取代理预测的操作–> 通过操作直接在环境执行–> 对每一步操作的结果进行记录。本节中将会对两个环境的实现分别进行描述。

4.3.1 Overcooked

- 游戏环境搭建：根据开源的 Overcooked 环境，使用 gym 库设置该实验的实验参数，并生成实验环境。根据游戏任务，游戏关键参数设置如下：
  其中，grid_dim、n_agent 和 map_type 一同初始化游戏的棋盘，表示 1 个游戏代理在尺寸为 7×7 的 A 类型棋盘上进行操作，而游戏将会根据 obs_radius 的大小将代理操作后的观测值以矩阵的形式返回。在该环境下有两个任务，一个是切番茄，另一个是组装并切好生菜与番茄，根据每个任务会有不同的动作模式，如向左、向右、停留、拿起刀、拿起番茄、拿起生菜等普通操作。

Parameter	Value
grid_dim	[7, 7]
map_type	"A"
n_agent	1
obs_radius	2
mode	"vector"

表 1. Overcooked 环境参数

- 加载代理并开始游戏：加载训练好的 Lora 模块与冻结参数的 LLaMa-7B 模型，从环境初始观测值开始，智能体对每个观测值状态进行行为的预测，然后循环将预测的动作在游戏中执行，并获取新观测值再次预测，直到循环超过 200 次或者达到目标时停止。
- 根据观测值获取代理预测的操作：环境中已经给出了将观测值转变为 prompts 与 action 的文本表示的代码，其实现的内容是检查各物体与智能体的状态，生成描述性的语句。行为预测的实现过程为：根据每个提示和对应的动作组合，生成完整的输入文本序列–> 使用 tokenizer 将文本序列转换为张量，标识序列中每个 token 的编码和标识实际内容和填充部分–> 输入编码后的文本序列，通过 LLaMa-7B-Lora 代理生成 logits–> 解析动作序列与对应 logits，并进行归一化。此时就可以得到预测的操作了。

4.3.2 Virtualhome

- 游戏环境搭建：Virtualhome 环境的搭建过程与 Overcooked 类似，使用了 gym 库设置该实验的实验参数，并生成实验环境。但是 Virtualhome 可选择的场地只有一个，可选择的智能体数量也只有一个，所以并不需要对环境参数做其他设置。
  然而，virtualhome 是一个开源的 3D 游戏，它的地图场景更加复杂，物体交互也更变得更为多样，所以对智能体的挑战会更大。在该环境下有两个任务，一个是将食物放进微波炉加热，另一个是准备牛奶与薯片打开电视观看，根据每个任务会有不同的动作模式，如拿起物体，放下物体，发现物体，进入厨房，进入起居室等普通操作。
- 加载代理并开始游戏：与 Overcooked 一样，加载训练好的 Lora 模块与冻结参数的 LLaMa-7B 模型，并进行预测。
- 根据观测值获取代理预测的操作：虽然两个游戏环境不同，但是对游戏给出的观测值进行行为预测的过程是相同的，所以在得到观测值的描述性语句了以后，可以同样使用写好的操作预测函数进行操作的获取。

4.4 MC Dropout 不确定性量化

在强化学习中，价值函数的预测通常受到噪声或模型的不确定性影响，通过量化不确定性，可以更好地评估模型预测的可靠性，从而减少因预测误差导致的决策风险，同时也可以帮助平衡探索与利用，通过高不确定性指导策略更倾向于探索。在这里则是采用了 MC Dropout 方法量化模型计算过程中的不确定性。该方法通过在模型预测过程中多次启用 Dropout，实现 Monte Carlo 采样，以进一步让 Critic 学习分布并量化不确定性。

4.4.1 Dropout 模块的引入

在原来的架构中，Critic 在 LLaMa-7B 后加入了三个 MLP 层，逐步将提取的特征映射到隐藏维度并进行降维，最后使用 ReLU 激活函数增加非线性，最终输出一个单一的标量值作为价值估计。本实验则是在 MLP 第二层后增加 Dropout，通过在前向传播中随机丢弃部分神经元的激活值来模拟不同的模型行为，从而获得多样化的预测结果。

4.4.2 前向传播的调整

在原来的前向传播中，只是单一的对当前的状态进行一次价值估计，为当前状态分配一个数值。引入 MC Dropout 后，代码不仅可以输出状态的价值估计，还可以量化该估计值的不确定性。这为策略优化提供了更丰富的信息，可以动态调整对价值估计的信任度。使用多次使用 Monte Carlo 采样前向传播 [8]，每次随机丢弃神经元，通过 Dropout 模拟不同的模型行为，采样得到一组预测值计算均值与方差，其中输出均值作为价值估计，输出标准差表示不确定性。

4.4.3 基于不确定性的 PPO 优化调整

- 策略更新的采样与收集：环境中的状态、动作和奖励数据等采样数据是后续计算优势函数和优化策略的基础。在原来的方法中，价值估计只是单纯从 critic 输出的值，没有对其可信度进行权衡。

  $$\text{Adjusted Value} = V(s) - \text{Uncertainty Weight} \cdot \text{Uncertainty}(s)$$

  添加不确定性调整后，当不确定性较高时，会适当降低价值估计，使模型对高置信度预测赋予更高权重，此时的 Uncertainty Weight 设置为 0.1。
- 价值网络的更新：它是 PPO 算法的关键部分，用于通过优化价值损失函数，提升对状态价值的准确估计，从而为策略更新提供稳定的基准。价值网络的目标是估计当前状态 s 的价值 $V(s)$，即预测从当前状态开始所能获得的累计奖励的期望值。通过计算预测值 $V(s)$ 和目标值 $G(s)$ 的差距，最小化以下价值损失函数：

  $$L_v = \frac{1}{2}\left(V(s) - G(s)\right)^2$$

  引入 MC Dropout 后，在计算价值损失时对预测误差进行加权，修改后的未裁剪损失函数为：

  $$L_v = \frac{1}{2}\frac{(V(s) - G(s))^2}{1 + \text{Uncertainty}(s)}$$

  修改后的裁剪损失函数为：

  $$L_v = \frac{1}{2}\max\left(\frac{(V(s) - G(s))^2}{1 + \text{Uncertainty}(s)}, \frac{(\text{clip}(V(s)) - G(s))^2}{1 + \text{Uncertainty}(s)}\right)$$

  这种方法降低高不确定性对模型的影响，当模型预测的不确定性较高时（即 $\text{Uncertainty}(s)$ 较大），对应的误差被缩小，这降低了高噪声状态对模型的负面影响。同时也可以增加对高置信度预测的信任，当不确定性较低时（即模型对预测有较高的信心），误差几乎不受影响，模型仍会以较高权重学习这些状态。

5 实验结果分析

5.1 Word Normalization 与 Token Normalization 的比较

原论文中对 4 个游戏任务都进行了比较，如下图所示 2，然而在 Virtualhome 环境中进行食材准备模型的训练时，进行 15k 步就已经需要 1.5 小时，而论文中的步数已经达到 50k，其他的任务在达到收敛时已经执行 100k 步甚至还要多，由于算力资源有限，只对 Virtualhome 第一个任务进行归一化比较，并且步数设置为更小的 15k。

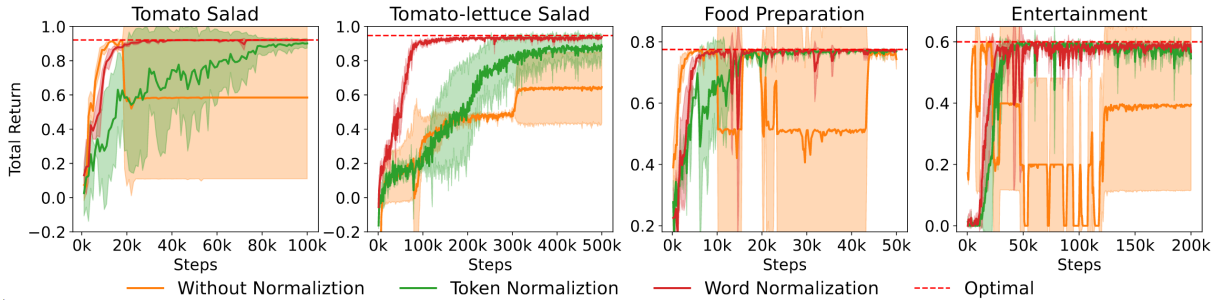

图 2. 原论文中 Word Normalization 与 Token Normalization 的比较

Virtualhome 的第一个任务是在房间内拿出食材放到微波炉中加热，从图中可以观察到，Token Normalization（绿色曲线）在训练的初始阶段展现了非常快速的收敛速度，其总回报迅速提升，并在较短的训练步数内达到接近稳定的水平。这表明 Token Normalization 在处理初期任务时具有较高的效率，能够快速学习环境中的策略。然而，随着训练的持续，Token Normalization 的表现趋于平稳，未能进一步提升总回报，表明其在后期的策略优化中可能受限，难以在复杂场景中进一步提升表现。

相较之下，Word Normalization（红色曲线）虽然在训练初期的总回报增长较为缓慢，但其整体曲线呈现出持续上升的趋势。特别是在中后期阶段，Word Normalization 的表现明显优于 Token Normalization，总回报显著更高。这说明 Word Normalization 在更长的时间内能够逐步优化策略，展现出更强的泛化能力和适应复杂任务的能力。

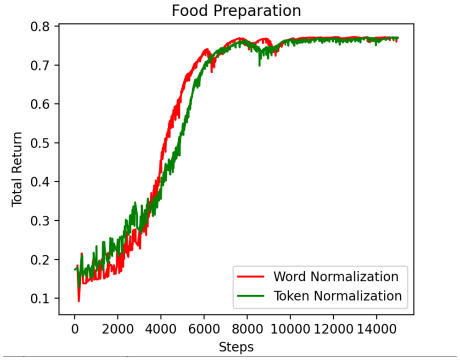

图 3. 复现的 Word Normalization 与 Token Normalization 的比较

从整体来看，Token Normalization 更适合那些需要快速收敛的任务，能够在短时间内学习到可接受的策略，适用于对实时性要求较高的场景。而 Word Normalization 则更适合对策略精度和长期表现要求较高的任务，其持续的优化能力使其在复杂的长期任务中表现更优。

这两种方法的差异为不同任务场景提供了选择依据，充分体现了 normalization 方法在强化学习策略设计中的重要性。并且，Token Normalization 与 Word Normalization 的走势与原论文中是一致的。

5.2 MC Dropout 的结果

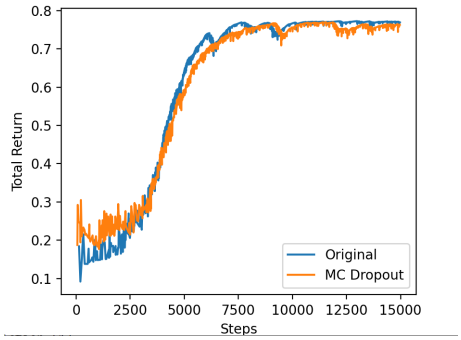

图 4. 原方法与引入 MC Dropout 方法的比较

这张图 4展示了在 Virtualhome 环境下，对第一个任务进行改进前后方法的训练对比，步数均设置为 15k。Original（蓝色曲线）与 MC Dropout（橙色曲线）两种方法在训练过程中总回报（Total Return）随步数（Steps）变化的对比情况。可以观察到，在训练的早期阶段，MC Dropout 方法的总回报增长速度明显快于原方法，表现出更高的样本效率。这种加速增长可能得益于 MC Dropout 能够通过量化模型预测的不确定性来调整策略，使其更快地聚焦于高价值的探索方向。在中期阶段，MC Dropout 仍然保持较高的总回报，相比原方法略有领先，表明其在策略优化过程中具有更高的稳定性和鲁棒性。同时，在这一阶段，原方法方法虽然也在持续提升总回报，但波动较大，可能反映出模型在探索过程中的不确定性较高或策略收敛性较差。

到训练后期，两种方法的总回报都逐渐趋于稳定，均接近于 0.8 的水平，表明在收敛性能上两者大致相当。然而，MC Dropout 方法的曲线波动显著较小，显示出更强的稳定性和对环境变化的适应能力。相比之下，原方法的曲线在部分区间依然存在显著波动，可能表明其策略在某些场景中缺乏稳健性。整体来看，MC Dropout 方法不仅加速了模型的学习过程，还有效提升了策略优化的稳定性和可靠性，为强化学习中需要快速探索和稳健优化的任务提供了一种更高效的解决方案。这一结果进一步说明，引入不确定性量化的策略调整方法在复杂决策任务中具有明显的优势。

这张表格 2展示了 Dropout 方法与原方法在两个任务中的性能对比，具体包括平均奖励（Reward Mean）、奖励标准差（Reward Std）、完成任务的平均步数（Step Mean）、步数标准差（Step Std）以及成功率（Success Rate）。其中对 overcooked 的 Tomato Salad 步数设置为 30k，Virtualhome 的 Food Preparation 步数设置为 15k。

无论是 Task 1: Food Preparation，还是 Task 2: Tomato Salad，Dropout 方法平均奖励为高于原方法，表明 Dropout 方法在奖励最大化方面更好一点。同时，Dropout 方法的奖励标准差低于原方法，表明 Dropout 方法的奖励波动更小，表现更加稳定。在完成任务的平均步数方面，Dropout 方法为相比原方法的更少，说明 Dropout 方法需要更少的步数即可完成

表 2. MC Dropout 和原方法的比较

Task	Method	Reward Mean	Reward Std	Step Mean	Step Std	Success Rate (%)
1	Dropout	0.7697	0.0187	6.11	0.51	100
	Original	0.7526	0.0560	6.61	1.74	100
2	Dropout	0.7195	0.1582	17.54	14.80	100
	Original	0.7110	0.1773	19.46	17.71	100

任务。此外，Dropout 方法的步数标准差为低于原方法的，进一步说明其在任务执行效率上的一致性更高。两种方法在该任务中的成功率均为 100%，表明两种方法都能够可靠完成任务。

综合来看，Dropout 方法在两个任务中的平均奖励均高于 Original 方法，且奖励和步数的标准差更低，表明其在任务执行的稳定性和效率上具有显著优势。尤其是对于需要减少任务执行时间和提升任务结果一致性的场景，Dropout 方法显示出了较高的实用价值。尽管成功率上两种方法均为 100%，但 Dropout 方法通过更高的奖励和更少的步数展现了其优越性。

6 总结与展望

这次研究中，成功复现出了作者的部分实验，虽然总体趋势相同，但是并不是完全相同，有一定的出入，达不到作者的精度，原因还有待分析。并且由于算力原因，部分实验没有实现，需要后续对该研究继续努力。提出的 MC Dropout 在两个任务中有所提升，但是稳定性还有很大的提升空间。

参考文献

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[3] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[4] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[5] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[6] Sarah A Wu, Rose E Wang, James A Evans, Joshua B Tenenbaum, David C Parkes, and Max Kleiman-Weiner. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432, 2021.

[7] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502, 2018.

[8] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Reinforcement learning in continuous action spaces through sequential monte carlo methods. *Advances in neural information processing systems*, 20, 2007.