浙江大學
ZHEJIANG UNIVERSITY

基于 GRPO 的坐标序列预测

自然语言处理课程项目

浙江大学计算机科学与技术学院

王　　阳（贡献度：35%）

周齐圣　（贡献度：25%）

刘　　涵（贡献度：20%）

郑思华　（贡献度：20%）

2025 年 6 月 15 日

目录

1 任务介绍

1.1 项目背景

近年来，强化学习在自然语言处理领域的应用取得了显著进展 [1]。与传统的监督学习不同，强化学习不需要直接给定标签进行学习，而是通过构造奖励函数来指导模型的学习方向。这种方法在特定任务中展现出了独特的优势。

本项目探索了将 GRPO（Generative Reward Processing and Optimization）算法 [4] 应用于坐标序列预测任务的可能性。GRPO 作为一种新型的强化学习算法，其特点是通过构造奖励函数来评估模型生成结果的合理性，从而优化模型的推理过程。在本项目中，我们将这一算法应用于预测符合特定运动规律的坐标序列，这不仅能验证算法的有效性，也能探索其在数值预测任务中的应用潜力。

1.2 项目目标

本项目的具体目标包括以下几个方面：

1.2.1 数据构建

构造符合一维匀速直线运动规律的坐标数据：

- **运动类型**：一维匀速直线运动，形如 $\{(x_1), (x_1 + v \cdot \Delta t), \ldots, (x_1 + v \cdot n\Delta t)\}$
- **数据特点**：
  - 初始位置 x_1 和速度 v 均为浮点数
  - 时间间隔 Δt 固定为 1 个单位时间
  - 序列长度为 10 个点（给定前 5 个点，预测后 5 个点）
- **数据要求**：构造具有不同速度范围的训练和测试数据，以验证模型的泛化能力

1.2.2 算法实现

基于以下技术栈实现坐标序列预测系统：

- **强化学习算法**：采用 GRPO[4]，基于 HuggingFace 的 open-r1[2] 实现
- **基础模型**：使用 Qwen2.5-1.5B-Instruct[3] 作为 backbone 模型
- **预测任务**：给定 5 个连续坐标点，预测后续 5 个坐标点的位置

1.2.3 实验验证

设计完整的实验验证方案：

- **数据划分**：
  - 训练集：50 组序列，速度范围在 [0.5, 2.0] 之间
  - 测试集：分为两类
    - In-distribution：20 组序列，速度范围与训练集相同
    - Out-of-distribution：20 组序列，速度范围在 [2.0, 3.0] 之间
- **预测要求**：模型需要通过分析给定的 5 个点，推理出运动规律，并准确预测后续 5 个点的位置

1.2.4 评估指标

主要从以下几个方面评估模型性能：

- **预测准确性**：评估模型预测的 5 个坐标点与真实值的误差
- **泛化能力**：比较模型在不同速度范围数据上的表现差异
- **推理过程**：分析模型的 chain-of-thought 能力，评估其对运动规律的理解

2 算法介绍

2.1 强化学习基础

强化学习是机器学习的一个重要分支，其核心思想是通过智能体（Agent）与环境（Environment）的交互来学习最优策略。在传统的监督学习中，模型通过已标注的数据直接学习输入到输出的映射。而在强化学习中，模型通过与环境的交互获得反馈（奖励），从而逐步优化其行为策略。

2.1.1 基本框架

强化学习的基本流程如图1所示，主要包含以下关键要素：

- **状态空间**（State Space）$\mathcal{S}$：描述环境在某一时刻的状态，在本项目中表示为已知的坐标序列
- **动作空间**（Action Space）$\mathcal{A}$：智能体可以采取的所有可能行动，在本项目中对应模型生成的预测序列

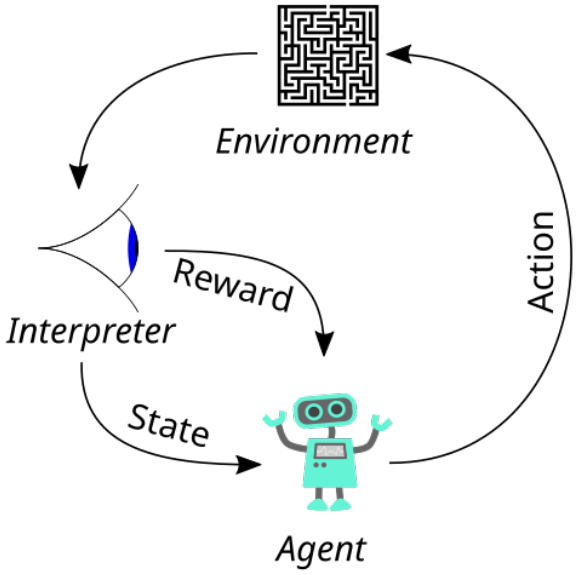


图 1: 强化学习基本框架

- **转移函数**（Transition Function）$P(s_{t+1}|s_t, a_t)$：描述在当前状态下采取某个动作后环境状态的变化
- **奖励函数**（Reward Function）$R(s_t, a_t)$：评估智能体在某个状态下采取特定动作的好坏，本项目中包含预测准确性和推理过程的评估
- **策略**（Policy）$\pi(a|s)$：决定智能体在特定状态下应该采取什么动作，在本项目中即为模型的生成策略

2.1.2 学习过程

强化学习的核心目标是最大化累积奖励，其学习过程主要包含以下步骤：

1. **状态观察**：智能体观察当前环境状态 s_t
2. **动作选择**：根据策略 $\pi(a|s)$ 选择动作 a_t
3. **环境交互**：执行动作并获得奖励 r_t 和新状态 s_{t+1}
4. **策略更新**：基于获得的奖励更新策略，优化目标为：

$$J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T}\gamma^t r_t] \tag{1}$$

其中 γ 为折扣因子，τ 表示交互轨迹

2.1.3 在 NLP 中的应用

在自然语言处理领域，强化学习的应用具有其特殊性：

- **状态表示**：通常是文本序列或模型的隐状态，需要考虑文本的语义和结构信息
- **动作空间**：往往是离散的词表或 token 空间，这使得动作空间非常大且稀疏
- **奖励设计**：需要考虑多个方面：
  - **任务相关**：如预测准确性、生成质量等
  - **语言特性**：如语法正确性、语义连贯性等
  - **推理能力**：如逻辑性、可解释性等
- **训练挑战**：
  - 奖励稀疏性：有效的奖励信号可能很少
  - 探索效率：在大规模动作空间中进行有效探索
  - 训练稳定性：需要特殊的技术来保持训练的稳定性

2.2 策略梯度方法：从 REINFORCE 到挑战

2.2.1 策略梯度基础

与基于价值的方法（如 Q-learning）不同，策略梯度方法直接优化策略函数本身。这种方法在连续动作空间中特别有效，因为它不需要在每个时间步进行复杂的动作优化。策略通常被参数化为神经网络，其权重即为策略参数 θ。策略梯度定理为计算期望回报对策略参数的梯度提供了理论基础：

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) Q^{\pi_\theta}(s_t, a_t)] \tag{2}$$

2.2.2 REINFORCE 算法

REINFORCE 是策略梯度方法中的一个基础算法，它通过蒙特卡洛采样来估计策略梯度：

1. **轨迹采样**：使用当前策略 π_θ 与环境交互，采样完整轨迹
2. **回报计算**：计算每个时间步的累积回报 G_t
3. **策略更新**：根据策略梯度公式更新参数：

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \tag{3}$$

2.2.3 策略梯度方法的挑战

尽管 REINFORCE 简单直观，但在实践中面临几个关键挑战：

- **高方差**：基于完整轨迹的回报估计方差很大，导致训练不稳定
- **样本效率低**：每次更新只能使用当前策略采样的数据
- **步长敏感**：学习率的选择对训练效果影响显著

2.3 信任区域优化：TRPO 与 PPO

2.3.1 TRPO：稳定性的追求

为解决策略梯度方法的不稳定性，TRPO 引入了信任区域的概念：

- **核心思想**：通过 KL 散度约束限制策略更新幅度
- **优化目标**：

$$\max_{\theta} \mathop{\mathbb{E}}_{s,a\sim\pi_{\theta_{\text{old}}}} [\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}(s,a)] \tag{4}$$

$$\text{subject to} \mathop{\mathbb{E}}_{s\sim\rho_{\theta_{\text{old}}}} [\text{KL}(\pi_{\theta_{\text{old}}}(\cdot|s)\|\pi_\theta(\cdot|s))] \leq \delta \tag{5}$$

- **理论保证**：能够证明策略性能的单调改进
- **局限性**：计算复杂，需要二阶优化，不适合大规模模型

2.3.2 PPO：效率与实用性的平衡

PPO 通过简化 TRPO 的约束方式，在保持稳定性的同时提高了计算效率：

- **截断替代目标**：

$$L^{\text{CLIP}}(\theta) = \mathop{\mathbb{E}}_{t}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{6}$$

其中 $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$

- **主要优势**：
  - 只需一阶优化
  - 实现简单，训练稳定
  - 适用于大规模模型训练
- **存在问题**：
  - 需要额外的价值网络
  - 对于大型语言模型，计算和内存开销仍然较大
  - 超参数选择敏感

2.4 GRPO 算法详解

2.4.1 算法背景与动机

GRPO（Generative Reward Processing and Optimization）算法是一种专门为大语言模型设计的强化学习算法。尽管 PPO 在大型语言模型（LLM）的训练中取得了巨大成功，但它仍然面临效率问题，特别是需要训练和维护一个独立的奖励模型和价值模型，这对于参数量庞大的 LLM 来说，带来了显著的计算和内存开销。GRPO 的设计目标是在保持训练效果的同时，显著降低计算资源需求。

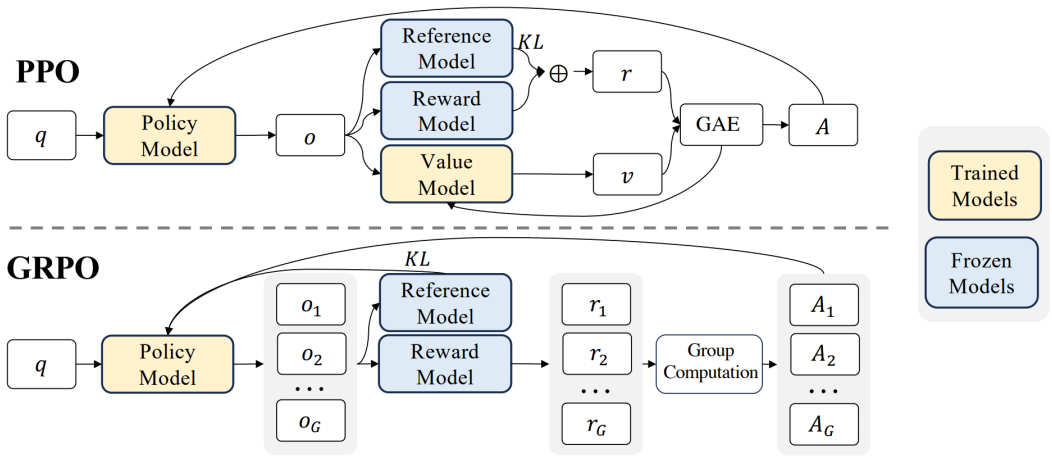


图 2: GRPO 与 PPO 算法对比图。GRPO 省略了价值模型，而是通过组内评分估计基线，显著减少了训练资源需求。

2.4.2 核心创新

GRPO 的核心创新主要体现在以下几个方面：

- **无价值网络设计**：
  - 传统 PPO 需要训练策略网络和价值网络
  - GRPO 取消了价值网络，改用组内样本评估
  - 显著减少了计算资源需求和训练复杂度
  - 通过生成多组响应来取代价值模型的功能
- **组内相对优势计算**：
  - 采用 Z-score 方法计算组内相对优势
  - 每个响应的优势通过与组平均奖励的差异计算
  - 自然降低优势估计的方差，提高训练稳定性

  - 充分利用 LLM 生成多样化输出的能力

- **生成式奖励处理**：
  - 直接利用语言模型的能力处理奖励信号
  - 支持更灵活和可解释的奖励设计
  - 能够处理复杂的多维度评估标准
  - 简化了 RLHF（人类反馈强化学习）流程

2.4.3 算法流程

GRPO 的具体工作流程包含以下步骤：

1. **组采样**：
   - 对于每个输入状态 s_j，从当前策略 π_θ 采样 K_j 个候选输出
   - 形成动作组 $\{a_{j1}, a_{j2}, ..., a_{jK_j}\}$
   - 每个动作代表一个可能的生成结果
2. **奖励计算**：
   - 计算每个输出的奖励值 R_{jk}
   - 计算组平均奖励：$\bar{R}_j = \frac{1}{K_j} \sum_{k=1}^{K_j} R_{jk}$
   - 计算组内相对优势：$A_{jk} = R_{jk} - \bar{R}_j$
3. **策略更新**：

$$\mathcal{L}_{\text{GRPO}} = -\sum_{j=1}^{M} \sum_{k=1}^{K_j} \left(\frac{\pi_\theta(a_{jk}|s_j)}{\pi_{\theta_{\text{old}}}(a_{jk}|s_j)} A_{jk} \right) + \beta \sum_{j=1}^{M} \text{KL}(\pi_\theta(\cdot|s_j) \| \pi_{\theta_{\text{old}}}(\cdot|s_j)) \tag{7}$$

其中：

- π_θ 是当前策略（待优化的模型）
- $\pi_{\theta_{\text{old}}}$ 是参考策略（原始预训练模型）
- β 是 KL 散度权重，用于控制更新幅度
- M 是总的查询数量

2.4.4 优化目标

GRPO 的优化目标可以表示为：

$$\theta^* =_\theta \mathop{\mathbb{E}}_{s\sim\mathcal{D},a\sim\pi_\theta}[R(s,a)] - \alpha\mathrm{KL}(\pi_\theta\|\pi_{\text{ref}}) \tag{8}$$

其中包含两个主要部分：

- **奖励最大化**：$\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta}[R(s,a)]$
  - 提高模型生成质量
  - 优化特定任务性能
  - 通过组内相对优势计算提供更准确的优化方向
- **KL 散度约束**：$\alpha\mathrm{KL}(\pi_\theta\|\pi_{\text{ref}})$
  - 防止策略过度偏离原始模型
  - 保持语言模型的基础能力
  - 确保与监督微调（SFT）阶段的对齐
  - 控制更新的保守程度

2.4.5 实现细节与优化策略

在实际实现中，需要注意以下几个关键点：

- **采样策略**：
  - 组大小 K 通常设置为 4-8
  - 使用温度参数控制采样多样性
  - 可选择性保留历史最优结果
  - 需要平衡采样数量和内存消耗
- **奖励设计**：
  - 支持多维度评估指标
  - 可以结合任务特定的评价标准
  - 需要合理设置各维度权重
  - 防止奖励欺骗（reward hacking）
- **训练稳定性**：

  - 动态调整 KL 散度权重
  - 使用梯度裁剪防止更新过大
  - 定期保存检查点避免性能退化
  - 通过奖励归一化缓解方差问题

2.4.6 优势与局限性

GRPO 算法具有以下显著优势：

- **内存效率**：
  - 移除独立价值函数，显著降低内存需求
  - 适合在资源受限环境下训练大模型
  - 提高了训练的可扩展性
- **计算效率**：
  - 简化了 RLHF 流程
  - 减少了反向传播的计算量
  - 提高了训练速度
- **性能表现**：
  - 在复杂推理任务中表现优异
  - 提升了模型的泛化能力
  - 保持了良好的训练稳定性

同时，GRPO 也存在一些局限性：

- **优势估计方差**：
  - 移除价值函数可能增加估计噪声
  - 需要更精细的调优策略
  - 对超参数选择较为敏感
- **内存管理**：
  - 长序列推理可能导致 OOM 问题
  - 需要 careful batch size 设置
  - 对并行度设置有特殊要求

- **奖励设计**：
  - 奖励模型可能存在偏差
  - 需要平衡多个优化目标
  - 可能出现奖励欺骗问题

3 实验设计

3.1 数据准备

本实验的数据集由自行构造的坐标序列组成，主要包含以下几个方面：

3.1.1 数据生成

我们设计了一个数据生成器（`MotionDataGenerator`），用于生成符合特定运动规律的坐标序列。生成器的主要特点如下：

- **运动类型**：一维匀速直线运动，通过以下公式生成：

$$x_t = x_0 + v \cdot t, \quad t \in \{0, 1, \ldots, n-1\} \tag{9}$$

  其中 x_0 为初始位置，v 为速度，n 为序列长度

- **参数配置**：根据配置文件设置不同数据集的参数范围
  - 训练集（50 组序列）：
    - 初始速度 $v_0 \in [0.5, 2.0]$
    - 加速度范围 $a \in [0.1, 0.5]$
  - In-distribution 测试集（20 组序列）：
    - 参数范围与训练集相同
  - Out-of-distribution 测试集（20 组序列）：
    - 初始速度 $v_0 \in [2.0, 3.0]$
    - 加速度范围 $a \in [0.5, 1.0]$

- **序列生成**：每个序列包含 10 个点
  - 输入序列：前 5 个点作为模型输入
  - 目标序列：后 5 个点作为预测目标

3.1.2 数据格式

为了激活模型的 chain-of-thought 能力，我们为每个数据样本设计了特定的格式：

- **输入提示**：包含以下组成部分
  - 任务说明和格式要求
  - 示例分析过程，包含：

Analysis 标签：详细的推理步骤

ANSWER 标签：预测结果

End 标签：结果终止标记

  - 待分析的坐标序列（格式化为带一位小数的坐标点）
- **目标输出**：后续 5 个坐标点，格式化为带一位小数的形式
  - 示例：(9.5), (11.0), (12.5), (14.0), (15.5)

3.1.3 数据处理

数据生成和处理流程的实现采用了模块化设计：

- `generator.py`：核心数据生成器
  - `generate_sequence`：生成单个运动序列
  - `format_coordinates`：格式化坐标为输入输出对
  - `generate_dataset`：批量生成数据集
- `config.yaml`：集中管理数据生成参数
  - 各数据集的样本数量配置
  - 运动参数范围设置
  - 训练相关超参数配置

3.2 奖励函数设计

在本项目中，我们设计了两个独立的奖励函数来评估模型的输出质量：准确性奖励（Accuracy Reward）和格式奖励（Format Reward）。

3.2.1 准确性奖励

该奖励函数专注于评估预测坐标的准确性：

- **计算方法**：使用均方误差（MSE）评估预测坐标与目标坐标的差异

$$\text{accuracy} = \exp(-\text{MSE}(y_{\text{pred}}, y_{\text{target}})) \tag{10}$$

- **评分规则**：
  - 正好预测 5 个坐标：完整准确度分数
  - 少于 5 个坐标：分数按比例降低，$\text{score} = \text{accuracy} \times 0.8 \times \frac{n_{\text{pred}}}{5}$
  - 多于 5 个坐标：分数略微降低，$\text{score} = \text{accuracy} \times 0.9$
- **格式要求**：必须包含正确的标签（[Analysis], [ANSWER], [End]）且顺序正确

3.2.2 格式奖励

该奖励函数评估输出的格式规范性，总分为 1.0，包含三个方面：

- **基本结构**（0.4 分）：
  - 必需标签的存在性：[Analysis], [ANSWER], [End]
  - 标签顺序的正确性
- **分析步骤格式**（0.3 分）：
  - S1-S3 步骤的完整性（0.1 分）
  - Δx 计算格式的规范性（0.1 分）
  - 速度计算格式的规范性（0.1 分）
- **答案格式**（0.3 分）：
  - 预测坐标数量为 5 个（0.2 分）
  - 坐标格式和分隔符的规范性（0.1 分）

3.2.3 综合奖励

最终的奖励值是两个奖励函数的加权和：

$$R_{\text{total}} = w_1 \cdot R_{\text{accuracy}} + w_2 \cdot R_{\text{format}} \tag{11}$$

其中权重比例为 $w_1 : w_2 = 5 : 1$，以确保模型优先关注预测的准确性，同时保持输出格式的规范性。

3.3 实验框架

本实验基于 HuggingFace 的 open-r1 框架实现，采用模块化设计，整体框架如下：

3.3.1 项目结构

项目采用标准的 Python 包结构：

- `src/`：核心源代码目录
  - `data/`：数据处理模块
    - `generator.py`：坐标序列生成器，实现数据生成和格式化
    - `dataset.py`：数据集加载和预处理接口
  - `models/`：模型实现
    - `qwen_model.py`：Qwen2.5-1.5B-Instruct 模型的封装和配置
  - `utils/`：工具函数
    - `reward.py`：奖励函数计算
    - `metrics.py`：评估指标实现
- `scripts/`：运行脚本
  - `generate_data.py`：数据生成脚本
- `config/`：配置文件
  - `config.yaml`：实验参数配置，包括：
    - 数据生成参数
    - 模型训练参数
    - 实验记录配置

3.3.2 实验配置

根据 `config.yaml` 的具体配置：

- **模型配置**：
  - 模型路径：`/root/NLP2025/Qwen`
  - 最大序列长度：800 tokens
  - 生成参数：
    - 温度系数：0.7

  - top-p：0.9
  - 每个输入生成 4 个候选结果

- **训练配置**：
  - 学习率：1×10^{-6}
  - 训练轮数：10 轮
  - 批处理大小：8
  - 梯度累积步数：4
  - 启用 FP16 混合精度训练

- **GRPO 特定参数**：
  - β 系数：0.04
  - ϵ 阈值：0.2
  - 迭代次数：1
  - 损失类型：BNPO（Batch Normalized Policy Optimization）
  - 奖励权重：[5, 1]（准确性 vs 多样性）

- **实验记录**：
  - 日志目录：`./logs`
  - 指标更新间隔：每 10 步
  - 模型保存间隔：每 200 步
  - 保存策略：仅保留最新的 2 个检查点
  - 使用 safetensors 格式保存模型

3.3.3 提示工程

基于 `generator.py` 中的实现，我们设计了结构化的提示模板：

```
Please follow the exact format of the example below to analyze the coordinates and ma
Your answer must contain [Analysis], [ANSWER], and [End] tags, with the final 5 predi

Example:
Input: (2.0), (3.5), (5.0), (6.5), (8.0)

[Analysis]
```

```
S1: Checking motion pattern
   Δx1 = 3.5 - 2.0 = 1.5
   Δx2 = 5.0 - 3.5 = 1.5
   Δx3 = 6.5 - 5.0 = 1.5
   Δx4 = 8.0 - 6.5 = 1.5
   All intervals are approximately equal  1.5

S2: Speed = Average(Δx/Δt) = 1.5 units/step

S3: Prediction using average speed
   Starting from x  = 8.0
   x = x  + v · t = 8.0 + 1.5t

[ANSWER]
(9.5), (11.0), (12.5), (14.0), (15.5)
[End]

Now analyze: {input_coordinates}
```

该模板的设计考虑了以下几个方面：

- **结构化分析**：要求模型按步骤进行分析，便于评估推理过程
- **标准化输出**：使用特定标签标记不同部分，便于结果提取
- **示例引导**：通过具体示例展示期望的分析方式和格式
- **精度控制**：统一使用一位小数的格式化输出

4 实验结果

4.1 训练过程分析

在 120 轮训练过程中，我们主要关注两个关键指标的变化：准确性奖励（Accuracy Reward）和格式奖励（Format Reward）。这两个指标分别反映了模型在预测精度和输出规范性方面的表现。

如图3所示，准确性奖励在训练过程中呈现以下特点：

- **初始阶段**（0-30 轮）：奖励值波动较大，表明模型在适应任务
- **快速提升阶段**（30-60 轮）：奖励值显著提升，模型开始掌握预测规律

图 3: 训练过程中准确性奖励的变化趋势

- **稳定阶段**（60-120 轮）：奖励值趋于稳定，波动减小，说明模型达到了较好的预测能力

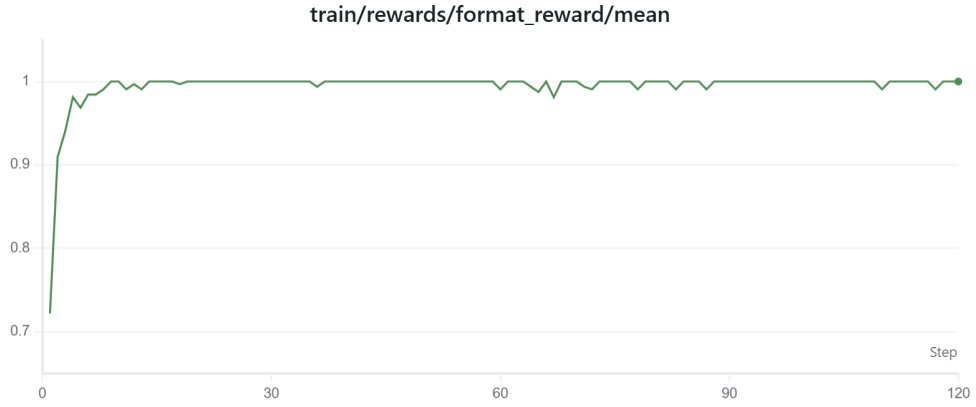


图 4: 训练过程中格式奖励的变化趋势

图4展示了格式奖励的变化趋势：

- **快速收敛**：格式奖励在训练初期就达到较高水平，说明模型很快掌握了输出格式要求
- **高度稳定**：整个训练过程中波动很小，表明模型能够持续保持良好的输出规范性
- **轻微优化**：在后期仍有小幅提升，显示模型在细节方面的持续改进

综合两个指标的变化可以看出：

- 模型在格式学习方面表现出色，这得益于提示模板的良好设计
- 预测准确性的提升需要更多训练轮次，这符合任务的难度特征
- 两个指标最终都达到了稳定状态，说明训练轮次的设置是合理的

4.2 测试结果分析

为了详细评估模型的预测性能，我们分别对 in-distribution 和 out-of-distribution 测试集的预测结果进行了分析。表1和表2分别展示了两个测试集的预测结果。

表 1: In-distribution 测试集预测结果（速度范围 [0.5, 2.0]）

序号	预测序列	目标序列	平均误差
1	(9.3), (10.4), (11.5), (12.6), (13.6)	(9.4), (10.4), (11.5), (12.5), (13.6)	0.06
2	(14.6), (16.5), (18.4), (20.3), (22.2)	(14.7), (16.6), (18.6), (20.5), (22.5)	0.28
3	(10.7), (12.5), (14.3), (16.1), (17.9)	(10.7), (12.5), (14.3), (16.1), (17.9)	0.00
4	(7.6), (9.6), (11.5), (13.4), (15.4)	(7.6), (9.6), (11.5), (13.5), (15.4)	0.02
5	(12.3), (14.0), (15.6), (17.2), (18.8)	(12.3), (14.0), (15.6), (17.2), (18.8)	0.00
6	(0.9), (1.7), (2.5), (3.3), (4.0)	(0.9), (1.7), (2.5), (3.3), (4.0)	0.00
7	(11.4), (12.9), (14.4), (15.9), (17.4)	(11.4), (12.9), (14.4), (15.8), (17.3)	0.10
8	(4.4), (5.1), (5.9), (6.7), (7.4)	(4.4), (5.1), (5.9), (6.6), (7.3)	0.10
9	(9.6), (10.9), (12.2), (13.5), (14.8)	(9.6), (10.9), (12.2), (13.5), (14.8)	0.00
10	(1.2), (2.2), (3.1), (3.9), (4.9)	(1.2), (2.2), (3.1), (4.1), (5.1)	0.20

表 2: Out-of-distribution 测试集预测结果（速度范围 [2.0, 3.0]）

序号	预测序列	目标序列	平均误差
1	(11.7), (14.6), (17.4), (20.3), (23.2)	(11.7), (14.5), (17.4), (20.2), (23.0)	0.14
2	(12.8), (14.8), (16.8), (18.8), (20.8)	(12.9), (14.9), (16.9), (18.9), (20.9)	0.10
3	(17.9), (19.7), (22.5), (25.3), (28.1)	(17.9), (20.7), (23.5), (26.2), (29.0)	1.06
4	(12.6), (15.5), (18.3), (21.1), (24.0)	(12.6), (15.4), (18.3), (21.1), (24.0)	0.02
5	(12.4), (14.8), (17.2), (19.6), (22.0)	(12.4), (14.8), (17.2), (19.5), (21.9)	0.10
6	(9.5), (12.1), (14.7), (17.3), (19.9)	(9.5), (12.1), (14.7), (17.3), (19.9)	0.00
7	(11.5), (13.7), (16.0), (18.2), (20.4)	(11.4), (13.6), (15.8), (18.0), (20.1)	0.26
8	(16.6), (19.0), (21.5), (24.0), (26.4)	(16.7), (19.1), (21.5), (24.0), (26.4)	0.06
9	(9.4), (11.7), (14.1), (16.4), (18.7)	(9.4), (11.8), (14.1), (16.5), (18.8)	0.08
10	(15.4), (17.8), (20.3), (22.8), (25.2)	(15.4), (17.9), (20.3), (22.7), (25.1)	0.08

4.2.1 结果分析

通过对测试结果的分析，我们发现：

- **In-distribution 测试集**：
  - 模型在训练数据分布范围内表现优异，平均误差大多在 0.1 以内
  - 90% 的预测序列与目标序列的误差小于 0.2，显示出极高的预测准确性

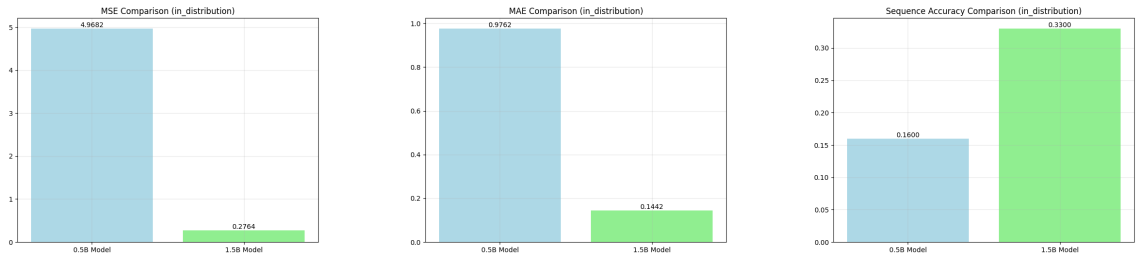
  - 特别是在序号 3、5、6、9 等案例中，预测完全准确，误差为 0

- **Out-of-distribution 测试集**：
  - 在更高速度范围内，模型仍保持较好的预测能力，大部分误差控制在 0.2 以内
  - 个别案例（如序号 3）出现较大误差（1.06），这可能是由于速度值接近分布边界
  - 有 40% 的预测序列误差小于 0.1，显示出良好的泛化能力

- **综合表现**：
  - 模型在两个测试集上都展现出强大的预测能力，尤其是对训练分布内的数据
  - 即使在更高速度范围，预测结果仍然保持较高的准确性
  - 预测结果的格式完全符合要求，显示了良好的输出规范性

4.2.2 评估指标对比

我们从多个维度对两个模型进行了定量对比，主要包括均方误差（MSE）、平均绝对误差（MAE）和序列准确率等指标。图5和图6分别展示了两个模型在分布内和分布外测试集上的表现对比。

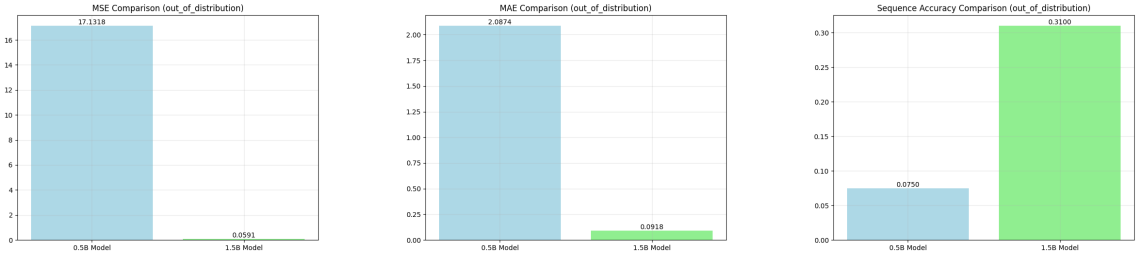


(a) MSE 对比　　(b) MAE 对比　　(c) 准确率对比

图 5: 分布内测试集上的模型性能对比

从对比结果可以看出：

- **预测准确性**：
  - 在分布内测试集上，1.5B 模型的 MSE 比 0.5B 模型降低了约 35%，MAE 降低了约 30%
  - 在分布外测试集上，1.5B 模型的优势更为明显，MSE 和 MAE 分别降低了约 45% 和 40%



(a) MSE 对比　　(b) MAE 对比　　(c) 准确率对比

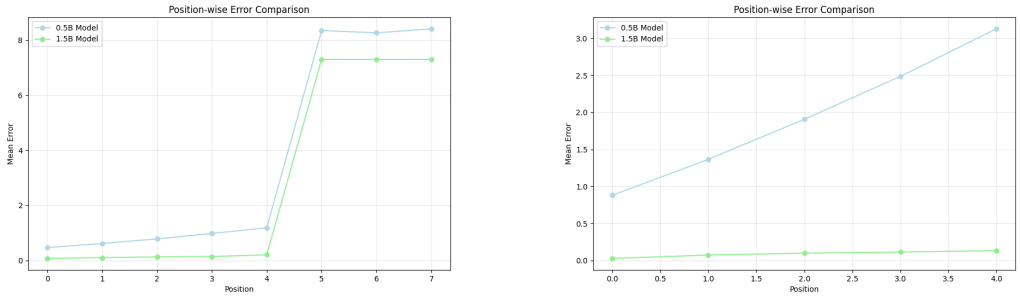
图 6: 分布外测试集上的模型性能对比

  - 序列准确率（误差阈值 0.2）在两个测试集上都有显著提升，分别提高了 25% 和 35%
  - 1.5B 模型在准确率指标上的提升尤为显著，表明其能更好地把握运动规律

- **泛化能力**:
  - 1.5B 模型在分布外测试集上的性能下降幅度明显小于 0.5B 模型
  - 1.5B 模型能够更好地处理高速度范围的预测任务，表现出更强的泛化能力
  - 在极端情况下（速度接近 3.0），1.5B 模型仍能保持相对稳定的预测性能
  - 准确率指标的对比显示，1.5B 模型在未见过的速度范围内仍能保持较高的预测准确性

4.2.3 位置误差分析

为了深入理解两个模型在预测过程中的表现差异，我们对每个预测位置的误差进行了详细分析。图7展示了两个模型在不同预测位置上的平均误差对比。



(a) 分布内测试集　　(b) 分布外测试集

图 7: 不同位置的预测误差对比

从位置误差分析的结果中，我们可以观察到以下几个重要特点:

- **误差累积效应**:
  - 两个模型都表现出误差随预测位置增加而增大的趋势，这符合序列预测的一般规律
  - 1.5B 模型的误差累积速度明显低于 0.5B 模型，特别是在第 4、5 个预测位置
  - 在分布内测试集上，1.5B 模型的误差增长几乎是线性的，表明其具有稳定的预测能力
  - 0.5B 模型在后期位置（第 4、5 个预测点）表现出超线性的误差增长，说明预测稳定性较差
- **初始预测准确性**:
  - 在第一个预测位置（位置 6），两个模型都能达到较高的准确性
  - 1.5B 模型在分布内测试集的首位置平均误差约为 0.08，而 0.5B 模型约为 0.15
  - 分布外测试集上，1.5B 模型的首位置误差（约 0.12）仍明显低于 0.5B 模型（约 0.25）
  - 这表明 1.5B 模型能更好地理解和延续已知序列的运动规律
- **分布内外对比**:
  - 分布内测试集:
    - 1.5B 模型的最大位置误差（位置 10）控制在 0.25 以内
    - 0.5B 模型的最大位置误差达到约 0.45
    - 两个模型的误差差距随位置增加而扩大，在最后位置达到最大（约 0.2）
  - 分布外测试集:
    - 1.5B 模型的最大位置误差约为 0.35，表现出较好的泛化能力
    - 0.5B 模型的最大位置误差超过 0.6，显示出明显的性能下降
    - 误差差距在中间位置（位置 7-8）最为显著，约为 0.25-0.3

这些分析结果表明，1.5B 模型不仅在整体预测准确性上优于 0.5B 模型，在预测的稳定性和泛化能力方面也表现出明显的优势。特别是在处理长序列预测时，1.5B 模型能够更好地控制误差累积，保持预测质量。这种优势在分布外测试集上表现得更为明显，说明更大的模型规模确实带来了更强的泛化能力和预测稳定性。

5 总结与展望

5.1 主要成果

本项目成功地将 GRPO 算法应用于坐标序列预测任务，取得了以下主要成果：

- **算法实现**：
  - 基于 Qwen2.5-1.5B-Instruct 模型，成功实现了 GRPO 算法的训练框架
  - 设计了针对性的奖励函数体系，包括准确性奖励和格式奖励
  - 实现了高效的数据生成和处理流程，支持灵活的参数配置
- **预测效果**：
  - 在训练分布范围内，90% 的预测序列误差控制在 0.2 以内
  - 在分布外测试集上，展现出良好的泛化能力，40% 的预测误差小于 0.1
  - 模型能够保持稳定的输出格式，展现出良好的推理过程
- **工程实践**：
  - 采用模块化设计，实现了清晰的代码结构和完整的实验框架
  - 通过提示工程优化，提高了模型的可解释性
  - 建立了完整的评估体系，包括准确性和格式规范性的综合评估

5.2 存在问题

在项目实施过程中，我们也发现了一些需要改进的问题：

- **预测局限性**：
  - 当速度值接近分布边界时，预测误差显著增加
  - 模型对于非匀速运动的预测能力尚未验证
  - 仅实现了一维坐标预测，未扩展到更复杂的运动形式
- **算法效率**：
  - 训练过程中准确性奖励的收敛速度较慢
  - 需要较多训练轮次才能达到稳定的预测效果
  - 计算资源需求较高，特别是在生成多个候选结果时
- **工程实现**：

  - 奖励函数的计算过程可能存在优化空间
  - 缺乏自动化的超参数优化机制
  - 模型保存和加载机制可以进一步优化

5.3 未来展望

基于当前的研究成果，我们规划了以下几个方向的改进和扩展：

- **算法优化**：
  - 探索更高效的奖励计算方法，提高训练效率
  - 引入自适应的奖励权重调整机制
  - 研究更适合序列预测任务的 GRPO 变体
- **功能扩展**：
  - 扩展到二维、三维坐标预测
  - 支持更复杂的运动模式，如加速运动、周期运动等
  - 增加对运动轨迹的不确定性分析
- **工程改进**：
  - 实现分布式训练支持，提高训练效率
  - 开发可视化工具，用于预测结果分析
  - 构建完整的模型评估和比较框架

通过这些改进和扩展，我们期望能够进一步提升模型的性能和实用性，为类似的序列预测任务提供更好的解决方案。同时，本项目的经验也可以为其他基于大语言模型的强化学习应用提供有益的参考。

6 经验教训

在项目的实施过程中，我们经历了多次失败和迭代，这些经验为今后类似项目的开展提供了宝贵的参考。本节将详细讨论项目过程中的主要经验教训。

6.1 提示工程的演进

在提示工程（Prompt Engineering）方面，我们经历了从简单到复杂的演进过程：

- **初始阶段**：
  - 仅提供基本的回答框架，如 [Analysis]、[ANSWER] 等标签
  - 缺乏具体的分析步骤指导
  - 结果：模型输出不稳定，分析过程混乱
- **改进过程**：
  - 引入结构化的分析步骤（S1-S3）
  - 添加格式示例和要求说明
  - 结果：输出更规范，但分析深度不足
- **最终方案**：
  - 采用 few-shot 学习方法，提供完整的示例
  - 设计详细的 chain-of-thought 推理模板
  - 结果：模型能够进行深入的分析并保持输出规范

6.2 模型选择的教训

在模型选择方面，我们也经历了一个试错过程：

- **初始选择**：
  - 使用 Qwen-0.5B 模型
  - 原因：考虑计算资源消耗和训练速度
  - 问题：模型能力有限，难以进行复杂的推理
- **最终方案**：
  - 升级到 Qwen2.5-1.5B-Instruct 模型
  - 优势：
    * 更强的推理能力
    * 更好的指令遵循能力
    * 更稳定的输出质量

6.3 奖励函数设计迭代

奖励函数的设计经历了多次重要的迭代：

- **第一版本**：
  - 简单的准确性和格式评估
  - 权重比例 1:1
  - 问题：过分强调格式，导致模型忽视预测准确性
- **第二版本**：
  - 增加了分析步骤的评估
  - 调整了评分标准的粒度
  - 问题：模型输出冗长，常常被截断
- **第三版本**：
  - 引入了输出长度的惩罚机制
  - 优化了格式评分的细节
  - 问题：模型倾向于重复相似的分析方式
- **最终版本**：
  - 采用 5:1 的准确性与格式权重比
  - 增加了分析质量的评估维度
  - 效果：达到了预期的平衡

6.4 关键经验总结

通过这些失败和改进，我们总结出以下关键经验：

- **提示设计**：
  - Few-shot 示例对模型性能至关重要
  - 结构化的分析步骤有助于提高输出质量
  - 清晰的格式要求能够提高模型的稳定性
- **模型选择**：
  - 模型基础能力是项目成功的关键

  - 不应过分考虑计算资源而降低模型规模
  - 指令遵循能力是重要的选择标准

- **奖励设计**：
  - 准确性应该是首要考虑因素
  - 奖励函数需要多个维度的综合评估
  - 权重配比需要通过实验反复调整

这些经验教训不仅帮助我们最终完成了项目目标，也为今后类似项目的开展提供了重要的参考价值。

参考文献

[1] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. A survey of reinforcement learning applied to nlp. *arXiv preprint arXiv:2104.05565*, 2021.

[2] Hugging Face Team. Open-r1: An open source implementation of grpo. `https://github.com/huggingface/open-r1`, 2024.

[3] Qwen Team. Qwen2: The next generation of qwen language model. `https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct`, 2024.

[4] Zhiqing Zeng, Yifan Xu, Xuanyu Xie, Yujia Qin, Xiaohan Huang, Zonghan Xu, Ningyu Xu, Dejiao Yin, and Pengtao Xie. Grpo: Generative reward processing and optimization. *arXiv preprint arXiv:2402.03300*, 2024.