RLT 项目实战

一、项目简介

Reinforcement Learning Teachers (RLT) 是 Sakana AI 开源的前沿强化学习项目，专注于通过创新的"教师-学生"强化学习范式显著提升大型语言模型的数学推理能力。

该项目通过测试时扩展 (Test Time Scaling) 技术，使用强化学习训练专门的"教师"模型来指导"学生"模型生成更高质量的推理过程。项目采用了基于 GRPO (Group-based Reinforcement Policy Optimization) 算法的创新奖励机制，结合 KL 散度优化和多阶段训练流程，在 AIME24、MATH500、GPQA 等权威数学推理基准上取得了显著性能提升。

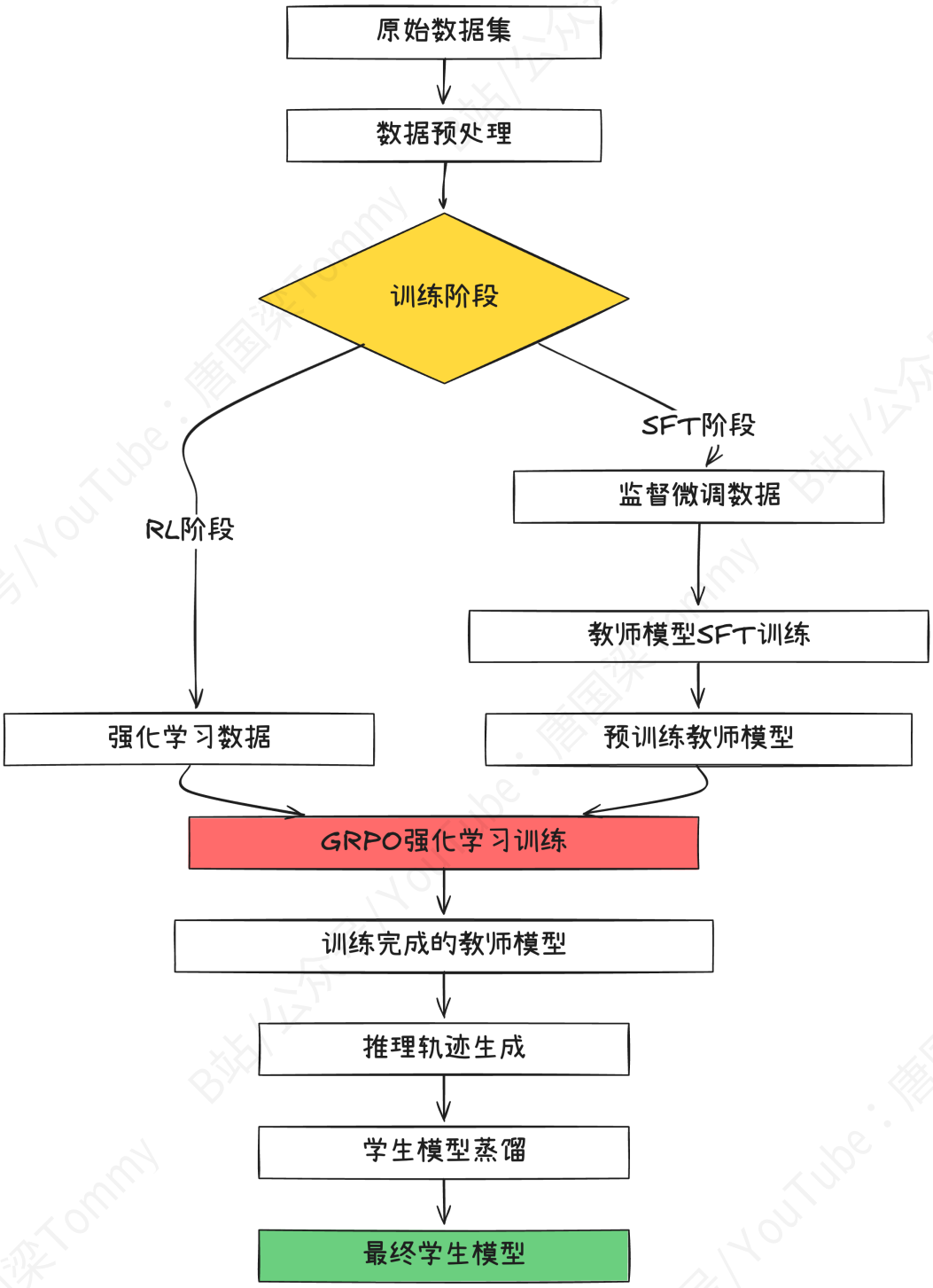
1. 核心特性

- 两阶段训练流程：监督微调(SFT) + 强化学习(RL)
- 测试时扩展：支持推理时间的动态扩展
- 高效推理：集成vLLM加速推理
- 分布式训练：支持多GPU并行训练
- 灵活配置：基于Hydra的模块化配置系统

2. 技术栈

技术层级	核心技术	版本/规格	作用与优势
深度学习框架	PyTorch	2.6.0	CUDA 12.4 优化，混合精度训练
模型库	Transformers	最新版	HuggingFace 生态集成
强化学习	TRL (定制)	1.5+	GRPO 算法实现与优化
推理引擎	vLLM	0.8.3	高并发推理，PagedAttention
分布式训练	DeepSpeed	ZeRO-3	千万级参数模型支持
注意力优化	Flash Attention	v2	内存效率提升 8x+
配置管理	Hydra	最新版	实验管理与超参数调优
模型优化	FlashInfer	CUDA 12.4	推理加速专用库
监控工具	Weights & Biases	集成版	实验跟踪与可视化

3. 训练流程



代码实现片段，process_single_reward 方法位于 trainers/teacher_rewards.py 脚本中。

```
def process_single_reward(self, chat, teacher_completion, ...):
    """
    处理单个样本的奖励计算 - 教师-学生模型训练的核心逻辑

    该函数通过比较学生模型和教师模型的输出分布来计算奖励信号，
    用于指导学生模型学习教师模型的推理过程和答案生成能力。
    """
```

```
    # ========== 第一步：计算学生对数概率 ==========
    # 使用学生模型计算对给定文本（学生对话）的token级对数概率
    student_log_probs = self.compute_batch_log_probs(
        text=chat, # 输入：学生对话文本
        student_model=True,  # 指定使用学生模型进行计算
        temperature=self.unbias_student_log_probs_temp
        # 温度参数：控制输出分布的平滑程度
        # 较高温度 -> 更平滑的分布，较低温度 -> 更尖锐的分布
    )
    # 返回：[batch_size, sequence_length] 的张量，包含每个token的对数概率

    # ========== 第二步：计算教师对数概率 ==========
    # 使用教师模型计算对给定文本（教师完成文本）的token级对数概率
    teacher_log_probs = self.compute_batch_log_probs(
        text=teacher_completion, # 输入：教师完成的文本
        student_model=False, # 指定使用教师模型进行计算
        temperature=self.teacher_gen_temperature # 教师模型的温度参数
    )
    # 返回：[batch_size, sequence_length] 的张量，包含每个token的对数概率

    # ========== 第三步：计算KL散度（核心步骤）==========
    # 计算教师分布P和学生分布Q之间的KL散度：KL(P||Q)
    # 这是衡量两个概率分布差异的重要指标
    thought_tokens_kl = self.estimate_kl(
        p_log_probs=teacher_log_probs, # P分布：教师模型的对数概率（目标分布）
        q_log_probs=student_log_probs, # Q分布：学生模型的对数概率（当前分布）
        use_schulman_kl_estimation=use_schulman_unbiased_estimate  # 是否使用Schulman的无偏
KL估计(推荐使用)
        # Schulman方法：KL(p||q) ≈ log(p/q) - 1 + q/p
        # 标准方法：KL(p||q) = log(p/q) = log(p) - log(q)
    )
    # 返回：[batch_size, sequence_length] 的张量，包含每个token位置的KL散度
    # KL散度越小 -> 学生分布越接近教师分布 -> 学生模型学习得越好

    # ========== 第四步：计算总奖励 ==========
    # 奖励函数由三个组件组成，每个组件衡量学生模型的不同方面

    # 4.1 对数概率奖励：衡量学生模型生成正确答案的能力
    log_prob_reward = (log_prob_scores * self.answer_log_prob_coeff).sum(-1)
    # log_prob_scores：经过归一化和降维处理的学生对数概率分数
    # answer_log_prob_coeff：答案对数概率的权重系数
    # .sum(-1)：对最后一个维度求和，得到每个样本的总分
    # 更高的对数概率 -> 更好的答案生成能力 -> 更高的奖励

    # 4.2 KL散度奖励：衡量学生模型推理过程与教师模型的相似性
    kl_reward = -1 * (kl_scores * self.kl_penalty_reward_coeff).sum(-1)
    # kl_scores：经过归一化和降维处理的KL散度分数
    # kl_penalty_reward_coeff：KL散度惩罚的权重系数
    # -1 *：取负值，因为我们希望最小化KL散度（KL散度越小，奖励越高）
    # 更小的KL散度 -> 学生更好地模仿教师的推理过程 -> 更高的奖励
```

```
    # 4.3 总奖励计算：线性组合各个奖励分量
    total_reward = log_prob_reward + kl_reward + match_reward
    # log_prob_reward：答案质量奖励（鼓励生成正确答案）
    # kl_reward：推理过程奖励（鼓励模仿教师的推理过程）
    # match_reward：格式匹配奖励（鼓励正确的输出格式）

    # 总奖励反映了学生模型在以下方面的综合表现：
    # - 答案准确性（通过对数概率衡量）
    # - 推理过程质量（通过KL散度衡量）
    # - 输出格式规范性（通过格式匹配衡量）

    return total_reward  # 返回最终的奖励信号，用于GRPO训练过程
```

两种KL散度估计方法的区别：

特性	标准方法	Schulman方法
偏差	无偏	无偏
方差	高方差	低方差
数值稳定性	差	好
训练稳定性	可能不稳定	更稳定
计算复杂度	简单	稍复杂

- **使用标准方法的情况：**
  - 分布P和Q非常接近
  - 有大量样本可以平均化方差
  - 需要最简单的实现
- **使用Schulman方法的情况：**
  - 分布P和Q可能相差较大
  - 样本数量有限
  - 需要稳定的训练过程（如RL中的策略更新）

总结：Schulman方法通过巧妙的数学技巧，在保持无偏性的同时显著降低了方差，特别适用于强化学习等需要稳定训练的场景。

二、项目实战

1. 环境准备

1.1 系统要求

- **操作系统**: Linux (推荐 Ubuntu 22.04)
- **Python**: 3.11
- **CUDA**: 12.8
- **GPU**: 8x H100 (最少2 x RTX 4090)
- **内存**: 建议 32GB+ RAM

1.2 Conda环境创建

```
# 创建新的conda环境
conda create -n rlt python=3.11 -y
conda init bash && source /root/.bashrc
conda activate rlt
conda install ipykernel
ipython kernel install --user --name=rlt

# 安装基础依赖
./scripts/install_08.sh
```

```
Successfully installed accelerate-1.4.0 aiohttp_cors-0.8.1 antlr4-python3-runtime-4.9.3 bitsandbytes-0.45.5 cachetools-5.5.2 colorful-0.5.7 contourpy-1.3.2 cycler-0.12.1 datasets-3.2.0 deepspeed-0.15.4 dill-0.3.8 distlib-0.3.9 fire-0.7.0 fonttools-4.58.5 gitdb-4.0.12 gitpython-3.1.44 google-api-core-2.25.1 google-auth-2.40.3 googleapis-common-protos-1.70.0 hf-transfer-0.1.9 hjson-3.1.0 hydra-core-1.3.2 importlib-metadata-8.7.0 kiwisolver-1.4.8 matplotlib-3.10.3 multiprocess-0.70.16 omegaconf-2.3.0 opencensus-0.11.4 opencensus-context-0.1.3 opentelemetry-api-1.34.1 opentelemetry-exporter-prometheus-0.55b1 opentelemetry-proto-1.34.1 opentelemetry-sdk-1.34.1 opentelemetry-semantic-conventions-0.55b1 pandas-2.3.1 peft-0.14.0 proto-plus-1.26.1 protobuf-5.29.5 py-spy-0.4.0 pyarrow-20.0.0 pyasn1-0.6.1 pyasn1-modules-0.4.2 pyparsing-3.2.3 pytz-2025.2 rsa-4.9.1 smart_open-7.3.0.post1 smmap-5.0.2 tensorboard-2.18.0 termcolor-3.1.0 transformers-4.51.1 trl-0.14.0 tzdata-2025.2 virtualenv-20.31.2 wandb-0.21.0 wrapt-1.17.2 xxhash-3.5.0 zipp-3.23.0
```

1.3 数据集与模型下载

- **模型下载:**

```
# 安装 git lfs
apt-get update
apt-get install -y curl vim sudo
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
sudo apt-get install git-lfs
git lfs install

# 设置国内镜像环境变量（可选，解决科学上网问题）
export HF_ENDPOINT=https://hf-mirror.com

git clone https://huggingface.co/Qwen/Qwen3-0.6B
```

- **数据集下载:**

```
pip install --upgrade huggingface_hub

huggingface-cli download --repo-type dataset --resume-download bespokelabs/Bespoke-Stratos-17k --local-dir /root/autodl-tmp/datasets/Bespoke-Stratos-17k
```

2. 快速开始

2.1 第一阶段：监督微调（SFT）

```
# 参数说明：
# 2 ：使用的GPU数量
# teacher_sft.yaml：SFT训练配置文件
# output_dir：指定保存预训练模型的路径


# 2块GPU训练(默认：DeepSpeed ZeRO-3)
HYDRA_FULL_ERROR=1 ./launch.sh 2 cfgs/run_cfg/teacher_sft.yaml \
  model_name_or_path=/root/autodl-tmp/models/Qwen3-0.6B \
  dataset_local_directory=/root/autodl-tmp/datasets/Bespoke-Stratos-17k \
  report_to=tensorboard \
  output_dir=results/sft_with_tensorboard \
  train_batch_size=4 \
  per_device_train_batch_size=1 \
  max_seq_length=4096 \
  gradient_checkpointing=true
```

```
# 训练开始后，在另一个终端启动tensorboard
tensorboard --logdir results/sft_with_tensorboard --port 6006 --host 0.0.0.0

# 在浏览器访问：http://localhost:6006
```

2.2 第二阶段：强化学习训练（RLT）

使用第一阶段的检查点进行RL训练：

```
# 在2张GPU上运行RL训练（1张用于vLLM服务器，1张用于训练）
HYDRA_FULL_ERROR=1 ./launch_with_server.sh 1 1 cfgs/run_cfg/teacher_rlt.yaml \
    model_name_or_path=/root/autodl-tmp/models/Qwen3-0.6B \
    student_model=/root/autodl-tmp/models/Qwen3-1.7B \
    dataset_id_or_path=/root/autodl-tmp/datasets/Bespoke-Stratos-17k \
    train_batch_size=8 \
    output_dir=results/my_rlt_model \
    report_to=tensorboard \
    offload
```

```
# 参数说明：
# 1：vLLM服务器使用的GPU数量
# 1：训练使用的GPU数量
# teacher_rlt.yaml：RL训练配置文件
# model_name_or_path：第一阶段训练好的模型路径
# output_dir：最终RLT模型保存路径
```

```
# 训练开始后，在另一个终端启动tensorboard
tensorboard --logdir results/my_rlt_model --port 6006 --host 0.0.0.0

# 在浏览器访问：http://localhost:6006
```

2.3 基于 vLLM 推理测试

```
# 启动vLLM服务器
python -m vllm.entrypoints.openai.api_server \
    --model results/my_rlt_model/checkpoint-50 \
    --host 127.0.0.1 \
    --port 8000 \
    --tensor-parallel-size 1
```

```
import requests
import json

def vllm_generate(prompt, max_tokens=1024):
    url = "http://127.0.0.1:8000/v1/completions"
    data = {
        "model": "results/my_rlt_model/checkpoint-50",
        "prompt": prompt,
        "max_tokens": max_tokens,
        "temperature": 0.7,
        "top_p": 0.9
    }

    response = requests.post(url, json=data)
    return response.json()["choices"][0]["text"]

# 测试
prompt = "请分析这个数学问题的解决步骤："
response = vllm_generate(prompt)
print(response)
```