RL Fine-Tuning of Language Models

CS 224R Default Project Specification

You will be exploring the implementation of several RL algorithms and exploring how to use these algorithms for fine-tuning large language models (LLMs) comprising of two main parts: (1) an implementation part focused around implementing two algorithms for fine-tuning large language models and (2) a more research oriented exploration, to explore a technique of your choosing and evaluate whether that additional technique improves performance.

We will start by implementing direct preference optimization (DPO) to how it can improve the model's (Qwen 2.5 0.5B Base) capabilities for instruction following and mathematical reasoning. Additionally, we will implement a second algorithm called RLOO with a parametric/rule-based reward function to study how online policy-gradient algorithms perform. Next, you will construct an evaluation setup to evaluate your learned policies versus relevant baselines. Finally, you will propose an extension of your choosing to explore as a researcher, how existing algorithmic ideas can be built upon. You will submit your extension to a common leaderboard to evaluate your performance!

Note on default project vs. custom project:

It is not intended for the default final project to require less effort or work than the custom final project. The default project simply reduces the overall difficulty of devising your own project idea and evaluation methods, allowing students to commit an equivalent amount of effort to the provided problem.

Contents

1. Initial Implementation

1.1. Background

Reinforcement Learning is a popular paradigm for LLM fine-tuning today. Two dominant approaches are prevalent today: (1) training on preference data containing relative feedback, helpful for tasks where a ground truth reward is hard to capture but can rank two responses well (e.g grading an essay might be quite tough but it might be easier to differentiate a lower quality essay from a higher quality essay), and (2) training with verifier-based rewards, used for tasks such as math and code reasoning. Below we will give some background on each paradigm.

Let's start with learning from preference data. Typically, before training on preference data, a pre-

trained model is fine-tuned on high-quality data from the task of interest via supervised fine-tuning (SFT), to obtain a "reference" model π_{ref}. Then, to fine-tune π_{ref} with human preferences, usually a preference dataset $\mathcal{D}_{\text{pref}} = \{\mathbf{x}^{(i)}, \mathbf{y}_w^{(i)}, \mathbf{y}_l^{(i)}\}$ is collected, where $\mathbf{x}^{(i)}$ denotes a prompt and $\mathbf{y}_w^{(i)}, \mathbf{y}_l^{(i)}$ denote preferred and dispreferred responses, often obtained by sampling from π_{ref}. Given a preference dataset, most fine-tuning pipelines assume the existence of an underlying reward function $r^*(\mathbf{x}, \cdot)$. One popular framework for this is the Bradley-Terry (BT) model [5], assuming that human preferences can be written as:

$$p^*(\mathbf{y}_1 \succ \mathbf{y}_2|\mathbf{x}) = \frac{e^{r^*(\mathbf{x},\mathbf{y}_1)}}{e^{r^*(\mathbf{x},\mathbf{y}_1)}+e^{r^*(\mathbf{x},\mathbf{y}_2)}} \tag{1}$$

Given this reward function r^*, preference fine-tuning aims to find the optimum of the reward r^*. While the ultimate goal of preference fine-tuning is to find the *unconstrained* optimum of the reward function, in practice, we often replace the reward function with a reward model. Since the reward model is erroneous, we apply a KL-divergence constraint on the policy to prevent exploitation in the reward model. To align our results with typical preference fine-tuning procedures, we will consider such a KL-constrained reward optimization as our fine-tuning goal:

$$\max_{\pi_\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{\text{pref}},\mathbf{y}\sim\pi_\theta(\cdot|x)}[r^*(\mathbf{x},\mathbf{y})] - \beta\mathbb{D}_{\text{KL}}[\pi_\theta(\cdot|\mathbf{x})||\pi_{\text{ref}}(\cdot|\mathbf{x})] \tag{2}$$

The regularizer, weighted by β, controls the deviation of π from π_{ref} under the reverse KL divergence.

Reward model training. In order to fine-tune an LLM policy $\pi_\theta(\mathbf{y}|\mathbf{x})$, Eq. (1) provides a convenient way to learn a reward model either explicitly (i.e., by fitting a parametric reward model $r_\phi(\mathbf{x}, \mathbf{y})$) or implicitly (i.e., via direct preference optimization (DPO) [22] or IPO [14], that re-purposes the log-likelihood $\log \pi_\theta(\mathbf{y}|\mathbf{x})$ of the policy to represent the reward $r_\theta(\mathbf{x}, \mathbf{y})$). Explicit reward models are trained using the following classification objective:

$$\max_{\phi} \mathbb{E}_{(\mathbf{x},\mathbf{y}_w,\mathbf{y}_l)\sim\mathcal{D}_{\text{pref}}} \left[\log \sigma \left(r_\phi(\mathbf{x}, \mathbf{y}_w) - r_\phi(\mathbf{x}, \mathbf{y}_l)\right)\right], \tag{3}$$

where σ is the logistic function. Contrastive learning objectives [22, 14] on the other hand repurposes $\log \pi_\theta(\mathbf{y}|\mathbf{x})$ as the implicit reward $r_\theta(\mathbf{x}, \mathbf{y})$:

$$r_\theta(\mathbf{x}, \mathbf{y}) = \beta \left[\log \pi_\theta(\mathbf{y}|\mathbf{x}) - \log \pi_{\text{ref}}(\mathbf{y}|\mathbf{x})\right]. \tag{4}$$

Policy learning. On-policy RL approaches such as PPO [25] and REINFORCE [36] explicitly sample new responses from the current snapshot of the learned policy, $\mathbf{y}_i \sim \pi_\theta(\cdot|\mathbf{x}_i)$, score them under the reward model, and perform a policy gradient update on parameters θ, for example:

$$\theta' \leftarrow \theta - \eta\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{\text{pref}},\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})} \left[\nabla_\theta \log \pi_\theta(\mathbf{y}|\mathbf{x}) \cdot \bar{r}_\phi(\mathbf{x}, \mathbf{y})\right] \tag{5}$$

is the gradient update employed by REINFORCE, where $\bar{r}_\phi(\mathbf{x}, \mathbf{y})$ corresponds to a normalized estimate of the reward model's predictions over a batch of samples drawn from the policy. Using a normalized reward estimate instead of directly the raw reward value helps reduce the variance of the policy gradient estimate as seen in RLOO [1]. High variance gradients slow down convergence and even sometimes lead to sub-optimal solutions in deep RL [17].

For more details:

Here is an additional set of references, looking at RLHF and RL optimization of LLMs:

1. CS 336 Lecture Notes: https://stanford-cs336.github.io/spring2024/
2. CSE 291 (AI Agents): https://pearls-lab.github.io/ai-agents-course/index.html

1.2. Description of Tasks

In this section, we describe the two tasks you will be asked to explore.

Instruction Following Instruction following in a large language model (LLM), as exemplified in Ultrafeedback [6], involves training the model to generate coherent, relevant, and helpful responses when given natural language instructions. This task evaluates how well the LLM can generalize to unseen instructions, maintain factual accuracy, and follow the intent of the prompt across a wide range of domains and difficulty levels.

Math Reasoning Math reasoning, as studied in Countdown [11], focuses on evaluating and improving the model's ability to perform multi-step mathematical problem solving using natural language. In Countdown, a set of numbers and a target number is provided and the model has to logically arrive at the set of operations that would allow us to transform the provided numbers into the target number. This task evaluates the model's ability to plan, decompose, and execute mathematical steps systematically, to arrive at the correct solution.

1.3. Data Loading and Construction (Difficulty : Easy, Time : Medium)

Data is generally the most crucial part of any machine learning system and it is good to interact with the data prior to training with any algorithm on it. In this project, you will explore two types of datasets: (1) preference datasets and (2) verifier-based datasets. In this section, you will be setting up the data pipeline to take a Huggingface dataset (https://huggingface.co/docs/datasets/en/index) and output a batch that can be used to take gradient updates or perform evaluation.

In these sections, we will generate datasets for three algorithms. First, we will need to generate datasets for Supervised Fine-Tuning (SFT), where we warm-start a pre-trained model with a set of desirable behaviors through filtered behavior cloning, where the state is an instruction and the target action is a desirable response. Next, we will generate datasets for preference optimization algorithms such as DPO and reward modeling with Bradley Terry where the model is provided with a prompt, preferred, and dispreferred responses. Finally, we will generate prompts for online RL algorithms such as RLOO, which are provided with a set of prompts for sampling.

Preference Datasets Preference datasets are typically structured as a paired corpus of prompts and responses (preferred responses and dispreferred responses). A standard way to structure such a dataset is with a chat template (https://huggingface.co/docs/transformers/main/en/chat_templating#advanced-adding-and-editing-chat-templates), which can be used to structure the conversation so far during training and evaluation.

Additionally, for language models, the text will need to be tokenized with the tokenizer from the pre-trained model which can allow for the text to be converted into language tokens for the query and completion, respectively. In this project, we will using the Huggingface Tokenizers (check out https://huggingface.co/learn/nlp-course/en/chapter2/4) to perform this tokenization for our dataset. You may want to consider padding and truncation to construct batches for training (see https://huggingface.co/docs/transformers/en/pad_truncation).

Additionally, for the algorithms that we will be employing for instruction following or reasoning, typically, the loss isn't applied to the query tokens during training. Therefore, in your dataset construction, it may be useful to construct an attention mask that will mask out the query tokens to

efficiently implement this.

You should use a dataloader that is efficient to prevent bottlenecks in training. One natural choice for this is a Pytorch Dataloader (https://pytorch.org/tutorials/beginner/basics/data_tutorial.html), which can allow you to sample batches during training to compute a loss function and propogate gradients. Note, a huggingface dataset should be directly compatible with a torch dataloader (https://huggingface.co/docs/datasets/v1.13.0/use_dataset.html).

For preference fine-tuning, we will be working with 2 public datasets:

- **SmolTalk [2] (dataset for SFT)**, a collection of high-quality chat responses from GPT-4o. We will use a filtered subset as found here: https://huggingface.co/datasets/HuggingFaceTB/smol-smoltalk
- **UltraFeedback [9] (dataset for DPO and RLOO)**, a preference dataset to study the instruction following abilities of LLMs (https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized). We will share the same set of prompts for Reward Modeling and Online Rl for optimization.

Verifier-Based Datasets Verifier-Based datasets are typically structured as a corpus of problem and solution pairs. Additionally, a rule-based reward function [7] is employed to check the correctness of the generated response. For verifier-based datasets, we will be working with countdown [11], a task to study the mathematics abilities of LLMs. Here a set of N numbers and a target number is provided and the goal is to recover the operations (e.g $+$, $-$, etc) that recover the target number. Please view the details below to setup the dataloader:

- **WarmStart Dataset (SFT)** [12] a dataset that can be used to increase the base model performance by biasing the on-policy rollouts of the model to desirable strategies such as backtracking and verification. The dataset can be found here: https://huggingface.co/datasets/Asap7772/cog_behav_all_strategies
- **Prompts Dataset (RLOO)** - For online RL, we need a set of prompts for on-policy sampling during optimization. We will use the dataset provided from TinyZero [20] (https://huggingface.co/datasets/Jiayi-Pan/Countdown-Tasks-3to4). We recommend using the same prompt format as the SFT dataset.
- **On-Policy Preference Dataset (DPO)** - Leveraging the same set of prompts as RLOO (https://huggingface.co/datasets/Jiayi-Pan/Countdown-Tasks-3to4), construct an on-policy preference dataset. This can be done by sampling 2 examples from the SFT model with a non-zero temperature and labeling a preferred and dispreferred response using the rule-based reward, filtering out for ties. We recommend using the same prompt format as the SFT dataset. We recommend revisiting this construction after you evaluate SFT using the rule-based reward. For sampling, we recommend using an inference server such as VLLM (https://docs.vllm.ai/en/latest/).

Rule-Based Reward Function You can leverage the rule-based reward function provided by Gandhi et al. [12] found here: https://github.com/kanishkg/cognitive-behaviors/blob/main/verl/utils/reward_score/countdown.py#L59. The reward is a weighted average of a format score and a verification score, where the format score encourages the LLM to provide a parseable answer.

Task Definition Your task is to take the preference and verifier datasets above and construct dataloaders amenable for fine-tuning. For a successful dataloader, consider what elements may be

necessary to use this dataset for SFT and RL. It may be beneficial to consider the methods presented in the next section and revisit your dataloader to ensure that the dataloader is constructed correctly and efficiently for RL fine-tuning.

Advice

Note that most errors stem from the dataset being defined in an ill manner. It may be worth your time to extensively test the dataloaders for both correctness and efficiency to prevent issues that might arise further in the project. We recommend using a smaller instance to test out these changes to save GPU credits.

1.4. Method Implementation (Difficulty : Medium, Time : High)

You will implement a set of algorithms for both parametric and rule based reward models. Note, you **must use the Qwen 2.5 0.5B Base model (https://huggingface.co/Qwen/Qwen2.5-0.5B)** for all experiments in this project. You **may not use any other model** (including the instruct model) as an initialization for fine-tuning for evaluation fairness, and doing so will result in a penalty and disqualify you from any leaderboard credit.

1.4.1. Supervised Fine-Tuning (SFT)

As described in Section 1.1, the first part of any RL pipeline in language is Supervised Fine-Tuning. Supervised Fine-Tuning is the same next-token prediction objective that is used in pre-training. However, no loss is applied to the query tokens. This supervised learning objective is optimized over queries x and completions y that are drawn from an expert distribution. In the context of preference datasets, the completion y is typically the preferred completion y_w and is alternatively referred to as Pref-FT. The objective can be written as follows:

$$\max_{\theta} \mathbb{E}_{x,y\in D} \sum_{t=1}^{|y|} \log \pi_\theta(y_t|x, y_{<t}) \tag{6}$$

Action Item

Implement the supervised fine-tuning objective for a pre-trained language model.

1.4.2. Preference Optimization (DPO)

Preference Optimization Objectives such as DPO parameterize the reward model as a function of the log-likelihood of the responses as seen in Section 1.1. We will be implementing the Direct Preference Optimization (DPO) Objective.

DPO In Rafailov et al. [22], through a clever reward parameterization, the constrained RL problem can be reformulated as a supervised preference classification problem on human preference data. More formally, the DPO loss is:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right]. \tag{7}$$

where x is the prompt, y_w is the preferred responses, y_l is the dispreferred responses, π_θ is the policy that is being optimized and π_{ref} is the reference policy.

Action Item

Implement the DPO objective, using the SFT model as π_{ref}.

1.4.3. Online Policy-Gradient (RLOO)

Several online RL algorithms additionally exist that are popular for fine-tuning language models. We will study one representative choice: REINFORCE Leave One-Out (RLOO) [1]. Additionally, for the preference dataset, where a ground-truth reward is unknown, we will learn a reward model using the Bradley Terry objective.

RLOO RLOO is a policy gradient estimator based on REINFORCE with a baseline to reduce variance of samples that is the weighted average of the reward of other samples from that policy. More formally the objective can be written as follows:

$$\frac{1}{k}\sum_{i=1}^{k}\left[R(y_{(i)},x)-\frac{1}{k-1}\sum_{j\neq i}R(y_{(j)},x)\right]\nabla\log\pi(y_{(i)}|x)\text{ for }y_{(1)},...,y_{(k)}\overset{i.i.d}{\sim}\pi_\theta(\cdot|x) \tag{8}$$

Action Item

Implement the Online Policy-Gradient algorithm of RLOO.

Preference Reward Modeling For preference datasets, we will need to learn a parametric reward function through the bradley terry objective. More formally, the objective can be written as follows:

$$\mathcal{L}_{\text{BT}}(\phi)=\max_{\phi}\ \mathbb{E}_{(\mathbf{x},\mathbf{y}_w,\mathbf{y}_l)\sim\mathcal{D}_{\text{pref}}}\left[\log\sigma\left(r_\phi(\mathbf{x},\mathbf{y}_w)-r_\phi(\mathbf{x},\mathbf{y}_l)\right)\right] \tag{9}$$

Action Item

Implement the Bradley-Terry reward modeling objective.

1.5. Evaluation Setup (Difficulty : Easy, Time : Low)

The next step is to set up an evaluation pipeline that can be used to test the efficacy of preference based and verifier based learning.

Model Sampling We highly recommend the use of a library for LLM inference such as VLLM (https://github.com/vllm-project/vllm) or SGLang (https://github.com/sgl-project/sglang) to make it more efficient for you to sample during evaluation. With an engine like this, you should find more throughput than using huggingface's sampling implementation of 'model.generate()'.

Evaluating Ultrafeedback For UltraFeedback, we can use a parametric reward model for scoring with the Llama 3.1 Nemotron 70B Reward Model (https://huggingface.co/nvidia/Llama-3.1-Nemotron-70B-Reward). We recommend using the hosted inference, where signing up provides you 100k free API calls to this model. To use the hosted inference, you can utilize the OpenAI API with an API Key provided from the link above to evaluate the performance of a given prompt and response:

Listing 1: Evaluating with the Nemotron 70B Reward

```python
from openai import OpenAI

client = OpenAI(
  base_url = "https://integrate.api.nvidia.com/v1",
  api_key = "$API_KEY_REQUIRED_IF_EXECUTING_OUTSIDE_NGC"
)

completion = client.chat.completions.create(
  model="nvidia/llama-3.1-nemotron-70b-reward",
  messages=[
    {"role": "user", "content": "I am going to Paris, what should I see?"},
    {"role": "assistant", "content": "Ah, Paris, the City of Light! There are
        so many amazing things to see and do in this beautiful city ..."}
  ],
)

print(completion) # -19.875
```

Note: For the same prompt, a response with a higher reward score has a response of higher quality than another response with a lower reward score, but the same cannot be said when comparing the scores between responses to different prompts. Thus, we recommend using the following approach for evaluation: score the learned model/policy's response with the reward model and compare the reward/score with a reference model's response to compute a win-rate, computed as follows:

1. Collect a set of prompts to evaluate
2. Then for each prompt, use the inference engine (e.g VLLM/SGLang) to sample responses for your trained model and a reference model (Qwen 2.5 0.5B Instruct)
3. Generate a reward score through Nemotron to evaluate the reward score for both your trained and reference models
4. For each prompt, construct a per-prompt win-rate binary label, where 1 corresponds to the reward of the trained model being higher and 0 corresponds to the reward of the reference model being higher.
5. Calculate the winrate as the average of the binary label over all prompts.

Evaluating Countdown Following TinyZero [20], we can use a two stage reward to evaluate the answer from the countdown task: (1) a format score to see if any answer was provided and (2) a verification score to check if the responses is correct. [1]

Expected Win-Rate and Accuracy Requirements for Ultrafeedback and Countdown For Ultrafeedback, we expect a **60% win-rate** or higher with DPO and RLOO on Ultrafeedback. For countdown, we expect a score of **0.3** or higher. Additionally, with RLOO and DPO, we expect a higher reported score than the SFT initialization. Note, that scoring requirements are subject to change and reference Ed for the latest information. The evaluation will be done with the leaderboard submission, which incorporates a held-out, private evaluation.

[1] https://github.com/kanishkg/cognitive-behaviors/blob/main/verl/utils/reward_score/countdown.py

2. Possible Extensions to the RL Methods Explored (Difficulty: High, Time: High)

Now, that we have explored some common instantiations for policy gradient objectives used in Fine-Tuning of Language Models, several approaches can be considered to improve the performance of these algorithms further. Below, we will provide a list of suggestions that you can consider to pursue as an extension of your choosing:

2.1. Synthetic Data Augmentation

Data is a crucial component of any ML system today. Two limiting factors in RL optimization that have been surfaced is (1) the initialization of the policy (Base/SFT model), where a poor initialization leads to poor downstream performance (see 'aha' moment in DeepSeek-AI et al. [7]) or (2) a reduction in diversity as RL training progresses which leads to model capacity challenges(see Pal et al. [19], Tajwar et al. [33]). One approach to address this is to increase the dataset diversity, which can be done through synthetic data augmentation. Consider how a diverse dataset can be efficiently collected to reduce issues in RL optimization, related to model initialization/capacity.

Some references to consider are Bai et al. [3], Lee et al. [16], Dong and Ma [8].

2.2. Improving Efficiency through Off-Policy Sampling

For online policy-gradient objectives such as RLOO or GRPO, the policy is typically trained in an on-policy fashion, where at each step samples from the current policy are used to update the policy. However, can we relax this by performing more gradient updates per sample from the policy? For environments such as Robotics or Agentic Coding Tasks, where sampling from the environment is difficult, extensive on-policy sampling is undesirable. Consider how the efficiency of the algorithm changes (performance vs sample reuse) as you consider approaches that are more off-policy.

Some references to consider are Tang et al. [34], Bartoldson et al. [4], Roux et al. [23].

2.3. Incorporating Test Time Inference

Once a policy is trained, we can offload some of the computation during test time to make the policy more robust. Explore strategies to make the policy more robust using a verifier or reward function for the Countdown and Ultrafeedback tasks. For UltraFeedback, note that there isn't a ground-truth reward function which may make scoring difficult/ill-formed. Are there strategies to make this work effectively, even when your reward function is underspecified?

Some references to consider are Snell et al. [29], Wang et al. [35], Zhang et al. [38].

2.4. Multi-Objective Optimization/Personalization

Preference-based RL usually focuses on optimizing a single scalar reward. Investigate multi-objective RL formulations where the policy simultaneously optimizes multiple signals, such as fluency, coherence, factual correctness, or style preservation. How do various scalarization methods (weighted sums, Pareto-optimality, scalarized utility functions) influence the policy training and downstream performance? Consider how rlhf may marginalize/under-represent certain users.

Some references to consider are Sorensen et al. [30], Meister et al. [18], Singh et al. [28].

2.5. Exploration

Language-based RL suffers from exploration inefficiencies due to the vast combinatorial space of language tokens. Consider developing novel exploration strategies specifically tailored for discrete token-based action spaces, potentially using intrinsic motivation signals (such as novelty detection, uncertainty estimation, or semantic embedding space exploration). Evaluate how these exploration strategies accelerate policy optimization and enhance generalization.

Some references to consider are Qu et al. [21], Xie et al. [37], Foster et al. [10].

2.6. Self-Play/Multi-Agent RL

One fundamental challenge in reasoning domains for LLMs is the lack of high-quality training data. Although RL partially mitigates this issue by alternating between the LLM generating solutions and finetuning them on correctly generated ones, performance quickly plateaus due to the scarcity of correct solutions (sparse rewards). To keep improving the models with limited data, one strategy that can be employed is multi-agent RL where cooperative/adversarial agents can be constructed to allow the for more diversity in the scenarios that are potentially explored in the environment and allow the model to improve based on the experiences of other agents in the environment.

Some references to consider are Dong and Ma [8], Subramaniam et al. [32], Silver et al. [27].

2.7. Effective Tool Use

Tools are functions that allow large language models (LLMs) to interact with external applications. They provide an interface for LLMs to access and retrieve information from outside sources. These tools can enable models to become more robust to queries that necessitate precision (e.g a calculator helping with math queries) or need up to date information (e.g a weather tool). One instantiation of this is that when you ask LLM a question that requires tool assistance, the model looks for the tools it has, and if a relevant one is found based on the tool name and description, it halts the text generation and outputs a structured response. This response, usually a JSON object, contains the tool's name and parameter values deemed fit by the LLM model. Now, you can use this information to execute the original function and pass the output back to the LLM for a complete answer. Explore, how you can optimize for tool use with RL, where a tool can help with queries where it is necessary.

Some references to consider are Jin et al. [15], Gehring et al. [13], Schick et al. [24].

2.8. Learning with a Curriculum

Curriculum learning has demonstrated effectiveness across numerous machine learning tasks. For instance, beginning with simpler tasks and progressively introducing more complex ones helps models incrementally develop robust knowledge. In reinforcement learning (RL), this can entail creating a dynamic curriculum of transitions for optimization, where the complexity or diversity of training examples adapts in real-time according to the policy's current performance.

Some references to consider are DeepSeek-AI et al. [7], Gandhi et al. [12], Soviany et al. [31].

2.9. Alternative Objectives and Policy Constraints

The preference optimization and online RL algorithms we selected are a small, representative subset of the algorithms considered. Consider tradeoffs of how different objectives and policy constraints would perform and benchmark them on a task of your choosing.

Some references to consider are Shao et al. [26], Zheng et al. [39], Xie et al. [37].

2.10. Choose Your Own Adventure

Consider an extension of your choosing that would ultimately lead to better downstream performance on either task: Math Reasoning (Countdown) and Instruction Following (Ultrafeedback).

3. Default Project Leaderboard

We will additionally host a leaderboard to compare approaches from different submissions. For the leaderboard, for fairness, we will **require use of the Qwen 2.5 0.5B (non-instruct) model for all submissions**. We will have a hidden evaluation set for fair comparison across all submissions. You can use the public validation set as a rough heuristic of what performance should be like on the private validation set as the queries should be from a similar distribution.

To score high on the leaderboard, we recommend iterating heavily on the design of your extension direction considering where potential limitations lie in your approach, tuning hyperparameters, and potentially collecting more data (synthetically). The top submissions in both Instruction Following (Ultrafeedback) and Math Reasoning (Countdown) will receive extra credit on the project.

More details coming soon on Ed! This will include details such as how to submit to the leaderboard, where to access the leaderboard, the number of total submissions, any potential errors if the leaderboard submission was unsuccessful, and up-to-date minimum thresholds for the basic implementation and extensions.

Advice on Leaderboard Submissions

You will have a limited number of submissions to the leaderboard, so please test out your model on the public validation set and be confident about your submission prior to any attempt.

4. Important Information

4.1. Project Mentors

All final projects will be assigned **a single mentor CA** to serve as a point of contact. Your project mentor should be your primary point of contact throughout the quarter as you work on your project. This individual will also be the person who ultimately grades your project milestone and final reports. Once your project mentor has been assigned, you should not visit other CAs at office hours to discuss your final project since much of your time will likely be spent simply bringing them up to speed; instead, **you should visit your project mentor's office hours for project-related issues and challenges**.

In your response to the project survey (Section **??**), you may choose to indicate specific members of the teaching staff who you think compliment your interests and/or are well-suited to guide your particular proposed project. To help you with this, listed below are the broad specialities of each CA (alphabetical order):

- **Andy Tang**: Robot Learning, Generalization, Exploration
- **Anikait Singh**: Imitation Learning, Offline RL, Meta RL, LLM Reasoning and Alignment
- **Annie Chen**: Imitation Learning, Offline RL, Continual/Lifelong RL, Robot Learning, Robustness
- **Ashish Rao**: Online RL, Continual Learning, LLMs Expressivity, sci-ML
- **Daniel Shin**: Offline RL
- **Fengyu Li**: RL, LLM

- **Haoyi Duan**: 3D Vision
- **Jensen Gao**: Imitation Learning, Offline/Online RL, Robot Learning
- **Jinny Chung**: Offline RL, MDP, Model-based RL, Bio-ML, Generative Models
- **Joy He-Yueya**: Imitation Learning, LLM Agents
- **Jubayer Ibn Hamid**: Imitation Learning, Representation Learning, Online RL, Robot Learning
- **Marcel Torne**: RL, Robot Learning
- **Pulkit Goel**: RL, Imitation Learning
- **Sergio Charles**: Generative Models, Graph Neural Networks, Autonomous RL
- **Shirley Wu**: RLHF, LLM Agents, Compound AI Systems
- **Sirui Chen**: Robot Learning
- **Sri Jaladi**: Imitation Learning, offline RL, Vision, Graph-based Models
- **Yash Kankariya**: Robotics
- **Zhen Wu**: Robot learning

Clearly, this list of reinforcement-learning topics is **not exhaustive** and, furthermore, each CA likely has expertise that extends beyond what is listed. That said, even if you find yourself matched with a project mentor whose specialities do not overlap with your chosen project area, your mentor is still broadly well-versed in reinforcement learning, well aware of what constitutes a solid final project for the course, and is ready to help you to the best of their abilities. We anticipate that some final projects will fall outside our areas of expertise as a teaching staff, and that is okay so long as there is clear connection to and application of core course topics.

You should think of your project mentor as an individual who can provide you with feedback as you hit checkpoints in your final project; it would not be in your best interest to seek out advising from your project mentor on a week-by-week basis. Instead, **consult your mentor when you have uncertainty on the path ahead or there are major decisions to be made** which can dramatically impact the contents of your final report.

4.2. Project Groups

Final projects for the course may be completed by **individual students or in teams of no more than 3 people**. We encourage everyone to work in groups with the expectation that the overall contributions of the final project be commensurate with the group size.

While the project survey will ask that your group for CA matching purposes, project groups are mutable up to and until the deadline of the project proposal. In other words, **your group is locked in once you submit your project proposal.**

4.3. Contributions

To help us (and yourselves) keep track of individual workloads, we ask that project proposals include **an anticipated breakdown of individual roles and responsibilities**. We will also ask for your final report to include a **similar breakdown of individual contributions** as well. It is perfectly fine for roles to adapt and change over the course of the project including deleted roles and newly added roles between the proposal and final report. If there are any changes in responsibilities, we ask that they be documented in the final report so we can ensure an equitable division of labor among group members.

Project groups are welcome to work with people not enrolled in the class, however we expect the contribution breakdowns in the project proposal and final report to reflect the work of all members involved (both internal and external). Final projects may be shared between CS224R and another class however, in this case, you should clearly indicate this in your project proposal and we will expect

a more ambitious project.

4.4. Honor Code and AI Tools Policy

You should work as a team independently and ensure academic integrity by clearly documenting which parts of your code or ideas were assisted by AI tools and which parts were developed independently.

For the **default project**: You are **not allowed to collaborate with AI tools** such as GitHub Co-Pilot and ChatGPT on the initial implementation. However, in the extension these tools can help you generate boilerplate code.

However, to maximize learning through active implementation, we expect that you use them **only as an aid rather than the complete solution**. For example, it is acceptable to leverage AI for standard coding tasks (e.g., constructing a data pipeline or debugging minor code issues). For essential components of your project, such as implementing deep reinforcement learning algorithms (e.g., Proximal Policy Optimization (PPO)), you should write the code independently. This ensures that you develop a deep understanding of the underlying principles and techniques.

Note: TAs may not look at students' code for either the default or custom final projects.

4.5. Submission Format

We recommend submitting proposals and reports using the provided latex templates below. This ensures consistent formatting for all submissions.

- Project Proposal: CS224R Project Proposal Template
- Milestone: CS224R Project Milestone Template
- Final Project: CS224R Final Report Template

For poster presentations, you are free to choose templates that effectively communicate your research findings, such as Stanford-LaTeX-Poster-Template. All submissions must adhere to the specified page limits in the instructions.

4.6. Using Late Days

With the exception of the project poster (which **cannot be submitted late** due to the live poster session) and the final project report (which **cannot be submitted late** due to the University deadline to submit final grades for the quarter), entire project groups may apply a maximum of two late days to any other graded component of the final project, for example, project milestone. Groups cannot aggregate collective late days and late days will be applied to all group members; this means that, to apply two late days, each group member must have two unused late days to spare. Assignment extensions granted by the Office of OAE do not apply to group assignments. However, if you are working individually they can apply.

4.7. Computing Resources

We will be providing **$100 in cloud computing credits** to each student for use on homework assignments and the final project, which should amount to more than 200 hours of GPU time (and much more CPU time), depending on the size of the GPU. Details about how to access these credits are forthcoming.

5. Deliverables and Grading

For the default project, we will have the same set of deliverables as the custom project. The project will count for **50**% of the course grade, broken down as follows:

- Project proposal – 10%
- Project milestone – 10%
- Poster presentation – 10%
- Final project report – 20%

5.1. Survey (4/16):

Everyone, either individually or as part of a larger group, should submit the Project Survey which will allow us to collect preliminary information on final project topics, determine the total number of groups, and assign project mentors. While project groups should, ideally, be finalized by this stage, we will allow group changes up to the project proposal deadline. Please clarify that you and your group are doing the default project, and not a custom project.

5.2. Project Proposal (4/25):

Focus your proposal on the extension of choice proposing a relevant idea or technique that could be useful to solve your extension of choice. Provide some evidence to why this may be an appropriate strategy to solve the extension of your choice.

Your proposal should have the following structure:

1. **Extension** 1/4 page. Explain the desired extension you wish to pursue in the default project and why the extension is relevant and important.
2. **Related Work** 1/2 page. Review the most relevant prior work, and highlight where those works fall short of meeting the extension described above.
3. **Technical Outline of Extension** 1/2 page. Explain your approach at a high-level, making clear the novel technical contribution, and describe the evaluation plan.
4. **Team Contributions** 1/4 page. Include a brief, high-level breakdown of the planned contributions of each individual group member.

Submission & Grading – Submit one proposal PDF per group to gradescope under the "Project Proposal" assignment that includes the (finalized) list of project group members. While the deadline for the project proposal is in late-April, we strongly encourage you to **begin developing your idea for the project sooner**. The grading for the proposal will be primarily intended to give you feedback on how to adjust/pivot your proposal to meet the project requirements.

Modifying Contributions – It is perfectly fine for items in your breakdown of team member contributions to remain, adapt, or even disappear between submitting the proposal and the final project. Our goal is to get as even a distribution of workload as possible amongst group members. Any changes to this list of contributions that you do make after submitting the proposal should be documented in your final project report.

5.3. Milestone (5/23):

Your milestone report should be a one-page document that showcases the completion of the initial implementation in section 1. This entails validating the data loading, implementation, and evaluation

of the algorithms referenced in section 1. Think about what metrics would be useful to present here when evaluating on the two tasks.

In your report, you are required to show the quantiative and qualitative performance of your policy, effectively reporting the quantitative metrics against the relevant baselines. Finally, submit your initial checkpoints for SFT, DPO, and RLOO to the leaderboard for both the Countdown and Ultrafeedback tasks (anonymously if you wish) and share your unique id in the report for review.

Submission & Grading – Submit one one-page milestone report PDF per group, with the names of all project members, to gradescope under the assignment "Milestone Report." The milestone report will be graded by your project mentor. The grading of the milestone report will be light as the primary goal is to provide you feedback so that there is a clear path going forward to the poster presentation and final report.

5.4. Poster Presentation (6/4)

Students are expected to prepare a research poster that **focuses on the extension that was explored in Section 2**. Ensure that the poster encapsulates the following details:

1. What is the problem?
2. Why is the problem interesting and important?
3. What are the key findings from my experiments?
4. What can audiences learn from this work? What is the limitation / future work?

Poster Dimensions – We recommend printing posters in landscape orientation with size 24" $\times$ 36". While other sizes are acceptable, this size will work best with the poster boards and easels we will provide for you at the poster session, so please make sure your poster isn't too much bigger or smaller than this.

Poster Printing – You are not required to do commercial poster printing; you can print the poster on smaller sheets (letter size or A4 size) using regular printers, and tape them together. Adobe Acrobat Reader automates this— look for the Poster options in the Print dialog.You are expected to print out the posters yourself, so please plan ahead! Here are some options available for 20"x30" poster printing. For each option, make sure to verify turnaround and services yourself—the info below is just gathered from the corresponding websites!

Some resources for printing:

- Lathrop Library Tech Desk: order online for a 3-day turn around.
- FedEx: approximately 2-day turnaround. Offers customizability (e.g. laminate the poster for greater durability; add grommets).
- CVS Photo: Affordable, same-day pick up on some options; can use code 50SITE for offer.
- Walgreens: claims to have same-day pickup for 24" x 36".
- Biotech Productions: specialized for research posters and offers same-day free delivery to Stanford campus.
- Walmart: Different printing options and templates; home-delivery option

Poster Exemptions – At least one group member must be present at the poster session, unless you have reached out via email to the course staff and received special permission to submit a video presentation. The only exception is if the project group is made up entirely of (remote) SCPD students, in which case, the group will be expected (by default) to submit a five-minute video presentation of

their poster instead. A Google Drive or YouTube link pointing to the video should be uploaded to gradescope under the “Project Poster Video” assignment.

Location & Time – The poster session will be held on 6/4/25, 9a.m-3p.m at the Burnham Pavilion.

5.5. Report (6/9):

The final report should be in the style of a research paper, preceded by the following:

1. A one-page extended abstract: The one-page extended abstract should summarize the main findings and accomplishments of your final project. The extended abstract should be attached as the first page of the full report.
2. The main paper: It should describe the extension in detail, discuss relevant prior work and how the project makes new contributions relative to these prior works, and discuss the results, including any relevant figures or plots.

Formatting & Contributions – Successful reports will have a main body that is about eight pages in length, but there are no hard length requirements or formatting requirements, except the one-page extended abstract. For project groups, please include a fully-updated breakdown of the contributions of individual team members and highlight why these adjustments were necessary from your original breakdown in the project proposal; this is as much a reflection on the research process for you as it is a tool for us to track equitable division of labor.

Submission & Grading – Submit one final report per group with the names of all project members to gradescope under the assignment “Final Project Report."

Leaderboard The leaderboard also closes on 6/9, synced with the deadline of the final report. Please submit your model for your extension to the leaderboard (optionally anonymously) and list your submission of choice to the final report for consistent evaluation of your model.

6. Expectation of the Default Project

The default project requires implementing, evaluating, and documenting a novel extension for the RL fine-tuning of language models. Note that the extension is half of the project grade and requires careful thought into its construction.

The *novelty* should entail exploration of an unanswered or partially answered problem (potentially guided by the sample extension directions presented in Section 2). This may include exploring or developing a new algorithmic idea that can lead to better performance, higher efficiency, or instill desirable properties into the language model that were not present before. Often students think that the goal is to build the highest performing model instead of doing science to figure out strengths and weaknesses of whatever they have tried. **Achieving state-of-the-art performance is not required for this project**. That being said, you are welcome to explore ideas and reason about approaches that will allow you to perform well on the leaderboard.

In light of the above goals, students are expected to prepare a proposal, prepare a milestone report, present their work in the final poster session, and submit the final report, with specific details as outlined in this document.

References

[1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024. URL https://arxiv.org/abs/2402.14740.

[2] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022. URL https://arxiv.org/abs/2212.08073.

[4] Brian R. Bartoldson, Siddarth Venkatraman, James Diffenderfer, Moksh Jain, Tal Ben-Nun, Seanie Lee, Minsu Kim, Johan Obando-Ceron, Yoshua Bengio, and Bhavya Kailkhura. Trajectory balance with asynchrony: Decoupling exploration and learning for fast, scalable llm post-training, 2025.

[5] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 00063444. URL http://www.jstor.org/stable/2334029.

[6] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with scaled ai feedback, 2024. URL https://arxiv.org/abs/2310.01377.

[7] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng

Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[8] Kefan Dong and Tengyu Ma. Stp: Self-play llm theorem provers with iterative conjecturing and proving, 2025. URL https://arxiv.org/abs/2502.00212.

[9] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback, 2024.

[10] Dylan J. Foster, Zakaria Mhammedi, and Dhruv Rohatgi. Is a good foundation necessary for efficient reinforcement learning? the computational role of the base model in exploration, 2025. URL https://arxiv.org/abs/2503.07453.

[11] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (sos): Learning to search in language, 2024. URL https://arxiv.org/abs/2404.03683.

[12] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL https://arxiv.org/abs/2503.01307.

[13] Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. Rlef: Grounding code llms in execution feedback with reinforcement learning, 2025. URL https://arxiv.org/abs/2410.02089.

[14] Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A General Theoretical Paradigm to Understand Learning from Human Preferences. *arXiv e-prints*, art. arXiv:2310.12036, October 2023. doi: 10.48550/arXiv.2310.12036.

[15] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

[16] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback, 2024. URL https://arxiv.org/abs/2309.00267.

[17] Jincheng Mei, Wesley Chung, Valentin Thomas, Bo Dai, Csaba Szepesvari, and Dale Schuurmans. The role of baselines in policy gradient optimization. *Advances in Neural Information Processing Systems*, 35:17818–17830, 2022.

[18] Nicole Meister, Carlos Guestrin, and Tatsunori Hashimoto. Benchmarking distributional alignment of large language models, 2024. URL https://arxiv.org/abs/2411.05403.

[19] Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. Smaug: Fixing failure modes of preference optimisation with dpo-positive, 2024. URL https://arxiv.org/abs/2402.13228.

[20] Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

[21] Yuxiao Qu, Matthew Y. R. Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning, 2025. URL https://arxiv.org/abs/2503.07572.

[22] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[23] Nicolas Le Roux, Marc G. Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, Alex Fréchette, Carolyne Pelletier, Eric Thibodeau-Laufer, Sándor Toth, and Sam Work. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms, 2025. URL https://arxiv.org/abs/2503.14286.

[24] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[26] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[27] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

[28] Anikait Singh, Sheryl Hsu, Kyle Hsu, Eric Mitchell, Stefano Ermon, Tatsunori Hashimoto, Archit Sharma, and Chelsea Finn. Fspo: Few-shot preference optimization of synthetic preference data in llms elicits effective personalization to real users, 2025. URL https://arxiv.org/abs/2502.19312.

[29] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

[30] Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, Tim Althoff, and Yejin Choi. A roadmap to pluralistic alignment, 2024. URL https://arxiv.org/abs/2402.05070.

[31] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey, 2022. URL https://arxiv.org/abs/2101.10382.

[32] Vighnesh Subramaniam, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains, 2025. URL https://arxiv.org/abs/2501.05707.

[33] Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data, 2024. URL https://arxiv.org/abs/2404.14367.

[34] Yunhao Tang, Taco Cohen, David W. Zhang, Michal Valko, and Rémi Munos. Rl-finetuning llms from on- and off-policy data with a single algorithm, 2025. URL https://arxiv.org/abs/2503.19612.

[35] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

[36] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992.

[37] Tengyang Xie, Dylan J. Foster, Akshay Krishnamurthy, Corby Rosset, Ahmed Awadallah, and Alexander Rakhlin. Exploratory preference optimization: Harnessing implicit q*-approximation for sample-efficient rlhf, 2024. URL https://arxiv.org/abs/2405.21046.

[38] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2025. URL https://arxiv.org/abs/2408.15240.

[39] Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. Secrets of rlhf in large language models part i: Ppo, 2023. URL https://arxiv.org/abs/2307.04964.