Literature Review - Synthesis of Sources
From RLHF to GRPO - Optimization Techniques for LLM Reasoning

1. RLHF & Challenges

Reinforcement Learning with Human Feedback (RLHF) has emerged as a leading paradigm for aligning large language models (LLMs) to human preferences, significantly advancing the interpretability and controllability of generative models. The seminal paper by Ouyang et al. (2022), known as InstructGPT, implemented RLHF using Proximal Policy Optimization (PPO), an advanced policy-gradient method (Schulman et al., 2017). Despite its widespread adoption, PPO-based RLHF exhibits several intrinsic limitations when applied to reasoning-intensive, long-horizon tasks (Lyu et al., 2023).

- Reward Model Dependency: PPO requires a separate reward model, $R_\phi(y|x)$, trained on human-labeled preference pairs. Such learned models are prone to overfitting, annotator bias, and can induce reward hacking (Askell et al., 2021).
- Value Estimation Challenges: PPO relies on estimating a value function $V_\Psi(x)$ to stabilize policy updates. However, estimating value functions in high-dimensional token-output spaces frequently results in instability or inaccurate credit assignment, especially in long-horizon settings (Engstrom et al., 2020).
- KL-Penalty Constraints: PPO employs KL-divergence regularization against a pretrained model distribution, constraining exploration and reducing the model's ability to discover novel reasoning trajectories (Ziegler et al., 2019).

To alleviate some of these issues, Rafailov et al. (2023) proposed Direct Preference Optimization (DPO). DPO eliminates explicit reward modeling and value estimation by directly optimizing the policy using pairwise comparisons of completions. Formally, DPO maximizes the likelihood ratio of preference pairs:

$$max_\theta E_{(x, y_w, y_l)\sim D}\left[log\sigma\left(\beta\, log\frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}\right)\right]$$

where y_w and y_l denote preferred and dispreferred completions respectively, and σ denotes the logistic sigmoid. While computationally simpler, DPO still collapses complex multi-step reasoning to pairwise scalar signals. This limitation becomes acute for reasoning tasks where correctness spans entire sequences rather than isolated tokens.

2. Group Relative Policy Optimization (GRPO)

Addressing the short-horizon limitation of scalar-based RLHF methods, Group Relative Policy Optimization (GRPO), introduced by DeepSeekMath (2024), reframes the optimization problem by ranking a sampled set of completions. The fundamental optimization objective of GRPO can be stated formally as:

- For each prompt x, sample a group of k completions $Y = \{y^{(1)}, y^{(2)}, \dots, y^{(k)}\}$.

- Rank these completions according to an external scoring function $R(y|x)$ reflecting correctness or reasoning quality (e.g., symbolic correctness, logical consistency, or partial credit).
- Update the policy parameters θθ by minimizing the following ranking-based loss:

$$L_{GRPO}(\theta) = -\sum_{i=1}^{k} w_i log\pi_\theta(y^{(i)}|x),$$

where w_i are monotonically decreasing rank weights (e.g., exponential or linear weights ensuring higher-ranked completions receive stronger gradient signals). Crucially, this group-based ranking bypasses the need for a learned reward model or value estimator, relying instead on deterministic or verifiable scoring criteria.

DeepSeek-R1 (2025) operationalized GRPO with enhancements, including adaptive rank reweighting, improved sampling heuristics (e.g., top-p or diverse beam sampling), and explicit compatibility with open-weight LLM architectures. Subsequently, GRPO was integrated into the Unsloth (2025) training framework, optimized specifically for LLaMA-family models. Unsloth delivers high-throughput sampling, memory-efficient fine-tuning via Low-Rank Adaptation (LoRA, Hu et al., 2021), and GPU pipeline optimization, significantly lowering the computational barriers to implementing GRPO.

In our project, we utilize the Unsloth-GRPO stack for reasoning-intensive mathematical tasks, particularly standardized test-style benchmarks such as MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021). These benchmarks emphasize long-horizon correctness, trajectory-level coherence, and structured logical derivations—attributes well-aligned with the GRPO optimization formulation.

3. Symbolic Reasoning & Execution-Guided Feedback

Parallel to developments in RL optimization, symbolic execution-based feedback has emerged as a rigorous and verifiable alternative to subjective human preference labeling. Early execution-guided reasoning frameworks, such as Toolformer (Schick et al., 2023) and ReAct (Yao et al., 2023), demonstrated that coupling LLM outputs to symbolic or code-based environments substantially improves accuracy and interpretability.

Further refinement arrived through Program of Thoughts (PoT, Chen et al., 2022) and Program-aided Language Models (PAL, Gao et al., 2022), where problems are translated to executable pseudo-code or Python expressions. Minerva (Lewkowycz et al., 2022) specifically trained LLMs on math-code generation tasks, executing generated Python code to verify outputs directly.

We explicitly extend this symbolic-execution lineage in our project by defining a deterministic oracle based on SymPy. Specifically, each sampled reasoning trajectory y is parsed via regular expressions into symbolic expressions. The correctness criterion is then rigorously formalized as:

$$R(y|x) = \begin{cases} 1, & \text{if SymPyEvaluate}(y) = A^*, \\ 0, & \text{otherwise} \end{cases}$$

where A^* denotes the verified symbolic or numerical ground truth solution. This verifiable feedback approach directly aligns with Cao et al. (2023)'s argument that verifiability provides a more accurate supervision signal than scalar preference ratings, particularly for tasks demanding exact reasoning or proof-like solutions.

Furthermore, we explore partial-credit evaluation inspired by Zelikman et al. (2022) and Wang et al. (2023):

$$R_{\text{partial}}(y|x) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\{s_t = s_t^*\},$$

where s_t are intermediate symbolic steps and s_t^* their corresponding ground-truth derivations. This fine-grained reward shaping complements GRPO's group ranking approach, enabling incremental improvement in intermediate reasoning fidelity.

4. Computational Efficiency: LoRA & Unsloth Framework

While GRPO presents significant theoretical advantages for symbolic reasoning tasks, its practical numerical implementation faces daunting computational challenges. Unlike conventional scalar-reward reinforcement learning, GRPO demands evaluating multiple candidate trajectories per prompt. Each sampled completion must be fully generated, token-by-token, parsed into symbolic form, and evaluated using symbolic mathematics libraries such as SymPy. Consequently, runtime and memory complexity scale as $O(kT)$, where k is the number of sampled trajectories (typically $k = 8$) and T the token-length of each trajectory (often thousands of tokens for standardized test-style reasoning tasks). To address this, we leverage two critical advancements:

- LoRA (Low-Rank Adaptation) (Hu et al., 2021; Dettmers et al., 2023): LoRA dramatically reduces computational overhead during fine-tuning by introducing low-dimensional parameter adapters into transformer attention layers. Rather than updating billions of dense model parameters directly, LoRA restricts updates to low-rank matrices (ranks typically between 8–64), reducing memory consumption from billions to millions of parameters. This memory-efficiency is crucial for maintaining a practical GPU footprint during iterative GRPO updates.

- Unsloth (2025): Unsloth provides specialized computational optimizations tailored explicitly for efficient GRPO fine-tuning with LLaMA-based architectures. Unlike generic RL or transformer fine-tuning frameworks, Unsloth directly targets the VRAM and throughput challenges posed by multi-trajectory sampling, ranking, and long-context generation inherent in GRPO. It achieves this through several advanced innovations:

1. Efficient Linear-Memory GRPO Algorithm:
    - Employs a linear memory algorithm inspired by Horace He's linear cross-entropy implementation, achieving approximately a 90% VRAM reduction compared to traditional implementations (e.g., TRL with Flash Attention 2). This drastically lowers VRAM requirements—essential for long sequences (20K tokens) and multiple sampled completions (8 per input).
2. Advanced Gradient Checkpointing via Unified Memory:
    - Implements an advanced gradient checkpointing strategy that asynchronously offloads intermediate transformer activations from GPU VRAM to CPU RAM (unified memory). This optimization alone saves around 372GB of VRAM at the cost of a negligible performance slowdown (~1%).
3. Optimized Multi-Completion Sampling and Ranking Kernels:
    - Integrates highly optimized CUDA kernels to fuse token sampling, log-probability scoring, and ranking operations, significantly improving GPU utilization and throughput. Critically, symbolic parsing and numeric scoring (via SymPy) are conducted after completions are generated, ensuring efficient GPU operation while maintaining precise symbolic correctness evaluation externally.
4. Unified GPU/CUDA Memory Space Utilization:
    - Shares GPU memory between inference and training engines (vLLM integration), further reducing VRAM overhead by around 16GB. This integration streamlines the pipeline, further enhancing efficiency and memory management.

By integrating these targeted optimizations, Unsloth allows GRPO fine-tuning under practical hardware constraints, effectively supporting long-context (up to 20,000 tokens per completion) and multiple-trajectory training (8-way sampling) on a single A100 GPU.

Other/Future Work