ARCANE: A Multi-Agent Framework for Interpretable and Configurable Alignment

Charlie Masters[1], **Marta Grześkiewicz**[2], **Stefano V. Albrecht**[1]

[1]DeepFlow, London, United Kingdom
[2]University of Cambridge, United Kingdom

Abstract

As agents based on large language models are increasingly deployed to long-horizon tasks, maintaining their alignment with stakeholder preferences becomes critical. Effective alignment in such settings requires reward models that are interpretable so that stakeholders can understand and audit model objectives. Moreover, reward models must be capable of steering agents at interaction time, allowing preference shifts to be incorporated without retraining. We introduce ARCANE, a framework that frames alignment as a multi-agent collaboration problem that dynamically represents stakeholder preferences as natural-language rubrics: weighted sets of verifiable criteria that can be generated on-the-fly from task context. Inspired by utility theory, we formulate rubric learning as a reconstruction problem and apply a regularized Group-Sequence Policy Optimization (GSPO) procedure that balances interpretability, faithfulness, and computational efficiency. Using a corpus of 219 labeled rubrics derived from the GDPVal benchmark, we evaluate ARCANE on challenging tasks requiring multi-step reasoning and tool use. The learned rubrics produce compact, legible evaluations and enable configurable trade-offs (e.g., correctness vs. conciseness) without retraining. Our results show that rubric-based reward models offer a promising path toward interpretable, test-time adaptive alignment for complex, long-horizon AI systems.

1 Introduction

As AI agents based on large language models (LLMs) take on longer-horizon, project-scale tasks with multiple specialized agents, maintaining alignment with stakeholder preferences becomes increasingly critical (Masters et al. 2025). Prior research has shown how cooperation can falter under partial observability, shifting incentives, or social feedback loops (e.g. Leibo et al. 2017; Jaques et al. 2019; Carichon et al. 2025). Recent studies on LLM-based debate and collaborative reasoning echo these dynamics: joint systems can exhibit sycophancy (Cau et al. 2025), persuasion failures (Wynn et al. 2025), and collective bias amplification (Ashery et al. 2025). Thus, alignment becomes a *system-level objective* that must hold under delegation, communication, and co-adaptation.

Recently, *rubrics* have emerged as a promising paradigm for evaluating and improving the alignment of agentic systems with human preferences. Rather than optimizing against a single opaque reward model, rubric-based approaches define structured natural-language criteria that decompose preferences into interpretable and verifiable dimensions, such as factual accuracy, reasoning depth, and clarity. Recent work has explored rubrics both as evaluation instruments and as training-time objectives: for example, using them directly as reward functions during reinforcement fine-tuning (Gunjal et al. 2025), or as expert evaluators in professional and educational domains (Wang et al. 2025b; Anghel et al. 2025). Large-scale frameworks such as *OpenRubrics* (Liu et al. 2025b) further demonstrate that rubrics themselves can be synthetically generated, enabling scalable, domain-specific supervision without extensive human labeling. Together, these developments suggest that rubrics can serve as interpretable, multi-dimensional reward models capable of guiding complex behavior.

However, existing methods largely treat rubric generation and rubric-conditioned optimization as separate processes, assuming rubrics are given rather than learned through interaction with stakeholders. This limits their adaptability to evolving or multi-objective preference contexts. To address this gap, we propose to frame rubric generation itself as a policy-optimization problem within a multi-agent alignment setting. Specifically:

- We design ARCANE (**A**daptive **R**ubric-based **C**ontrol of **A**gents via **N**atural-language **E**xchange),[1] a rubric-based alignment framework in which a manager agent learns to generate rubrics that align worker agents with stakeholder utilities through interactive preference elicitation.
- We propose a two-stage learning procedure to optimize the manager agent policy: initial supervised fine-tuning using synthetic manager–stakeholder dialogues, followed by reinforcement fine-tuning via regularized *Group Sequence Policy Optimization (GSPO)* (Zheng et al. 2025), incorporating penalty terms for rubric complexity and stakeholder interaction cost.
- We evaluate ARCANE on tasks from the GDPVal benchmark (Patwardhan et al. 2025), empirically showing that

[1]Link to ARCANE code can be found in the MA-Gym readme: https://github.com/DeepFlow-research/manager_agent_gym

the learned rubrics (1) effectively guide workers to maximize stakeholder preferences at test-time; (2) preserve rankings over outputs from the stakeholder preferences, and (3) are interpretable and effectively verifiable.

2 Related Work

Training-Time Alignment. Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al. 2022; Stiennon et al. 2022) remains the dominant paradigm for aligning language models with human preferences. Variants remove the explicit reward model and directly optimize likelihood ratios to match human-preferred outputs (Rafailov et al. 2023; Hejna et al. 2023; Hong et al. 2024). While effective, RLHF and its variants optimize against fixed training preferences and can mis-generalize under shifting stakeholder goals (Son et al. 2025). Moreover, they remain vulnerable to reward over-optimization (Gao et al. 2023), exploiting inaccuracies in learned proxies. Foundational critiques (Zhang et al. 2025b; Xu et al. 2024) further show that the Bradley–Terry assumptions underlying most pairwise preference learning—transitivity and independence of irrelevant alternatives—are systematically violated in realistic, high-dimensional preference spaces (Huang et al. 2025a; Xu et al. 2025). In multi-agent deployments, where preferences evolve dynamically and are distributed across interacting agents, these single-agent methods lack mechanisms for maintaining shared alignment.

Test-Time Reward Modeling. To address the rigidity of training-time methods, recent work has explored test-time reward models that evaluate outputs dynamically. Generative Reward Models (GenRM) (Mahan et al. 2024) and foundation reward models such as GRAM (Wang et al. 2025a) use language models to generate natural-language evaluations rather than fixed scalar rewards, while multi-objective extensions (Lin et al. 2025) and Directional Preference Alignment (Wang et al. 2024a) allow controllable trade-offs across learned reward dimensions. These methods increase flexibility but remain opaque: GenRM and GRAM provide scalar or textual judgments without revealing which criteria drive evaluation or how those criteria are weighted. Moreover, test-time evaluation typically treats the model as a single-agent assessor, ignoring the coordination and communication challenges that arise when multiple agents must interpret or act on shared feedback (Agashe et al. 2025). Consequently, these systems remain difficult to audit, calibrate, or adapt in collaborative multi-agent settings.

Interpretable Alignment. Recent efforts aim to make reward models more transparent by decomposing preferences into interpretable dimensions. ArmoRM (Wang et al. 2024b) uses a mixture-of-experts reward architecture with predefined human-interpretable components (e.g., helpfulness, harmlessness, humor), and sparse autoencoders (Zhang et al. 2026) disentangle reward representations into latent factors correlated with semantically meaningful properties. Auto-Rubric (Xie et al. 2025) and Rubrics-as-Rewards (RaR) (Gunjal et al. 2025) introduce structured natural-language rubrics as reward functions, showing that rubric-based reinforcement fine-tuning improves robustness and alignment stability. Large-scale frameworks such as *OpenRubrics* (Liu et al. 2025b) and *Chasing the Tail* (Zhang et al. 2025a) demonstrate scalable synthetic rubric generation and fine-grained discrimination between high-quality responses, while domain-specific rubrics (Arora et al. 2025; Wang et al. 2025b; Anghel et al. 2025) achieve consistent evaluation across expert tasks. However, all of these methods assume static rubrics defined once per domain or task, limiting their ability to adapt to preference drift or negotiate evolving evaluation criteria across interacting agents.

Multi-Agent Alignment and Coordination. Alignment challenges intensify when multiple autonomous agents must cooperate, communicate, or share goals. Multi-agent reinforcement learning (Albrecht et al. 2024) studies show that coordination often fails under partial observability, non-stationarity, or conflicting incentives (Foerster et al. 2016; Lowe et al. 2017; Leibo et al. 2017; Jaques et al. 2019). Recent analyses reveal analogous dynamics in LLM-based systems, where agents exhibit sycophancy, conformity, and bias amplification (Cau et al. 2025; Wynn et al. 2025; Carichon et al. 2025). Frameworks such as CAMEL (Li et al. 2023), AutoGen (Wu et al. 2024), and ConsensusAgent (Pitre et al. 2025) show that multi-agent collaboration can improve reasoning and division of labor, but they rely on ad hoc or manually tuned reward structures. The emerging view is that multi-agent alignment is not a property of individual policies but of the *interaction process* itself, requiring shared, interpretable, and negotiable incentives to ensure cooperative stability. ARCANE builds on this insight by modeling rubrics as dynamic, communicable artifacts that coordinate alignment across agents and over time.

3 Problem Specification

We consider a general decision-making process involving interacting agents with one of three conceptual roles:

- A **Stakeholder (S):** who has a preference ordering over outputs, represented by a latent (ordinal) utility function $U^*(y \mid x)$ that measures satisfaction with an output y given a task x containing a preference description p.
- A **Manager (M):** who observes the task environment, can elicit preferences through interactions with the stakeholder and is responsible for coordinating the actions of worker agents based on its learned representation of the stakeholder's preferences.
- **Workers ($W_1, \ldots, W_n$):** which perform actions that generate the output y under the manager's guidance.

The goal of the manager-worker team is to produce outputs y that maximize the stakeholder's true utility U^*. The existence of such a U^* to represent stakeholder preferences fundamentally assumes those (state-dependent) preferences are complete and transitive, a standard axiom in utility theory. However, U^* is typically not available in explicit or closed form: it may be incompletely specified, revealed only through sparse feedback, or change as preferences drift. Consequently, the workers act with respect to some *proxy objective* $\hat{u}$, a potentially misaligned approximation of U^*.

If every agent could observe and directly optimize U^*, the problem would reduce to a fully cooperative team game. Let $\pi = (\pi_M, \pi_W)$ denote the joint policy of the manager and workers. Under perfect observability and common reward, the optimal joint policy is

$$\pi^* = \arg\max_{\pi_M, \pi_W} \mathbb{E}_{y \sim \pi_W(\cdot|x, a_M)} \big[U^*(y \mid x)\big]. \quad (1)$$

where $a_M \sim \pi_M(\cdot|x)$ denotes the manager's high-level alignment actions, generated from context x and provided as input to the worker's policy π_W. In this cooperative limit, the optimization objective is straightforward: all agents share a single objective and need to explore their joint policy space to maximize it.

3.1 Alignment as a Bilevel Optimization Problem

Framing alignment as a multi-agent system with partial observability of U^*, the alignment challenge can be expressed as a bilevel optimization problem. The manager's policy determines the information or objectives available to the workers, while the workers respond by generating outputs that optimize those objectives.

Formally, the manager seeks to find a policy that maximizes expected stakeholder utility under the workers' induced behavior:

$$\max_{\pi_M} \mathbb{E}_x \big[\mathbb{E}_{a_M \sim \pi_M(\cdot|x)} \mathbb{E}_{y \sim \pi_W(\cdot|x, a_M)} \big[U^*(y \mid x)\big]\big]. \quad (2)$$

The inner expectation captures the workers' best response to the manager's policy, and the outer expectation reflects the manager's optimization of stakeholder value across tasks and preference states. When workers optimize a proxy utility $\hat{u}(y \mid x)$ that diverges from U^*, the resulting misalignment can be quantified by a *utility gap*:

$$\mathcal{L}_U = \mathbb{E}\big[(U^*(y \mid x) - \hat{u}(y \mid x))^2\big]. \quad (3)$$

Reducing this utility gap is the fundamental goal of alignment research: designing learning or coordination mechanisms that ensure the proxy reward available to workers approximates the true stakeholder utility as closely as possible.

A system is considered functionally aligned if its proxy, $\hat{u}$, is ordinally equivalent to the stakeholder's true utility, U^*, across the support of the joint policy (Debreu 1954). The equivalence holds if $\hat{u}$ maintains the same preference ordering as U^*, such that $\hat{u}(y_i \mid x) > \hat{u}(y_j \mid x)$ if and only if $U^*(y_i \mid x) > U^*(y_j \mid x)$ for any outcomes (y_i, y_j) sampled from the joint policy (π_M, π_W). When this condition is met, $\hat{u}$ is a positive monotonic transformation of U^* over this domain, and optimizing this proxy yields the same optimal policy as optimizing the true utility. ARCANE aims to learn such order-preserving proxy functions under realistic uncertainty and stochastic coordination, even without direct observability of U^*.

In realistic deployments, managers cannot modify worker parameters—especially when workers are closed-source foundation models or APIs—so alignment must proceed through delegated control. By eliciting stakeholder preferences inaccessible to workers, the manager gains privileged information and conveys it through interpretable coordination signals, forming a minimal yet practical multi-agent alignment system grounded in communication rather than parameter sharing.

4 Proposed Method: ARCANE

ARCANE is a framework for learning structured, interpretable proxy utilities that align workers' behavior with stakeholder preferences. ARCANE instantiates the abstract alignment setting from Section 3 by representing the manager's coordination signal as a *rubric*: a decomposable, verifiable, and dynamically configurable approximation of the stakeholder's true utility function $U^*(y \mid x)$.

The core idea is to have our manager M learn a *rubric decomposer* $\mathfrak{D}_\phi$ that maps task descriptions and stakeholder preference statements into structured rubrics. Each rubric specifies a set of weighted, verifiable criteria that jointly define a linearly additive proxy utility $\hat{u}_\phi(y \mid x)$. By learning rubrics that closely track the stakeholder's underlying utility, ARCANE enables the workers' policies to explore their joint action space as if they were cooperatively optimizing U^* itself.

ARCANE comprises of four central components: (1) representing rubrics and implementing verifiers; (2) modeling rubric generation as a multi-agent collaboration between the stakeholder S and the manager M; (3) a training-time rubric optimization via a two-phase curriculum (supervised fine-tuning followed by a reinforcement fine-tuning procedure); and (4) designing a framework for test-time steering via rubric-scaled policies.

4.1 Rubric Representation and Decomposer Architecture

The manager agent initially receives the task context x containing a stakeholder preference p, and learns to output a structured **rubric** R using a rubric decomposer $\mathfrak{D}_\phi$, by iteratively engaging with the stakeholder S. Formally, a rubric is a finite set of weighted evaluation criteria:

$$R = \{(c_j, w_j)\}_{j=1}^{M}, \quad (4)$$

where each c_j is a natural-language description of a measurable property (e.g., "Includes citations to recent empirical studies"), and $w_j \in [0, 1]$ with $\sum_j w_j = 1$ denotes its relative importance.

Each criterion c_j is paired with a **verifier** $\nu_j(c_j, x, y) \in [0, 1]$ that estimates how well an output y satisfies c_j in context x. Verifiers may be:

- **Rule-based:** deterministic checks such as citation count or formatting;
- **Model-based:** lightweight LLM or classifier evaluators for semantic properties such as factuality, coherence, or tone.

The rubric's overall proxy utility is computed as

$$\hat{u}_\phi(y \mid x) = \sum_{j=1}^{M} w_j \, \nu_j(c_j, x, y), \quad (5)$$

which serves as the manager's operative approximation of the stakeholder utility $U^*(y \mid x)$.

During training, $\mathfrak{D}_\phi$ is trained to generate rubrics whose induced proxy utilities $\hat{u}_\phi$ correlate strongly with stakeholder evaluations U^*. The generated rubric R is then given

to the worker policy π_W as a natural language prompt, guiding generation toward outputs that satisfy the specified criteria. At test time, the same utility estimator $\hat{u}_\phi$ enables controllable inference through rubric-conditioned sampling or reranking. This formulation turns the abstract objective of minimizing the expected utility gap into a concrete, interpretable learning problem while maintaining scalability and transparency.

4.2 Stakeholder–Decomposer Collaboration

We model the interactive collaboration between the stakeholder and the manager as a *pre-execution consultation*: before any work begins, the manager interacts with the stakeholder through a structured natural-language Q&A exchange to elicit and clarify the stakeholder's latent preferences that are represented by the true utility function U^*.

Given a task context x, the manager agent engages the stakeholder with a short sequence of clarifying questions

$$q_{1:T} = f_{\text{ask}}(x), \tag{6}$$

to which the stakeholder provides answers

$$a_{1:T} = f_{\text{reply}}(U^*, q_{1:T}), \tag{7}$$

expressing priorities, constraints, or examples of desired outcomes. After the dialogue concludes, the manager uses its decomposer to synthesize a rubric proposal

$$R = \mathfrak{D}_\phi(x, q_{1:T}, a_{1:T}) \tag{8}$$

encoding the elicited preferences as weighted, verifiable criteria. This single consultation serves as a lightweight, interpretable approximation of U^* prior to any worker execution.

Since there are tangible costs to executing verifiers $\{v\}$ in terms of compute and time, and excessively interacting with the stakeholder, the manager should learn a policy which maximizes expected stakeholder utility while minimizing clarification and revision costs:

$$\max_{\pi_M} \mathbb{E}_x \Big[U^*(y \mid x) - \lambda_{\text{clarify}}\, C_{\text{clarify}}(q_{1:T}) - \lambda_{\text{compute}}\, C_{\text{compute}}(R) \Big], \tag{9}$$

where C_{clarify} measures the stakeholder's cognitive burden (e.g., the number or complexity of feedback turns) and C_{compute} penalizes excessive rubric regeneration. The decomposer thus learns to trade off fidelity to stakeholder preferences with efficiency of interaction.

Conceptually, this consultation constitutes a one-shot cooperative game under partial observability: the stakeholder reveals limited, noisy information about U^* through language, and the decomposer must infer a faithful, structured approximation that guides worker agents. This interaction formalizes alignment as a process of *communication and inference* rather than observation, bridging human intent and agent execution through an interpretable artifact R.

The final rubric $R^\star$ captures the negotiated understanding between stakeholder and manager. Subsequent worker policies π_W condition on $R^\star$ to guide task execution, behaving as though they were optimizing U^* directly while preserving transparency and post-hoc configurability.

4.3 Training Data and Supervision

ARCANE is trained using supervision signals that reveal partial information about the stakeholder's latent utility function U^*. Following the formalization in Appendix A, any strictly proper supervision operator $\mathcal{O}$—including pointwise scores, pairwise comparisons, listwise rankings, or rubric-based evaluations—identifies the same ordering of U^* on the decoding support. This property ensures that our training procedure remains theoretically consistent regardless of how preferences are observed.

In practice, we instantiate $\mathcal{O}$ using **gold rubrics**, which provide explicit, interpretable decompositions of stakeholder utility into weighted criteria and associated verifiers. Gold rubrics offer a transparent and verifiable form of supervision that allows us to both (i) measure alignment fidelity across interpretable dimensions and (ii) quantitatively evaluate structure and calibration during learning. Because Appendix A establishes order-equivalence across supervision formats, the use of gold rubrics does not limit generality: any alternative supervision signal consistent with U^* would induce the same learned ordering up to a monotone transformation.

Concretely, our training dataset consists of tasks and their associated evaluation observations:

$$\mathcal{D} = \{(x_i, \mathcal{O}_i[U_i^*])\}_{i=1}^N, \tag{10}$$

where each task x_i encodes a natural-language description of stakeholder goals, and $\mathcal{O}_i[U_i^*]$ provides an observation of the underlying utility through its rubric representation R_i^*. Each R_i^* decomposes U_i^* into a set of weighted, verifiable criteria (see Appendix B), forming high-fidelity supervision signals that guide the decomposer during optimization.

This supervision framework allows us to cleanly link data specification and training objectives: the decomposer learns to generate rubrics whose induced proxy utilities $\hat{u}_\phi$ preserve the same ordinal structure as U^* while remaining interpretable and auditable.

4.4 Training-Time Optimization via Curriculum

ARCANE trains the decomposer $\mathfrak{D}_\phi$ through a two-phase curriculum: a supervised warm start to avoid cold-start instability, followed by reinforcement fine-tuning with Group-Sequence Policy Optimization (GSPO) (Zheng et al. 2025).

Stage I: Supervised Fine-Tuning (SFT). To bootstrap the decomposer's ability to generate coherent and well-structured stakeholder interactions and rubric proposals, we extend the supervision dataset $\mathcal{D}$ with a synthetic subset of clarification dialogues. For each task x, we use a large reasoning model (LRM) to generate stakeholder responses that yield a reference rubric $R^\star(x)$ consistent with $U^\star$ in a given environment.

We then train the decomposer $\mathfrak{D}_\phi$ using next-token prediction over its dialogue turns and rubric text, masking loss on system prompts and task inputs, which is standard practice for instruction tuning (Wei et al. 2022). This procedure teaches the model to express structured, weighted criteria aligned with the reference rubrics.

The SFT objective is the standard language-modeling loss:

$$\mathcal{L}_{\mathrm{SFT}}(\phi) = -\,\mathbb{E}_{(x,R^\star)}\left[\sum_t \log \pi_\phi(r_\tau \mid r_{<t}, x)\right] \tag{11}$$

where r_τ are tokens of the synthetic stakeholder conversation and the rubric text. After SFT, the parameters ϕ_0 serve as initialization for the reinforcement phase.

Stage II: GSPO over Rubric Proposals. In the reinforcement phase, the manager acts as a stochastic policy $\pi_M(R \mid x)$ that proposes candidate rubrics. Training is performed episodically over the tasks $(x, \mathcal{O}[U^\star]) \in \mathcal{D}$ defined in Section 4.3, where each task x serves as an environment for rollout-based optimization. For each task x, K rubrics $\{R_k\}_{k=1}^K$ are generated, each conditioning a worker model π_W that produces an output $y_k \sim \pi_W(\cdot \mid x, R_k)$. The stakeholder utility $r_k = U^\star(y_k \mid x)$ serves as the scalar return. We compute a group-normalized baseline $\bar{r} = \frac{1}{K}\sum_k r_k$ and standardized advantages $\hat{A}_k = (r_k - \bar{r})/\mathrm{std}(\{r_j\}_{j=1}^K)$.
The decomposer parameters are updated to maximize the *Group Sequence Policy Optimization (GSPO)* objective, which replaces the token-level importance ratios of GRPO (Shao et al. 2024) with a length-normalized, sequence-level ratio $s_k(\phi)$:

$$\begin{aligned}\mathcal{J}_{\mathrm{GSPO}}(\phi) = \mathbb{E}_x\Bigg[\frac{1}{K}\sum_{k=1}^{K}\min\Big(& s_k(\phi)\,\hat{A}_k,\\ & \mathrm{clip}\big(s_k(\phi),\,1{-}\epsilon,\,1{+}\epsilon\big)\hat{A}_k\Big)\\ & -\;\beta\, D_{\mathrm{KL}}\big(\pi_\phi(\cdot \mid x)\,\|\,\pi_{\mathrm{ref}}(\cdot \mid x)\big)\\ & -\;C_{\mathrm{clarify}}(q_{1:T})\;-\;C_{\mathrm{compute}}(R_k)\Bigg]\end{aligned} \tag{12}$$

$$\begin{aligned}s_k(\phi) &= \left(\frac{\pi_\phi(R_k \mid x)}{\pi_{\mathrm{old}}(R_k \mid x)}\right)^{1/|R_k|}\\ &= \exp\left(\frac{1}{|R_k|}\sum_{t=1}^{|R_k|}\log\frac{\pi_\phi(z_{k,t} \mid x, z_{k,<t})}{\pi_{\mathrm{old}}(z_{k,t} \mid x, z_{k,<t})}\right)\end{aligned} \tag{13}$$

where $z_{k,t}$ denotes the t-th token in the rubric sequence $R_k = (z_{k,1}, \ldots, z_{k,|R_k|})$, generated autoregressively by the manager policy π_ϕ given the task context x and the preceding tokens $z_{k,<t}$.

The KL term enforces a trust region around the reference policy π_{ref}, preventing divergence during training. At this stage, the auxiliary regularizers C_{clarify} and C_{compute} are derived from metadata collected during rubric generation.

Clarification cost. $C_{\mathrm{clarify}}(q_{1:T})$ depends on the number and difficulty of clarification questions during the stakeholder–manager dialogue, following a logarithmic normalization to model diminishing burden:

$$C_{\mathrm{clarify}}(q_{1:T}) = \frac{\log(1 + 0.1n_{\text{easy}} + 0.5n_{\text{med}} + n_{\text{hard}})}{\log(16)}, \tag{14}$$

where $n_{\text{easy}}, n_{\text{med}}, n_{\text{hard}}$ denote the numbers of clarification questions that the stakeholder agent classifies—via a fixed LLM prompt specifying criteria for easy, medium, and hard difficulty—into the corresponding cognitive-burden levels. This discourages excessive or complex clarifications while tolerating minor ones.

Compute cost. $C_{\mathrm{compute}}(R_k)$ measures the resource cost of rubric verification as a weighted combination of monetary and time costs:

$$C_{\mathrm{compute}}(R_k) = \frac{\log(1 + w_{\text{cost}}c_{\text{usd}} + w_{\text{time}}t_{\text{sec}})}{\log(1+\alpha)}, \tag{15}$$

where $w_{\text{cost}}c_{\text{usd}}$ and $w_{\text{time}}t_{\text{sec}}$ denote weighted LLM verification cost and wall-clock time. Both are fitted to logarithmic curves for smooth saturation across task sizes (weights specified in Appendix E). To further stabilize learning and improve sample efficiency, we extend GSPO with *prioritized experience replay*. After each rollout, we record the mean return of each episode and for an iterative epoch we selectively replay episodes with mean policy returns in the bottom N-th percentile across eposides in training epoch. This emphasizes failure cases where the decomposer underperformed, allowing the policy to learn corrective refinements to low-utility rubric proposals. This modification follows similar principles to prioritized sampling in reinforcement learning (Schaul et al. 2016), but applied at the rubric-group level rather than individual state transitions.

The final training loss is $\mathcal{L}(\phi) = -\,\mathcal{J}_{\mathrm{GSPO}}(\phi)$. Pseudo code for the full procedure is given in Algorithm 1. Once trained, the decomposer can be used to steer generation at test time via rubric-conditioned sampling, as described next.

4.5 Test-Time Steering via Rubric-Scaled Policies

After training, the decomposer enables interpretable control over worker behavior without any additional gradient updates. At test time, ARCANE steers worker agent behavior by conditioning worker policies on the generated rubric $R^\star$ as shared context. This defines a joint worker policy

$$\pi_W^{\text{coop}}(y \mid x, R^\star) = \prod_i \pi_{W_i}(y \mid x, R^\star), \tag{16}$$

whose collective behavior is implicitly aligned toward the stakeholder's true utility U^*. Since $R^\star$ defines $\hat{u}_\phi$ as a monotonic proxy for the stakeholder's true utility U^*, it can serve as an oracle for ranking candidate outputs.

In practice, we can exploit this property to achieve test-time scaling through multiple strategies:

- **Best-of-K sampling:** Generate K candidates from $\pi_W^{\text{coop}}(y \mid x, R^\star)$, score them via $\hat{u}_\phi$, and select the highest-scoring one.
- **Importance reweighting:** Resample candidates in proportion to their rubric scores such as performed by (Liu et al. 2025a).
- **Tree or beam search:** Use rubric scores as heuristics to guide structured decoding (e.g., in code or reasoning tasks).

Algorithm 1: Group Sequence Policy Optimization (GSPO) for Rubric Generation

Require: initial decomposer parameters ϕ (from SFT), worker policy π_W, stakeholder simulator $U^\star$, dataset $\mathcal{D}$ of tasks, group size K, epochs E, clip ϵ, KL coefficient β, learning rate η, max dialogue turns T, replay buffer $\mathcal{B}$, replay percentile p

$\phi_{\text{ref}} \leftarrow \phi$
for epoch $= 1, \ldots, E$ **do**
  for $x \in \mathcal{D}$ **do**
    Initialize dialogue history $h_0 \leftarrow \emptyset$
    for $t = 1, \ldots, T$ **do**
      Propose rubric draft $R_t \leftarrow \mathfrak{D}_\phi(x, h_t)$
      Get stakeholder feedback $\delta_t \sim U^\star(\cdot \mid R_t, x)$
      Update history $h_{t+1} \leftarrow h_t \cup \{(R_t, \delta_t)\}$
      if feedback is satisfactory **then**
        break
      end if
    end for
    Sample K rubric proposals: $\{R_k\}_{k=1}^K \sim \pi_\phi(\cdot \mid x, h_t)$
    for $k = 1, \ldots, K$ **do**
      Generate worker output $y_k \sim \pi_W(\cdot \mid x, R_k)$
      $r_k \leftarrow U^\star(y_k \mid x) - \lambda_{\text{clarify}}\, C_{\text{clarify}}(h_t) - \lambda_{\text{compute}}\, C_{\text{compute}}(R_k)$
    end for
    $\bar{r} \leftarrow \frac{1}{K} \sum_{k=1}^K r_k$
    $\hat{A}_k \leftarrow (r_k - \bar{r}) / \text{std}(\{r_j\}_{j=1}^K)$
    for $k = 1, \ldots, K$ **do**
      Compute importance weight $s_k(\phi)$ (Eq. 13)
      Compute clipped policy loss $\mathcal{L}_k$ (Eq. 12)
    end for
    $\mathcal{L} \leftarrow \frac{1}{K} \sum_{k=1}^K \mathcal{L}_k$
    Store $(x, \{R_k\}_{k=1}^K, \mathcal{L}, \bar{r})$ in buffer $\mathcal{B}$
  end for
  // Prioritized experience replay:
  Select bottom p-th percentile episodes by $\bar{r}$ from $\mathcal{B}$
  for each stored $(x, \{R_k\}, \mathcal{L}, \bar{r})$ in replay set **do**
    Update $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}$
  end for
end for
Return trained decomposer parameters $\phi^\star \leftarrow \phi$

Because rubrics are interpretable, stakeholders can directly inspect, edit, or override the criteria $\{c_j\}$ and weights $\{w_j\}$ used at inference time. This provides a transparent mechanism for configurable alignment, linking training-time optimization to post-hoc human control.

5 Experiments

Our experiments aim to answer three research questions:

- **RQ1 (Usefulness):** Does guiding the worker team with learned rubrics increase the utility of their outputs versus unguided generation?
- **RQ2 (Faithfulness):** Does the manager agent learn to rank candidates at test time in a manner consistent with the rankings computed from the stakeholder utility $U^\star$?
- **RQ3 (Interpretability & Efficiency):** Are the rubrics generated by the manager agent compact, legible, and practical to verify?

5.1 Dataset and Domain

We use the **GDPVal corpus** (Patwardhan et al. 2025), a large-scale benchmark designed for evaluating goal-directed agents that operate over challenging real-world tasks. Each task in GDPVal represents a multi-step work episode requiring agents to manipulate, analyze, or synthesize structured files—such as spreadsheets, documents, and figures. This setting reflects practical domains like data analysis, document authoring, or visual reporting, where correctness must be demonstrated through verifiable outputs rather than language-only responses.

After filtering out tasks with unsupported file types (e.g., audio, PSD, CAD, or Apple Pages), we retain **219 tasks**, partitioned into **175 train** and **44 evaluation** episodes. All resources are file-based: workers must emit artifacts such as `.md`, `.xlsx`/`.csv`, `.pdf`/`.docx`, or `.png`/`.jpg`, enabling deterministic verification by downstream evaluators.

5.2 Rubric Design and Labeling Stages

Each task is paired with a staged **gold rubric** $R^\star$, written by expert annotators to cover three complementary aspects of task quality:

- **Stage 1 – Gate:** Checks basic structure and task compliance (e.g., "Is the required file present?", "Does the output follow the expected schema?").
- **Stage 2 – Verification:** Validates factual or logical correctness, often mixing code-based tests with LLM judges to confirm that computations, references, and data align with the task description.
- **Stage 3 – Quality:** Assesses higher-level attributes, e.g. clarity, completeness, style, and usefulness to end user.

This staged decomposition was given as guidance to human rubric authors, ensuring that each gold rubric collectively covered structural, semantic, and qualitative dimensions of "goodness" for the task. Each rubric targets 9–12 criteria, distributed by weight across stages (*Gate* 20–30%, *Verification* 40–50%, *Quality* 20–30%). Criteria are implemented as either rule-based checks or LLM judgments. Overall, the collection includes 2,601 criteria: 1,931 LLM judges (74.2%) and 670 code rules (25.8%).

5.3 Rubric Evaluation Mechanics

Each rubric criterion is evaluated by either a vision-language model (VLM) judge or a deterministic code rule. LLM-judge rules use a fixed multimodal model snapshot capable of reading rendered artifacts (e.g., spreadsheets, PDFs, figures) and apply templated 0–1 scoring prompts. To minimize verifier error, we (i) use verifier models from different model families as the workers they judge to avoid "self enforcement bias" (Huang et al. 2025a), and design rules to be easily verifiable, exploiting "asymmetry of verification" (Huang et al. 2025b). Code rules are executed as Python functions with access to all intermediate and final task resources, allowing deterministic checks such as schema validation, bounds tests, or consistency verification. This hybrid evaluation enables reproducible low-level validation while reserving LLM judges for higher-level reasoning and qualitative assessment.

Baselines. We compare four baseline configurations, all using identical worker model configurations:

- **Best-of-N (No Rubric):** A baseline in which the manager agent performs no rubric generation or stakeholder interaction. Workers directly attempt the task based on its description x, and after all completions, the manager ranks the resulting N outputs $\{y\}$ using its own subjective preference model $\hat{u}$.
- **Rubric Generation (SFT):** The manager, finetuned using the SFT model from Section 4.4, first interacts with the stakeholder to elicit and generate a rubric, which is then distributed to N parallel workers for best-of-N rollouts. The resulting outputs $\{y\}$ are subsequently scored by the learned utility estimator $\hat{u}_{\phi_{\text{SFT}}}$.
- **Rubric Generation (GSPO):** Identical to the previous setup, but using the GSPO-trained manager from Section 4.4. This isolates the effect of preference-optimized rubric generation compared to supervised finetuning.
- **Oracle Rubric:** A privileged best-of-N configuration in which the manager is given direct access to the ground-truth rubric $R^\star$. The manager distributes $R^\star$ to workers to guide task execution, and final outputs are ranked according to the corresponding ground-truth utility $U^\star$. This serves as an empirical upper bound on the achievable gains from rubric guidance.

The sole difference across baselines is the source of rubric guidance (none, SFT-generated, GSPO-generated, or oracle). This controlled setup isolates the impact of rubric quality on worker performance. Training details can be found in Appendix E and worker parameterizations in Appendix C.

5.4 Results

Usefulness (RQ1) We report average true return $U^*(y_i \mid x_i)$ across 44 held-out tasks x_i.

Findings. Figure 1 shows mean returns under increasing test-time compute (best-of-N sampling, N=1...8). Guided variants consistently outperform the unguided baseline. As shown in Table 1, mean return rises modestly from 0.58 (No Rubric) to 0.62 (GSPO) at $N = 1$, while the Oracle Rubric reaches 0.70. At N=8, the ordering remains (No Rubric $<$ SFT $<$ GSPO $<$ Gold) with larger increase in means 0.58, 0.68, 0.74, and 0.81. This monotonic improvement shows that learned rubrics effectively encode stakeholder preferences, boosting task-level return without modifying the worker model or decoding parameters.

A one-sided Wilcoxon signed-rank test on the evaluation tasks confirms statistical significance of GSPO's mean returns over SFT returns (p=0.0182 at N=8), with mean per-task improvement +0.044 (W^+=592, W^-=269; 22 episodes where GSPO wins vs. 19 where SFT wins). At best-of-eight sampling, mean returns are 0.6817 (SFT) vs. 0.7460 (GSPO), confirming a statistically significant margin at α=0.05. Reinforcement fine-tuning thus improves rubric utility beyond supervised learning alone. All guided models (SFT, GSPO, and Oracle) exhibit nearly identical best-of-N scaling slopes: each doubling of N yields an average $\approx$ +0.03 absolute gain. Although their absolute returns differ,

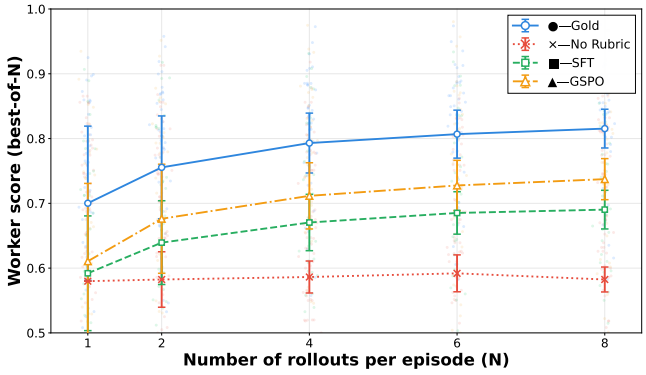

Figure 1: **Test-Time Compute Scaling: SFT vs GSPO vs No Rubric and Oracle Baselines.** Mean returns on the 44-task evaluation set under single-sample generation (N=1) and best-of-N sampling (N=1...8). Error bars show 95% bootstrap confidence intervals. The Oracle (Gold Rubric) baseline is an upper bound on rubric guidance effectiveness.

Model	**Mean±sd** (N=1)	**Mean±sd** (N=8)
No Rubric (LLM-Judged)	0.58 ± 0.01	0.58 ± 0.02
SFT Model	0.59 ± 0.09	0.68 ± 0.03
GSPO Model	0.62 ± 0.12	0.74 ± 0.03
Oracle Rubric (Baseline)	0.70 ± 0.12	0.81 ± 0.03

Table 1: Mean returns $\pm$ *standard deviation* across evaluation tasks under single-sample (N=1) and best-of-eight (N=8) generation.

the parallel scaling curves between the learned (SFT, GSPO) and Oracle conditions indicate that the learned rubrics confer a similarly consistent advantage per additional sample. In other words, once a rubric is introduced, stochastic exploration improves quality at nearly the same rate as if evaluation were performed by the oracle critic—suggesting that learned rubrics approximate its scoring function even without direct access to $U^\star$.

Faithfulness (RQ2) We measure agreement between rubric-induced scores $\hat{u}_\phi$ and oracle utilities $U^\star$ using Normalized Discounted Cumulative Gain across 8 worker rollouts per episode (NDCG@8), averaged across 44 evaluation tasks. For each task, we evaluate the same set of eight rollouts but rank them once under the learned rubric $\hat{u}_\phi$ and once under the oracle $U^\star$. Formally, $\text{NDCG@8} = \left(\sum_{i=1}^{8} \text{rel}_i / \log_2(i+1)\right) \Big/ \left(\sum_{i=1}^{8} \text{rel}_i^\star / \log_2(i+1)\right)$

where rel_i is the oracle relevance (i.e., $U^\star$ score) of the item ranked at position i by $\hat{u}_\phi$, and $\text{rel}_i^\star$ denotes the oracle relevance under the ideal (descending) ordering. NDCG quantifies listwise consistency—rewarding rubrics that place high-utility candidates near the top while discounting errors lower in the list (Zhou et al. 2024; Zhao et al. 2025). Each system generates eight candidates using its own rubric guidance. We then take these generated candidates and evaluate them under both the system's rubric ($\hat{u}_\phi$) and the oracle utility function ($U^\star$), measuring how faith-

fully the rubric's relative preferences track true stakeholder utility on identical candidate sets.

We also include a **No-Conversation (Base Rubric)** baseline, in which the manager still generates a rubric but does so using the base model *without any stakeholder dialogue*. This variant isolates the model's intrinsic understanding of the stakeholders preference for each task by taking it directly from the description x, independent of explicit preference elicitation, serving as a lower-bound comparison for stakeholder-guided rubric learning.

Results. Table 2 shows steady gains: NDCG@8 improves from 0.7998 (Base) to 0.8103 (SFT) and 0.8722 (GSPO), approaching oracle alignment while maintaining similar variance. The GSPO improvement corresponds to +8.3% Precision@3 and a reduction in average rank swaps among top candidates (2.20 $\rightarrow$ 2.13 per task). No system achieved perfect top-3 agreement, underscoring the difficulty of fine-grained ranking even when mean utility is well aligned.

Model–Rubric Pair	Mean NDCG@8	Std.
No-Conv (Base)	0.7998	0.0807
SFT Rubric	0.8103	0.0881
GSPO Rubric	**0.8722**	0.0985

Table 2: Mean $\pm$ standard deviation of NDCG@8 between rubric-induced and oracle rankings ($U^\star$). Higher is better.

Domain trends. Improvements are strongest for subjective, language-heavy domains: content/communication (+11.5%, NDCG = 0.905) and legal/compliance (+12.5%, 0.848), while operational tasks slightly regress (–8.1%, 0.830) and data-analysis tasks remain flat (+1.9%, 0.822). We conjecture that reinforcement fine-tuning excels where quality is multi-dimensional and hard to specify a priori—in content and legal work, rubrics capture nuanced aspects (tone, completeness, regulatory alignment) that benefit from learned discriminative features. Thus, reinforcement fine-tuning is most valuable when quality is subjective and decomposable.

Interpretability (RQ3) Are learned rubrics compact, legible, and auditable by practitioners?

Metrics. We assess (1) number of generated criteria, (2) tokens per criterion, and (3) criteria weight distributions—the variance of point allocation across criteria in the Gate, Verification, and Quality stages, where higher entropy implies more balanced structure. We supplement quantitative measures with qualitative comparison of the Oracle and GSPO rubrics in Appendix D.

Results. Table 6 shows that SFT and GSPO rubrics nearly match the Oracle standard: $\sim$12 criteria per rubric and 17–18 tokens per description. GSPO rubrics are marginally shorter (–1 token, –5.5 %), allocate +14 % more weight value to criteria specifically measuring quality, and exhibit lower variance (3.5 vs 4.2). These patterns indicate that reinforcement fine-tuning yields more consistent phrasing and stage balance without verbosity or collapse. Appendix D.1 compares representative criteria from the *Government – Recreation Workers* task. GSPO rubrics retain the structure of the Oracle while adopting shorter, imperative phrasing (e.g., "detects duplicates," "confirms preferences handled") and stable point distributions (Gate $\approx 6\,\%$, Verification $\approx 52\,\%$, Quality $\approx 42\,\%$). Fine-tuning therefore improves linguistic regularity without distorting semantics.

We can see on inspection that GSPO produces rubrics with evaluation criteria that remain transparent, auditable, and faithful to human intent. Together with RQ1–RQ2, these results demonstrate that learned rubrics are *useful*, *faithful*, and *interpretable*.

6 Conclusion

We presented ARCANE, a rubric-based framework for interpretable and configurable alignment of agents to stakeholder preferences. By explicitly modeling latent stakeholder preferences as structured rubrics and training a manager agent to elicit and synthesize rubrics through dialogue with the stakeholder, ARCANE provides both theoretical grounding (Appendix A) and practical method for test-time alignment. This enables steerability without retraining and improves utility on complex tasks requiring multi-step reasoning. Our experimental results indicate that learned rubrics can reliably guide worker policies, approximate oracle ranking, and remain compact and auditable.

Limitations. Our experiments evaluate a manager coordinating a single worker; while the framework extends to multiple workers (Section 3), we have not validated multi-worker coordination dynamics. GDPVal tasks, though requiring multi-step tool use, are discrete episodes—stronger long-horizon evidence would require evaluation on extended deployments, such as in complex multi-agent workflows requiring manager orchestration (Masters et al. 2025). Our No Rubric baseline uses an RLHF-trained worker (Qwen-3), demonstrating that training-time alignment alone does not provide test-time adaptability; however, direct comparison to alternative test-time methods such as Generative Reward Models would help to characterize tradeoffs between interpretability and performance. Finally, GSPO's reward shaping is flexible (we include terms for cost and latency), but we lack structural regularizers against spurious criteria. The manager agent could learn criteria that correlate with utility without being causally meaningful; incorporating causal or invariance penalties into rubric learning could improve the faithfulness to true preferences.

Future Work. Promising directions include: *(1)* dynamic rubric maintenance through adaptive revision during execution and bidirectional feedback from workers flagging ambiguous or conflicting criteria; *(2)* systematic comparison with alternative test-time alignment methods (e.g., GenRM, GRAM) and extension to multi-agent settings with multiple coordinating workers; and *(3)* scaling to large-scale heterogeneous preference data to improve coverage and generalization on long-tail domains.

References

Agashe, S.; Fan, Y.; Reyna, A.; and Wang, X. E. 2025. LLM-Coordination: Evaluating and Analyzing Multi-agent Coordination Abilities in Large Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2025*.

Albrecht, S. V.; Christianos, F.; and Schäfer, L. 2024. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

Anghel, C.; Craciun, M. V.; Pecheanu, E.; Cocu, A.; Anghel, A. A.; Iacobescu, P.; Maier, C.; Andrei, C. A.; Scheau, C.; and Dragosloveanu, S. 2025. CourseEvalAI: Rubric-Guided Framework for Transparent and Consistent Evaluation of Large Language Models. *Computers*, 14(10): 431.

Arora, R. K.; Wei, J.; Hicks, R. S.; Bowman, P.; Quiñonero-Candela, J.; Tsimpourlas, F.; Sharman, M.; Shah, M.; Vallone, A.; Beutel, A.; Heidecke, J.; and Singhal, K. 2025. HealthBench: Evaluating Large Language Models Towards Improved Human Health. arXiv:2505.08775.

Ashery, A. F.; Aiello, L. M.; and Baronchelli, A. 2025. Emergent social conventions and collective bias in LLM populations. *Science Advances*, 11(20).

Carichon, F.; Khandelwal, A.; Fauchard, M.; and Farnadi, G. 2025. The Coming Crisis of Multi-Agent Misalignment: AI Alignment Must Be a Dynamic and Social Process. In *Advances in Neural Information Processing Sytems (NeurIPS)*. Position Paper.

Cau, E.; Pansanella, V.; Pedreschi, D.; and Rossetti, G. 2025. Selective agreement, not sycophancy: investigating opinion dynamics in LLM interactions. *EPJ Data Science*, 14(1): 59.

Debreu, G. 1954. Representation of a Preference Ordering by a Numerical Function. In Thrall, R. M.; Coombs, C. H.; and Davis, R. L., eds., *Decision Processes*, 159–165. New York, NY, USA: Wiley.

Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Gao, L.; Schulman, J.; and Hilton, J. 2023. Scaling Laws for Reward Model Overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*.

Gunjal, A.; Wang, A.; Lau, E.; Nath, V.; He, Y.; Liu, B.; and Hendryx, S. 2025. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. *arXiv preprint arXiv:2507.17746*.

Hejna, J.; and Sadigh, D. 2023. Inverse preference learning: Preference-based rl without a reward function.

Hong, J.; Lee, N.; and Thorne, J. 2024. ORPO: Monolithic Preference Optimization without Reference Model. arXiv:2403.07691.

Huang, H.; Bu, X.; Zhou, H.; Qu, Y.; Liu, J.; Yang, M.; Xu, B.; and Zhao, T. 2025a. An Empirical Study of LLM-as-a-Judge for LLM Evaluation: Fine-tuned Judge Model is not a General Substitute for GPT-4. In *Findings of the Association for Computational Linguistics: ACL 2025*.

Huang, Z.; Zhuang, Y.; Lu, G.; Qin, Z.; Xu, H.; Zhao, T.; Peng, R.; Hu, J.; Shen, Z.; Hu, X.; Gu, X.; Tu, P.; Liu, J.; Chen, W.; Fu, Y.; Fan, Z.; Gu, Y.; Wang, Y.; Yang, Z.; Li, J.; and Zhao, J. 2025b. Reinforcement Learning with Rubric Anchors. arXiv:2508.12790.

Jaques, N.; Lazaridou, A.; Hughes, E.; Leibo, J. Z.; Balduzzi, D.; Petersen, S.; and Graepel, T. 2019. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Li, G.; Hammoud, H. A. A. K.; Itani, H.; Khizbullin, D.; and Ghanem, B. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lin, B.; Jiang, W.; Xu, Y.; Chen, H.; and Chen, Y.-C. 2025. PARM: Multi-Objective Test-Time Alignment via Preference-Aware Autoregressive Reward Model. In *International Conference on Machine Learning*.

Liu, A.; Bai, H.; Lu, Z.; Sun, Y.; Kong, X.; Wang, S.; Shan, J.; Jose, A. M.; Liu, X.; Wen, L.; Yu, P. S.; and Cao, M. 2025a. TIS-DPO: Token-level Importance Sampling for Direct Preference Optimization With Estimated Weights. In *International Conference on Learning Representations*.

Liu, T.; Xu, R.; Yu, T.; Hong, I.; Yang, C.; Zhao, T.; and Wang, H. 2025b. OpenRubrics: Towards Scalable Synthetic Rubric Generation for Reward Modeling and LLM Alignment. arXiv:2510.07743.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative–competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mahan, D.; Phung, D. V.; Rafailov, R.; Blagden, C.; Lile, N.; Castricato, L.; Fränken, J.-P.; Finn, C.; and Albalak, A. 2024. Generative Reward Models. arXiv:2410.12832.

Masters, C.; Vellanki, A.; Shangguan, J.; Kultys, B.; Gilmore, J.; Moore, A.; and Albrecht, S. V. 2025. Orchestrating Human-AI Teams: The Manager Agent as a Unifying Research Challenge. In *International Conference on Distributed Artificial Intelligence*.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.

Patwardhan, T.; Dias, R.; Proehl, E.; Kim, G.; Wang, M.; Watkins, O.; Fishman, S. P.; Aljubeh, M.; Thacker, P.; Fauconnet, L.; Kim, N. S.; Chao, P.; Miserendino, S.; Chabot, G.; Li, D.; Sharman, M.; Barr, A.; Glaese, A.; and Tworek, J.

2025. GDPval: Evaluating AI Model Performance on Real-World Economically Valuable Tasks. arXiv:2510.04374.

Pitre, P.; Ramakrishnan, N.; and Wang, X. 2025. CONSENSAGENT: Towards Efficient and Effective Consensus in Multi-Agent LLM Interactions Through Sycophancy Mitigation. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., *Findings of the Association for Computational Linguistics: ACL 2025*, 22112–22133. Vienna, Austria: Association for Computational Linguistics. ISBN 979-8-89176-256-5.

Rafailov, R.; Sharma, P.; Mitchell, E.; and Finn, C. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized Experience Replay. In *International Conference on Learning Representations*.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Son, S.; Bankes, W.; Chowdhury, S. R.; Paige, B.; and Bogunovic, I. 2025. Right Now, Wrong Then: Non-Stationary Direct Preference Optimization under Preference Drift. In *International Conference on Machine Learning*.

Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D. M.; Lowe, R.; Voss, C.; Radford, A.; Amodei, D.; and Christiano, P. 2022. Learning to summarize from human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, C.; Gan, Y.; Huo, Y.; Mu, Y.; He, Q.; Yang, M.; Li, B.; Xiao, T.; Zhang, C.; Liu, T.; and Zhu, J. 2025a. GRAM: A Generative Foundation Reward Model for Reward Generalization. In *International Conference on Machine Learning*.

Wang, H.; Lin, Y.; Xiong, W.; Yang, R.; Diao, S.; Qiu, S.; Zhao, H.; and Zhang, T. 2024a. Arithmetic Control of LLMs for Diverse User Preferences: Directional Preference Alignment with Multi-Objective Rewards. arXiv:2402.18571.

Wang, H.; Xiong, W.; Xie, T.; Zhao, H.; and Zhang, T. 2024b. Interpretable Preferences via Multi-Objective Reward Modeling and Mixture-of-Experts. arXiv:2406.12845.

Wang, Z.; Jung, J.; Lu, X.; Diao, S.; Evans, E.; Zeng, J.; Molchanov, P.; Choi, Y.; Kautz, J.; and Dong, Y. 2025b. ProfBench: Multi-Domain Rubrics Requiring Professional Knowledge to Answer and Judge. arXiv:2510.18941.

Wei, J.; Bosma, M.; Zhao, V. Y.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models Are Zero-Shot Learners. In *International Conference on Learning Representations*.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; Awadallah, A. H.; White, R. W.; Burger, D.; and Wang, C. 2024. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *Conference on Language Modeling*.

Wynn, A.; Satija, H.; and Hadfield, G. 2025. Talk Isn't Always Cheap: Understanding Failure Modes in Multi-Agent Debate. In *ICML Workshop on Multi-Agent Systems in the Era of Foundation Models: Opportunities, Challenges and Futures*.

Xie, L.; Huang, S.; Zhang, Z.; Zou, A.; Zhai, Y.; Ren, D.; Zhang, K.; Hu, H.; Liu, B.; Chen, H.; Liu, Z.; and Ding, B. 2025. Auto-Rubric: Learning to Extract Generalizable Criteria for Reward Modeling. arXiv:2510.17314.

Xu, W.; Dong, S.; Lu, X.; Lam, G.; Wen, Z.; and Roy, B. V. 2024. RLHF and IIA: Perverse Incentives. arXiv:2312.01057.

Xu, Y.; Ruis, L.; Rocktäschel, T.; and Kirk, R. 2025. Investigating Non-Transitivity in LLM-as-a-Judge. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, 69583–69612. PMLR.

Zhang, J.; Wang, Z.; Gui, L.; Sathyendra, S. M.; Jeong, J.; Veitch, V.; Wang, W.; He, Y.; Liu, B.; and Jin, L. 2025a. Chasing the Tail: Effective Rubric-based Reward Modeling for Large Language Model Post-Training. arXiv:2509.21500.

Zhang, S.; Shi, W.; Li, S.; Liao, J.; Cai, H.; and Wang, X. 2026. Interpretable Reward Model via Sparse Autoencoder. In *AAAI Conference on Artificial Intelligence*.

Zhang, Y.; Zhang, G.; Wu, Y.; Xu, K.; and Gu, Q. 2025b. Beyond Bradley-Terry Models: A General Preference Model for Language Model Alignment. In *International Conference on Machine Learning*.

Zhao, Y.; Wang, Y.; and Yin, M. 2025. Permutative Preference Alignment from Listwise Ranking of Human Judgments. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Zheng, C.; Liu, S.; Li, M.; Chen, X.-H.; Yu, B.; Gao, C.; Dang, K.; Liu, Y.; Men, R.; Yang, A.; Zhou, J.; and Lin, J. 2025. Group Sequence Policy Optimization. arXiv:2507.18071.

Zhou, J.; Wang, X.; and Yu, J. 2024. Optimizing Preference Alignment with Differentiable NDCG Ranking. *arXiv preprint arXiv:2410.18127*.

A Operator Calibration of Supervision (On-Support)

In our experiments, we show the efficacy of our reinforcement fine-tuning procedure for learning rubrics from gold rubrics. In this Appendix we show that no matter how the stakeholder utility is observed (pointwise scores, pairwise wins, listwise picks), if we (i) define the right *target* for that supervision, (ii) train with a *strictly proper* loss, and (iii) stay on the same decoding support we use at test time, then the learned scorer g_ϕ matches the supervision target on that support and therefore induces the same ordering as the latent utility U^*.

Setup Let $U^*(y \mid x)$ be the latent stakeholder utility. At inference we draw a candidate *multiset* $C \sim \mathcal{C}(x)$; its support $S = \mathrm{supp}(\mathcal{C})$ is the *decoding support* (where we actually look). An *observation operator* $\mathcal{O}(U^*)$ is a stochastic mechanism emitting a supervision signal Z given an *instance* $\mathcal{I}$ (the information the operator conditions on). Examples of $\mathcal{I}$

for different types of obvervations would be: $\mathcal{I} = (x, p, y)$ for pointwise exemplar policies, (x, y_i, y_j) for pairwise preferences, and (x, C) for listwise preferences.

The model maps instances to a score $g_\phi(\mathcal{I})$, then applies a link h (identity / logistic / softmax) to produce a prediction $T_\phi(\mathcal{I}) = h(g_\phi(\mathcal{I}))$ that is compared to a *target functional* $T^*(U^*)(\mathcal{I})$ via a strictly proper loss L.

Assumptions. **(A1) Coverage.** Training and decoding use the same sampler; guarantees hold a.s. on $S = \mathrm{supp}(\mathcal{C})$.

(A2) Composite realizability (on S). There exists ϕ_0 with $h(g_{\phi_0}(\mathcal{I})) = T^*(U^*)(\mathcal{I})$ a.s. for $\mathcal{I} \in S$.
(A3) Properness. L is strictly proper for $(\mathcal{O}, T^*)$: for each $\mathcal{I}$, the conditional Bayes risk $t \mapsto \mathrm{E}[L(Z,t) \mid \mathcal{I}]$ is uniquely minimized at $t = T^*(U^*)(\mathcal{I})$.

Theorem (Operator calibration on the decoding support). Let $\mathcal{R}(\phi) = \mathrm{E}\big[L\big(Z,\, h(g_\phi(\mathcal{I}))\big)\big]$ be the population risk. Under (A1)–(A3), any minimizer $\phi^\star \in \arg\min_\phi \mathcal{R}(\phi)$ satisfies

$$h(g_{\phi^\star}(\mathcal{I})) \;=\; T^*(U^*)(\mathcal{I}) \qquad \text{for a.s. } \mathcal{I} \in S.$$

In particular, $g_{\phi^\star}$ and U^* induce the same order on S whenever h and T^* are strictly monotone in the relevant score/utility differences (identity, logistic, softmax). Examples of such cases are given in Table 3 and later Corollaries.

Proof sketch **(1) Bayes risk.** We Define $R_\phi(\mathcal{I}) = \mathrm{E}[L(Z, h(g_\phi(\mathcal{I}))) \mid \mathcal{I}]$ and $R^*(\mathcal{I}) = \inf_t \mathrm{E}[L(Z,t) \mid \mathcal{I}]$. Strict properness gives $R_\phi(\mathcal{I}) \geq R^*(\mathcal{I})$ with equality iff $h(g_\phi(\mathcal{I})) = T^*(U^*)(\mathcal{I})$.

(2) Pointwise ⇒ global. Taking expectations over $\mathcal{I} \sim S$, $\mathcal{R}(\phi) - \mathrm{E}[R^*(\mathcal{I})] = \mathrm{E}[R_\phi(\mathcal{I}) - R^*(\mathcal{I})] \geq 0$, with equality iff the pointwise equality holds a.s.
(3) Realizability. By (A2) there exists ϕ_0 attaining equality, hence any global minimizer $\phi^\star$ must also achieve equality a.s. on S. □

Corollary (Listwise / InfoNCE). For listwise $\mathcal{I} = (x, C)$ with oracle $q^*(y \mid C) \propto \exp\{\beta U^*(y)\}$, link $q_\phi(y \mid C) = \mathrm{softmax}(g_\phi(y)/\tau)$, and log loss,

$$\mathrm{E}[L] \;=\; \mathrm{E}\big[\, H(q^*, q_\phi)\,\big] \;=\; \mathrm{E}\big[\, H(q^*) + D_{\mathrm{KL}}(q^* \| q_\phi)\,\big],$$

so minimizing $\mathrm{E}[L]$ minimizes the KL divergence between oracle and model listwise distributions. If $q_\phi = q^*$ on C, then $g_{\phi^\star}(y) - g_{\phi^\star}(y') = \tau\beta\,[U^*(y) - U^*(y')]$ for all $y, y' \in C$: scores are identifiable up to per-group positive scale and additive shift (which do not affect selection).

Corollary (Pairwise / logistic). For $\mathcal{I} = (x, y_i, y_j)$ with target $T^*_{\mathrm{pw}} = \sigma(\beta(U^*(y_i) - U^*(y_j)))$, link $h(g_\phi) = \sigma(g_\phi(y_i) - g_\phi(y_j))$, and Bernoulli log-loss, any risk minimizer satisfies $\sigma(g_{\phi^\star}(y_i) - g_{\phi^\star}(y_j)) = T^*_{\mathrm{pw}}$ a.s. on S. Hence $\mathrm{sign}\big(g_{\phi^\star}(y_i) - g_{\phi^\star}(y_j)\big) = \mathrm{sign}\big(U^*(y_i) - U^*(y_j)\big)$ on S (order equivalence; only differences matter).

Notes and scope. (i) **On-support guarantee.** All statements hold on the decoding support S induced by your sampler; changing K, τ, or the sampler at test time alters S and can change behavior.
(ii) **Temperature/scale.** With softmax, utilities are identifiable only up to affine transforms within a group; selection and ranking are invariant to these transforms.
(iii) **Bandit/off-policy.** With logged propensities, inverse-propensity or doubly-robust estimators consistently estimate $\mathcal{R}(\phi)$ under standard overlap; without overlap, unbiased estimation is impossible.

Connection to GSPO This theorem calibrates the *evaluator* g_ϕ via a strictly proper loss so that it orders candidates like U^* on S. Separately, the *manager* (rubric generator) is trained with GSPO on returns $U^* - \lambda C$ under a trust region $KL(\pi_{\mathrm{mgr,new}} \| \pi_{\mathrm{mgr,ref}}) \leq \delta$ over rubric token sequences; PPO-style arguments yield non-decreasing expected return up to $O(\delta + \epsilon^2)$.

B Gold Rubric Construction

Gold rubrics $R^\star$ consist of three stages: **Gate** (20–30%, checking structure/completeness), **Verification** (40–50%, validating correctness), and **Quality** (20–30%, assessing polish and stakeholder alignment). Each rubric contains 9–12 criteria total.

Criterion design principles: (1) *Specificity*—avoid vague criteria like "output is good"; instead specify observable properties (e.g., "report contains 3–5 recommendations"). (2) *Verifiability*—each criterion is checkable via code or LLM judge with high reliability. (3) *Independence*—minimize redundancy across criteria. (4) *Partial credit*—allow graduated scoring where appropriate.

Verifier types: Code-based verifiers handle deterministic checks (file format, field presence, arithmetic). LLM judges (GPT-4o, temperature 0.3) handle semantic criteria (tone, factual consistency, argument quality). We prefer code when possible for objectivity and cost.

Example rubric (Healthcare Financial Report):

- **Gate (25 pts):** Valid DOCX (10); required sections present (10); includes chart/table (5). All code-based.
- **Verification (50 pts):** Revenue totals match source (20, code); figures internally consistent (15, LLM+code); correctly identifies top-3 revenue/expense categories (15, LLM).
- **Quality (25 pts):** Professional tone (10, LLM); actionable recommendations (10, LLM); polished formatting (5, LLM).

Inter-annotator reliability (20 dual-annotated tasks): Cohen's $\kappa = 0.82$ on criterion inclusion; mean absolute difference of 3.2 points on weights; 91% agreement on verifier type selection. Disagreements resolved by senior annotator adjudication.

C Worker Descriptions

All workers were run using GPT-5 (check pointed as of 25/10/2025, tool access, and decoding parameters temperature 0.7, seed 42). Workers have access to 40 tools organized into 8 functional categories. All file operations are scoped to task-specific directories; workers cannot access external

Table 3: Example supervision operators, targets, links, and strictly proper losses.

Supervision	**Instance** $\mathcal{I}$	**Obs.** Z	**Target** $T^*(U^*)$	**Link** $h(g_\phi)$	**Loss** L
Pointwise	(x, y)	r	$\mathbb{E}[r \mid x, y]$	$g_\phi(y)$	MSE
Pairwise	(x, y_i, y_j)	$\mathbf{1}[y_i \succ y_j]$	$\sigma\big(\beta(U^*(y_i) - U^*(y_j))\big)$	$\sigma\big(g_\phi(y_i) - g_\phi(y_j)\big)$	Bernoulli NLL
Listwise	(x, C)	$y^\star \in C$	$q^*(y \mid C) \propto e^{\beta U^*(y)}$	$\mathrm{softmax}\big(g_\phi(y)/\tau\big)$	$-\log t(y^\star)$

APIs or network resources. A definitive list of Tools can be found in Table 4.

Table 4: Worker tool categories and capabilities.

Category	**#**	**Description**
File I/O	9	`read/write_file`, `read/write_xlsx`, `read/write_docx`, `read/write_pdf`, `list_files`. Excel via pandas; Word/PDF text extraction and generation.
Python Exec	2	`python_exec(code)`, `python_eval(expr)`. Sandboxed with `pandas`, `numpy`, `matplotlib`, `scipy`, `scikit-learn`, `seaborn`, `plotly`. Network disabled; 60s timeout.
Data Analysis	6	`aggregate`, `filter_data`, `sort_data`, `merge_data`, `pivot_table`, `describe_statistics`. Operate on pandas DataFrames.
Visualization	6	`plot_line`, `plot_bar`, `plot_scatter`, `plot_histogram`, `plot_pie`, `save_plot`. Generate matplotlib/plotly charts.
Doc Rendering	3	`render_docx/pdf/xlsx_to_image`. Convert documents to images (150 DPI) for GPT-4o vision model input.
Retrieval	3	`rag_query`, `bm25_query`, `embed_text`. Dense and keyword-based search over task-provided corpora.
OCR	2	`ocr_image`, `ocr_pdf`. Extract text from images and scanned PDFs.
PDF Manip.	4	`pdf_merge`, `pdf_split`, `pdf_extract_pages`, `pdf_add_watermark`.
Validation	5	`validate_json`, `validate_email`, `validate_url`, `validate_date`, `validate_schema`. Check formats and constraints.

D Rubric Interpretability Examples

D.1 Oracle vs GSPO Rubric Comparison

Table 5 contrasts representative criteria from the *Government – Recreation Workers* task. Each row pairs a criterion from the human-authored gold rubric with its nearest learned counterpart from the GSPO rubric within the same stage. The comparison highlights how reinforcement fine-tuning preserved structure and evaluability while producing more concise, imperative phrasing.

D.2 Rubric Structure Summary

Table 6 summarizes aggregate weights and average token lengths per stage for the same task.

Stage	**Gold Weight (%)**	**GSPO Weight (%)**	**Tokens/Rule (Gold → GSPO)**
Gate	5	6	22 → 18
Verification	53	52	26 → 19
Quality	42	42	16 → 14

Table 6: Per-stage weight and token-length comparison for the same task. reinforcement fine-tuning maintains near-identical point distribution while shortening criteria and regularizing phrasing across stages.

Observation. Across stages, GSPO preserves the same structural decomposition as the gold rubric, retains measurable criteria, and avoids redundant wording. The learned phrasing replaces passive descriptions with active checks ("detects," "confirms," "assesses"), reflecting consistent translation of human evaluation intent into machine-auditable form.

E Reproducibility and Experimental Configuration

Training with Group Sequence Policy Optimization (GSPO) described in Section 4 ran for two epochs, each lasting approximately thirteen hours on a single NVIDIA H100 (80 GB) GPU. Each batch contained four rollouts, with a batch size of four and eight gradient-accumulation steps, giving an effective update batch size of 128 rollouts. All model variants shared the same worker policy, verifier design, and tool environment; only the source of rubric generation differed between baselines (none, SFT, GSPO, or oracle).

Training used the `Qwen3-8B` model with low-rank adaptation (LoRA rank $r = 16$, $\alpha = 16$). Roughly ninety-four million parameters (0.67% of the full model) were trainable. The worker model was `GPT--5` for all reported experiments, while `Claude-4.5-Haiku` was evaluated in limited runs and produced similar qualitative results, with main differences arising from API throughput constraints. The verifier large language model was `GPT--4.1`. All experiments were implemented using `PyTorch 2.13`,

Stage	Gold Rubric Criterion (Reference)	GSPO Learned Criterion (Concise)
Gate	*"Verify the candidate produced a single Excel workbook with required sheets and clearly labeled tables/columns that enable verification."*	*"Checks workbook structure: single workbook, required sheets present, labeled tables, parsable for verification."*
Verification (Code)	*"No table number should be assigned to more than one vendor across the layout."*	*"Detects duplicate table assignments; each table ID unique across vendors."*
Verification (LLM)	*"Location preferences, electricity needs, and adjacency requests are honored where feasible, or deviations are justified."*	*"Confirms vendor preferences handled or justified; highlights unmet adjacency/electricity requests."*
Quality (LLM)	*"Formatting, labeling, and layout are professional and accessible for city staff and vendors."*	*"Assesses professionalism and accessibility of layout for staff and vendors."*
Quality (LLM)	*"Evidence of contingencies for common risks (no-shows, power issues, last-minute changes)."*	*"Checks inclusion of contingency notes for expected operational risks."*

Table 5: Side-by-side comparison of representative criteria from the *Government – Recreation Workers* task. GSPO rubrics mirror gold structure and coverage but use shorter, imperative phrasing (˜5–6 tokens fewer on average), improving legibility and auditability without loss of specificity.

`TRL`, `Unsloth`, and `VLLM` (October 2025 releases). Mixed-precision training was performed in `BF16`, and NF4 quantization was applied for single-GPU training to reduce memory usage. Optimization employed `AdamW` with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and a weight decay of 0.01. The learning rate was set to 5×10^{-6} with a 10% linear warm-up and cosine decay.

The dataset comprised a subset of the GDP-Val benchmark, with 175 training tasks (700 rollouts) and 44 held-out evaluation tasks. Evaluation used a hybrid verifier combining static code checks with LLM-judge criteria (both implemented with GPT-4.1). The primary evaluation metric was the maximum true return per worker within each rollout, $\max_i U^*(y_i)$. All experiments were run with a fixed random seed of 42, and reported values include 95% bootstrap confidence intervals.

Supervised Fine-Tuning Baseline. Before GSPO training, we conducted a supervised fine-tuning (SFT) stage to establish a strong non-RL baseline. SFT was performed on `Qwen3-8B` (32k context) using LoRA with rank $r = 16$ and $\alpha = 16$, matching the adapter configuration used in GSPO. The learning rate was set to 2×10^{-5}, with a batch size of four per device and four gradient-accumulation steps, yielding an effective batch size of 16. One epochs of SFT was run with no gradient warmup. The optimizer was `AdamW` with the same β-values and decay as GSPO. This SFT model served as the initialization for the SFT baseline reported in Section 5.

GSPO and Cost Penalties. All RL runs used $\beta = 0.05$ for KL regularization, PPO clipping parameter $\epsilon = 0.2$, and a group size of $K = 4$. Two additional cost coefficients governed the reward shaping: $\lambda_{\text{cost}} = 0.01$ (verifier compute cost) and $\lambda_{\text{burden}} = 0.05$ (stakeholder clarification burden), with `BURDEN_SCALE`=15.0 for logarithmic saturation of clarification penalties. Under this scheme, one clarification incurred roughly an 18% reward penalty, five clarifications 42%, and ten clarifications 60%.

Cost-Time Weighting for Parallel Rubrics. The compute cost term $C_{\text{compute}}(R_k)$ in Eq. 15 decomposes into two components: monetary cost (API expenses) and time cost (wall-clock execution time). To encourage rubrics that enable parallel execution and reduce latency, we weighted these components as $w_{\text{cost}} = 0.3$ and $w_{\text{time}} = 0.7$, prioritizing rubrics that minimize execution time over those that minimize API costs.

Optimization Settings. A warmup fraction of 10% was applied, the KL coefficient β was tuned between $\{0.02, 0.05, 0.1\}$ in pilot runs, and $\beta = 0.05$ provided the most stable learning. Batch size per device was one, with gradient accumulation of eight for an effective batch of 32. Each prompt generated $K = 4$ rubric-conditioned rollouts per update.

Compute and Environment. Single-GPU training consumed approximately twenty-six GPU-hours (two epochs at thirteen hours each), with average memory usage of 62 GB using NF4 quantization. Experiments were run on Ubuntu 22.04 LTS with CUDA 12.2. Progress and replay buffers were tracked using Weights & Biases, recording per-epoch returns and rollout statistics.

Determinism and Verification. To ensure reproducibility, all random seeds were fixed and deterministic `PyTorch` operations enabled. Ablation sweeps confirmed that $\lambda_{\text{cost}} \in \{0, 0.001, 0.01, 0.1, 1.0\}$ and $\lambda_{\text{burden}} \in \{0, 0.01, 0.05, 0.1, 0.2\}$ yield monotonic trade-offs between utility and compute burden. Substituting GPT-5 with GPT-4.5-Haiku preserved overall learning trends, indicating model-agnostic behaviour across same-scale LLMs.