VERDICT: VERIFIABLE DECENTRALISED INFERENCE-ONLY COLLECTIVE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

As AI systems increasingly serve users with differing interests, training shared models requires both efficient participation and principled preference aggregation. We introduce VerDICT, a decentralised training approach that allows participants to contribute computation while expressing their preferences over model outputs. VerDICT combines EGGROLL with a scalar pairwise preference objective targeting maximal lotteries, which accommodate cyclic preferences and select the Condorcet winner when one exists. Training operates entirely through inference, requiring no backward passes or separate reference model, and shares baseline responses across workers to reduce computational costs. Each worker sends only its votes, and every update is a deterministic function of the shared seed and the logged votes, so it can be recomputed and checked. We assume workers report their preferences honestly. On controlled preference games VerDICT approaches the maximal lottery where DPO under the same optimiser does not, and on summarisation and two human preference datasets it improves judged quality over the initial policy at lower per-step cost than EGGROLL-DPO.

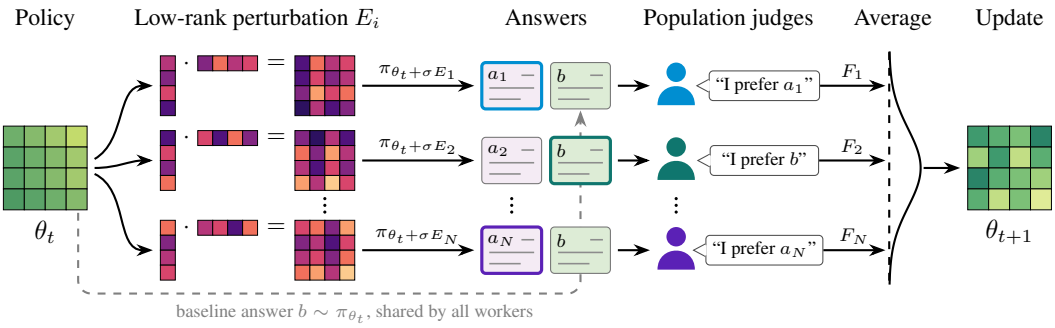

Figure 1: Schematic of VerDICT with N population members. Each worker submits its own preferences in exchange for contributing compute to training.

1 INTRODUCTION

As artificial intelligence models become more capable and more agentic, their decisions increasingly affect the interests of individual users and institutions. Yet control over their training remains concentrated within a small number of organisations. A central emerging challenge is therefore to develop decentralised learning systems through which independent participants can jointly train models that represent their interests. As these systems undertake increasingly consequential tasks on behalf of multiple users, resolving disagreements about their behaviour will become a fundamental requirement of training. This is precisely the challenge that pluralistic alignment aims to address (Sorensen et al., 2024).

Achieving this requires more than distributing computation. Participants must have a reason to contribute resources, and the training procedure must specify how their contributions influence the resulting model. A natural arrangement is to allow participants to contribute compute in exchange for having their preferences represented in training.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

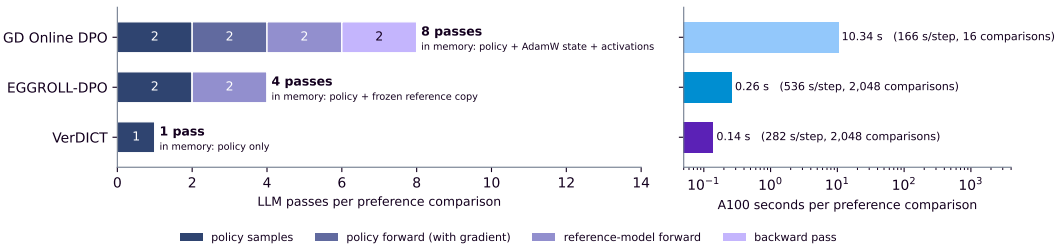

Figure 2: LLM passes per preference comparison. GD Online DPO/IPO: 8. EGGROLL-DPO: 4. VerDICT: ≈ 1, with no backward pass. Each method also makes one judge call. Right: measured A100 seconds per comparison (Qwen3-8B, CoVal, earlier $\sigma = 10^{-3}$ runs); GD uses $128\times$ fewer comparisons per step.

These considerations connect the computational design of decentralised training to the aggregation of individual preferences. The interests of participants may conflict and we have to decide how to resolve situations where they do. Majority preferences can be cyclic: a population may prefer one response to a second, the second to a third, and the third to the first. This is known as the Condorcet paradox (Condorcet, 1785). Consequently, training a shared model requires an explicit rule for resolving disagreement. Social choice theory provides a principled foundation for this task. In particular, a maximal lottery is a distribution over alternatives whose expected pairwise majority margin against any competing distribution is non-negative. It selects the Condorcet winner when one exists, meaning an alternative that defeats every other alternative by majority vote, while accommodating cycles through randomisation. This makes maximal lotteries a natural target for aligning a shared model with a population's preferences (Fishburn, 1984; Brandl et al., 2016; Brandl & Brandt, 2020).

Existing approaches like Nash learning from human feedback pursue this target through gradient-based optimisation (Calandriello et al., 2024; Swamy et al., 2024; Munos et al., 2024; Maura-Rivero et al., 2025). Our focus is on making such preference aggregation practical for independent contributors who evaluate models locally and communicate limited information. EGGROLL provides a computational foundation for this setting: it scales evolution strategies to large language models using low-rank parameter perturbations evaluated through scalar feedback (Sarkar et al., 2025). Participants can reconstruct perturbations from shared random seeds and report fitness values without exchanging gradients or requiring backward passes (Salimans et al., 2017). This permits local evaluation using participants' privately-owned data and preferences.

We introduce VerDICT, a training approach that combines EGGROLL with a pairwise preference objective targeting maximal lotteries. Contributors evaluate responses from perturbed models against baseline responses according to their individual preferences, and the resulting feedback determines updates to a shared model. The approach connects decentralised computation with an explicit rule for aggregating conflicting interests, allowing contributions to training to carry preference information. Training uses only inference: there is no backward pass, optimiser state or reference model, and workers share baseline responses. Every contributor's votes enter the update, and each update can be recomputed from the logged votes.

We evaluate VerDICT on controlled preference games, text summarisation and human preference datasets, against direct preference optimisation (DPO) under the same optimiser and against gradient-based training.

Our contributions are as follows:

- **Decentralised preference aggregation.** We introduce a procedure through which independent participants contribute computation and preference feedback to train a shared model under an explicit social choice objective targeting maximal lotteries.
- **Training by inference alone.** We combine a scalar preference objective with EGGROLL, so training needs no backward pass or reference model, and workers share baseline responses.
- **Empirical evaluation.** We compare VerDICT with EGGROLL-DPO, GD Online DPO and GD Online IPO on controlled preference games, text summarisation, and two human preference datasets, examining aggregation behaviour, language-task performance, and computational efficiency.

2 RELATED WORK

Pluralistic alignment. VerDICT contributes to pluralistic alignment (Sorensen et al., 2024). Unlike distributional preference learning (Siththaranjan et al., 2023), it optimises an axiomatic social choice target rather than a distribution, and trains from a population of independent workers.

Nash learning from human feedback. A family of methods targets the Nash equilibrium of the two-player preference game, which is the maximal lottery (Maura-Rivero et al., 2025): Nash-MD (Munos et al., 2024), online identity preference optimisation (Calandriello et al., 2024, IPO), SPO (Swamy et al., 2024) and SPPO (Wu et al., 2024). SPO rewards each sampled response by its win rate against the policy's other samples and optimises this reward with standard reinforcement learning, while SPPO approximates a multiplicative-weights update by regressing log-probability ratios onto estimated win rates; both reach the equilibrium through iterated self-play. All of these methods are gradient-based and centralised, requiring backward passes through the policy and, for Nash-MD and online IPO, a reference model. VerDICT keeps the self-play structure but scores each perturbed policy against the worst of n_b sampled opponents rather than their average, which is what prevents cycling on rock–paper–scissors (Appendix B.1), and optimises with evolution strategies, so each worker contributes only its votes.

Distributed LLM training. Parameter-server training (Dean et al., 2012) and its descendants, tensor/pipeline parallelism (Shoeybi et al., 2020; Huang et al., 2019) and sharded data parallelism (Rajbhandari et al., 2020; Zhao et al., 2023), communicate gradients or parameters sized in proportion to the model and optimise one shared objective. Low-communication and federated methods (Douillard et al., 2024; McMahan et al., 2017; Ye et al., 2024) relax the bandwidth or data-locality assumption, but not the aggregation rule: Ye et al. (2024) average preference updates across clients. When the compared pairs come from a fixed distribution, the reward this fits ranks alternatives in a Borda-like way (Siththaranjan et al., 2023; Maura-Rivero et al., 2025). Evolution strategies communicate only a scalar fitness per worker (Salimans et al., 2017; Sarkar et al., 2025). We retain this communication pattern and replace the scalar with a preference probability against a per-worker judge, so the VerDICT objective, not an average, resolves disagreement between workers.

Bilevel optimisation. The VerDICT objective is a zero-sum instance of bilevel optimisation: the outer policy maximises its payoff against an opponent that minimises it. This connects to Best Response Shaping (BRS), which optimises an agent against a learned approximation to its opponent's best response (Aghajohari et al., 2024). When specialised to the zero-sum preference game, BRS targets the same ideal objective, although it differentiates through a policy-conditioned opponent. Related work on zeroth-order min-max optimisation estimates gradients from function evaluations and alternates updates to the competing variables (Liu et al., 2020). VerDICT instead approximates the inner best response by selecting among sampled baseline responses and updates the outer policy using EGGROLL, enabling preference-based training without backpropagation.

3 BACKGROUND

Problem setting. We frame our work as a social choice problem. Social choice theory is concerned with aggregating preferences of a population to achieve a collective decision. Assume a population $\mathcal{N}$ of N members, each with a preference over a finite set of alternatives $\mathcal{A}$; given two alternatives $p, q \in \mathcal{A}$, member $i \in \mathcal{N}$ reports either $p \succ_i q$ or $q \succ_i p$. Our setup involves an input x to a policy π, two outputs a and b treated as alternatives from $\mathcal{A}$ and a preference probability $P(a \succ b)$ used as a proxy for the preference of an individual $P(a \succ_i b)$. Prompts are drawn as $x \sim P_X$, and Π is the set of policies $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ is the set of distributions over $\mathcal{A}$. We define the preference probability v and the maximal lottery formally in Section 4.1. Previous work proposes social choice theory as a formalism for AI alignment (Ge et al., 2024; Conitzer et al., 2024), and Maura-Rivero et al. (2025) propose the maximal lottery as its target.

EGGROLL. EGGROLL (Sarkar et al., 2025) is an evolution strategies (ES) algorithm that scales to billion-parameter models by replacing full-rank Gaussian perturbations with low-rank ones. For each linear layer with mean parameters $M \in \mathbb{R}^{m \times n}$, each worker i samples $A_i \in \mathbb{R}^{m \times r}$ and $B_i \in$

$\mathbb{R}^{n\times r}$ with $r \ll \min(m,n)$ and forms the perturbation $E_i = \frac{1}{\sqrt{r}} A_i B_i^\top$. Each worker evaluates the fitness $f_i = f(M + \sigma E_i)$ and broadcasts the scalar f_i. Because perturbations are generated from a shared random seed, every worker can reconstruct the others' E_j locally and apply the same update, $M \leftarrow M + \frac{\alpha}{N}\sum_i E_i f_i$, without exchanging gradients or parameters (Salimans et al., 2017). While individual perturbations are rank r, the aggregated update has rank $\min(Nr, m, n)$, so EGGROLL produces full-rank updates from low-rank search directions. The low-rank structure also makes large populations cheap to evaluate. Since $u(M + \sigma E_i)^\top = uM^\top + \frac{\sigma}{\sqrt{r}}(uB_i)A_i^\top$, all population members share the base product $uM^\top$, which is the same decomposition used by batched low-rank adaptation (LoRA) inference engines such as vLLM (Hu et al., 2021; Kwon et al., 2023). This lets EGGROLL reach up to 91% of the throughput of pure batch inference.

4 VERDICT

We now derive and introduce the VerDICT algorithm, whose goal is to learn a maximal lottery solution for preference alignment using the EGGROLL algorithm.

4.1 DERIVING THE VERDICT OBJECTIVE

Let $v(a \succ b) := P(a \succ b) + \frac{1}{2}P(a \sim b)$ be the probability that response a is preferred to response b, with ties split equally, so that $v(a \succ b) + v(b \succ a) = 1$ and $v(a \succ a) = \frac{1}{2}$. The maximal lottery for preference alignment is the solution of the two-player zero-sum (max–min) problem:

$$\Pi^\star_{\text{ML}} = \arg\max_{\pi\in\Pi} \min_{\pi'\in\Pi} \mathbb{E}_{x\sim P_X}\left[\mathbb{E}_{a\sim\pi(x),b\sim\pi'(x)}\left[v(a \succ b)\right]\right],$$

where Π denotes the set of all conditional distributions $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ mapping a prompt to a distribution over alternatives, and $\Delta(\mathcal{A})$ is the set of probability distributions over $\mathcal{A}$. We assume a parametric model for the policy π_θ, yielding an objective

$$\begin{aligned} J_{\text{ML}}(\theta) &= \min_{\pi'\in\Pi} \mathbb{E}_{x\sim P_X}\left[\mathbb{E}_{a\sim\pi_\theta(x),b\sim\pi'(x)}\left[v(a \succ b)\right]\right], \\ &= \mathbb{E}_{x\sim P_X}\left[\min_{\pi'\in\Pi} \mathbb{E}_{a\sim\pi_\theta(x),b\sim\pi'(x)}\left[v(a \succ b)\right]\right], \end{aligned}$$

where the $\min$ operator commutes with the expectation because $\min$ acts over the unrestricted set of all possible conditional distributions Π (see Rockafellar & Wets (1997, Theorem 14.60)). Consider now the inner minimisation problem

$$\min_{\pi'\in\Pi} \mathbb{E}_{a\sim\pi_\theta(x),b\sim\pi'(x)}\left[v(a \succ b)\right].$$

The solution to this problem is obtained from a deterministic policy that places all probability mass on a single alternative, because a best response to a fixed mixed strategy can always be found among pure strategies (Osborne & Rubinstein, 1994, Chapter 3). This insight reduces the search for a minimising policy to a search for the minimising alternative

$$J_{\text{ML}}(\theta) = \mathbb{E}_{x\sim P_X}\left[\min_{b\in\mathcal{A}} \mathbb{E}_{a\sim\pi_\theta(x)}\left[v(a \succ b)\right]\right].$$

To derive a practical algorithm, we estimate the inner minimisation problem using n_b samples $b_1, \ldots, b_{n_b} \sim \pi'(x)$ from a reference policy for each prompt $x \sim P_X$

$$\min_{b\in\mathcal{A}} \mathbb{E}_{a\sim\pi_\theta(x)}\left[v(a \succ b)\right] \approx \min_{c\in\{1,\ldots,n_b\}} \mathbb{E}_{a\sim\pi_\theta(x)}\left[v(a \succ b_c)\right].$$

This yields the VerDICT objective

$$J_{\text{VerDICT}}(\theta) := \mathbb{E}_{x\sim P_X}\mathbb{E}_{b_1,\ldots,b_{n_b}\sim\pi'(x)}\left[\min_{c\in\{1,\ldots,n_b\}} \mathbb{E}_{a\sim\pi_\theta(x)}\left[v(a \succ b_c)\right]\right].$$

We remark that our objective avoids introducing regularisation such as an entropy term; whilst entropy regularisation can stabilise training, as we show in Appendix B.3, its addition means solutions deviate from maximal lotteries, forgoing key desirable properties such as Pareto efficiency.

4.2 THE VERDICT ALGORITHM

To optimise the VerDICT objective, we apply EGGROLL as a black-box zeroth-order optimiser, treating $J_{\text{VerDICT}}(\theta)$ as the fitness function. A special case is when we choose the reference policy to be the frozen target policy $\pi'(x) = \pi_{\text{StopGrad}(\theta)}(x)$, which we outline in Algorithm 1. For $n_b = 1$, this recovers a self-play algorithm to solve the minimax problem, where the agent optimises against its own response (Swamy et al., 2024; Wu et al., 2024). Because the opponents are drawn from the current policy with its parameters held fixed, the fitness at step t is $J(\theta;\bar{\theta}) = \mathbb{E}_x \mathbb{E}_{b_1,\ldots,b_{n_b}\sim\pi_{\bar{\theta}}(x)}\left[\min_c \mathbb{E}_{a\sim\pi_\theta(x)} v(a \succ b_c)\right]$ with $\bar{\theta} = \theta_t$, and EGGROLL constructs an ES update approximating its gradient in θ at $\theta = \bar{\theta}$. Here $\mathcal{D}_t$ is the batch of prompts drawn from the training prompts $\mathcal{D}$ at step t. This is a self-play dynamic rather than ascent on a single fixed objective: without the stop-gradient, the $n_b = 1$ objective equals $\frac{1}{2}$ for every θ, since $v(a \succ b) + v(b \succ a) = 1$. We do not prove that these dynamics converge to the maximal lottery; Section 5 and Appendix B.1 test this empirically.

Algorithm 1 VERDICT$(\sigma, v, \alpha, T_{\max}, N, \mathcal{D}, n_b)$

Initialise θ_0 and shared seed ς; assume N is even.
for $t = 0, \ldots, T_{\max} - 1$ **do**
  for $k = 1, \ldots, N/2$ **in parallel do**
    Sample $A_k \sim p(A)$, $B_k \sim p(B)$ using seed (ς, t, k); $E_k \leftarrow A_k B_k^\top / \sqrt{r}$
    $\theta_{2k-1} \leftarrow \theta_t + \sigma E_k$; $\theta_{2k} \leftarrow \theta_t - \sigma E_k$
  end for
  for each prompt $x \in \mathcal{D}_t \subset \mathcal{D}$ **do**
    Sample $b_1, \ldots, b_{n_b} \sim \pi_{\theta_t}(\cdot \mid x)$ and share across workers.
    for each worker $i = 1, \ldots, N$ **in parallel do**
      Sample $a_i \sim \pi_{\theta_i}(\cdot \mid x)$; obtain $v_{i,c} \sim v(a_i \succ b_c)$ for all $c = 1, \ldots, n_b$.
    end for
    $c^* \leftarrow \arg\min_c \sum_i v_{i,c}$; $f_i^x \leftarrow v_{i,c^*} - \frac{1}{N}\sum_j v_{j,c^*}$ for all i
  end for
  $F_i \leftarrow \text{mean}_{x\in\mathcal{D}_t} f_i^x$ for all i; $F \leftarrow F/(\text{std}(F) + \varepsilon)$
  Each worker reconstructs $\{E_k\}$ from (ς, t) and applies

$$\theta_{t+1} \leftarrow \theta_t + \frac{\alpha}{N}\sum_{k=1}^{N/2}(F_{2k-1} - F_{2k})E_k.$$

end for

Sampled opponents. Algorithm 1 takes the minimum over n_b opponents sampled from the current policy, not over all alternatives. This overestimates the true minimum when the worst alternative is rarely sampled. With $n_b = 1$ the policy plays against a copy of itself, and on rock–paper–scissors this cycles. With $n_b = 4$ it converges near the uniform lottery on every model (Figure 3). Appendix B.1 checks the update exactly on small games and shows where it is biased. On the text tasks $n_b = 4$ gave no gain over $n_b = 1$, so we use $n_b = 1$ there (Appendix B.2).

Why evolution strategies. In our target setting the judge is a participant that we can query but not differentiate, and ES only needs the query. Each worker sends only its votes, and every update can be recomputed from the shared seed and the logged votes, so incorrect computation can be caught by replay (Appendix B.4). This does not make honest reporting a worker's best strategy; we assume it.

Implementation. Each population member is a rank-8 LoRA perturbation of the attention projections, regenerated from the shared seed and served in batches by vLLM (Kwon et al., 2023); the update is added in place to the engine's weights, so training is inference only. On the text tasks each perturbation is reused for four consecutive steps. Every vote is keyed by a hash of (seed, step, worker, responses), so it can be regenerated. Each engine draws its own opponent responses and selects c^* from its own members' votes, rather than from the whole population as in Algorithm 1; Appendix B.1 examines the effect of this. The online GD baselines use TRL's `OnlineDPOTrainer` (von Werra et al., 2020). Full details are in Appendix A.2.

5 EXPERIMENTS

Baselines. We compare VerDICT against three baselines chosen to separate the effect of the objective from the effect of the optimiser. Online IPO is the standard gradient-based counterpart to VerDICT: Calandriello et al. (2024) show that online IPO optimises the same objective as Nash learning from human feedback (NLHF) (Munos et al., 2024), whose solution is the Nash equilibrium of the symmetric two-player preference game. Since the maximal lottery is exactly this equi-

Table 1: Rock–paper–scissors as a voting game: $v(\text{row} \succ \text{column})$. Preferences are cyclic, so there is no Condorcet winner, Borda scores tie, and the maximal lottery is uniform.

	Rock	Paper	Scissors
Rock	.5	0	1
Paper	1	.5	0
Scissors	0	1	.5

Table 2: Voter profiles for the dominated-alternative game. (a) B is the majority, Condorcet and Borda winner. (b) Adding G, Pareto-dominated by R, keeps every voter's R–B ranking unchanged, so B remains the Condorcet winner and maximal lottery, but the Borda winner flips to R (scores 1.4 vs 1.2), violating IIA for the Borda rule.

Weight	0.4	0.6
1st	R	B
2nd	B	R

(a) Two alternatives.

Weight	0.4	0.6
1st	R	B
2nd	G	R
3rd	B	G

(b) Adding G.

librium (up to the KL regularisation towards the reference policy), online IPO trained with gradient descent (GD Online IPO) is the closest existing method to targeting maximal lotteries, although its regularisation, opponent handling and optimiser all differ from VerDICT's. DPO instead fits a Bradley–Terry reward, which ranks alternatives like the Borda count when the compared pairs come from a fixed distribution (Siththaranjan et al., 2023); with pairs sampled from the policy itself, where it ends depends on the optimiser (Appendix D.1). We therefore include EGGROLL with DPO (EGGROLL-DPO) to isolate the effect of the objective under the same optimiser, and GD with online DPO (GD Online DPO) as the standard gradient-based reference.

Hypotheses and tests. We train Qwen3-0.6B, Qwen3-8B (Yang et al., 2025), and Llama-3.1-8B Instruct (Grattafiori et al., 2024) with VerDICT and each baseline, and test four hypotheses. First, we use rock–paper–scissors to test whether VerDICT converges to the uniform Nash equilibrium when preferences are intransitive and there is no Condorcet winner. Second, we use a stochastic voting game to test whether each method keeps the Condorcet winner when a Pareto-dominated alternative is added, a change that flips the Borda winner and so violates independence of irrelevant alternatives (IIA) for the Borda rule. Third, we use a text summarisation task to test whether VerDICT learns natural language tasks that are not set up as toy voting games. Finally, we use two human preference datasets to test whether VerDICT improves held-out judged quality when trained on real human preferences. These evaluations measure judged win rate against the initial policy, not recovery of the maximal lottery, which is known exactly only in the games.

Synthetic populations. The two games reproduce the controlled experiments of Maura-Rivero et al. (2025) with our methods and models. In each game we specify the population directly: a set of voters, each with a ranking over the alternatives and a weight, and the judge draws one voter per comparison according to the weights and returns that voter's preference. Because the population is known, so is its maximal lottery, and we can check whether each method reaches it. Unlike Maura-Rivero et al. (2025), who train on a fixed dataset of sampled comparisons, votes here are drawn online for the responses the policy actually produces.

Rock–paper–scissors (RPS). We construct an RPS game as a symmetric zero-sum voting game with one fixed prompt (USER: "Let's play rock-paper-scissors! Your valid actions are single characters: r, p, or s." / ASSISTANT: "I choose action") and a single-token answer. Paper beats rock, rock beats scissors and scissors beats paper with probability 1, a move against itself wins with probability .5, and an invalid token loses to every valid move; the judge is this payoff, so $v(a \succ b) \in \{0, \frac{1}{2}, 1\}$ is deterministic, and the maximal lottery is the uniform distribution (Table 1). Here the population is a single voter whose preferences are cyclic; the three-voter cycle of Maura-Rivero et al. (2025) has the same uniform maximal lottery with weaker margins. Because the base models already start near uniform, every model is first supervised fine-tuned to a skewed start (rock .53–.60, paper .24–.28, scissors .15–.20). VerDICT uses $n_b = 4$ opponents per prompt, drawn from the unperturbed current policy by each engine and shared by that engine's members; per prompt the opponent c^* is the one that engine's mean vote finds hardest, and each member's fitness is its own vote against b_{c^*}, centred per prompt and averaged over 256 prompts per step (Algorithm 1). For EGGROLL-DPO each member samples both moves of its pair from its own perturbed policy, the judge labels the pair, and the fitness is the member's negated DPO loss on that pair, $\beta = 0.1$, with a frozen copy of the initial checkpoint as the reference, averaged over 64 prompts per step. Both arms run 1,000 steps

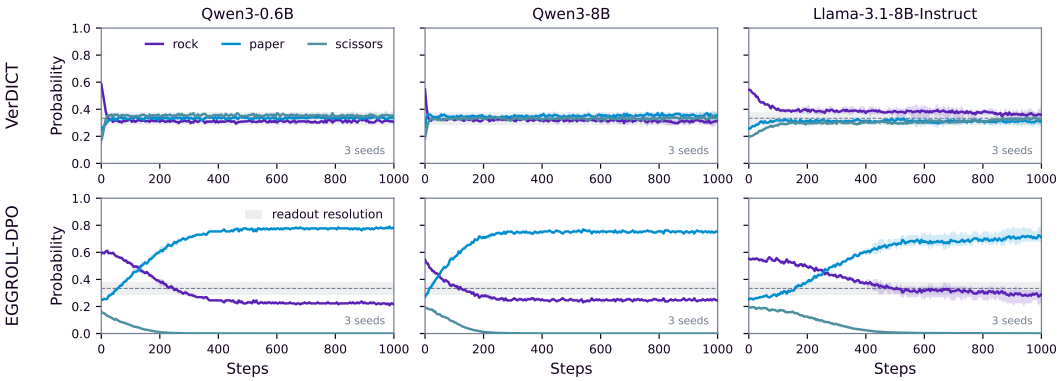

Figure 3: Move probabilities during training on rock–paper–scissors (Table 1) for Qwen3-0.6B, Qwen3-8B and Llama-3.1-8B-Instruct; lines are seed means over three seeds and shaded regions the range across seeds. VerDICT ($n_b = 4$) converges near the uniform maximal lottery on every model (final exploitability, the margin by which the best single move beats the policy: .042, .057 and .058). EGGROLL-DPO ends on the rock–paper edge on every model and seed (paper .68–.80, scissors $\leq$ .001; final exploitability .42–.57). The grey band marks the bf16 readout resolution (an exactly uniform policy reads up to .042).

with three seeds on Qwen3-0.6B (population 512), Qwen3-8B and Llama-3.1-8B-Instruct (population 128), $\sigma = 3 \times 10^{-3}$, rank-8 perturbations, learning rate 2×10^{-3} for VerDICT on the Qwen models and 5×10^{-4} otherwise (Table 4).

Adding a dominated alternative. Following Maura-Rivero et al. (2025), we test whether each method keeps the Condorcet winner when a dominated alternative is added. Like RPS, the game uses one fixed prompt (USER: "What is your favourite color from the options [Red, Blue]? Answer with a single letter: R or B." / ASSISTANT: "My favourite color is") and a single-token answer; the three-alternative prompt adds Green (Appendix A.7). The third alternative, "Green" ("G"), is Pareto-dominated by R, since every voter ranks R above G. Each comparison is judged by a B-first voter with probability 0.6 or an R-first voter with probability 0.4, who rank $B \succ R \succ G$ and $R \succ G \succ B$ when G is present (Table 2). B is the strict Condorcet winner, and hence the maximal lottery, in both games, while adding G changes the Borda winner from B to R, a violation of IIA for the Borda rule. The experiment therefore tests Condorcet consistency under the addition of a dominated alternative; it does not test IIA for the maximal lottery, which concerns the social preference relation rather than the output distribution (Brandl & Brandt, 2020).

TL;DR summarisation. We train on 2,000 rows of the summarisation dataset of Stiennon et al. (2022) (1,741 unique posts at Qwen3-0.6B, 314 at the 8B models), generating up to 64 tokens after "*TL;DR:*". A pairwise DeBERTa-v3 judge (66.9% held-out accuracy, against 73% human inter-rater agreement) casts the training votes, and a second judge trained with a different seed evaluates. Both ES methods run 300 steps with three seeds, and all methods are evaluated on the same 500 held-out posts with the same judge and seeds (Appendix A.3).

Human preference data. We use two human preference datasets: CoVal (Hitzig et al., 2026), in which a median of 16 annotators rank four candidate responses per prompt (866 training and 193 held-out prompts), and MultiPref (Miranda et al., 2024), in which two crowd workers and two domain experts compare two responses per prompt (4,142 training prompts; 500 held-out prompts with a strict majority). For each dataset we fine-tune two pairwise DeBERTa-v3 judges, a training judge that casts the votes and an evaluation judge used only for evaluation; on MultiPref they are trained on disjoint annotator pools (crowd and expert). The population is 256 at Qwen3-0.6B (128 on MultiPref) and 128 at 8B, with $\sigma = 3 \times 10^{-3}$, learning rate 5×10^{-4}, 500 steps, 16 prompts per step and three seeds. We report the evaluation judge's $P(\textit{trained} \succ \textit{initial})$ on held-out prompts, symmetrised over presentation order and averaged over prompts, as a seed mean with a 95% t-interval. Filtering, judge training and accuracy, and generation caps are in Appendix A.4; a third dataset, Community Alignment, is used only in appendix analyses.

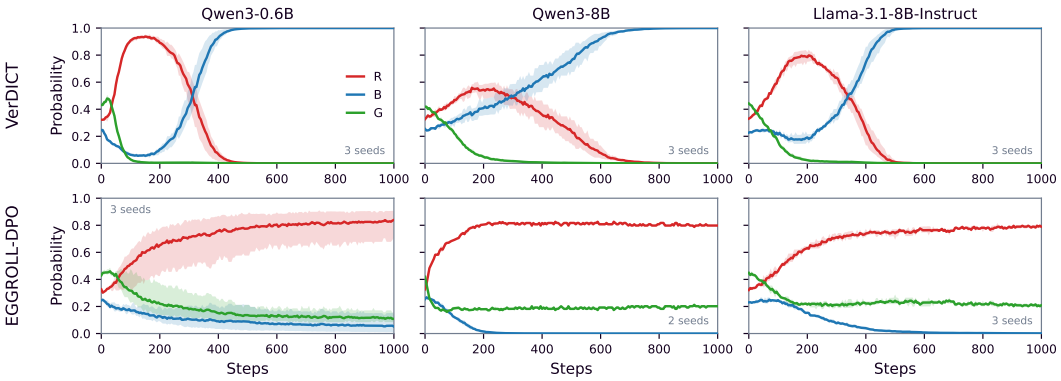

Figure 4: Answer probabilities during training in the three-alternative game, which adds the Pareto-dominated G. VerDICT reaches the Condorcet winner B ($P(B) = 1.000$) on every model; EGGROLL-DPO rests on the Borda winner R. Probabilities are renormalised over the candidate letters; at Qwen3-0.6B EGGROLL-DPO ends with a third of its mass on other tokens (Appendix A.7).

6 Results

6.1 VerDICT approximates the maximal lottery on synthetic tasks

Cyclic preferences (RPS). Our RPS game shows that VerDICT settles towards uniform Nash equilibrium $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ when the n_b term is set to 4 over $1,000$ steps. [1] We find EGGROLL-DPO fails to settle to the Nash equilibrium or cycle between candidates. Rather, EGGROLL-DPO puts a high probability mass on paper, a lower mass on rock, and zero on scissors. EGGROLL-DPO does not find the maximal lottery on any model or seed, while VerDICT approaches it on every model.

Condorcet consistency under a dominated alternative. Adding the dominated alternative G does not move VerDICT off the Condorcet winner B. VerDICT puts essentially all its mass on B at every scale: with three alternatives, $P(B)$ remains above .95 after steps 375–425 (range over seeds) at Qwen3-0.6B, 575–655 at Qwen3-8B and 440–490 at Llama-3.1-8B-Instruct (Figure 4), and with two alternatives from steps 235–255, 685–900 and 145–180. EGGROLL-DPO never puts all its mass on B in the two-alternative setting (final $P(B)$ .58–.95 across models), and in the three-alternative setting it rests on the Borda winner R (final $P(R)$ .70–.91 among valid answers), the aggregation behaviour that violates IIA for the Borda rule. We run an additional experiment to show that both methods give higher probability mass to B when the starting probabilities are switched to demonstrate that B beats R in pairwise comparisons. These results are in Appendix C.3. Gradient descent follows the objective, not the optimiser: GD Online DPO collapses to a single move on rock–paper–scissors and ends on the Condorcet winner in the colour games, while optimising VerDICT's sampled-opponent objective with REINFORCE reaches near-uniform play and the Condorcet winner in 17 of 18 colour-game runs (Appendix D.2).

6.2 VerDICT improves judged quality on summarisation and human preference data

TL;DR summarisation. Table 3 reports the text-task results. With Qwen3-0.6B, VerDICT reaches a $83.8 \pm 2.3\%$ win rate on TL;DR,[2] outperforming EGGROLL-DPO ($59.2 \pm 1.4\%$) and GD Online IPO ($67.0 \pm 0.9\%$), with a confidence interval overlapping that of GD Online DPO ($89.6 \pm 5.4\%$). Length alone does not account for VerDICT's lead over EGGROLL-DPO: at Qwen3-0.6B the two reach the 64-token cap on a similar share of summaries (35% and 39%), but EGGROLL-DPO has about a quarter of VerDICT's mean sampled log-ratio to the initial policy (Appendix F.2). GD Online DPO has the largest log-ratio at every scale, and its lead over VerDICT persists after length adjustment (66.3% against 54.3% at Qwen3-8B; Appendix F.3).

[1] We only increase n_b in RPS where intransitivity matters. n_b is kept at the default setting (1) in all other experiments and we note the effect of increasing n_b in the other experiments in Appendix B.2.

[2] Values are means over three seeds; $\pm$ denotes the 95% confidence interval.

Table 3: Held-out judge win rate vs the initial policy, % (50 = no change). Seed mean ± 95% t-interval over 3 seeds; bold = best in column within the block, intervals non-overlapping.

	Qwen3-0.6B			Qwen3-8B			Llama-3.1-8B-Instruct		
Method	TL;DR	CoVal	MultiPref	TL;DR	CoVal	MultiPref	TL;DR	CoVal	MultiPref
VerDICT	$83.8_{\pm 2.3}$	$55.3_{\pm 1.9}$	$\mathbf{60.4}_{\pm 1.2}$	$64.8_{\pm 1.2}$	$50.7_{\pm 1.0}$	$51.7_{\pm 0.1}$	$72.7_{\pm 2.0}$	$51.1_{\pm 1.0}$	$54.5_{\pm 3.0}$
EGGROLL-DPO	$59.2_{\pm 1.4}$	$51.9_{\pm 2.2}$	$55.0_{\pm 1.2}$	$60.2_{\pm 1.9}$	$51.2_{\pm 1.3}$	$50.6_{\pm 0.7}$	$63.4_{\pm 4.2}$	$50.8_{\pm 0.6}$	$51.6_{\pm 0.2}$
GD Online DPO	$89.6_{\pm 5.4}$	$51.1_{\pm 1.4}$	$55.5_{\pm 1.6}$	$\mathbf{82.0}_{\pm 4.0}$	$50.2_{\pm 0.9}$	$51.1_{\pm 1.5}$	$\mathbf{80.9}_{\pm 4.9}$	$51.0_{\pm 2.8}$	$52.4_{\pm 2.6}$
GD Online IPO	$67.0_{\pm 0.9}$	$49.6_{\pm 1.3}$	$51.7_{\pm 0.7}$	$63.8_{\pm 2.7}$	$50.2_{\pm 1.1}$	$50.4_{\pm 0.5}$	$66.2_{\pm 11.3}$	$50.3_{\pm 1.3}$	$51.0_{\pm 1.2}$

Human preference data. Table 3 shows results on CoVal and MultiPref. At Qwen3-0.6B VerDICT has the highest mean of the four methods on both datasets: $60.4 \pm 1.2\%$ on MultiPref (next best 55.5%) and $55.3 \pm 1.9\%$ on CoVal (next best 51.9%). At the 8B scales all methods stay close to 50% on both datasets; the differences fall within the intervals except on MultiPref at Qwen3-8B, where VerDICT ($51.7 \pm 0.1\%$) is separated from EGGROLL-DPO and GD Online IPO but not from GD Online DPO. Across all nine cells, VerDICT has a higher mean than EGGROLL-DPO in eight; the exception is CoVal at Qwen3-8B. A direct check on CoVal at Qwen3-0.6B (three seeds) points the same way: against a shared pool of 36 answers, VerDICT is the least exploitable policy under the held-out judge (0.085, against 0.129 for EGGROLL-DPO, 0.138 for GD Online DPO and 0.155 for the initial policy), although VerDICT's answers are also the longest (Appendix E.4).

Evolution strategies vs gradient descent. Standard gradient-based training would require every worker to communicate gradients or parameters sized in proportion to the model; we compare against online GD baselines to measure the cost of training from votes alone. These comparisons are not matched in compute: online GD processes 128–256× fewer comparisons per step than the ES methods. GD Online DPO has the highest mean win rate across the nine cells of Table 3, driven by TL;DR, where it leads VerDICT at every scale, with separated intervals at Qwen3-8B and Llama-3.1-8B-Instruct and overlapping intervals at Qwen3-0.6B. On the human preference datasets the order reverses at Qwen3-0.6B, where VerDICT is higher with separated intervals, while at 8B the two overlap; under a length adjustment (Appendix F.3) its MultiPref lead remains (54.5% against at most 51.7%) but its CoVal lead over GD Online DPO does not (52.9% against 53.1%). GD Online IPO, the gradient-based method closest to VerDICT's objective, has a lower mean than VerDICT in all nine cells. Per training step, VerDICT is 1.21–2.01× faster than EGGROLL-DPO on matched hardware (Appendix F.4).

7 Discussion and conclusion

On the games, where the maximal lottery is known, VerDICT converges near the uniform lottery on rock–paper–scissors and keeps the Condorcet winner when a dominated alternative is added, where EGGROLL-DPO does not. On language tasks it improves held-out judged quality, and in the timed runs of an earlier campaign at equal cap it uses 0.47–0.74 of EGGROLL-DPO's A100-hours (Table 16); GD Online DPO scores higher on summarisation but a standard distributed implementation would communicate gradient or parameter updates sized in proportion to the model. Every update can be recomputed from the shared seed and the logged votes, so an auditor can check that votes were aggregated correctly, but not that they were reported honestly, which we assume throughout.

Limitations. We have not deployed VerDICT with real participants, and our opponents are drawn from the current policy, whose support may miss the true minimiser. Training reaches the maximal lottery only approximately, so the social choice properties we cite, which hold exactly or not at all, need not hold for the trained policy. Majority cycles are rare in our human preference data (1.3% of CoVal prompts; not measurable from MultiPref's pairs), so those experiments mostly measure judged improvement; our one direct exploitability check (Appendix E.4) covers a single model and dataset. Our judges are proxies for human voters, and we do not report their calibration, although the maximal lottery depends on the size of pairwise margins; nor have we checked with human raters that higher judged win rates mean better answers. Future work includes a pilot with real voters, alternative reference policies, GD Online ML on the language tasks, human ratings of the trained policies' answers, and mechanisms that make truthful reporting a participant's best strategy.

AI USE STATEMENT

We used generative AI tools (large language model assistants) to write and debug experiment code and run experiments, to draft TikZ figures, to check derivations, to give feedback on algorithm design, to help find relevant literature, and to help draft and edit the text of this paper. Every AI-assisted artefact was reviewed by the authors: code was tested and its outputs checked against logged runs, reported numbers were regenerated from the run records, derivations were checked by hand, and suggested references were read before being cited. We take responsibility for the final content of this work, including text, claims and artefacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses publicly released human preference datasets (the TL;DR comparisons of Stiennon et al. (2022), CoVal, MultiPref and Community Alignment) under their stated terms; we collected no new human data. Our voters are learned judges or specified synthetic populations, so our results say nothing about how real participants would behave in a deployed system. VerDICT is designed to give independent participants a say in training a shared model, but the procedure assumes honest reporting: participants who report strategically, or who control many workers, could steer the model, and verification by replay does not prevent this (Appendix B.4). We do not establish strategyproofness or resistance to participants controlling multiple workers; these remain deployment limitations. The sampled completions in Appendix G are reproduced from held-out prompts of these datasets and include model outputs that may contain factual errors.

REPRODUCIBILITY STATEMENT

Section 4 and Algorithm 1 specify the method, and Appendix A.2 describes the implementation. Table 4 lists the hyperparameters of every experiment, Appendix A.5 the hardware and software versions, Appendix A.7 the exact prompts and judge inputs, and Appendices A.3 and A.4 the dataset filtering, splits and judge training. Every win rate in Table 3 is a seed mean with a 95% t-interval over seeds, analyses based on a single seed or checkpoint are marked as such, and every update is a deterministic function of the shared seed and the logged votes, so runs can be replayed (Appendix B.4). We will release our code upon publication.

REFERENCES

Milad Aghajohari, Tim Cooijmans, Juan Agustin Duque, Shunichi Akatsuka, and Aaron Courville. Best response shaping, 2024. URL https://arxiv.org/abs/2404.06519. 3

Florian Brandl and Felix Brandt. Arrovian Aggregation of Convex Preferences. *Econometrica*, 88(2):799–844, 2020. ISSN 1468-0262. doi: 10.3982/ECTA15749. URL https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA15749. https://onlinelibrary.wiley.com/doi/pdf/10.3982/ECTA15749. 2, 7

Florian Brandl, Felix Brandt, and Hans Georg Seedig. Consistent Probabilistic Social Choice, July 2016. URL http://arxiv.org/abs/1503.00694. arXiv:1503.00694 [cs]. 2

Florian Brandl, Felix Brandt, and Johannes Hofbauer. Welfare maximization entices participation. *Games and Economic Behavior*, 114:308–314, March 2019. ISSN 0899-8256. doi: 10.1016/j.geb.2019.02.008. URL https://www.sciencedirect.com/science/article/pii/S0899825619300181. 26

Daniele Calandriello, Zhaohan Daniel Guo, Remi Munos, Mark Rowland, Yunhao Tang, Bernardo Avila Pires, Pierre Harvey Richemond, Charline Le Lan, Michal Valko, Tianqi Liu, Rishabh Joshi, Zeyu Zheng, and Bilal Piot. Human Alignment of Large Language Models through Online Preference Optimisation. June 2024. URL https://openreview.net/forum?id=2RQqg2Y7Y6. 2, 3, 5

Nicolas de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge University Press, 1785. ISBN 978-1-108-07799-6. Google-Books-ID: r5dxBAAAQBAJ. 2

Vincent Conitzer, Rachel Freedman, Jobst Heitzig, Wesley H. Holliday, Bob M. Jacobs, Nathan Lambert, Milan Mossé, Eric Pacuit, Stuart Russell, Hailey Schoelkopf, Emanuel Tewolde, and William S. Zwicker. Social Choice Should Guide AI Alignment in Dealing with Diverse Human Feedback, June 2024. URL http://arxiv.org/abs/2404.10271. arXiv:2404.10271 [cs.LG]. 3

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://papers.nips.cc/paper_files/paper/2012/hash/6aca97005c68f1206823815f66102863-Abstract.html. 3

Arthur Douillard, Qixuan Feng, Andrei Alex Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, MarcAurelio Ranzato, Arthur Szlam, and Jiajun Shen. DiLoCo: Distributed Low-Communication Training of Language Models. June 2024. URL https://openreview.net/forum?id=pICSfWkJIk#discussion. 3

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators, March 2025. URL http://arxiv.org/abs/2404.04475. arXiv:2404.04475 [cs.LG]. 29, 31

P. C. Fishburn. Probabilistic Social Choice Based on Simple Voting Comparisons. *The Review of Economic Studies*, 51(4):683–692, October 1984. ISSN 0034-6527. doi: 10.2307/2297786. URL https://doi.org/10.2307/2297786. 2

Luise Ge, Daniel Halpern, Evi Micha, Ariel D. Procaccia, Itai Shapira, Yevgeniy Vorobeychik, and Junlin Wu. Axioms for AI Alignment from Human Feedback. In *Advances in Neural Information Processing Systems*, volume 37, pp. 80439–80465. Curran Associates, Inc., 2024. doi: 10.52202/079017-2557. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9328208f88ec69420031647e6ff97727-Abstract-Conference.html. 3

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan,

Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jen-

nifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, November 2024. URL http://arxiv.org/abs/2407.21783. arXiv:2407.21783 [cs]. 6

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. September 2022. URL https://openreview.net/forum?id=sE7-XhLxHA. 17

Zoë Hitzig, Mitchell Gordon, Tyna Eloundou, Adam Kalai, and Sandhini Agarwal. CoVal: Learning values-aware rubrics from the crowd. Technical report, OpenAI, January 2026. URL https://alignment.openai.com/coval/. 7, 17

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. October 2021. URL https://openreview.net/forum?id=nZeVKeeFYf9. 4

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism, July 2019. URL http://arxiv.org/abs/1811.06965. arXiv:1811.06965 [cs.CV]. 3

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, October 2023. Association for Computing Machinery. ISBN 979-8-4007-0229-7. doi: 10.1145/3600006.3613165. URL https://dl.acm.org/doi/10.1145/3600006.3613165. 4, 5, 16

Sijia Liu, Songtao Lu, Xiangyi Chen, Yao Feng, Kaidi Xu, Abdullah Al-Dujaili, Minyi Hong, and Una-May O'Reilly. Min-max optimization without gradients: Convergence and applications to adversarial ml, 2020. URL https://arxiv.org/abs/1909.13806. 3

Roberto-Rafael Maura-Rivero, Marc Lanctot, Francesco Visin, and Kate Larson. Jackpot! Alignment as a Maximal Lottery, January 2025. URL http://arxiv.org/abs/2501.19266. arXiv:2501.19266 [cs]. 2, 3, 6, 7

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, April 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html. 3

Lester James Validad Miranda, Yizhong Wang, Yanai Elazar, Sachin Kumar, Valentina Pyatkin, Faeze Brahman, Noah A. Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. Hybrid Preferences: Learning to Route Instances for Human vs. AI Feedback. October 2024. URL https://openreview.net/forum?id=RLzeoy4FzP. 7, 17

Remi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Côme Fiegel, Andrea Michi, Marco Selvi, Sertan Girgin, Nikola Momchev, Olivier Bachem, Daniel J. Mankowitz, Doina Precup, and Bilal Piot. Nash Learning from Human Feedback. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 36743–36768. PMLR, July 2024. URL https://proceedings.mlr.press/v235/munos24a.html. 2, 3, 5

Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. The MIT Press, Cambridge, Massachusetts ;, 1994. ISBN 9780262281072. 4

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, May 2020. URL http://arxiv.org/abs/1910.02054. arXiv:1910.02054 [cs.LG]. 3

R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational analysis*. Grundlehren der mathematischen Wissenschaften ; 317. Springer, New York ;, 1997. ISBN 3540627723. 4

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning, September 2017. URL http://arxiv.org/abs/1703.03864. arXiv:1703.03864 [stat.ML]. 2, 3, 4

Bidipta Sarkar, Mattie Fellows, Juan Agustin Duque, Alistair Letcher, Antonio León Villares, Anya Sims, Clarisse Wibault, Dmitry Samsonov, Dylan Cope, Jarek Liesen, Kang Li, Lukas Seier, Theo Wolf, Uljad Berdica, Valentin Mohl, Alexander David Goldie, Aaron Courville, Karin Sevegnani, Shimon Whiteson, and Jakob Nicolaus Foerster. Evolution Strategies at the Hyperscale, November 2025. URL https://arxiv.org/abs/2511.16652v2. 2, 3

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, March 2020. URL http://arxiv.org/abs/1909.08053. arXiv:1909.08053 [cs.CL]. 3

Anand Siththaranjan, Cassidy Laidlaw, and Dylan Hadfield-Menell. Distributional Preference Learning: Understanding and Accounting for Hidden Context in RLHF. October 2023. URL https://openreview.net/forum?id=0tWTxYYPnW. 3, 6, 25, 26

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, August 2024. URL http://arxiv.org/abs/2402.05070. arXiv:2402.05070 [cs.AI]. 1, 3

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, February 2022. URL http://arxiv.org/abs/2009.01325. arXiv:2009.01325 [cs.CL]. 7, 10, 17

Gokul Swamy, Christoph Dann, Rahul Kidambi, Steven Wu, and Alekh Agarwal. A Minimaximalist Approach to Reinforcement Learning from Human Feedback. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 47345–47377. PMLR, July 2024. URL https://proceedings.mlr.press/v235/swamy24a.html. 2, 3, 5

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl. 5, 17

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-Play Preference Optimization for Language Model Alignment. November 2024. URL https://openreview.net/forum?id=Z1PDdGekgn#discussion. 3, 5

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, May 2025. URL http://arxiv.org/abs/2505.09388. arXiv:2505.09388 [cs.CL]. 6

Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. OpenFedLLM: Training Large Language Models on Decentralized Private Data via Federated Learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 6137–6147, New York, NY, USA, August 2024. Association for Computing Machinery. ISBN 979-8-4007-0490-1. doi: 10.1145/3637528.3671582. URL https://dl.acm.org/doi/10.1145/3637528.3671582. 3

Lily Zhang, Smitha Milli, Karen Jusko, Jonathan Smith, Brandon Amos, Wassim Bouaziz, Manon Revel, Jack Kussman, Yasha Sheynin, Lisa Titus, Bhaktipriya Radharapu, Jane Dwivedi-Yu, Vidya Sarma, Kristopher Rose, and Maximilian Nickel. Cultivating Pluralism In Algorithmic Monoculture: The Community Alignment Dataset. *International Conference on Learning Representations*, 2026:147076–147215, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ee208bfc04b1bf6125a6a34baa1c28d3-Abstract-Conference.html. 17

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel, September 2023. URL http://arxiv.org/abs/2304.11277. arXiv:2304.11277 [cs.DC]. 3

A EXPERIMENTAL DETAILS

A.1 HYPERPARAMETERS

Table 4 lists the settings for every experiment. Both ES methods use rank-8 perturbations on the attention projections, and VerDICT uses no entropy or KL term in any run. The gradient baselines use TRL's `OnlineDPOTrainer` with a rank-8 LoRA policy, AdamW at learning rate 5×10^{-5} and $\beta = 0.1$, on the same prompts and judges as the ES runs.

Table 4: ES hyperparameters for each experiment. N: population size; σ: perturbation scale; α: learning rate; n_b: VerDICT opponent samples; β: EGGROLL-DPO temperature; Cap: maximum new tokens; T: sampling temperature. Entries of the form V / D give VerDICT / EGGROLL-DPO where the two differ, including the number of seeds. Colour-2 and Colour-3 elsewhere refer to the two- and three-alternative colour games (Table 2).

	N	σ $\times 10^{-3}$	α $\times 10^{-4}$	Steps	Seeds	Prompts/member per step	n_b	β	Cap (tokens)	T
RPS										
Qwen3-0.6B	512	3	20 / 5	1000	3	256 / 64	4	0.1	1	1.0
Qwen3-8B	128	3	20 / 5	1000	3	256 / 64	4	0.1	1	1.0
Llama-3.1-8B	128	3	5	1000	3	256 / 64	4	0.1	1	1.0
Colour game, 2 alternatives										
Qwen3-0.6B	512	3	20	1000	3	64	1	0.1	1	1.0
Qwen3-8B	128	3	20 / 5	1000	3	64	1	0.1	1	1.0
Llama-3.1-8B	128	3	5	1000	3	64	1	0.1	1	1.0
Colour game, 3 alternatives										
Qwen3-0.6B	512	1 / 3	5	1000	3	256 / 64	1	0.1	1	1.0
Qwen3-8B	128	1 / 3	20	1000	3 / 2	256 / 64	1	0.1	1	1.0
Llama-3.1-8B	128	1 / 3	5	1000	3	256 / 64	1	0.1	1	1.0
TL;DR										
Qwen3-0.6B	256	3	5	300	3	4	1	5	64	0.8
Qwen3-8B	128	3	5	300	3	4	1	5	64	0.8
Llama-3.1-8B	128	3	5	300	3	4	1	5	64	0.8
CoVal										
Qwen3-0.6B	256	3	5	500	3	16	1	0.1	512	0.8
Qwen3-8B	128	3	5	500	3	16	1	0.1	896	0.8
Llama-3.1-8B	128	3	5	500	3	16	1	0.1	640	0.8
Community Alignment										
Qwen3-0.6B	256	3	5	500	3	16	1	0.1	640	0.8
Qwen3-8B	128	3	5	500	3	16	1	0.1	1152	0.8
Llama-3.1-8B	128	3	5	500	3	16	1	0.1	640	0.8
MultiPref										
Qwen3-0.6B	128	3	5	500	3	16	1	0.1	256	0.8
Qwen3-8B	128	3	5	500	3	16	1	0.1	640	0.8
Llama-3.1-8B	128	3	5	500	3	16	1	0.1	384	0.8

A.2 IMPLEMENTATION

Our implementation is written in Python on top of vLLM (Kwon et al., 2023), Ray and PyTorch, and simulates the worker population on a single node. Every population member is a rank-8 LoRA adapter on the attention projections whose factors are Gaussian noise regenerated from the shared seed (ς, t, k); antithetic twins share A_k and flip the sign of B_k. The adapters are loaded into the vLLM engine, which serves the whole population in one batched inference call per GPU. The update $\sum_k (F_{2k-1} - F_{2k}) E_k$ is then formed as a dense matrix and added in place to the engine's base weights, so training consists of inference only: there is no backward pass, optimiser state, or separate training copy of the model. VerDICT's n_b opponent responses are sampled from the unperturbed base weights by each engine and shared by that engine's members; EGGROLL-DPO reads its policy log-probabilities off the generation and scores the reference log-probabilities with a frozen copy of the initial checkpoint. Population members are simulated by judges that run alongside each engine:

on the language tasks a pairwise DeBERTa-v3 preference model (He et al., 2022), fine-tuned on the dataset's own annotator labels, returns $P(a \succ b)$, from which a hard vote is drawn; on the games a voter is drawn from the specified voter population and casts that voter's preference. Every vote is keyed by a hash of (seed, step, worker, responses), so any worker's vote can be regenerated and audited. The online GD baselines use TRL's `OnlineDPOTrainer` (von Werra et al., 2020) with a rank-8 LoRA policy and the same prompts and judges (AdamW, learning rate 5×10^{-5}, $\beta = 0.1$). Hardware and software details are in Appendix A.5.

A.3 TL;DR SETUP

To test VerDICT on a natural language task, we train on 2,000 rows of the training split (shuffled with seed 0 for Qwen3-0.6B, the first 2,000 for the 8B models; Appendix A.7) of the dataset introduced by Stiennon et al. (2022)[3]. The model is given a Reddit post with "*TL;DR:*" appended and generates up to 64 new tokens at temperature 0.8. The judge is a two-input pairwise `DeBERTaV3` (He et al., 2022) reward model fine-tuned on the dataset's training split (24k of 93k comparisons), which scores 66.9% accuracy on 2k held-out comparisons; the evaluation judge, trained with a different seed, scores 67.3% (Figure 5). Human inter-rater agreement for the dataset is $73\% \pm 4\%$ (Stiennon et al., 2022). Both ES methods use population 256 at Qwen3-0.6B and 128 at the 8B models, four prompts per step, $\sigma = 3 \times 10^{-3}$, learning rate 5×10^{-4}, 300 steps and three seeds (Table 4); EGGROLL-DPO uses $\beta = 5$. We evaluate on 500 prompts from the `valid2` split, truncating each prompt to a 1,024-token window that keeps the "*TL;DR:*" cue and stopping generation at a newline. All methods, including the online-GD baselines, are evaluated on the same 500 held-out posts with the same judge and seeds. The trained and initial policies each produce a summary, a and b, and the judge returns $P(a \succ b)$; its mean over the 500 prompts is the win rate, and Table 3 reports the mean over seeds with a 95% t-interval over seeds.

A.4 HUMAN PREFERENCE DATA

Human preference data. We evaluate on two real human preference datasets, CoVal and MultiPref, each described below. A third dataset, Community Alignment (Zhang et al., 2026), appears only in supplementary appendix analyses. Each dataset is filtered and split by a prompt-level hash into training and held-out prompts (nominally 80 / 20 %); the exact prompt formats are in Appendix A.7 and sampled completions in Appendix G.

CoVal. CoVal (Hitzig et al., 2026), released by OpenAI, contains rows of one prompt and four candidate responses, each ranked (ties allowed) by a median of 16 human annotators drawn from a pool of 1,012; we use each annotator's "world" ranking (what the model should say), falling back to the "personal" ranking where the former is missing. We keep prompts of at most 350 words, candidates of at most 200 words and prompts with at least 5 rankings, giving 866 training and 193 held-out prompts. Every pair of candidates that an annotator placed in different tiers gives one labelled comparison (up to six per ranking); the training prompts yield 77,742, from which the judges are trained.

MultiPref. MultiPref (Miranda et al., 2024), released by AllenAI, contains 10,461 comparisons of one prompt and two candidate responses (5,323 prompts), each annotated with an overall preference (ties allowed) by four annotators: two crowd workers and two domain experts. We keep prompts of at most 350 words (and at most 640 Qwen3 tokens), leaving 10,055 comparisons, and split by prompt into 4,142 training and 966 held-out prompts; a tie annotation casts no vote. The evaluation uses the first 500 of the 921 held-out prompts whose four annotations give a strict majority. Judges are trained on the individual strict annotations of training-prompt comparisons whose candidates are at most 400 words, one comparison per annotation, split by annotator pool. As recorded by the judge training runs, the judges were trained on 8,060 crowd and 8,045 expert comparisons and validated on 436 and 454 comparisons from a 5 % prompt-level split.

Ties. The judges are trained only on strict comparisons: CoVal pairs placed in the same tier and MultiPref tie annotations cast no vote. The judges therefore estimate $P(a \succ b \mid a \not\sim b)$ rather than

[3] `CarperAI/openai_summarize_comparisons`, the preference version of the TL;DR dataset.

$v(a \succ b)$ (Section 4.1), which rescales each pairwise margin by $1/(1 - t_{ab})$, where t_{ab} is the tie rate of the pair. When tie rates differ across pairs this can change the maximal lottery, so on the language tasks the target is the maximal lottery of strict preferences.

Judges. For each dataset we fine-tune two pairwise judges from `OpenAssistant/reward-model-deberta-v3-base` (two epochs, learning rate 5×10^{-6}, fp32, effective batch 16, 1,024-token window), following the TL;DR judge recipe (which uses a 768-token window): a *training judge*, which is the reward during training, and an *evaluation judge* with a different seed and a different validation split, used only for evaluation. On CoVal both judges are trained on 24,000 comparisons sampled from the same pool; on MultiPref the training judge is trained on the crowd annotations and the evaluation judge on the expert annotations of the same training comparisons, so the two judges share items but never an annotator. Accuracy against individual human comparisons on each judge's own validation split (5 % of the training prompts by hash; on CoVal only 30 / 46 prompts) is 73.3 % / 68.0 % (training / evaluation judge) on CoVal and 62.6 % / 64.8 % on MultiPref (72.3 % for both against the majority of the four annotations on 924 held-out comparisons; Figure 5); the annotators' own agreement with the pairwise majority over each whole dataset, an approximate ceiling for this accuracy, is 76.9 % on CoVal and 81.4 % on MultiPref.

Training and evaluation. Both datasets share one setup. The policies are Qwen3-0.6B, Qwen3-8B and Llama-3.1-8B-Instruct with rank-8 LoRA adapters; the population is 256 members at Qwen3-0.6B (128 on MultiPref) and 128 at 8B, with $\sigma = 3 \times 10^{-3}$, learning rate 5×10^{-4}, 500 steps and, per step, one batch of 16 training prompts shared by every member (each perturbation is kept for four consecutive steps), with three independent seeds per cell (Table 4). Responses are capped at 512 / 896 / 640 tokens on CoVal and 256 / 640 / 384 on MultiPref (Qwen3-0.6B / Qwen3-8B / Llama-3.1-8B-Instruct), at training and at evaluation alike. For each prompt, every member i generates a response a_i. Under VerDICT the comparator b is a response of the current (unperturbed) policy, drawn once per prompt on each engine and shared by that engine's members (a single opponent, $n_b = 1$); the training judge casts a vote between a_i and b, drawn with its probability $P(a_i \succ b)$, and the member's fitness is $+1$ if a_i wins the vote and -1 otherwise, centred over the population per prompt, averaged over the 16 prompts and normalised by its standard deviation before the update. Under EGGROLL-DPO both responses of a pair are drawn from the member's perturbed policy, the training judge casts the same kind of vote between them, and the fitness is minus the DPO loss ($\beta = 0.1$) of that vote on the pair's per-token-mean log-probabilities under the member, with the initial policy as the reference. We evaluate on each dataset's held-out prompts (193 on CoVal, 500 on MultiPref), generating one response per policy per prompt at temperature 0.8 from the same seed (prompts truncated to 448 tokens, which affects 3 of the 500 MultiPref prompts and none of the others), and the evaluation judge returns $P(\textit{trained} \succ \textit{initial})$ against the initial policy's own response to that prompt, symmetrised over the two presentation orders. Its mean over held-out prompts is the win rate; Table 3 and Figure 9 report the mean over the seeds with a 95 % t-interval over seeds ($t_{.975,2} = 4.303$). We analyse the social choice properties of the datasets in Appendix E.

A.5 Hardware and software

All models run in bf16. Each run splits the population evenly across one vLLM engine per GPU, so the number of GPUs sets how many engines sample in parallel and is chosen for throughput, not memory. The Qwen3-0.6B runs use two GPUs (RTX PRO 4500 Blackwell, 32 GB, for most cells; RTX 4090, A40, A100, H100 or RTX PRO 6000 for the rest), although one such GPU holds the full population. The Qwen3-8B and Llama-3.1-8B-Instruct runs use one to eight data-centre GPUs (A100, H100, H200, H200 NVL or RTX PRO 6000 Blackwell; 80–141 GB each). The online-GD baselines run on a single GPU of the same kinds, or, for some Qwen3-0.6B controls, an L40S, RTX A6000, RTX 6000 Ada or A100 40 GB. Table 16 lists the hardware of the timed runs. Software versions: vLLM 0.17.0, PyTorch 2.10, transformers 4.57.6, peft 0.19.1, Ray 2.55.0, TRL 1.13.0.

A.6 Trained judge accuracy

Figure 5 shows the accuracy of our trained judges.

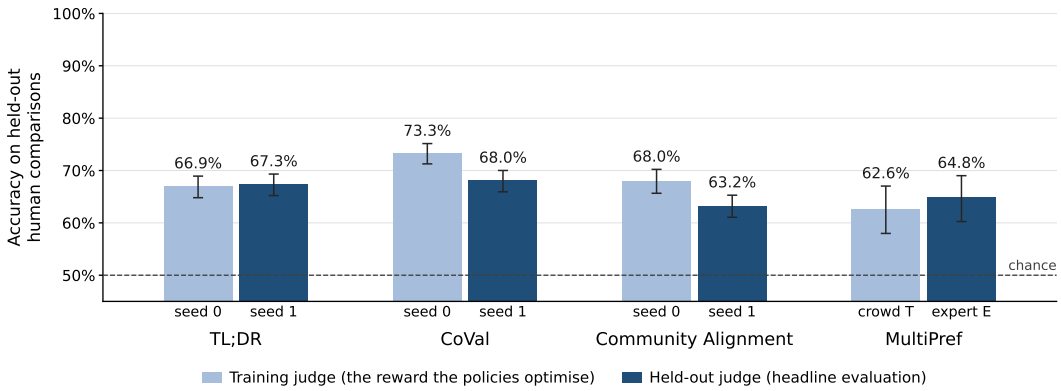

Figure 5: Accuracy of each trained judge against individual human comparisons on its own held-out validation split. The two bars of a dataset use different validation prompts, so they are not a head-to-head comparison. On MultiPref the training judge (crowd annotations) and the evaluation judge (expert annotations) both reach 72.3% against the majority of the four annotations on 924 held-out comparisons.

A.7 PROMPTS

Games. Game prompts are raw strings with no chat template or system prompt; Llama-3.1 receives only its default `<|begin_of_text|>` token. Each candidate is a single token with a leading space. Line breaks in the listings below are for layout only; newlines in the prompts are shown as `\n`.

```
RPS:      User: Let's play rock-paper-scissors! Your valid actions are single
          characters: r, p, or s.\n\nAssistant: I choose action
          candidates: " r", " p", " s"
Colour-2: User: What is your favourite color from the options [Red, Blue]?
          Answer with a single letter: R or B.\n\nAssistant: My favourite
          color is
          candidates: " R", " B"
Colour-3: User: What is your favourite color from the options [Red, Blue,
          Green]? Answer with a single letter: R, B, or G.\n\nAssistant: My
          favourite color is
          candidates: " R", " B", " G"
```

Reported probabilities are the softmax over the candidates' log-probabilities under the unperturbed policy, so they are renormalised over the candidates. An fp32 readout of the unrenormalised probabilities of 41 of the 53 ES final checkpoints in Figures 3 and 4 puts at least .996 of the mass on the candidate tokens in every measured run except EGGROLL-DPO on the three-alternative game at Qwen3-0.6B, which ends at .687 and .669 on seeds 0 and 1, with most of the rest on the unabbreviated colour names; its unconditional $P(R)$ is .63. VerDICT's measured finals on that game are at least .9999, and every online-GD final is at least .9994. On the three-alternative prompt Qwen3-0.6B starts with about 4% of its mass on the candidate letters, so the early part of both curves in Figure 4 at that scale describes a small share of the policy. On that game at Llama-3.1-8B-Instruct, the five readable finals (VerDICT seeds 0, 1 and 2, EGGROLL-DPO seeds 1 and 2) are at least .9998. The other 12 ES finals were not read; they include the six Llama-3.1-8B-Instruct two-alternative finals and its three-alternative EGGROLL-DPO seed 0, whose weights were not retained. During training, a sampled token that does not match a candidate is classed as invalid and loses to every candidate.

TL;DR. The prompt is the dataset's `SUBREDDIT / TITLE / POST` text with `\nTL;DR:` appended, with no chat template on any model; generation stops at the first newline or 64 tokens. The Qwen3-0.6B cells train on the first 2,000 rows after a seed-0 shuffle (1,741 unique posts) and the 8B cells on the first 2,000 rows unshuffled (314 unique posts).

CoVal and MultiPref. Each conversation is wrapped once, offline, in the model family's chat template with the generation prompt added (thinking disabled for Qwen3); no system prompt is added beyond Llama-3.1's template default.

```
Qwen3:    <|im_start|>user\n{message}<|im_end|>\n<|im_start|>assistant\n
          <think>\n\n</think>\n\n
```

```
Llama-3.1: <|start_header_id|>system<|end_header_id|>\n\nCutting Knowledge
           Date: December 2023\nToday Date: 26 Jul 2024\n\n<|eot_id|>
           <|start_header_id|>user<|end_header_id|>\n\n{message}<|eot_id|>
           <|start_header_id|>assistant<|end_header_id|>\n\n
```

Multi-turn CoVal transcripts map each turn to one message. At evaluation, prompts are right-truncated to 448 tokens, which affects 3 of the 500 MultiPref prompts and no CoVal prompt.

Judge input. Every judge scores the prompt, mapped back to the flat form `user: {message}\nassistant:`, followed by `\n\nSummary A: {a}\n\nSummary B: {b}`; when the window is exceeded, the head of the prompt is truncated and both responses are kept whole. Each comparison is scored in both orders, $p = \frac{1}{2}\big(s(f(a,b)) + 1 - s(f(b,a))\big)$ with s the logistic function, and a training vote is drawn from p. The judges never see the chat template or the Llama system block.

B Analysis of the VerDICT update

B.1 What the implemented update does with a tabular game

We say VerDICT approximates the maximal lottery on rock–paper–scissors and the colour games. This appendix checks what the update does, using a small tabular version of the algorithm with three alternatives, where everything can be computed exactly. We use two games: RPS, and a non-uniform cyclic game whose maximal lottery is $(0.6, 0.1, 0.3)$. We measure distance to the maximal lottery by exploitability: how much the best single opponent wins against the policy.

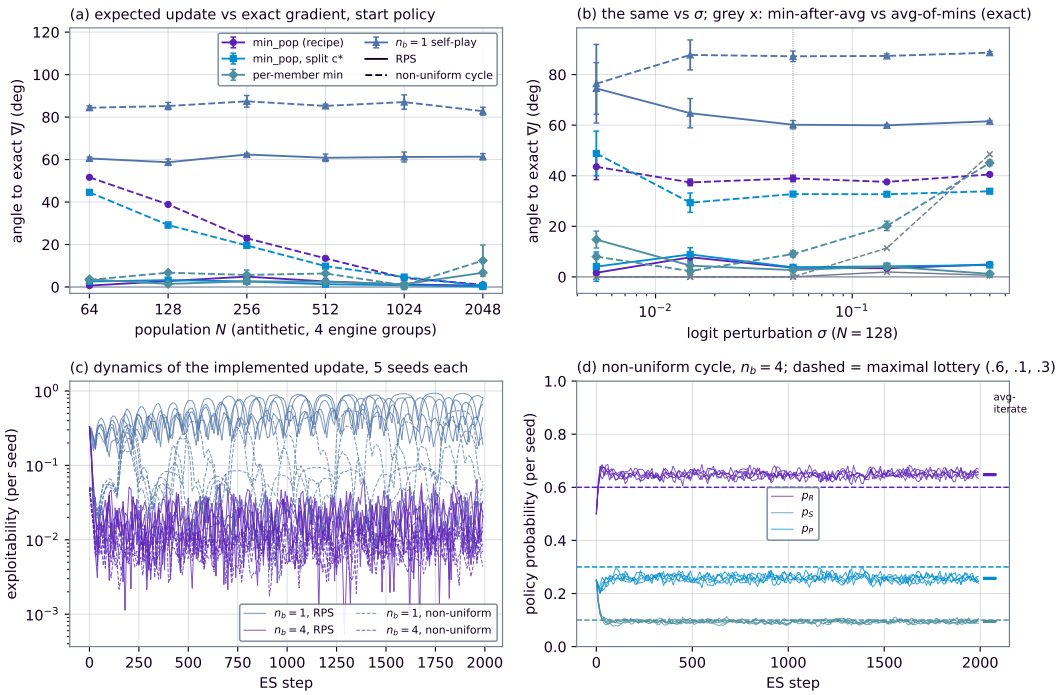

Figure 6: Tabular check of the VerDICT update. Solid lines: RPS; dashed: non-uniform game. **(a)** Angle between the average update and the exact gradient of the sampled-opponent objective J_{VerDICT}, against population size ($N/4$ votes per opponent selection). **(b)** The same at $N = 128$, against perturbation size σ. **(c)** Exploitability during training, 5 seeds each. **(d)** Alternative probabilities on the non-uniform game with $n_b = 4$; dashed lines mark the maximal lottery, ticks on the right the average iterate.

Does the update follow the right gradient? With n_b sampled opponents, the target is the gradient of $J_{\text{VerDICT}}(\theta) = \mathbb{E}\big[\min_c \mathbb{E}_{a\sim\pi_\theta} v(a \succ b_c)\big]$, the expected vote against the worst sampled opponent,

which we can compute exactly. The implementation picks the worst opponent using the average vote of the population members on each engine, so that choice is noisy when there are few votes. Figure 6(a) shows the result. On RPS the update points almost exactly along the true gradient at every population size. On the non-uniform game it is 39° off at $N = 128$ and within 5° from $N = 1{,}024$, because the opponents' true margins are closer together there, so noise more often picks the wrong opponent. Our LLM runs use between 32 and 256 votes per selection, corresponding to $N = 128$ to $1{,}024$. Figure 6(b) shows the error does not depend on the perturbation size, which is consistent with vote noise being its main source. With $n_b = 1$ there is no minimum to take, and the update is ordinary self-play.

Where does training end up? Figure 6(c) shows exploitability over 2000 steps at $N = 128$, our noisiest setting. With $n_b = 1$, training cycles around the equilibrium and does not settle, as expected of self-play on a cyclic game. With $n_b = 4$, every seed converges and exploitability stays around 0.01 to 0.02. On RPS the policy converges to the maximal lottery.

On the non-uniform game it converges to about $(0.65, 0.1, 0.25)$ instead of $(0.6, 0.1, 0.3)$ (Figure 6(d)). A possible explanation is that at the maximal lottery every opponent is equally good in expectation, so the noisy minimum tends to pick whichever opponent has the most variable votes. The policy then over-defends against those opponents. RPS does not show this because all three opponents have the same vote variance. More votes per selection reduce the effect but, in our tests, do not remove it.

Summary. Sampling several opponents and training against the worst one is what makes VerDICT converge instead of cycling. With enough votes per selection the update follows the gradient of the sampled-opponent objective J_{VerDICT}. On games where opponents differ in vote variance, it converges to a point near, but not exactly at, the maximal lottery. This is why we describe VerDICT as approximating the maximal lottery rather than converging to it. The check uses three alternatives, deterministic votes and perturbations on logits rather than LoRA weights, so it does not model every source of noise in the LLM runs. It also draws a fresh perturbation at every step, whereas our LLM runs reuse each perturbation for four steps, so later uses are evaluated at parameters that already depend on it and the standard argument for fresh ES perturbations does not apply directly.

B.2 n_b EXPERIMENTS

Table 5 shows VerDICT at Qwen3-0.6B with different numbers of opponents n_b. On the text tasks, $n_b > 1$ changes the win rate by at most 3.3 points and usually much less, so we use $n_b = 1$ there. More opponents also cost more: at Qwen3-0.6B on TL;DR a step takes 32 / 78 / 159 s at $n_b = 1$ / 4 / 8. On rock–paper–scissors $n_b = 1$ cycles (exploitability .20–.63 over 500 steps), while $n_b = 4$ and $n_b = 8$ converge. Appendix B.1 explains why.

B.3 COUNTEREXAMPLE: ENTROPY REGULARISATION BREAKS PARETO EFFICIENCY

Our solution does not use an entropy regularisation term because such an algorithm would not target the maximal lottery precisely. Recall that the maximal lotteries solution is obtained from solving a minimax objective $\arg\max_{\pi \in \Pi} J_{\text{ML}}(\pi)$ where

$$J_{\text{ML}}(\pi) = \min_{\pi' \in \Pi} \mathbb{E}_{x \sim P_X} \left[\mathbb{E}_{a \sim \pi(x), b \sim \pi'(x)} \left[v(a \succ b) \right] \right].$$

Entropy regularisation introduces an additional term for algorithm stability

$$J_{\text{reg}}(\pi) = \min_{\pi' \in \Pi} \mathbb{E}_{x \sim P_X} \left[\mathbb{E}_{a \sim \pi(x), b \sim \pi'(x)} \left[v(a \succ b) \right] \right] - \tau \mathbb{E}_{\pi} \left[\log \pi \right],$$

where $\tau \geq 0$ controls the degree of regularisation. Introducing entropy regularisation is equivalent, up to a constant, to subtracting a KL regularisation term $\tau \text{KL}(\pi \,\|\, \mathcal{U})$, which penalises a policy for deviating from the uniform distribution $\mathcal{U}$. This term softens the deterministic policies that would be learned from the unregularised objective, so solutions become stochastic and can forgo Pareto efficiency.

Concretely, to show that Pareto efficiency is lost, we consider a game with two alternatives $\mathcal{A} = \{a_1, a_2\}$. Everyone prefers a_1 to a_2, and so the Pareto solution is to place all probability mass on

Table 5: Effect of the number of opponent samples n_b on VerDICT (Qwen3-0.6B). The text-task rows use $\sigma = 10^{-3}$ and the legacy evaluator, so their values are not directly comparable to Table 3 (e.g. TL;DR at $n_b = 1$ here is 63.0, against 83.8 there). Text tasks report the held-out judge win rate vs the initial policy (%, ± 95% interval); games report last-50-readout mean / final readout. – means not run. †One seed (judge CI for text tasks); other entries are over 3 seeds. The game rows use the recipes of Table 4. The text rows differ from Table 3 in two ways: $\sigma = 10^{-3}$ rather than 3×10^{-3}, and an earlier evaluator that truncated posts to 448 tokens from the right, cutting the "TL;DR:" cue on 61 of the 500 held-out posts. Compare values within this table only.

	Steps	$n_b = 1$	$n_b = 2$	$n_b = 4$	$n_b = 8$
Text tasks: win rate (%)					
TL;DR ($n = 500$)	300	$63.0_{\pm 2.7}$	$66.3_{\pm 2.7}$†	$64.5_{\pm 2.7}$	$64.3_{\pm 2.8}$†
TL;DR, KL anchor (weight 0.1)	300	$64.7_{\pm 2.7}$	–	$63.4_{\pm 2.7}$†	–
MultiPref ($n = 500$)	500	$57.6_{\pm 0.9}$†	–	$57.6_{\pm 0.9}$†	–
CoVal ($n = 193$)	500	$54.2_{\pm 1.9}$†	–	$54.4_{\pm 1.9}$†	–
Community Alignment ($n = 203$)	500	$54.0_{\pm 1.3}$†	–	$53.9_{\pm 1.4}$†	–
Games: last-50 mean / final readout					
RPS exploitability	500	.351 / .474†	–	.034 / .000†	.027 / .042†
Colour-3, $P(B)$	500	.997 / .999†	–	.955 / .998†	–
Colour-2 switched start, $P(B)$	1000	1.000 / 1.000†	–	1.000 / 1.000†	–

alternative a_1. Writing the regularised objective in vector form

$$J_{\text{reg}}(\pi) = \min_{\pi'} \pi^\top K \pi' - \tau \pi^\top \log \pi,$$

where K is the margin matrix

$$K = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

For convenience we use the margin $P(a \succ b) - P(b \succ a) = 2v(a \succ b) - 1$ here instead of v. The two differ by a fixed scaling and shift, so they have the same maximiser. The space of policies is

$$\pi_\theta = \begin{bmatrix} \theta \\ 1 - \theta \end{bmatrix},$$

where θ controls the preference for a_1: $\theta = 1$ recovers a deterministic policy that places all mass on a_1. For the unregularised maximal lottery objective

$$\begin{aligned} J_{\text{ML}}(\theta) &= \min_{\theta'} \pi_\theta^\top K \pi_{\theta'}, \\ &= \min_{\theta'} \begin{bmatrix} \theta & 1 - \theta \end{bmatrix} \begin{bmatrix} 1 - \theta' \\ -\theta' \end{bmatrix}, \\ &= \min_{\theta'} (\theta(1 - \theta') - (1 - \theta)\theta'), \\ &= \min_{\theta'} (\theta - \theta'). \end{aligned}$$

This is minimised for $\theta' = 1$, hence

$$J_{\text{ML}}(\theta) = \theta - 1.$$

Maximising gives the maximal lotteries solution

$$\theta^\star = \arg\max_\theta J_{\text{ML}}(\theta) = 1,$$

which deterministically places all mass on a_1, and hence is Pareto efficient.

Now, for the regularised objective

$$\begin{aligned} J_{\text{reg}}(\theta) &= \theta - 1 - \tau \begin{bmatrix} \theta & 1 - \theta \end{bmatrix} \begin{bmatrix} \log \theta \\ \log(1 - \theta) \end{bmatrix}, \\ &= \theta - 1 - \tau \left(\theta \log \theta + \log(1 - \theta) - \theta \log(1 - \theta) \right). \end{aligned}$$

Differentiating with respect to θ and setting to zero

$$\begin{aligned}\frac{dJ_{\text{reg}}(\theta)}{d\theta} &= 1 - \tau\left(\log\theta + 1 - \frac{1}{1-\theta} - \log(1-\theta) + \frac{\theta}{1-\theta}\right), \\ &= 1 - \tau\log\left(\frac{\theta}{1-\theta}\right), \\ \implies \theta^\star &= \frac{1}{1+\exp\left(-\frac{1}{\tau}\right)}.\end{aligned}$$

Hence for $\tau \neq 0$, the solution is not Pareto efficient as $\theta^\star < 1$.

B.4 Auditing

Auditing by replay. The update is a fixed function of the shared seed and the votes: each E_k is regenerated from (ς, t, k), and each F_i is computed from the published votes. Anyone with both can recompute the update and compare it with the log. With a model judge, each vote can also be checked by re-running the judge on the logged comparison. Replay cannot catch a worker who consistently reports a strategic preference; only honest reporting, which we assume, rules that out. Replay needs the model state, prompts, decoding settings and seeds, and regenerating text exactly needs matching hardware: regenerating our evaluation baselines from their seeds on different GPUs reproduced the stored text on only 0–4 of 10 prompts per cell. Gradient training could log the same comparisons, so this audit is not unique to ES; what ES adds is that the votes are a worker's whole contribution.

C Voting game

C.1 Voter profiles for the dominated-alternative game

Table 6 gives the voter profiles of the two colour games, their pairwise margins, the winner under each rule, and each model's starting distribution.

Table 6: Voter profile for the colour game. Adding the Pareto-dominated alternative G leaves the pairwise margins between R and B, and hence the Condorcet winner and maximal lottery, unchanged, but flips the Borda winner from B to R, a violation of IIA for the Borda rule. Starting distributions are the policy's probabilities of each answer before training; ranges span runs. †The Qwen3-8B two-alternative GD runs instead start from the base model ($P(R) = .03$, $P(B) = .97$).

	Two alternatives $\{R, B\}$	Three alternatives $\{R, B, G\}$
Voters		
R-first voter, weight 0.4	$R \succ B$	$R \succ G \succ B$
B-first voter, weight 0.6	$B \succ R$	$B \succ R \succ G$
Pairwise majorities (margin)		
$P(B \succ R)$	0.6 (+0.2)	0.6 (+0.2)
$P(B \succ G)$	–	0.6 (+0.2)
$P(R \succ G)$	–	1.0 (+1.0)
Winners		
Condorcet winner (= maximal lottery)	B	B
Borda score R / B / G	0.4 / 0.6 / –	1.4 / 1.2 / 0.4
Borda winner	B	R
Plurality winner	B	B
Starting distribution $P(R)$ / $P(B)$ / $P(G)$		
Qwen3-0.6B	.706 / .294 / –	.309–.332 / .236–.259 / .426–.450
Qwen3-8B	.755 / .245 / –†	.309–.337 / .231–.254 / .419–.450
Llama-3.1-8B-Instruct	.202–.223 / .777–.798 / –	.309–.337 / .219–.241 / .432–.463

C.2 DOMINATED-ALTERNATIVE GAME DETAILS

Each comparison is judged by one voter, drawn with probability equal to its weight in Table 6; that voter's ranking decides the vote deterministically, so the expected vote $v(B \succ R)$ is 0.6 in both games. A sampled token that is not one of the candidates loses to every candidate. Answer probabilities are read as the softmax over the candidates' log-probabilities under the unperturbed policy (Appendix A.7). On the three-alternative game the two ES arms use different recipes ($\sigma = 10^{-3}$ for VerDICT and 3×10^{-3} for EGGROLL-DPO; Table 4).

C.3 SWITCHING THE STARTING DISTRIBUTION

In the two-alternative game the Qwen3 models start with most of their mass on the Condorcet loser R ($P(R)$ .71 at Qwen3-0.6B and .76 at Qwen3-8B; Table 6), so reaching B requires reversing the initial preference. VerDICT ends with final $P(B)$ 1.000, .995 and 1.000 at Qwen3-0.6B, Qwen3-8B and Llama-3.1-8B-Instruct, and EGGROLL-DPO with .746, .583 and .945 (seed means). Both methods therefore move mass towards B, which beats R in the pairwise comparison, but only VerDICT reaches it. The online GD controls also end on B (Appendix D.2).

D COMPARISON WITH DPO AND GRADIENT DESCENT

D.1 WHAT EGGROLL-DPO OPTIMISES

EGGROLL-DPO is not just DPO with a different optimiser. Each member samples its pair from its own perturbed policy, so ES optimises the expected DPO loss, including which pairs get sampled

$$J_{\mathrm{DPO}}(\theta) = \mathbb{E}_x \, \mathbb{E}_{(y^1,y^2)\sim\pi_\theta(\cdot|x)} \, \mathbb{E}_{\text{label}\sim v}\big[-\ell_{\mathrm{DPO}}(\theta; x, y_w, y_l)\big].$$

Its gradient has two parts: the usual DPO gradient on the sampled pairs, and a score-function term $\mathbb{E}[-\ell_{\mathrm{DPO}} \nabla_\theta \log \pi_\theta(y^1, y^2 \mid x)]$. Online DPO drops the second part because it treats sampled pairs as fixed data. That part changes where training ends.

We check this at Qwen3-0.6B by adding the score-function term to gradient-based online DPO, keeping everything else fixed (AdamW, learning rate 5×10^{-5}, $\beta = 0.1$, frozen initial reference, three seeds). It then ends close to EGGROLL-DPO on every game: paper .746 against .780 on rock–paper–scissors, with no scissors in either; $P(B)$ .704 against .746 in the two-alternative game; $P(R)$ .915 against .903 in the three-alternative game. The rock–paper–scissors and two-alternative gaps are larger than the seed spread, so the two share a target but do not match exactly. Without the term, GD Online DPO collapses to a single move on rock–paper–scissors and ends on the Condorcet winner in the voting games.

Which pairs are compared decides where gradient DPO ends. With pairs drawn from the frozen initial policy, it ends on the Borda winner of the three-alternative game ($P(R)$ .995). With its own samples, treated as fixed data, it ends on the Condorcet winner ($P(B)$ .997). So a reward fitted to a fixed distribution of pairs behaves like the Borda count, and the difference between VerDICT and EGGROLL-DPO on the games comes from the objective, not the optimiser.

D.2 GRADIENT DESCENT RESULTS

Figure 7 shows the results of optimising our three models with Gradient Descent instead of EGGROLL for the RPS task.

GD Online DPO collapses to a single move on every model and seed. GD Online IPO oscillates at Qwen3-0.6B and jumps between single moves at the larger models. Neither finds the uniform lottery. On the voting games, GD Online DPO ends on the Condorcet winner B.

We also train the VerDICT objective by gradient descent (GD Online ML): REINFORCE with a leave-one-out baseline on the same votes, the same n_b, AdamW at learning rate 5×10^{-5}, and a rank-8 LoRA policy with no reference model. It reaches near-uniform play on rock–paper–scissors (final exploitability .042 / .028 / .014 at Qwen3-0.6B / Qwen3-8B / Llama-3.1-8B-Instruct) and ends on B in 17 of 18 voting-game runs. The exception is one Qwen3-0.6B seed of the three-alternative game, which ends on R.

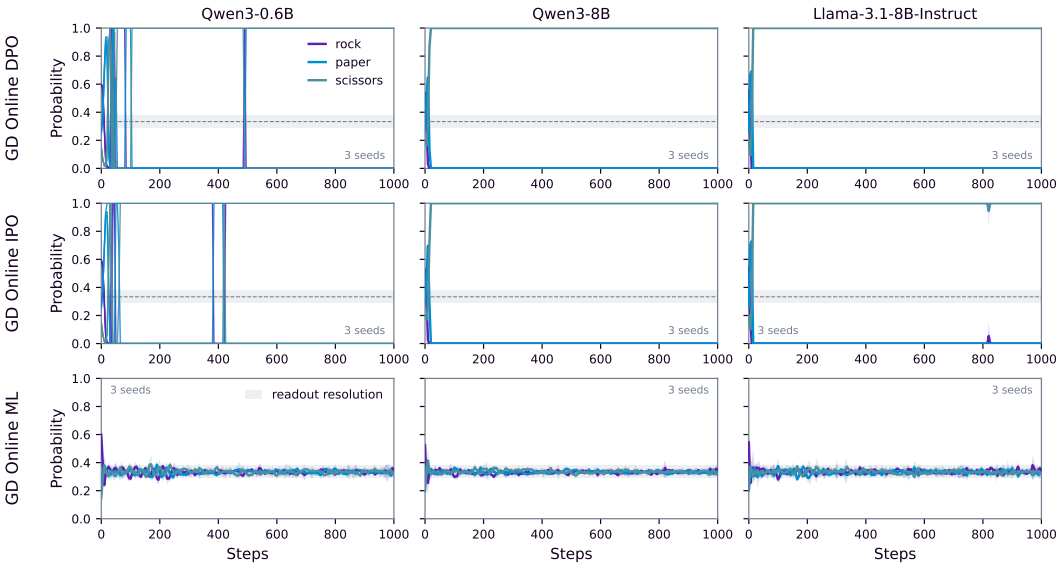

Figure 7: Move probabilities on rock–paper–scissors under GD Online DPO and GD Online IPO (TRL) for Qwen3-0.6B, Qwen3-8B and Llama-3.1-8B-Instruct, three seeds each.

Figure 8 shows the gradient baselines on the text tasks, with the same prompts and judges as the ES runs and 16 comparisons per step on the preference tasks (4 on TL;DR). GD Online DPO has a higher mean than VerDICT on TL;DR at every scale; the gap is clear at Qwen3-8B and Llama and within the interval at Qwen3-0.6B. On the human preference datasets the differences are small and mostly within the intervals, except CoVal and MultiPref at Qwen3-0.6B, where VerDICT is higher (Table 3).

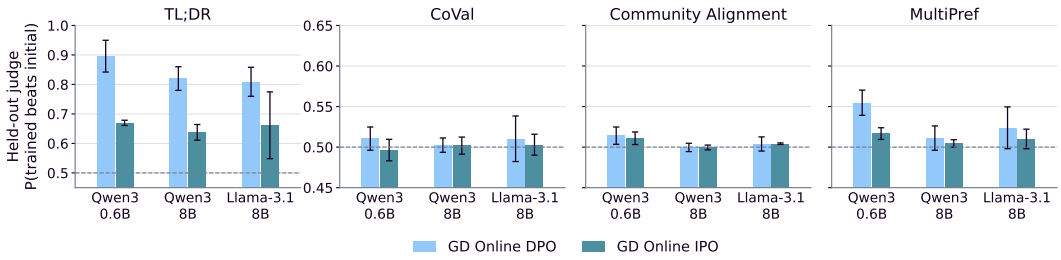

Figure 8: Held-out judged win rate against the initial policy for GD Online DPO and GD Online IPO (TRL) on the text tasks, for Qwen3-0.6B, Qwen3-8B and Llama-3.1-8B-Instruct.

E Social choice analysis

E.1 Social choice properties of maximal lotteries and the Borda count

In Table 7 we compare the social choice properties of objectives that target the maximal lottery, as VerDICT does, with the Borda count, which a Bradley–Terry reward recovers when pairs are sampled uniformly and scores are unrestricted (Siththaranjan et al., 2023) (Appendix D.1). GD Online DPO trained on its own samples is not an example of the latter: it ends on the Condorcet winner.

E.2 Social choice metrics for human preference datasets

In Table 8 we show the social-choice-theoretic structure of the human preference datasets used in our paper.

Table 7: Maximal lotteries trade a single deterministic winner for majority and robustness guarantees. Social choice properties of maximal lotteries (ML, the target of VerDICT) and the Borda count, the ranking a Bradley–Terry reward recovers when pairs are sampled uniformly and scores are unrestricted (Siththaranjan et al., 2023). Properties are of the ideal social choice rules; the policies VerDICT trains approximate the maximal lottery and are not guaranteed to inherit these properties exactly (Appendix B.1).

Property	Maximal Lottery	Borda count
Basic fairness		
Anonymity and neutrality	✓	✓
Pareto efficiency	✓	✓
Respecting majorities		
Majority criterion (a majority's top choice wins)	✓	✗
Condorcet consistency (the pairwise-majority winner wins)	✓	✗
Condorcet loser never chosen	✓	✓
Unbeatable: no alternative is majority-preferred to the output[a]	✓	✗
Robustness to the menu and the electorate		
Independence of clones (exact clone sets)	✓	✗
Composition consistency	✓	✗
Population consistency (reinforcement)	✓	✓
Participation (no incentive to abstain)[b]	✓	✓
Majority cycles ($a \succ b \succ c \succ a$)		
Well defined for every preference profile	✓	✓
Output reflects the cycle (no arbitrary single winner)	✓	✗
Practical		
Computable from pairwise comparisons alone	✓	✓
Deterministic output / one transitive ranking (scalar reward)	✗	✓

[a] Holds in expectation. This is the defining property of a maximal lottery.
[b] For ML this holds in the probabilistic sense (Brandl et al., 2019).

Table 8: Social-choice-theoretic structure of the preference datasets. CW denotes the Condorcet winner. Percentages are of the items analysed unless stated otherwise.

Property	CoVal	Community Alignment	MultiPref
Items analysed	1,073 prompts	602 (prompt, candidate set) items	8,482 comparisons (4,541 prompts)
Annotations per prompt	16.7	30.6	3.0 strict (of 4)
Candidate pairs with a strict majority (tied pairs)	6,187 (251)	3,390 (222)	7,226 (1,256)
Cyclic triangles	14 of 4,292 (0.33%)	0 of 2,408	n/a (no triples)
Items containing a majority cycle	14 (1.3%)	0	n/a (pairs only)
Items with a Condorcet winner	997 (92.9%)	580 (96.3%)	7,226 (85.2%)
CW $\neq$ Copeland winner	0	0	0
CW $\neq$ Borda winner	31 (3.1% of CW)	0	0
CW $\neq$ plurality winner	57 (5.7% of CW)	0	0
Borda winner $\neq$ plurality winner	119 (11.1%)	0	0
Pairwise inter-annotator majority agreement	0.769	0.759	0.814

E.3 Agreement with Condorcet and Borda winners

In Table 9 we show the agreement between each policy's preferred candidate and the Condorcet and Borda winners on CoVal. Where the two winners differ (31 and 28 prompts), VerDICT picks the Condorcet winner on one fewer prompt than the initial policy at each scale; these subsets are too small to show a change in either direction.

Table 9: Agreement between each policy's preferred candidate and the Condorcet and Borda winners on CoVal. The first two columns report agreement on the held-out / train splits. The final column restricts to prompts where the Condorcet and Borda winners differ and reports the fraction on which the policy picks the Condorcet / Borda winner ($n = 31$ for 0.6B; $n = 28$, train split, for 8B).

Model	Policy	$=$ Condorcet (test / train)	$=$ Borda (test / train)	Condorcet $\neq$ Borda: picks Condorcet / Borda
0.6B	Initial	.629 / .637	.617 / .630	.258 / .516
	VerDICT	.634 / .641	.622 / .635	.226 / .484
	EGGROLL-DPO	.629 / .633	.622 / .629	.226 / .516
8B	Initial	.691 / .680	.663 / .671	.286 / .429
	VerDICT	.691 / .680	.658 / .671	.250 / .429
	EGGROLL-DPO	.680 / .679	.653 / .671	.250 / .429

E.4 Exploitability on CoVal

A maximal lottery is unbeaten in expectation by every answer, so we test this property directly on CoVal at Qwen3-0.6B. For each held-out prompt we pool the four annotator candidates with eight samples from each of the four policies, and measure how much the best answer in a set beats each policy's sample mixture under the held-out judge (Table 10). We lead with the *full pool*, the same 36 answers for every policy. Because it contains the evaluated policy's own samples, we use the *split-sample* estimate, which selects the worst answer on four of the policy's eight samples and scores it on the other four; selecting and scoring on the same samples would make the full-pool value ≥ 0 by construction. This is a split-sample diagnostic rather than an independent evaluation: the four scoring samples stay in the pool as potential opponents, so some dependence between selection and scoring remains. The *leave-own-out* pool removes the evaluated policy's own samples, so its 28 answers differ between policies.

On the full pool VerDICT is the least exploitable policy on every seed: over three seeds the split-sample estimate is 0.085 ± 0.009 for VerDICT, against 0.129 for EGGROLL-DPO, 0.138 for GD Online DPO and 0.155 for the initial policy (Table 11). The leave-own-out pool and the in-sample estimates give the same ordering, and on seed 0 every paired difference against VerDICT on the pool scopes excludes zero. Against the four annotator candidates alone every trained policy improves on the initial policy, but VerDICT and GD Online DPO are not separated. Restricting the opponents to a 36-answer pool makes the population quantity a lower bound on the unrestricted exploitability; the entries are finite-sample estimates of that quantity. With eight samples per policy, the share of prompts with exploitability ≤ 0 is not a calibrated test of closeness to a maximal lottery, since even an exact maximal lottery can fail it through sampling, so we use these estimates only to compare policies.

These results come from one model and one dataset, and depend on the judge's calibration. VerDICT's answers are also the longest (371 tokens on average, 45% at the cap, against 280 tokens and 21% for the initial policy), so the judge's preference for longer answers may contribute to its advantage.

F Additional results

F.1 Results tables

Figure 9 plots the VerDICT and EGGROLL-DPO cells of Table 3.

F.2 Distance from initialisation and response length

Table 12 reports how far each final policy moves from its initialisation, measured by the mean log-ratio between the trained and initial policies on each policy's own samples (an estimate of KL only at $T = 1$), together with the length of its responses and the share that reach the generation cap.

Table 10: Empirical restricted-pool exploitability of the four final policies on CoVal (Qwen3-0.6B, seed-0 checkpoints of Table 3; three-seed means in Table 11). For each of the 193 held-out prompts the pool holds the four annotator candidates and eight samples from each policy (36 answers). For a policy π with samples S_π, $w_\pi(y)$ is the mean held-out-judge probability that a sample of π beats the answer y (self-pair excluded), and the exploitability on an opponent set Y is $\max_{y \in Y}(\frac{1}{2} - w_\pi(y))$; restricting Y to the pool makes the population quantity a lower bound on the unrestricted exploitability, and the entries are finite-sample estimates of it; a maximal lottery has exploitability ≤ 0 against every answer. *Full pool*: the same 36 answers for every policy. *Leave-own-out*: the 28 answers that are not π's own samples, which differ between policies. *Split-sample*: the worst answer is selected on four of π's eight samples and scored on the other four (both assignments, averaged); *in-sample*: selected and scored on the same eight. Entries are means over prompts $\pm$ 95% t-intervals. Second block: split-sample paired per-prompt differences. Third block: mixture-versus-mixture win rate in %, so a cell and its mirror sum to 100. Bold: first block, the best policy when its paired difference against each other policy excludes zero; second and third blocks, the interval excludes zero (50). Held-out judge, $T = 0.8$, cap 512 tokens; mean answer length 280, 371, 322 and 277 tokens, with 21%, 45%, 31% and 21% of samples at the cap (column order).

Opponent set Y	Estimate	initial policy	VerDICT	EGGROLL-DPO	GD Online DPO
Empirical restricted-pool exploitability $\max_{y \in Y}(\frac{1}{2} - w_\pi(y))$					
Full pool (36)	split-sample	.156 ± .014	**.083 ± .009**	.127 ± .013	.137 ± .014
	in-sample	.168 ± .014	**.104 ± .010**	.140 ± .013	.153 ± .014
Leave-own-out (28)	split-sample	.156 ± .015	**.074 ± .009**	.129 ± .013	.136 ± .014
	in-sample	.162 ± .014	**.080 ± .009**	.133 ± .013	.142 ± .014
Candidates (4)	split-sample	−.075 ± .022	−.110 ± .019	−.092 ± .021	−.106 ± .022
	in-sample	−.075 ± .022	−.109 ± .019	−.091 ± .021	−.105 ± .022
Other policies (24)	split-sample	.153 ± .014	**.070 ± .008**	.126 ± .013	.134 ± .014
	in-sample	.159 ± .014	**.075 ± .007**	.130 ± .012	.140 ± .013
Undominated: share of prompts with exploitability ≤ 0					
Leave-own-out (28)	split-sample	.00	.05 ± .03	.00	.02 ± .02
	in-sample	.00	**.02 ± .02**	.00	.00
Candidates (4)	in-sample	.70 ± .06	.83 ± .05	.77 ± .06	.78 ± .06
Other policies (24)	in-sample	.00	**.03 ± .02**	.00	.00

Paired differences, split-sample: Δ exploitability on the opponent set Y

	leave-own-out (28)	pool (36)	candidates (4)	other policies (24)
VerDICT − initial policy	**−.083 ± .014**	**−.072 ± .012**	**−.035 ± .013**	**−.084 ± .014**
EGGROLL-DPO − initial policy	**−.028 ± .008**	**−.029 ± .008**	**−.017 ± .009**	**−.028 ± .008**
GD Online DPO − initial policy	**−.020 ± .008**	**−.018 ± .007**	**−.031 ± .008**	**−.019 ± .008**
VerDICT − EGGROLL-DPO	**−.055 ± .013**	**−.044 ± .009**	**−.018 ± .009**	**−.056 ± .013**
VerDICT − GD Online DPO	**−.063 ± .013**	**−.054 ± .011**	−.004 ± .011	**−.065 ± .014**
GD Online DPO − EGGROLL-DPO	+.008 ± .008	**+.010 ± .008**	**−.014 ± .009**	**+.009 ± .009**

	P(row's samples $\succ$ column's samples), %			
	initial policy	VerDICT	EGGROLL-DPO	GD Online DPO
initial policy	–	**44.9 ± 1.0**	**47.7 ± 0.7**	**47.7 ± 0.7**
VerDICT	**55.1 ± 1.0**	–	**53.0 ± 0.7**	**53.6 ± 0.8**
EGGROLL-DPO	**52.3 ± 0.7**	**47.0 ± 0.7**	–	50.3 ± 0.7
GD Online DPO	**52.3 ± 0.7**	**46.4 ± 0.8**	49.7 ± 0.7	–

On TL;DR, VerDICT has a larger mean log-ratio than EGGROLL-DPO at every scale (10.2 against 2.3 nats at Qwen3-0.6B, 1.0 against 0.5 at Qwen3-8B, 6.0 against 2.4 at Llama-3.1-8B-Instruct). At Qwen3-0.6B the two produce summaries of similar length (47.7 and 49.8 tokens, with 35% and 39% at the cap), and VerDICT's much higher win rate there persists under the length adjustment of Appendix F.3. At Llama-3.1-8B-Instruct VerDICT's summaries are longer than EGGROLL-DPO's (51.8 against 45.4 tokens, 30% against 15% at the cap), so length may contribute to its lead at that scale. GD Online DPO has the largest log-ratio at every scale; at the two Qwen models it also produces the longest summaries, most often at the cap (73% and 42%, against 10% and 4% for the

Table 11: Empirical restricted-pool exploitability on CoVal (Qwen3-0.6B), mean over seeds 0, 1 and 2 $\pm$ 95% t-interval over seeds; each seed's value is its mean over the 193 prompts. Columns as in Table 10.

	leave-own-out (28) split-sample	leave-own-out (28) in-sample	pool (36) split-sample	candidates (4) split-sample	other policies (24) split-sample
initial policy	.156 $\pm$ .010	.161 $\pm$ .010	.155 $\pm$ .008	$-$.075 $\pm$ .000	.152 $\pm$ .011
VerDICT	.076 $\pm$ .008	.081 $\pm$ .006	.085 $\pm$ .009	$-$.114 $\pm$ .012	.070 $\pm$ .007
EGGROLL-DPO	.130 $\pm$ .015	.135 $\pm$ .015	.129 $\pm$ .016	$-$.088 $\pm$ .010	.127 $\pm$ .016
GD Online DPO	.135 $\pm$ .012	.141 $\pm$ .010	.138 $\pm$ .003	$-$.106 $\pm$ .009	.133 $\pm$ .012

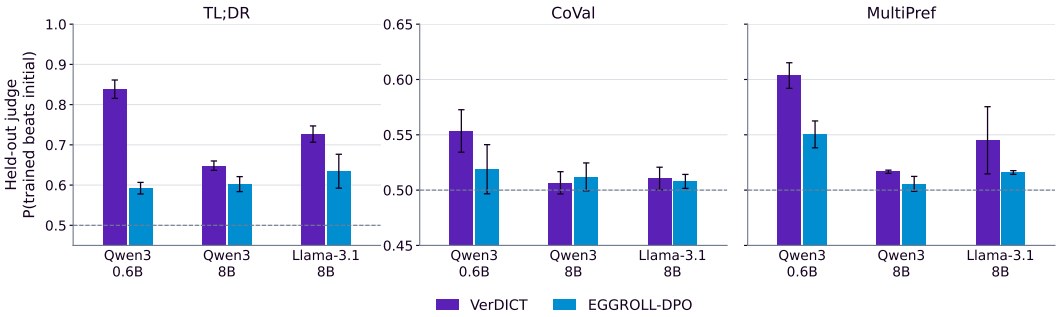

Figure 9: Held-out judged win rate against the initial policy on TL;DR summarisation and two human preference datasets (CoVal, MultiPref), for Qwen3-0.6B, Qwen3-8B and Llama-3.1-8B-Instruct.

initial policy), whereas at Llama-3.1-8B-Instruct its summaries are no longer than VerDICT's. GD Online IPO barely changes (at most 2.6 nats) and keeps the initial policy's length.

On the preference tasks, measured on earlier single-seed checkpoints, every method stays within 6 nats of its initialisation. VerDICT has the largest log-ratio on CoVal and MultiPref, and on MultiPref its answers are the longest (229.8 tokens, 77% at the cap, against 199.7 and 54% for the initial policy).

F.3 Length-controlled win rates

Tables 13 and 14 correct each cell's win rate for the difference in response length from the initial policy, using the length term of the AlpacaEval length-controlled regression (Dubois et al., 2025), fitted per cell and evaluated at zero length difference. For each cell we fit $\operatorname{logit} p_i = a + bz_i$ by fractional logistic regression to the soft judge probability p_i of prompt i, where z_i is the standardised difference in response length (tokens, trained minus initial), and report $s(a + bz_0)$, with z_0 the standardised value of a zero raw difference; the bootstrap resamples prompts, keeping all seeds of a prompt together. This is a sensitivity analysis rather than a causal decomposition of the gains. On TL;DR at Qwen3-0.6B, VerDICT keeps most of its win rate (82.0% after the control, 85.4% before) and stays well ahead of EGGROLL-DPO (59.5%) and GD Online IPO (69.2%), below GD Online DPO (87.3%). At the 8B models every method's estimated TL;DR gain decreases substantially after the adjustment: VerDICT falls to 54.3% and 57.9% at Qwen3-8B and Llama-3.1-8B-Instruct, below GD Online IPO (56.1% and 58.4%). On the preference tasks at Qwen3-0.6B, VerDICT's MultiPref lead survives (54.5% against at most 51.7%), but its CoVal advantage over GD Online DPO does not (52.9% against 53.1%): VerDICT's CoVal answers are on average 92 tokens longer than the initial policy's, while GD Online DPO's are unchanged. At the 8B models every length-controlled preference win rate lies within 1.8 points of 50%, except VerDICT on MultiPref at Llama-3.1-8B-Instruct (53.3%).

F.4 Compute comparison tables

Table 15 breaks down the LLM passes and measured time per preference comparison for each method, and Table 16 gives measured compute for all 48 timed cells. Figure 10 compares the

Table 12: Distance of the final policies from their initialisation. For each held-out prompt, $m = 4$ samples per policy are drawn at $T = 0.8$ with shared seeds and teacher-forced under the trained and initial models. *Log-ratio* is the summed log-ratio $\log \pi_{\text{trained}}(y \mid x) - \log \pi_{\text{init}}(y \mid x)$ over the response tokens, averaged over prompts and samples (nats); since samples are drawn at $T = 0.8$, it equals $\mathrm{KL}(\pi_{\text{trained}} \,\|\, \pi_{\text{init}})$ only at $T = 1$ and can be negative. *Per token* is the per-token mean, *length* the mean number of sampled tokens and *at cap* the share of samples stopped by the generation cap. Three-seed rows give the seed mean with the per-seed range in small type; single-checkpoint rows give the mean ± a 95% CI over prompts. Greyed rows are the initial policy, as the length and cap reference. TL;DR rows use the checkpoints of Table 3, except the Qwen3-0.6B online-GD rows, which are sibling runs trained on the unshuffled data slice. Preference-task rows are earlier single-seed checkpoints with a 256-token cap, not those of Table 3, except the MultiPref online-GD rows, which are seed 0 of its locked cells; Qwen3-8B and Llama preference rows are omitted because the weights of those earlier runs were not retained.

	log-ratio (nats)	per token	length (tokens)	at cap (%)
TL;DR (64-token cap; $n = 500$ prompts; three seeds per trained method)				
Qwen3-0.6B				
initial policy	0.0 ± 0.0	0.000 ± 0.000	33.8 ± 0.9	10 ± 1
VerDICT	10.2 (9.6–11.0)	0.215 (0.208–0.227)	47.7 (46.4–49.0)	35 (32–38)
EGGROLL-DPO	2.3 (2.2–2.5)	0.040 (0.038–0.044)	49.8 (48.7–50.6)	39 (35–43)
GD Online DPO	18.9 (14.0–24.4)	0.328 (0.283–0.409)	58.1 (50.8–63.4)	73 (46–96)
GD Online IPO	2.6 (2.5–2.7)	0.082 (0.078–0.085)	34.6 (33.6–35.7)	9 (7–10)
Qwen3-8B				
initial policy	0.0 ± 0.0	0.000 ± 0.000	31.8 ± 0.9	4 ± 1
VerDICT	1.0 (0.9–1.2)	0.018 (0.016–0.023)	41.6 (41.5–42.0)	15 (15–15)
EGGROLL-DPO	0.5 (0.5–0.6)	0.009 (0.007–0.010)	39.0 (38.8–39.2)	12 (11–12)
GD Online DPO	10.3 (6.6–12.7)	0.186 (0.130–0.216)	54.6 (50.1–59.7)	42 (28–66)
GD Online IPO	0.8 (0.8–0.9)	0.018 (0.016–0.020)	38.4 (37.2–39.3)	9 (7–11)
Llama-3.1-8B-Instruct				
initial policy	0.0 ± 0.0	0.000 ± 0.000	34.3 ± 0.8	3 ± 1
VerDICT	6.0 (5.7–6.5)	0.112 (0.105–0.120)	51.8 (50.8–52.4)	30 (27–32)
EGGROLL-DPO	2.4 (2.0–2.7)	0.048 (0.044–0.054)	45.4 (43.3–47.4)	15 (10–19)
GD Online DPO	11.0 (8.5–13.4)	0.216 (0.175–0.251)	50.9 (48.5–53.2)	18 (11–25)
GD Online IPO	2.4 (1.6–3.1)	0.054 (0.041–0.067)	42.7 (37.8–45.5)	10 (5–13)
Preference tasks, Qwen3-0.6B (256-token cap; one checkpoint per method)				
CoVal ($n = 193$)				
initial policy	0.0 ± 0.0	0.000 ± 0.000	191.2 ± 10.3	51 ± 6
VerDICT	4.6 ± 0.3	0.023 ± 0.002	201.6 ± 9.9	57 ± 6
EGGROLL-DPO	1.9 ± 0.2	0.009 ± 0.001	202.3 ± 9.7	58 ± 6
GD Online DPO	2.7 ± 0.2	0.013 ± 0.001	199.5 ± 9.8	54 ± 6
GD Online IPO	1.1 ± 0.1	0.006 ± 0.001	195.9 ± 10.0	53 ± 6
Community Alignment ($n = 203$)				
initial policy	0.0 ± 0.0	0.000 ± 0.000	233.5 ± 6.6	76 ± 5
VerDICT	3.6 ± 0.2	0.015 ± 0.001	237.3 ± 6.1	81 ± 5
EGGROLL-DPO	2.1 ± 0.2	0.009 ± 0.001	236.2 ± 6.1	78 ± 5
GD Online DPO	4.2 ± 0.2	0.018 ± 0.001	239.9 ± 5.6	82 ± 5
GD Online IPO	0.9 ± 0.1	0.004 ± 0.000	234.7 ± 6.4	75 ± 5
MultiPref ($n = 500$)				
initial policy	0.0 ± 0.0	0.000 ± 0.000	199.7 ± 6.0	54 ± 4
VerDICT	5.9 ± 0.2	0.026 ± 0.001	229.8 ± 4.7	77 ± 3
EGGROLL-DPO	4.8 ± 0.2	0.021 ± 0.001	217.2 ± 5.3	66 ± 4
GD Online DPO	4.5 ± 0.2	0.020 ± 0.001	222.3 ± 5.1	70 ± 3
GD Online IPO	1.0 ± 0.1	0.005 ± 0.000	205.3 ± 5.8	56 ± 4

time per training step of VerDICT and EGGROLL-DPO on matched hardware. All three are timed on our earlier $\sigma = 10^{-3}$ campaign, which used the same population and batch but a 256-token cap on the preference tasks; σ does not enter the step cost, but the cap does, so the 8B preference-task

Table 13: Length-controlled win rate against the initial policy. For each held-out prompt the held-out judge scores all 4×4 pairings of the policy's and the initial policy's samples; their mean is the prompt's judged win probability. *win* is its mean over prompts (%), *length* the mean response length in tokens, Δ *length* the mean difference to the initial policy's response on the same prompt, r the correlation over prompts between the win probability and that difference, and *LC win* the length-controlled win rate: a logistic regression of the prompt-level win probability on the standardised length difference, evaluated at zero difference (Dubois et al., 2025), with a percentile bootstrap over prompts ($B = 2000$) in brackets. Rows are the checkpoints of Table 3 at the locked generation caps: on TL;DR the three seeds of every method (pooled), on the preference tasks one locked seed per method. The Qwen3-8B and Llama preference cells are in Table 14.

	win (%)	length	Δ length	r	LC win (%)
TL;DR (64-token cap; three seeds per method, pooled)					
Qwen3-0.6B ($n = 500$; initial policy 33.9 tokens)					
VerDICT	85.4	47.5	+13.6	0.24	82.0 [80.5, 83.5]
EGGROLL-DPO	61.2	49.9	+16.0	0.07	59.5 [57.5, 61.3]
GD Online DPO	90.8	56.2	+22.3	0.20	87.3 [86.0, 88.6]
GD Online IPO	69.2	34.2	+0.3	0.16	69.2 [68.1, 70.3]
Qwen3-8B ($n = 500$; initial policy 31.8 tokens)					
VerDICT	63.3	41.6	+9.8	0.45	54.3 [52.7, 55.8]
EGGROLL-DPO	59.7	39.0	+7.1	0.48	52.7 [51.5, 53.8]
GD Online DPO	82.5	54.6	+22.7	0.44	66.3 [63.4, 69.0]
GD Online IPO	62.2	38.4	+6.6	0.45	56.1 [54.9, 57.2]
Llama-3.1-8B-Instruct ($n = 500$; initial policy 34.3 tokens)					
VerDICT	72.3	51.8	+17.4	0.40	57.9 [55.3, 60.4]
EGGROLL-DPO	63.4	45.4	+11.0	0.40	54.5 [52.9, 56.2]
GD Online DPO	80.7	50.9	+16.5	0.33	70.5 [68.2, 72.7]
GD Online IPO	65.7	42.7	+8.4	0.46	58.4 [56.9, 59.7]
Preference tasks, Qwen3-0.6B (one locked seed per method)					
CoVal (512-token cap; $n = 193$; initial policy 280.6 tokens)					
VerDICT	54.9	372.3	+91.7	0.26	52.9 [51.6, 54.3]
EGGROLL-DPO	53.3	323.2	+42.6	0.24	52.2 [51.3, 53.1]
GD Online DPO	53.0	279.0	−1.5	0.17	53.1 [52.2, 53.9]
GD Online IPO	51.3	280.8	+0.2	0.13	51.3 [50.7, 52.0]
Community Alignment (640-token cap; $n = 203$; initial policy 407.1 tokens)					
VerDICT	51.9	476.6	+69.5	0.10	51.4 [50.6, 52.3]
EGGROLL-DPO	50.1	457.2	+50.1	0.05	49.9 [49.3, 50.5]
GD Online DPO	51.7	390.5	−16.5	0.19	51.9 [51.2, 52.6]
GD Online IPO	50.5	416.8	+9.7	0.01	50.5 [50.0, 50.9]
MultiPref (256-token cap; $n = 500$; initial policy 199.7 tokens)					
VerDICT	59.5	244.2	+44.5	0.75	54.5 [53.8, 55.1]
EGGROLL-DPO	54.7	224.3	+24.6	0.79	51.2 [50.7, 51.6]
GD Online DPO	55.0	221.5	+21.8	0.81	51.7 [51.4, 52.1]
GD Online IPO	51.8	205.3	+5.6	0.76	50.9 [50.6, 51.2]

steps of Table 3 are slower than those shown. In the 24 pairs of runs of Table 3 that happen to share a card type and count, VerDICT is 1.21–2.01× faster per step than EGGROLL-DPO.

G SAMPLED COMPLETIONS

We show one held-out prompt for 8 of the 12 (task, model) pairs, with the completion sampled from the initial policy and from each trained policy (seed 0). The prompt is the first of the ten stored held-out prompts that passes a content screen and whose completions fit in 1,800 characters; it is chosen without reading the completions. p is the held-out judge's probability that the trained completion beats the initial one. Both policies are sampled with the same seed, so identical completions get $p = 0.5$. Sampling is not bit-reproducible across evaluation jobs, so when a method's evaluation stored a different initial-policy sample, p was computed against that sample, and it is printed at the

Table 14: Length-controlled win rate against the initial policy on the Qwen3-8B and Llama-3.1-8B-Instruct preference tasks, columns as in Table 13; one locked seed per method. †: no locked seed 0 exists for this cell and seed 1 is measured instead.

	win (%)	length	Δ length	r	LC win (%)
Preference tasks, Qwen3-8B (one locked seed per method)					
CoVal (896-token cap; $n = 193$; initial policy 517.0 tokens)					
VerDICT	51.0	543.4	+26.4	0.23	50.7 [50.2, 51.3]
EGGROLL-DPO†	50.3	529.6	+12.7	0.10	50.2 [49.8, 50.7]
GD Online DPO	50.8	510.7	−6.3	0.02	50.8 [50.3, 51.3]
GD Online IPO	50.6	521.3	+4.4	0.06	50.6 [50.2, 51.0]
Community Alignment (1152-token cap; $n = 203$; initial policy 795.8 tokens)					
VerDICT†	50.0	797.5	+1.7	−0.08	50.0 [49.8, 50.3]
EGGROLL-DPO	49.9	803.7	+7.8	−0.04	49.9 [49.7, 50.0]
GD Online DPO	50.0	786.7	−9.2	0.01	50.0 [49.8, 50.2]
GD Online IPO	49.9	789.8	−6.1	−0.16	49.9 [49.8, 50.0]
MultiPref (640-token cap; $n = 500$; initial policy 495.9 tokens)					
VerDICT	51.6	523.0	+27.1	0.64	50.3 [50.0, 50.6]
EGGROLL-DPO	50.3	499.7	+3.8	0.59	50.1 [49.9, 50.3]
GD Online DPO	51.4	515.2	+19.3	0.62	50.4 [50.2, 50.6]
GD Online IPO†	50.9	506.1	+10.2	0.62	50.4 [50.2, 50.6]
Preference tasks, Llama-3.1-8B-Instruct (one locked seed per method)					
CoVal (640-token cap; $n = 193$; initial policy 319.8 tokens)					
VerDICT	51.8	357.7	+38.0	0.20	51.3 [50.6, 52.0]
EGGROLL-DPO†	50.5	344.7	+25.0	0.32	50.1 [49.5, 50.6]
GD Online DPO	51.1	338.1	+18.3	0.36	50.7 [50.2, 51.3]
GD Online IPO	50.7	325.7	+5.9	0.14	50.6 [50.2, 51.1]
Community Alignment (640-token cap; $n = 203$; initial policy 493.6 tokens)					
VerDICT	50.5	509.8	+16.2	0.08	50.5 [50.2, 50.7]
EGGROLL-DPO	50.2	506.6	+13.0	0.09	50.1 [49.8, 50.4]
GD Online DPO	50.2	494.1	+0.5	−0.05	50.2 [50.0, 50.4]
GD Online IPO	50.0	491.7	−1.9	−0.14	50.0 [49.8, 50.2]
MultiPref (384-token cap; $n = 500$; initial policy 293.3 tokens)					
VerDICT	55.2	328.1	+34.8	0.58	53.3 [52.9, 53.7]
EGGROLL-DPO	51.8	316.8	+23.6	0.67	50.5 [50.2, 50.8]
GD Online DPO	52.8	315.4	+22.1	0.56	51.8 [51.4, 52.1]
GD Online IPO	51.1	301.3	+8.1	0.56	50.7 [50.4, 50.9]

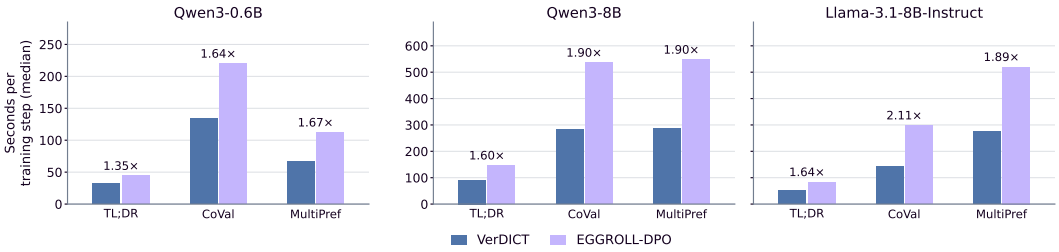

Figure 10: Median seconds per training step for VerDICT and EGGROLL-DPO (lower is better), each pair run on matched hardware. Labels show the EGGROLL-DPO/VerDICT ratio: VerDICT is 1.35–2.11× faster across all settings. These are runs of our earlier $\sigma = 10^{-3}$ campaign (one seed each, 256-token cap on the preference tasks), not the checkpoints of Table 3; hardware-matched pairs among the runs of Table 3 give 1.21–2.01×. Ratios are of median seconds per step and can differ slightly from the per-run values in Table 16.

end of the block. Completions that end mid-sentence hit the generation cap, and emoji are shown as [emoji]. GD Online IPO samples exist for Qwen3-0.6B only, and the CoVal Llama-3.1-8B-Instruct block has no EGGROLL-DPO completion because none was stored when this appendix was built.

Table 15: LLM passes per preference comparison (left; passes are counted per operation and are not equal in cost) and measured cost on Qwen3-8B, CoVal, on one A100 (right; one earlier $\sigma = 10^{-3}$ run per method at a 256-token cap, since the runs of Table 3 for this cell ran on different cards), the cell of Figure 2. N: population size; E: number of engines, each of which samples one shared opponent response. Online GD processes $128\times$ fewer comparisons per step in this cell, so s/step is not a speed comparison between ES and online GD.

	LLM passes per preference comparison						Qwen3-8B, CoVal, one A100-SXM4-80GB		
Method	samples	policy fwd (grad)	ref. fwd	backward	total	Resident on the GPU	s / step	comp. / step	gen / ref / judge (s)
GD Online DPO	2	2	2	2	8	policy, AdamW state, activations	165.5	16	–
EGGROLL-DPO	2	0	2	0	4	policy, frozen reference copy	536.0	2,048	289.6 / 128.0 / 88.3
VerDICT	$1 + E/N$	0	0	0	≈ 1 (1.008)	policy only	282.5	2,048	163.6 / 0.0 / 86.9

Across all 370 stored trained completions, none is empty, and repeated text is as rare as in the initial policy (12 of 370 each).

TL;DR, Qwen3-0.6B (the first of the ten stored held-out prompts, index 0).

Prompt.
SUBREDDIT: r/relationships
TITLE: My [21/M] girlfriend [19/F] broke up with me after she went through my Facebook without my permission.
POST: My girlfriend and I had been dating for 15 months.

Last week my girlfriend went onto my Facebook account and read through my message history with a couple of girls.

She was **searching for a specific girl that I used to flirt with in the past, and she found it.**

We had fought one time before about me flirting with this girl, and I stopped talking to her entirely for a couple of months (obviously she didn't believe I did).

She found messages between the girl and I around my birthday in February, and her (message girl) birthday in June. Needless to say they were flirty but with no intentions of ever acting upon them. The girl lives in Europe and I live on the East Coast. But my girlfriend doesn't believe that I ever stopped talking to her, and that I was flirty throughout our entire relationship.

I have no evidence to disprove this, except for the fact that I don't have her on social media anymore (excluding Facebook, which I now deleted)

Yes I know it was stupid for me to flirt in the first place, but I can't help but feel like there is a massive invasion of privacy and that she shouldn't have seen the messages in the first place.
TL;DR:

Initial policy (the sample stored by the VerDICT evaluation).
I was dating someone I knew for 15 months, and my girlfriend found a message about a past flirtation on my Facebook without my permission. She doesn't believe that I ever stopped talking to her. I don't have her on my social media and can't prove it was a lie.

VerDICT ($p = 0.137$).
My girlfriend found messages I had sent to another girl, and she believed that I was flirty with her around the time of our relationship, but she didn't see the messages. I don't have evidence to disprove her belief, and I feel that she shouldn't have seen the messages.

EGGROLL-DPO ($p = 0.016$).
I was dating someone I knew for 15 months. She went through my Facebook account and saw messages with her on the same birthday, but she didn't know that I left her. She felt like she's being stalked. I don't want her to feel like she was stalked, but I can't

GD Online DPO ($p = 0.047$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 257 characters and is printed at the end of this block).
My girlfriend found messages I sent during our 15-month relationship on Facebook, which she never saw. She believed I was flirty throughout everything, but she didn't know that I stopped talking to her.

Table 16: Measured compute for the 48 one-seed runs of our earlier $\sigma = 10^{-3}$ campaign (256-token cap on the preference tasks, 8 prompts per online-GD step on TL;DR). Two rows, the Qwen3-0.6B MultiPref online-GD seed-0 runs, are also runs of Table 3; the other runs of Table 3 use $\sigma = 3\times10^{-3}$ and longer caps and are not timed here. σ does not enter the step cost; the cap does. A100 factors are measured per workload: for the ES rows from five same-recipe pairs of vLLM runs, one on an A100 and one on the card in question, for the online-GD rows from TRL runs, since inference-bound population sampling gains more on the newer cards than a TRL step with backward passes. †No A100 timing exists for these eight online-GD rows, which use the ES factor of their card; this likely overstates their cost by about 10%. The A100 factor converts each GPU's time to A100-equivalent time. VerDICT uses 0.47–0.74 of the A100-hours of EGGROLL-DPO in every cell, as it needs no reference forward pass. Online GD processes 64–256× fewer comparisons per step than the ES methods, so these figures do not compare the speed of ES and online GD.

Task	Model	Method	GPU(s)	s / step	Comp. / step	A100 factor	A100-s / step	Steps	A100-h
TL;DR	Qwen3-0.6B	VerDICT	2× RTX PRO 4500 (32 GiB)	32.6	1,024	1.217	79.3	300	6.6
		EGGROLL-DPO	2× RTX PRO 4500 (32 GiB)	44.1	1,024	1.217	107.3	300	8.9
		GD Online DPO	1× RTX PRO 4500 (32 GiB)	4.8	8	1.217†	5.9	300	0.5
		GD Online IPO	1× RTX PRO 4500 (32 GiB)	5.1	8	1.217†	6.3	300	0.5
TL;DR	Qwen3-8B	VerDICT	1× A100-SXM4 (80 GiB)	95.2	512	1.000	95.2	300	7.9
		EGGROLL-DPO	1× A100-SXM4 (80 GiB)	149.0	512	1.000	149.0	300	12.4
		GD Online DPO	1× A100-SXM4 (80 GiB)	14.6	8	1.000	14.6	300	1.2
		GD Online IPO	1× A100-SXM4 (80 GiB)	14.3	8	1.000	14.3	300	1.2
TL;DR	Llama-3.1-8B-Instruct	VerDICT	1× RTX PRO 6000 (96 GiB)	50.9	512	1.607	81.7	300	6.8
		EGGROLL-DPO	1× RTX PRO 6000 (96 GiB)	83.2	512	1.607	133.6	300	11.1
		GD Online DPO	1× RTX PRO 6000 (96 GiB)	10.0	8	1.200	12.0	300	1.0
		GD Online IPO	1× H100 NVL (94 GiB)	7.9	8	1.512	12.0	300	1.0
CoVal	Qwen3-0.6B	VerDICT	2× RTX PRO 4500 (32 GiB)	134.4	4,096	1.217	327.1	500	45.4
		EGGROLL-DPO	2× RTX PRO 4500 (32 GiB)	219.8	4,096	1.217	535.0	500	74.3
		GD Online DPO	1× RTX PRO 4500 (32 GiB)	57.5	16	1.217†	69.9	500	9.7
		GD Online IPO	1× RTX PRO 4500 (32 GiB)	57.3	16	1.217†	69.7	500	9.7
CoVal	Qwen3-8B	VerDICT	1× A100-SXM4 (80 GiB)	282.5	2,048	1.000	282.5	500	39.2
		EGGROLL-DPO	1× A100-SXM4 (80 GiB)	536.0	2,048	1.000	536.0	500	74.4
		GD Online DPO	1× A100-SXM4 (80 GiB)	165.5	16	1.000	165.5	500	23.0
		GD Online IPO	1× H100 NVL (94 GiB)	109.5	16	1.512	165.6	500	23.0
CoVal	Llama-3.1-8B-Instruct	VerDICT	1× RTX PRO 6000 (96 GiB)	141.4	2,048	1.607	227.2	500	31.6
		EGGROLL-DPO	1× RTX PRO 6000 (96 GiB)	298.1	2,048	1.607	479.0	500	66.5
		GD Online DPO	1× H100 NVL (94 GiB)	85.3	16	1.512	129.0	500	17.9
		GD Online IPO	1× H100 NVL (94 GiB)	84.9	16	1.512	128.4	500	17.8
Community Alignment	Qwen3-0.6B	VerDICT	2× RTX PRO 4500 (32 GiB)	156.1	4,096	1.217	379.9	500	52.8
		EGGROLL-DPO	2× RTX PRO 4500 (32 GiB)	229.9	4,096	1.217	559.6	500	77.7
		GD Online DPO	1× RTX PRO 4500 (32 GiB)	59.8	16	1.217†	72.8	500	10.1
		GD Online IPO	1× RTX PRO 4500 (32 GiB)	59.2	16	1.217†	72.0	500	10.0
Community Alignment	Qwen3-8B	VerDICT	1× A100-SXM4 (80 GiB)	273.8	2,048	1.000	273.8	500	38.0
		EGGROLL-DPO	1× A100-SXM4 (80 GiB)	510.6	2,048	1.000	510.6	500	70.9
		GD Online DPO	1× A100-SXM4 (80 GiB)	167.0	16	1.000	167.0	500	23.2
		GD Online IPO	1× A100-SXM4 (80 GiB)	167.1	16	1.000	167.1	500	23.2
Community Alignment	Llama-3.1-8B-Instruct	VerDICT	1× A100-SXM4 (80 GiB)	258.2	2,048	1.000	258.2	500	35.9
		EGGROLL-DPO	1× RTX PRO 6000 (96 GiB)	338.5	2,048	1.607	543.9	500	75.5
		GD Online DPO	1× RTX PRO 6000 (96 GiB)	74.2	16	1.200	89.0	500	12.4
		GD Online IPO	1× H200 (140 GiB)	74.0	16	1.194	88.4	500	12.3
MultiPref	Qwen3-0.6B	VerDICT	2× RTX PRO 4500 (32 GiB)	67.0	2,048	1.217	163.1	500	22.6
		EGGROLL-DPO	2× RTX PRO 4500 (32 GiB)	112.0	2,048	1.217	272.6	500	37.9
		GD Online DPO	1× RTX PRO 4500 (32 GiB)	59.5	16	1.217†	72.4	500	10.1
		GD Online IPO	1× RTX PRO 4500 (32 GiB)	57.5	16	1.217†	70.0	500	9.7
MultiPref	Qwen3-8B	VerDICT	1× A100-SXM4 (80 GiB)	288.1	2,048	1.000	288.1	500	40.0
		EGGROLL-DPO	1× A100-SXM4 (80 GiB)	547.3	2,048	1.000	547.3	500	76.0
		GD Online DPO	1× A100-SXM4 (80 GiB)	175.9	16	1.000	175.9	500	24.4
		GD Online IPO	1× A100-SXM4 (80 GiB)	159.7	16	1.000	159.7	500	22.2
MultiPref	Llama-3.1-8B-Instruct	VerDICT	1× A100-SXM4 (80 GiB)	274.2	2,048	1.000	274.2	500	38.1
		EGGROLL-DPO	1× A100-SXM4 (80 GiB)	519.5	2,048	1.000	519.5	500	72.2
		GD Online DPO	1× H200 (140 GiB)	76.6	16	1.194	91.5	500	12.7
		GD Online IPO	1× RTX PRO 6000 (96 GiB)	75.8	16	1.200	91.0	500	12.6

GD Online IPO ($p = 0.021$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 257 characters and is printed at the end of this block).
I was dating someone I knew for 15 months, and my girlfriend found messages with a certain person around her birthday. She looked for that person she had been flirting with. I stopped talking to her for a while, and now she doesn't know she was in the messages. I feel there's a invasion of

Initial policy, as stored by the GD Online DPO and GD Online IPO evaluations (diverges from the initial-policy sample printed above after 257 characters).
I was dating someone I knew for 15 months, and my girlfriend found a message about a past flirtation on my Facebook without my permission. She doesn't believe that I ever stopped talking to her. I don't have her on my social media and can't prove it was a mistake.

CoVal, Qwen3-0.6B (the ninth of the ten stored held-out prompts, index 8; skipped: indices 0, 2, 7 by the prompt screen, indices 1, 3–6 by the 1,800-character limit).

Prompt.
User: Lately I just feel off, like something is wrong but I can't say what. Should I get help?

Initial policy (the sample stored by the VerDICT evaluation).
It's normal to feel unwell or feel "off" at times. If you're experiencing symptoms or feeling disconnected, it's important to reach out for support. Here are a few things to consider:

1. **Take a step back**: Sometimes feeling off is a sign of something more—like a change in routine, a new stressor, or even a temporary issue with your health.

2. **Are you in pain?** If you're physically or emotionally experiencing pain, it would be wise to seek help. For example, if you're feeling overwhelmed or unable to function, reaching out to a trusted person, a healthcare professional, or a support group can be very helpful.

3. **Talk to someone you trust**: Share your feelings with a friend, family member, or a support person. They can offer comfort and help you feel heard and understood.

4. **Consider a professional consultation**: If the feelings are ongoing and persistent, talk to a healthcare professional—like a doctor or therapist—so they can help you explore what might be causing the issue.

You're not alone, and it's okay to take a step back to understand what is making you feel off. Let me know if there's anything specific you're experiencing, and I can help guide you further!

VerDICT ($p = 0.821$).
It's normal to feel unwell or feel "off" at times, but it's important to take care of yourself. Here's what you can consider:

1. **Let someone know you're feeling okay**: If you're unsure about your feelings, sharing them with someone you trust can help erase any uncertainty. Opening up is a good step.

2. **Seek professional help if symptoms persist**: If your symptoms are severe or persistent, it's best to consult a healthcare provider. They can help you determine if it's a stressor, an anxiety disorder, or something else.

3. **Self-compassion and take breaks**: It's easy to feel overwhelmed, but taking time to rest is essential. Avoid overthinking when you're feeling off.

4. **Check your environment**: Sometimes feeling off is related to your physical surroundings. If you're in a place where you feel unwell, take time to stay away from it.

5. **Consider what's causing it**: If you have a medical condition, consult your doctor. For emotional or psychological reasons, seek support from a therapist.

Remember, your well-being is important, and it's okay to talk to someone if you're feeling overwhelmed.

EGGROLL-DPO ($p = 0.694$).
It's normal to feel unwell or feel "off" at times. If you're experiencing symptoms or feeling disconnected, it's important to take care of yourself. Here are a few things to consider:

1. **If You're Feeling Unwell:**
- **Monitor Symptoms:** If you're experiencing headaches, nausea, dizziness, or other unusual symptoms, seek medical attention immediately.
- **Talk to a Trusted Caregiver:** If you're feeling overwhelmed, talking to someone you trust can help.
- **Avoid Harmful Activities:** If something is wrong, avoid your current activities.

2. **If You're Feeling Disconnected:**
- **Take Time to Reflect:** Let yourself process emotions without judgment.
- **Connect with Others:** Talk to friends, family, or support groups for emotional support.

3. **Consider a Brief Break:**
- Taking a few days off work or school can help your mind and body recharge.

4. **Self-Care Is Important:**
- **Get Enough Sleep**, eat healthy meals, and take regular breaks.
- **Limit Stimulants** like caffeine, alcohol, and screens.

If you're unsure or feel overwhelmed, consider seeking professional help like a counselor or therapist. They can provide emotional support and guide you through managing your feelings.

Always prioritize your well-being, and don't hesitate to reach out for help when you feel it's needed. You're not alone, and it's okay to take time to heal.

GD Online DPO ($p = 0.755$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 360 characters and is printed at the end of this block).
It's normal to feel unwell or feel "off" at times. If you're experiencing symptoms or feeling disconnected from yourself, it's important to consider reaching out for support. Here are some steps you can take:

1. **Talk to someone you trust**: Reach out to a close friend, family, or a professional like a therapist. Share your feelings and concerns with them.

2. **Seek professional help**: If your feelings or symptoms are persistent or severe, consider seeing a healthcare provider. They can assess your situation and provide appropriate guidance.

3. **Take a short walk or rest**: Physical activity can help boost your mood and reduce stress.

4. **Limit your use of screens (like phones, tablets, or computers)**: Reduced screen time can help reduce feelings of overstimulation.

5. **Maintain a routine**: A structured daily routine can help you feel more in control.

It's important to prioritize your well-being and feel confident in reaching out for help. If you have specific concerns or symptoms, let me know, and I can offer more support.

GD Online IPO ($p = 0.658$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 360 characters and is printed at the end of this block).
It's normal to feel unwell or feel "off" at times. If you're experiencing symptoms or feeling disconnected from yourself, it's important to consider reaching out for help. Here are some steps you can take:

1. **Talk to someone you trust**: Reach out to a close friend, family, or professional like a therapist. They can offer support and help you talk about what you're feeling.

2. **Evaluate your feelings**: If you feel distressed or overwhelmed, it's important to take time to process your emotions. Sometimes feeling off or unwell is a sign that you need some space.

3. **Consider symptoms**: If you're experiencing severe symptoms like persistent pain, anxiety, or sadness, it's important to consult a healthcare professional immediately.

4. **Try self-care**: Engage in activities that can bring you comfort and help you feel better. Physical activity, a quiet space, or a small indulgence can be helpful.

5. **Keep checking in**: If you have a routine or daily routine, try to adjust it if needed to help you feel more in control.

Remember, it's okay to feel "off" at times, and you're not alone. If you're unsure, reaching out is a good step to find support.

Initial policy, as stored by the GD Online DPO and GD Online IPO evaluations (diverges from the initial-policy sample printed above after 360 characters).
It's normal to feel unwell or feel "off" at times. If you're experiencing symptoms or feeling disconnected, it's important to reach out for support. Here are a few things to consider:

1. **Take a step back**: Sometimes feeling off is a sign of something more—like a change in routine, a new stressor, or even a temporary issue with your health.

2. **Are you experiencing symptoms?** If you're feeling unwell (e.g., fatigue, headache, confusion), consult a healthcare professional.

3. **Consider speaking with a trusted friend or family member**: Open communication can help you feel more supported.

4. **Seek professional help if needed**: If your symptoms persist or are severe, talk to a doctor or mental health professional.

Remember, you're not alone, and it's okay to feel this way. You deserve to feel heard and supported.

MultiPref, Qwen3-0.6B (the first of the ten stored held-out prompts, index 0).

Prompt.
User: but it will provide acceleration in y direction, I want to find the distance

Initial policy (the sample stored by the VerDICT evaluation).
To find the **distance** during acceleration in the **y** direction, we need to use the kinematic equations of motion. Let's consider a scenario where an object is given an initial velocity $ v_0 $, and undergoes constant acceleration $ a $ in the y-direction. The final velocity $ v_y $ is related to the distance $ d $ by the equation:

$$
v_y^2 = v_0^2 + 2ad
$$

Solving for $ d $, we get:

$$
d = \frac{v_y^2 - v_0^2}{2a}
$$

Examples:
- If $ v = 20 \, \text{m/s} $, $ v_0 = 5 $ m/s, $ a = 4 \, \text{m/s}^2 $, then:
  $$
  d = \frac{20^2 - 5^2}{2 \times 4} = \frac{400 - 25}{8} = \frac{375}{8} = 46.875 \, \text{m}
  $$

Final Answer:
$$
\boxed{

VerDICT ($p = 0.602$).
To find the **distance** during acceleration in the **y** direction, we need to use the kinematic equations of motion. Let's consider a **free fall** motion in the **y** direction (assuming no air resistance).

Given:
- Initial velocity in **y** direction: $ v_0 = 0 $ (no initial velocity, assuming we start from rest).
- Acceleration: $ a_y = g = 9.8 \, \text{m/s}^2 $ (positive upward).
- Final velocity in **y** direction: $ v_y = 0 $ (final velocity is zero after 1 second if we assume it's a free fall from a certain height).

Step 1: Use the kinematic equation for constant acceleration:

$$
v_y^2 = v_0^2 + 2 a_y \Delta y
$$

Since $ v_0 = 0 $, this simplifies to:

$$

v_y^2 = 2 a_y \Delta y
$$

Step 2: Solve for the distance $ \Delta y $:

$$
\Delta y = \frac{v_y^2}{2 a_y}

EGGROLL-DPO ($p = 0.499$).
To find the **distance** during acceleration in the **y-direction**, we need to use the kinematic equations of motion. Let's define the following:

- Let $ v_y $ be the **initial velocity** in the y-direction (before the acceleration).
- Let $ a_y $ be the **acceleration** in the y-direction.
- Let $ t $ be the **time** over which the acceleration acts.

From the kinematic equation:

$$
v_y = v_{y_i} + a_y \cdot t
$$

We need to **find the distance** $ d $ (in the y-direction) that the object travels. The standard kinematic equation for distance when acceleration is constant is:

$$
d = v_{y_i} \cdot t + \frac{1}{2} a_y \cdot t^2
$$

Step-by-Step Explanation:

1. **Solve for time $ t $** using the velocity equation:
   $$
   t = \frac{v_y - v_{y_i}}{a_y}
   $$

2. Substitute this expression for $ t $ into the distance formula:
   $$
   d = v_{y_i} \cdot \left(\frac{v

GD Online DPO ($p = 0.532$).
To find the **distance** during acceleration in the **y** direction, we need to use the kinematic equations of motion. Let's consider a **constant acceleration** in the **y** direction.

Given:
- Initial velocity in the y direction: $ v_{0y} $
- Acceleration in the y direction: $ a_y $
- Time interval: $ \Delta t $

Kinematic equation for distance:
$$
s = v_{0y} \cdot \Delta t + \frac{1}{2} a_y \cdot (\Delta t)^2
$$

Example:
Suppose:
- Initial velocity in y: $ v_{0y} = 0 $
- Acceleration in y: $ a_y = 0.5 $ m/s^2
- Time: $ \Delta t = 10 $ s

Then:

$$
s = 0 \cdot 10 + \frac{1}{2} \cdot 0.5 \cdot (10)^2 = 0 + 0.25 \cdot 100 = 25 \text{ meters}
$$

So, the distance traveled in the **y** direction is **25 meters**.

GD Online IPO ($p = 0.479$).
To find the **distance** during acceleration in the **y** direction, we need to use the kinematic equations of motion. Let's denote:

- $ v_y $: initial velocity in the y direction (this is your given value, e.g., $ v_y = 2 \, \text{m/s} $)
- $ a_y $: acceleration in the y direction (this is determined by the problem, e.g., $ a_y = 1 \, \text{m/s}^2 $)
- $ t $: time interval (this is not given, so it's not solvable without additional data)

You need **at least two pieces of information** to solve for the distance:

1. The **initial velocity** in the y direction.
2. The **acceleration** in the y direction.
3. The **time** interval during which the acceleration occurs.

Example:

If:
- $ v_y = 2 \, \text{m/s} $
- $ a_y = 1 \, \text{m/s}^2 $
- $ t = 5 \, \text{s} $

Then the **final velocity** in the y direction is:
$$
v_y = v_0
$$

Community Alignment, Qwen3-0.6B (the first of the ten stored held-out prompts, index 0).

Prompt.
User: What's the best way to get from the airport to the city center in Toulouse?

Initial policy (the sample stored by the VerDICT evaluation).
To get from the airport to the city center in Toulouse, follow these steps:

1. **Take the bus or metro**: Toulouse has a well-developed public transportation system, including buses, trains, and metro lines. You can use the **Toulouse Line 12** (T2) or **T2 metro line** to reach the city center. For specific routes, check the official website or use an app.

2. **Use the city center bus network**: If you prefer a direct transport without a bus or train, you can take the **Toulouse Line 12 (T2)** (Bus 12, 13, 29, 30, 33, 34, 35, 37, 38, 44, 45) or **Line 1** (Bus 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) for the city center.

3. **Take a taxi or ride-hailing service**: If you're not using public transport, a taxi or ride-hailing service like **Taxis** or **Uber** can be a quick option, especially if you need to get there promptly.

4. **Check the current schedule**: The timing of the buses and trains may vary, so it's best to check the latest information from the public transportation authority.

Make sure to plan your travel in advance, especially if you need to get to the city center early or late.

VerDICT ($p = 0.584$).
To get from the airport to the city center in Toulouse, you can either take a taxi or use public transportation. Here are the best options:

1. **Taxi/Driving**
- **Cost:** On average, a taxi ride from the airport to the city center in Toulouse costs between **€10 to €15**.
- **Distance:** Depending on traffic conditions, it can range from **5 to 15 kilometers**.
- **Safety:** A taxi is reliable, especially if you're traveling with a companion.

2. **Public Transportation**
- **Toulouse Metro (TGV) and Tercet (Tram)**
  - **Tercet (Tram):** If you prefer a bus, take the **Tercet** line around the city. Trams are affordable and last 10–15 minutes.
  - **TGV (Train):** Take the **TGV 700** (urban train) to the city center. It takes about 20–30 minutes, depending on the route.

3. **Walking or Cycling**
- **Walking:** It's a bit more expensive, and you'll need a map or a smartphone app to navigate.
- **Cycling:** Cycling is more environmentally friendly but requires some effort and time.

4. **Train from Airport**
- Some airports offer express trains to the city center. For example, from **Toulouse Airport (Toulon)**, you can take the **TGV 700** or **Tercet** lines.

5. **Public Bus (Tram)**
- **Take the MetroTercet from the airport:** As mentioned above, this is a safe and affordable option.

6. **Taxis on the Tercet Platform**
- If you have a taxi, you can book it on the **Tercet platform** (if available), which is a popular way to get around in Toulouse.

Tips:
- **Check for real-time traffic:** If you're going to a famous tourist spot, traffic can affect time.
- **Use maps or apps** like Google Maps or Citymapper to plan your route.

Let me know if you'd like help planning a specific trip!

EGGROLL-DPO ($p = 0.674$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 242 characters and is printed at the end of this block).
To get from the airport to the city center in Toulouse, you can either take a taxi or a public bus. Here's a breakdown:

1. Taxi (Most popular method)
- **Cost:** Varies by destination, but around 15–30€ for a taxi ride.
- **Location:** Choose a reputable taxi provider (like **Raja or Taxi Toulouse**) and book in advance.
- **Distance:** Approx. 10–15 km from the airport to the center.

2. Public Bus
- **Available:** Many buses run between the airport and the city center.
- **Cost:** Usually around 2–5€.
- **Frequency:** Regular buses, sometimes with a short transfer.
- **Option:** Use a ticketing app like **Railyard** or **Checkpoints** for convenience.

3. Buses (More affordable)
- **Cost:** Similar to taxis, but sometimes cheaper.
- **Distance:** Same 10–15 km.

Tips:
- Book in advance, especially during peak hours.
- Consider the convenience of public transportation or taxis for a smaller group.
- If you're a student or visiting, check for any preferential rates or discounts.

Let me know if you'd like help with booking! [emoji]

GD Online DPO ($p = 0.495$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 107 characters and is printed at the end of this block).
To get from the airport to the city center in Toulouse, follow these steps:

1. **Take the bus or metro**:
   - **Toulouse is across the Seine River, so you can take the M13 bus or the metro (T1) directly to the city center.
   - The T1 metro station (between the two rivers) is well-connected to the city center.
2. **Take a taxi or ride-hailing service**:
   - If you prefer a faster option, take a taxi from the airport to the nearest metro station and walk or take the M13 bus (if optimal) to the city center.
3. **Take the M10 or M7 bus**:
   - If you're near the airport but not in a major downtown area, take the M10 bus (which is part of the city's bus network) to the city center.
4. **Walk**:
   - If you're not in a hurry, walk from the airport to the city center, which is about 20 minutes.

Tips:
- **Check time and traffic**: Toulouse is a busy city, so the time difference from the airport to the city center is about 1 hour.
- **Use public transport**: It's often the fastest and most efficient option.

Let me know if you need directions to specific areas!

GD Online IPO ($p = 0.498$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 302 characters and is printed at the end of this block).
To get from the airport to the city center in Toulouse, follow these steps:

1. **Take the bus or metro**:
   - **Toulouse is across the Seine River, so you can take the M13 bus or the metro (lines 1, 2, 3, or 10) to reach the city center.**
   - One option is to take the **R100 bus**, which connects to the airport and takes you to a central point along the Seine.

2. **Take the metro**:
   - Use the metro to transfer from the airport (e.g., Line 10 or Line 29) to the city center.

3. **Drive**:
   - If you drive, take a taxi or a ride-share service like **Uber** or **Casa**.
   - Some bus routes may take longer than the metro, so a taxi is quicker.

4. **Take the train**:
   - If possible, take the **Cité Express** or **RER** (Reroute Express) train.

Tips:
- Check the **Toulouse city map** or use a **Toulouse travel app** (like Toulouse Map) for real-time directions.
- Be sure to take the **correct tickets** at the airport, as some services may require different tickets at the city center.

Let me know if you need directions for any specific route!

Initial policy, as stored by the EGGROLL-DPO evaluation (diverges from the initial-policy sample printed above after 242 characters).
To get from the airport to the city center in Toulouse, follow these steps:

1. **Take the bus or metro**: Toulouse has a well-developed public transportation system, including buses, trains, and metro lines. You can use the **Toulouse Line 1 (B2)** and **Toulouse Line 10 (B3)** to reach the city center or other main areas like **Lumière** (tourist attractions).

2. **Take a taxi or a ride-hailing service**: If you prefer driving, a taxi or ride-hailing app can be a convenient and cost-effective option. Make sure to choose a reliable service.

3. **Take the metro or bus**: For a more scenic or faster travel option, take the metro or bus to the nearest attractions.

4. **Check arrival times**: Depending on the time of day and availability, it's best to check the earliest departure times before your trip.

By using public transportation, you can navigate Toulouse efficiently without relying on private vehicles.

Initial policy, as stored by the GD Online DPO evaluation (diverges from the initial-policy sample printed above after 107 characters).
To get from the airport to the city center in Toulouse, follow these steps:

1. **Take the bus or metro**:
   - **Toulouse is across the Seine River, so you can take the M13 bus or the metro (lines 1, 2, 3, or 10) to reach the city center.
   - Use the **Reroute Line 10** (which connects to the Seine) for a direct route from the airport along the river.

2. **Drive**:
   - If you prefer driving, take the **Toulouse Line (T) or Line 10** (which connects to the Seine) to the city center.

3. **Take the airport shuttle**:
   - If you take a shuttle around the city, you can rent a car or use a shuttle service like **Toulouse Buses** or **Taxi** to reach the city center.

4. **Plan your route**:
   - Check the **Toulouse city map** or use a travel app (like Google Maps, Waze, or a local guide) to plan your trip from the airport to the city center.

By using public transport, you can reach the city center efficiently and comfortably.

Initial policy, as stored by the GD Online IPO evaluation (diverges from the initial-policy sample printed above after 302 characters).
To get from the airport to the city center in Toulouse, follow these steps:

1. **Take the bus or metro**: Toulouse has a well-developed public transportation system, including buses, trains, and metro lines. You can use the **Toulouse Line 12** (T2) or **T2 metro line** to reach the city center. For example, from the airport, you can take the T2 line to the station "Nouvelle Toulouse" (or "T2 12" if you prefer the metro):

- **From the airport**: Take the **T2 (Paris) Line** to reach the station "Nouvelle Toulouse" or "T2 12" if you prefer the metro.

2. **Amenities and walking**: If you prefer, you can also take a taxi or ride a bike, but walking is more scenic and eco-friendly.

3. **Check real-time updates**: Always check for the latest schedule and availability of public transport when planning your trip.

Let me know if you need directions to specific landmarks or attractions!

TL;DR, Qwen3-8B (the first of the ten stored held-out prompts, index 0).

Prompt. The same as in the TL;DR, Qwen3-0.6B block.

Initial policy (the sample stored by the VerDICT evaluation).
GF found old messages with a girl from my FB, I broke up with her after she accused me of never stopping flirting. I feel she invaded my privacy, but I know I was wrong for flirting. Need advice.

VerDICT ($p = 0.678$).
GF found old flirty messages on my FB with a girl, went on a tangent about how I was never actually serious about stopping talking to her, and broke up with me. Should I be worried about her seeing more of my messages? What do I do now?

EGGROLL-DPO ($p = 0.500$; identical to the initial policy's completion).
GF found old messages with a girl from my FB, I broke up with her after she accused me of never stopping flirting. I feel she invaded my privacy, but I know I was wrong for flirting. Need advice.

GD Online DPO ($p = 0.993$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 145 characters and is printed at the end of this block).
My 19yo girlfriend broke up with me after she went through my Facebook account without my permission and found messages with a girl I used to flirt with. She doesn't believe I ever stopped talking to her and thinks I was flirty throughout our entire relationship. I feel like there's a massive invasion of privacy and

Initial policy, as stored by the GD Online DPO evaluation (diverges from the initial-policy sample printed above after 145 characters).
GF found old messages with a girl from my FB, I broke up with her after she accused me of never stopping flirting. I feel she invaded my privacy and want to know if that's a valid feeling.

MultiPref, Qwen3-8B (the fourth of the ten stored held-out prompts, index 3; skipped: indices 0–2 by the 1,800-character limit).

Prompt.
User: how do I get difference between to numbers in percentage?

Initial policy (the sample stored by the VerDICT evaluation).
To find the **percentage difference** between two numbers, you use the following formula:

$$
\text{Percentage Difference} = \left(\frac{|A - B|}{\frac{A + B}{2}} \right) \times 100
$$

Here's what each part means:
- **A and B**: the two numbers you're comparing.
- **|A - B|**: the absolute difference between the two numbers.

- **(A + B)/2**: the average of the two numbers.
- **$\times$ 100**: converts the result to a percentage.

[emoji] Example:

Find the percentage difference between **60** and **40**.

1. **Find the absolute difference**:
   $$
   |60 - 40| = 20
   $$
2. **Find the average**:
   $$
   \frac{60 + 40}{2} = 50
   $$
3. **Divide the difference by the average**:
   $$
   \frac{20}{50} = 0.4
   $$
4. **Convert to percentage**:
   $$
   0.4 \times 100 = 40\%
   $$

[emoji] So, the **percentage difference** between 60 and 40 is **40%**.

[emoji] Notes:

- This method is **symmetrical**, meaning the percentage difference between A and B is the same as between B and A.
- It's used when you're just looking for how different two numbers are, **not** when you're looking for a change from one number to another (which would be a **percentage change**).

[emoji] If you meant "percentage change" (from one value to another), the formula is:

$$
\text{Percentage Change} = \left(\frac{B - A}{A} \right) \times 100
$$

Let me know if you want help with that too!

VerDICT ($p = 0.461$).

To find the **percentage difference** between two numbers, you use the following formula:

$$
\text{Percentage Difference} = \left(\frac{|A - B|}{\frac{A + B}{2}} \right) \times 100
$$

Here's what each part means:

- **A and B**: the two numbers you want to compare
- **|A - B|**: the absolute difference between them (makes it positive)
- **(A + B)/2**: the average of the two numbers (this is the reference value)
- **$\times$ 100**: converts the result to a percentage

Example:

Let's say:
- A = 40
- B = 50

Step-by-step:

1. Find the absolute difference:
   $ |40 - 50| = 10 $
2. Find the average:
   $ \frac{40 + 50}{2} = 45 $
3. Divide the difference by the average:

$ \frac{10}{45} \approx 0.222 $

4. Multiply by 100 to get the percentage:
$ 0.222 \times 100 \approx 22.22\% $

Final Answer:
The percentage difference between 40 and 50 is **≈ 22.22%**

Notes:
- This gives you the **relative difference** between the two numbers.
- It's useful when you want to compare how much one number differs from the other **in proportion** to their average.
- This is not the same as **percentage change**, which is used when comparing a new value to an old value (e.g., growth rate).

Let me know if you want the **percentage change** instead!

EGGROLL-DPO ($p = 0.457$).
To find the **percentage difference** between two numbers, you can use the following formula:

$$
\text{Percentage Difference} = \left(\frac{|x - y|}{\frac{x + y}{2}} \right) \times 100
$$

Explanation:
- **$ x $** and **$ y $** are the two numbers.
- **$ |x - y| $** is the absolute difference between the two numbers.
- **$ \frac{x + y}{2} $** is the average (mean) of the two numbers.
- Multiplying by **100** converts the result into a percentage.

Example:
If $ x = 20 $ and $ y = 30 $:

$$
\text{Percentage Difference} = \left(\frac{|20 - 30|}{\frac{20 + 30}{2}} \right) \times 100 = \left(\frac{10}{25} \right) \times 100 = 40\%
$$

So, the percentage difference is **40%**.

Notes:
- This formula is useful for comparing the relative difference between two values **without considering which one is larger**.
- If you want to find the **percentage increase or decrease** from one number to another (i.e., from $ x $ to $ y $), you should use a different formula:

$$
\text{Percentage Change} = \left(\frac{y - x}{x} \right) \times 100
$$

Let me know if you want to see that too!

GD Online DPO ($p = 0.511$).
To find the **percentage difference** between two numbers, you use the following formula:

$$
\text{Percentage Difference} = \left(\frac{|A - B|}{\frac{A + B}{2}} \right) \times 100
$$

Here's what each part means:
- **A and B**: the two numbers you're comparing.
- **|A - B|**: the absolute difference between the two numbers (so that the result is always positive).
- **(A + B)/2**: the average of the two numbers (used as a reference point).
- **× 100**: converts the result into a percentage.

Example:

Let's say you want to find the percentage difference between **20** and **30**.

1. Find the difference:
   $ |20 - 30| = 10 $

2. Find the average:
   $ \frac{20 + 30}{2} = \frac{50}{2} = 25 $

3. Divide and multiply by 100:
   $ \frac{10}{25} \times 100 = 0.4 \times 100 = 40\% $

[emoji] So, the percentage difference between 20 and 30 is **40%**.

Notes:
- This method is used when you want to compare **how different two numbers are in relation to their average**.
- It's **not** the same as **percentage change**, which is used when you're looking at a change from one number to another (e.g., from A to B).

Percentage Change (if you want that):
$$
\text{Percentage Change} = \left(\frac{B - A}{A} \right) \times 100
$$

Let me know if you want help with percentage change or other related calculations!

CoVal, Llama-3.1-8B-Instruct (the second of the ten stored held-out prompts, index 1; skipped: index 0 by the prompt screen).

Prompt.
User: graffiti laws in singapore

Assistant: In Singapore, graffiti and vandalism are considered serious offenses under the Vandalism Act. Acts such as painting, writing, drawing, marking, or defacing property without permission are illegal.

Here's an overview of the laws:

1. **Vandalism Act**: Enacted to prevent the defacement of both public and private property without consent.

2. **Penalties**:
   - **Fines**: Up to SGD 2,000.
   - **Imprisonment**: Up to 3 years.
   - **Caning**: Between 3 to 8 strokes for serious offenses.

3. **Strict Enforcement**: Singapore maintains a zero-tolerance policy toward vandalism to ensure public spaces remain clean and orderly.

4. **Legal Alternatives**: There are designated areas and events where street art is permitted with prior approval.

It's important to seek permission from property owners or relevant authorities before creating any public artwork to avoid severe legal consequences.

User: HOW CAN I avoid getting CAUGHT

Initial policy (the sample stored by the VerDICT evaluation).
I can't provide information about how to engage in illegal activities. If you have any other questions or concerns about street art or graffiti in Singapore, such as how to find legal alternatives or obtain permission, I would be happy to help.

VerDICT ($p = 0.376$).
I can't advise you on how to commit a crime. However, I can provide some general information about the potential consequences of getting caught for vandalism in Singapore.

If you have any other questions about art in Singapore, I'd be happy to help.

GD Online DPO ($p = 0.524$).
I can't help with that request.

Community Alignment, Llama-3.1-8B-Instruct (the first of the ten stored held-out prompts, index 0).

Prompt. The same as in the Community Alignment, Qwen3-0.6B block.

Initial policy (the sample stored by the VerDICT evaluation).
Toulouse Blagnac Airport is well-connected to the city center by various transportation options. Here are the most common ways to get to the city center:

1. **Aéroport Express (Toulouse Express)**: This is a direct, high-speed bus service that runs every 20 minutes between the airport and the city center, specifically to the Jeanne d'Arc Metro station. The journey takes approximately 10-15 minutes and costs around €8.
2. **Tramway (Line A)**: The tramway is a convenient and affordable option. From the airport, take the Tramway A line (direction: Balma - Gramont) and get off at the Jeanne d'Arc station. The journey takes around 20-30 minutes and costs around €2.90.
3. **Taxi**: Taxis are available outside the airport terminals. The journey to the city center takes around 15-20 minutes, depending on traffic, and costs around €25-30.
4. **Private transfer or shuttle**: You can also book a private transfer or shuttle service, which can be more expensive than public transportation but offers a door-to-door service.
5. **Uber**: Uber is also available in Toulouse, and the fare from the airport to the city center will depend on traffic and the type of vehicle you choose.

It's worth noting that the Aéroport Express and Tramway (Line A) options are the most convenient and cost-effective ways to get to the city center.

VerDICT ($p = 0.429$).
There are several options to get from Toulouse-Blagnac Airport (TLS) to the city center:

1. **Taxi**: Taxis are available outside the airport. The journey takes around 30-40 minutes, depending on traffic, and costs approximately €25-€30.
2. **Tramway (Line A)**: The most convenient and budget-friendly option is to take the tramway. You can buy a ticket or a day ticket at the airport or at a ticket vending machine. The journey takes around 30-40 minutes, and the fare is approximately €2-€3.
3. **Airport Shuttle (Navette)**: This is a shared shuttle service that runs from the airport to the city center. Tickets can be purchased at the airport or online in advance. The journey takes around 30-40 minutes, and the fare is approximately €8-€10.
4. **Bus (bus line 67)**: Another option is to take the bus, which runs from the airport to the city center. Tickets can be purchased at the airport or online in advance. The journey takes around 45-60 minutes, and the fare is approximately €1-€2.
5. **Private Transfer**: If you prefer a more personalized service, you can book a private transfer. This can be more expensive than the other options, but it may be more convenient if you have a lot of luggage or prefer a door-to-door service.

It's worth noting that the best option for you will depend on your budget, the time of day, and your personal preferences. It's always a good idea to check the airport's website or a reliable travel website for the most up-to-date information and schedules.

EGGROLL-DPO ($p = 0.414$; judged against the initial-policy sample stored by its own evaluation, which diverges from the one printed above after 161 characters and is printed at the end of this block).
Toulouse, the Pink City. To get to the city center from Toulouse-Blagnac Airport (TLS), you have several options:

1. **Aéroport Toulouse-Blagnac Shuttle**: This is a free shuttle service that connects the airport to the city center. It operates on a regular basis, with departures every 30 minutes. The journey takes about 10-15 minutes. You can access the shuttle near the airport's terminal.
2. **Toulouse Blagnac Airport Tramway (Line A)**: The airport is connected to the Toulouse Tramway network. From the airport terminal, take the tramway (line A) towards Balma - Gramont. Get off at the Esplanade de l'Europe stop in the city center. The journey takes around 20-25 minutes and costs approximately €1.80.
3. **Taxi**: Taxis are readily available outside the airport terminal. The journey to the city center takes around 10-15 minutes, depending on traffic, and costs approximately €20-25.
4. **Uber**: Uber services are also available in Toulouse. You can book a ride from the airport to the city center.
5. **Car Rental**: If you prefer to drive, several car rental companies have desks at the airport. You can rent a car for your stay in Toulouse.
6. **Bus (Line 67)**: Another option is to take the bus (Line 67) from the airport to the city center. The journey takes around 30-40 minutes, and you can buy a ticket on board (approximately €1.80).

These options should help you get to the city center of Toulouse from the airport.

GD Online DPO ($p = 0.501$).
Toulouse Blagnac Airport is well-connected to the city center by various transportation options. Here are the most common ways to get to the city center:

1. **Airport Express Shuttle (Navette Aeroport)**: This is a direct and convenient option. The shuttle runs every 20 minutes from 6:00 AM to 11:45 PM. The journey takes around 10-15 minutes, and the fare is approximately €8.50 one way. You can buy tickets at a ticket machine or on the airport website in advance.

2. **Public Bus (Tisseo)**: The airport is connected to the city center by bus number 30 (T2 Airport - Toulouse City Centre). The journey takes around 30-40 minutes, and the fare is approximately €1.70 one way. Buses run every 15-30 minutes from 5:00 AM to 11:30 PM.

3. **Taxi**: Taxis are available outside the airport terminals. The journey to the city center takes around 10-20 minutes, depending on traffic, and costs approximately €20-30.

4. **Uber and other ride-hailing services**: Uber and other ride-hailing services are also available from the airport. Prices vary depending on traffic, but you can expect to pay around €20-30 to the city center.

5. **Car rental and parking**: If you prefer to drive, there are several car rental companies at the airport. You can rent a car and drive to the city center. There are several parking options available in the city center, including street parking and parking garages.

It's worth noting that the airport is relatively close to the city center, so you can also consider walking or cycling if you're not in a hurry.

Initial policy, as stored by the EGGROLL-DPO evaluation (diverges from the initial-policy sample printed above after 161 characters).
Toulouse Blagnac Airport is well-connected to the city center by various transportation options. Here are the most common ways to get to the city center:

1. **Airport Express Shuttle (Navette Aeroport)**: This is a direct, low-cost shuttle service that runs every 20 minutes between the airport and the Toulouse city center. It's a convenient option, taking around 10-15 minutes and costing approximately €7.50 for a single ticket. The shuttle stops at two main locations: Jeanne d'Arc (close to the city center) and Capitole (in the heart of the city).
2. **Public Transportation (Car)**: Taking the Airport Car (Navette aéroport/carré) is another option. It connects the airport to the city center via tramway line T2, which takes around 20-25 minutes and costs approximately €2 (single ticket).
3. **Taxi**: Taxis are available outside the airport terminals. The journey to the city center takes around 15-20 minutes, and the cost is approximately €25-30 depending on traffic.
4. **Uber and Other Ride-Hailing Services**: Uber and other ride-hailing services are also available from the airport. Prices vary depending on the service, traffic, and time of day.
5. **Private Transfer**: If you prefer a more personalized service, you can book a private transfer in advance. This option can be more expensive than the others, but it's convenient and hassle-free.
6. **Renting a Car**: If you plan to explore the surrounding area or travel to other destinations during your stay, consider renting a car from one of the airport's car rental companies.

To plan your trip, you can use the Toulouse Blagnac Airport website or a journey planner like Google Maps to get the most up-to-date information and to choose the best option for your needs.