

Self-Distilling Mathematical Reasoning in Small Language Models

Jörn Boehnke

Jacob Brophy

Ashwin A Nair

Abstract

Small base language models (0.5B–3B parameters) often fail to produce structured, scoreable output in zero-shot mathematical reasoning, leaving reward- and preference-based optimization with little signal to learn from. Yet the same models reach 80–90% scoreability under few-shot prompting, so the reasoning capability is present but not expressed by the zero-shot policy. We propose Advantage-Weighted Direct Preference Optimization (AWDPO), a self-distillation method that trains models to reproduce their own few-shot reasoning behavior without exemplars at inference time. AWDPO weights preference updates by the reward gap between an elicited few-shot trace and the model’s own zero-shot attempt, and stabilizes training with a dynamically scaled likelihood anchor. On GSM8K with Qwen-2.5 base models, AWDPO recovers over 90% of the accuracy of fully supervised chain-of-thought fine-tuning using four chain-of-thought exemplars rather than 7,473 annotated traces, and the resulting models generalize zero-shot to SVAMP, ASDiv, and MATH-500 without additional training. Methods that do not train on elicited traces fare far worse, with DPO and GRPO reaching 0% at 0.5B and answer-only supervised fine-tuning never exceeding 14.4%. We then examine the source of these gains by comparing AWDPO against single-round self-distillation on few-shot generated traces from each base model, with parameterization held fixed. The self-distillation objective comes within 2.2 points at 0.5B and 1.5B in-domain under the matched-LoRA comparison and transfers better out-of-domain at those scales, suggesting that much of the cold-start improvement can be recovered by distilling elicited behavior without preference optimization.

1 Introduction

Mathematical problem-solving is a widely used benchmark for reasoning in large language models (LLMs) (Cobbe et al., 2021). Recent work has shown that reinforcement learning (RL) can substantially improve mathematical reasoning (DeepSeek-AI et al., 2025), but these methods depend on model outputs being sufficiently structured to yield reliable reward signals. For small base models, this assumption often fails.

As we show empirically, 0.5B–3B parameter base models frequently produce unstructured, repetitive, or malformed zero-shot outputs, yielding parseable answers in fewer than 1–8% of cases (Section 4.1, Figure 1). Yet the same models reach 80–90% scoreability under few-shot chain-of-thought prompting (Wei et al., 2022). This creates a *cold-start regime*: useful reasoning behavior is accessible under prompting but poorly expressed by the zero-shot policy, making parser-based reward and preference optimization difficult.

We propose **Advantage-Weighted Direct Preference Optimization (AWDPO)**, which treats this few-shot/zero-shot gap as a self-distillation problem. AWDPO uses few-shot prompting to elicit structured traces from the model itself, weights preference updates by the reward gap between few-shot and zero-shot responses, and anchors training on correct few-shot traces. It requires only four externally supplied CoT demonstrations rather than a full rationale training set.

On GSM8K (Cobbe et al., 2021), four chain-of-thought exemplars are sufficient to recover over 90% of the accuracy of a supervised fine-tune using 7,473 generated reasoning traces, with zero-shot transfer to additional mathematical reasoning benchmarks.

Our central finding, however, is that much of this improvement can be reproduced by directly distilling few-shot-elicited behavior. We compare AWDPO against single-round self-distillation: supervised fine-tuning on the model’s own correct few-shot traces under the zero-shot prompt. With exemplars, questions, answer keys, and parameterization matched, self-distillation closely approaches AWDPO at the smaller Qwen scales and often transfers better out-of-domain, while larger differences emerge at 3B. These results show that simple self-distillation of few-shot-elicited behavior can reproduce much of the cold-start improvement, while the additional gap between self-distillation and AWDPO is smaller and model-dependent.

In summary, our contributions are:

1. We characterize a cold-start regime in small base models where zero-shot outputs are rarely scoreable despite substantially stronger few-shot behavior.
2. We introduce AWDPO, a reward-weighted self-distillation method that transfers few-shot reasoning into zero-shot behavior using four chain-of-thought exemplars and answer supervision.
3. We show through a matched self-distillation comparison that most of the gain comes from eliciting and distilling correct few-shot behavior rather than from the preference objective itself.

2 Related Work

Aligning Language Models with Preferences. Our work builds on Direct Preference Optimization (DPO) (Rafailov et al., 2023), a stable alternative to RLHF pipelines (Ouyang et al., 2022). AWDPO departs from standard DPO in two ways that place it within existing families rather than outside them. First, it replaces the binary preference label with a continuous reward-gap weight. Using reward or advantage magnitude in alignment objectives has precedent in Reward Difference Optimization (Wang et al., 2024), which reweights preference pairs using predicted reward differences, and Advantage-Induced Policy Alignment (Zhu et al., 2023), which directly incorporates estimated advantages into its policy-alignment loss. Second, our implementation drops the reference model, which forfeits DPO’s equivalence to a KL-regularized RLHF optimum and makes the objective a weighted contrastive loss. This places it in the reference-free family alongside CPO (Xu et al., 2024), whose behavior-cloning term plays the same role as our MLE anchor, ORPO (Hong et al., 2024), and SimPO (Meng et al., 2024). What distinguishes our setting is the source of the preference signal: pairs are generated by the model itself under two prompting conditions and scored against ground-truth answers, requiring no preference annotation or learned reward model. We train with LoRA (Hu et al., 2022), which matches full fine-tuning on many tasks while modifying a small fraction of parameters.

Distilling Context and Chain-of-Thought Reasoning. Chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022) elicits step-by-step reasoning that improves accuracy on math tasks. Several works transfer this ability to smaller models by generating rationales from a larger teacher and fine-tuning students on them (Magister et al., 2022; Li et al., 2023; Wang et al., 2023). We instead use the model’s own prompted behavior, on base models that lack instruction following. The closest prior work is context distillation, which internalizes behavior elicited by additional context

so it can be produced without that context (Askell et al., 2021; Bhargava et al., 2024). Snell et al. (2022) study this directly for in-context exemplars and step-by-step reasoning, which is the same premise AWDPO is built on. Also related is STaR (Zelikman et al., 2022), which iteratively filters and fine-tunes on a model’s own correct solutions, and single-round rejection-sampling fine-tuning (Yuan et al., 2023). We measure the performance of our preference formulation against these methods: the single-round self-distillation baseline of Section 3.4 is context distillation instantiated with our exemplars and training questions, and Sections 4.4 and 4.5 report how AWDPO compares against it.

3 Method

3.1 Advantage-Weighted Preference Loss

AWDPO uses a preference objective in the style of Direct Preference Optimization (Rafailov et al., 2023), with two departures from the standard formulation: the preference strength is a continuous reward gap rather than a binary label, and in all reported experiments we train without a reference model. This subsection describes the preference component of the objective. The MLE anchor and the dynamic weight that combines them are described in Section 3.2, and the full objective is given in Equation 2.

Sampling and pairing. For each training question x we draw G few-shot responses $y_1^{\text{fs}}, \dots, y_G^{\text{fs}} \sim \pi_\theta(\cdot \mid \text{prompt}(x, K))$, generated with K chain-of-thought exemplars in context, and G zero-shot responses $y_1^{\text{zs}}, \dots, y_G^{\text{zs}} \sim \pi_\theta(\cdot \mid x)$, generated from the question alone. These are combined into G preference pairs by matching on sample index: $(y_j^{\text{fs}}, y_j^{\text{zs}})$ for $j = 1, \dots, G$. Samples are not matched by similarity, length, or any other criterion, so as a result each pair is an independent draw from the joint of the two generation distributions. We use $G = 6$ throughout.

Batch construction. A question contributes to training only if at least one of its G few-shot responses yields a correct final answer. Questions for which no few-shot response is correct are skipped and replaced by further draws from the training set. The single-round self-distillation baseline (Section 3.4) applies the same correctness filter, so the two methods train on comparable question subsets.

Objective. Both responses in a pair are scored with the compound rules-based reward R of Section 3.3, and we take the signed reward gap

$$d_j(x) = R(x, y_j^{\text{fs}}) - R(x, y_j^{\text{zs}}).$$

We transform this gap with a hyperbolic tangent and decompose the result into a magnitude and a sign,

$$w_j(x) = |\tanh d_j(x)| \in [0, 1], \quad s_j(x) = \text{sign}(\tanh d_j(x)) \in \{-1, 0, +1\},$$

giving the preference loss

$$\mathcal{L}_{\text{AWDPO}}(\theta) = -\mathbb{E}\left[w_j(x) \log \sigma(s_j(x) \beta \Delta_\theta(x, j))\right], \quad (1)$$

$$\Delta_\theta(x, j) = \log \pi_\theta(y_j^{\text{fs}} \mid x) - \log \pi_\theta(y_j^{\text{zs}} \mid x),$$

where the expectation is over pairs drawn as described above. Both log-probabilities in Δ_θ are evaluated under the *zero-shot* prompt x , including that of the few-shot response. Few-shot

prompting is therefore used only to obtain candidate structured solutions, so the quantity being optimized is the zero-shot policy’s relative preference between them. This conditioning applies to the preference term only. The MLE anchor is conditioned on the few-shot prompt, as described in Section 3.2.

Three properties follow from this construction. First, the few-shot response is not assumed to be preferred. In practice, this means that when the zero-shot response receives the higher reward, $s_j(x) = -1$ and the update moves the policy toward y_j^{zs} instead. Second, because $w_j(x) \geq 0$, the loss is bounded below by zero. This is why the weight and sign are separated rather than using the signed gap $d_j(x)$ directly as a multiplier, which would leave the objective unbounded below on pairs where the zero-shot response scores higher. Third, pairs whose responses receive equal reward have $w_j(x) = 0$ and drop out of the update entirely, so the weighting acts as a soft filter, with the tanh bounding $w_j(x)$ to $[0, 1]$ so that no single large reward gap dominates a batch.

We set $\beta = 0.1$ in all experiments. Our implementation supports an optional reference model, in which case Δ_θ becomes the difference of policy-to-reference log-ratios for the two responses, but all results reported in this paper are obtained without one.

3.2 Parameterization and Anchoring

In the cold-start regime, the preference term in Equation 1 operates on a signal that is sparse and partly self-generated. Two additional components complete the method: a low-rank parameterization and a likelihood term on correct few-shot traces.

Low-rank adaptation We train AWDPO with LoRA (Hu et al., 2022) at rank $r = 64$ and scaling $\alpha = 64$, applied to the attention and MLP projection matrices (target modules listed in Appendix A.4). This restricts updates to a low-rank subspace and reduces the number of trainable parameters.

MLE anchor We add a maximum-likelihood term on correct few-shot traces,

$$\mathcal{L}_{\text{MLE}}(\theta) = -\mathbb{E}\left[\log \pi_\theta(y_j^{\text{fs}} \mid \text{prompt}(x, K))\right],$$

taken over those few-shot responses whose extracted final answer is correct. Two aspects of this term differ from the preference component. First, its conditioning: the anchor is evaluated under the *few-shot* prompt, whereas Δ_θ in Equation 1 is evaluated under the zero-shot prompt. The preference term transfers prompted behavior into the zero-shot policy, meaning that the anchor is intended to keep the prompted behavior itself from degrading as that transfer proceeds, since the few-shot generations are what supply the training signal. Second, it does not depend on the zero-shot policy producing scoreable output, so it remains available when the preference signal is sparse. Because a question enters a batch only if at least one of its few-shot responses is correct (Section 3.1), the anchor is active on every batch.

Combining the two terms. The anchor is weighted by a coefficient computed from the current batch’s scalar loss values,

$$\lambda_{\text{dyn}} = \begin{cases} \mathcal{L}_{\text{AWDPO}}/\mathcal{L}_{\text{MLE}} & \text{if } \mathcal{L}_{\text{MLE}} > 0, \\ 0.05 & \text{otherwise.} \end{cases}$$

Both losses are read as detached scalars, so λ_{dyn} enters the update only as a multiplier. The fallback constant is rarely used in practice, since a question enters a batch only if at least one of its few-shot responses is correct and the anchor is therefore active on every batch. The full objective is

$$\mathcal{L}_{\text{total}}(\theta) = \mathcal{L}_{\text{AWDPO}}(\theta) + \lambda_{\text{dyn}} \mathcal{L}_{\text{MLE}}(\theta). \quad (2)$$

This ratio keeps the two scalar loss values on a comparable scale, which prevents either term from numerically dominating the sum when they differ by orders of magnitude.

3.3 Reward Function

We use a compound, rules-based reward rather than a learned one. It primarily rewards correctness of the final answer, with smaller bonuses for using the required output format and for surface indicators of stepwise reasoning, and small penalties for malformed output. A full breakdown of the components and their weights is given in Appendix A.3.

The reward is distinct from answer-scoreability as defined in Section 4.1. Formatting and reasoning-shape terms are computable for outputs from which no final answer can be extracted, so a generation may receive a nonzero reward while remaining unscoreable, and the reward gap $d(x)$ between two unscoreable responses is generally nonzero.

3.4 Single-Round Self-Distillation

The step in AWDPO of using few-shot prompting to obtain structured traces from the base model can be paired with a much simpler objective. We take single-round self-distillation as our primary comparison because it provides a simpler way to exploit the same few-shot/zero-shot training setup.

Procedure. For each training question x we sample one few-shot response $y^{\text{fs}} \sim \pi_{\theta}(\cdot \mid \text{prompt}(x, K))$ using the same $k \in \{2, 3, 4\}$ exemplars as AWDPO, and retain it if its extracted final answer matches the ground truth. The model is then fine-tuned on the retained traces, conditioned on the *zero-shot* prompt:

$$\mathcal{L}_{\text{SD}}(\theta) = -\mathbb{E}[\log \pi_{\theta}(y^{\text{fs}} \mid x)]. \quad (3)$$

Training is a single pass: traces are generated once from the base model and not resampled as the policy changes.

Equation 3 has the same functional form as the MLE anchor of Section 3.2, but differs in conditioning. The anchor is evaluated under the few-shot prompt and is intended to preserve prompted behavior. Here the same traces are scored under the zero-shot prompt, which is what transfers that behavior into the zero-shot policy.

Relation to AWDPO. Self-distillation uses the same question bank, answer keys, exemplar pool, correctness criterion, and zero-shot inference target as AWDPO. It differs in the optimization objective and in trajectory generation: self-distillation samples and filters few-shot traces once from the base model, whereas AWDPO samples multiple few-shot and zero-shot responses and refreshes them during training. We report self-distillation both with full fine-tuning and with the same LoRA configuration as AWDPO, so that the latter comparison controls for parameterization while retaining these procedural differences.

```

1: Input: base policy  $\pi_\theta$ , questions  $\mathcal{D}$ ,  $K$  exemplars, reward  $R$ , group size  $G$ , batch size  $B$ .
2: while not converged do
3:    $\mathcal{B} \leftarrow \emptyset$ 
4:   while  $|\mathcal{B}| < B$  do
5:     Draw  $x$  from  $\mathcal{D}$ .
6:     Sample  $y_{1..G}^{\text{fs}} \sim \pi_\theta(\cdot \mid \text{prompt}(x, K))$  and  $y_{1..G}^{\text{zs}} \sim \pi_\theta(\cdot \mid x)$ .
7:     if no  $y_j^{\text{fs}}$  has a correct final answer then
8:       Discard  $x$  and continue.
9:     end if
10:     $\mathcal{B} \leftarrow \mathcal{B} \cup \{x\}$ 
11:  end while
12:  for each  $x \in \mathcal{B}$  and each  $j \in \{1, \dots, G\}$  do
13:     $d_j \leftarrow R(x, y_j^{\text{fs}}) - R(x, y_j^{\text{zs}})$ 
14:     $w_j \leftarrow |\tanh d_j|$ ,  $s_j \leftarrow \text{sign}(\tanh d_j)$ 
15:     $\Delta_j \leftarrow \log \pi_\theta(y_j^{\text{fs}} \mid x) - \log \pi_\theta(y_j^{\text{zs}} \mid x)$ 
16:     $\ell_j^{\text{pref}} \leftarrow -w_j \log \sigma(s_j \beta \Delta_j)$ 
17:  end for
18:   $\mathcal{L}_{\text{AWDPO}} \leftarrow \text{mean}(\ell^{\text{pref}})$ 
19:   $\mathcal{L}_{\text{MLE}} \leftarrow -\text{mean}[\log \pi_\theta(y_j^{\text{fs}} \mid \text{prompt}(x, K))]$  over correct  $y_j^{\text{fs}}$ 
20:   $\lambda_{\text{dyn}} \leftarrow \mathcal{L}_{\text{AWDPO}} / \mathcal{L}_{\text{MLE}}$  if  $\mathcal{L}_{\text{MLE}} > 0$ , else 0.05 ▷ detached
21:   $\theta \leftarrow \theta - \eta \nabla_\theta (\mathcal{L}_{\text{AWDPO}} + \lambda_{\text{dyn}} \mathcal{L}_{\text{MLE}})$ 
22: end while

```

Algorithm 1: Advantage-Weighted Direct Preference Optimization (AWDPO). Log-probabilities in Δ_j are evaluated under the zero-shot prompt; the MLE anchor is evaluated under the few-shot prompt.

3.5 Training Algorithm

Algorithm 1 summarizes the full procedure.

Training terminates early if no few-shot response is correct for 100 consecutive draws, which we treat as a collapse condition and report in Section 5.

4 Experiments

4.1 The Cold-Start Problem

Reward- and preference-based fine-tuning requires that a non-trivial fraction of model outputs be scorable under the reward function. We show that small base models fail to meet this requirement zero-shot, while satisfying it under few-shot prompting with the same weights.

Throughout the paper, scoreability refers specifically to answer-scoreability. Following the rule-based verification setting used in DeepSeek-R1 (DeepSeek-AI et al., 2025), we require the final answer to appear in a machine-parseable format. Given a generation, we take the contents of the final **<answer>** block and attempt to parse the stripped contents directly as an integer. The output is scoreable if this succeeds, and correct if the parsed integer matches the ground-truth answer. The same extractor is used for three purposes: measuring scoreability in this section, computing accuracy in Section 4.4, and deciding which few-shot traces are eligible to enter training. The

compound training reward (Appendix A.3) additionally includes formatting and reasoning-shape terms, but these are distinct from scoreability: an output may receive formatting bonuses or penalties independently of whether a final answer can be extracted from it.

Figure 1 tracks answer-scoreability over the course of AWDPO training. We adopt the XML-tagged answer format used in the reward construction of DeepSeek-R1 (DeepSeek-AI et al., 2025). At initialization, zero-shot outputs are rarely scoreable: under 1% for the 0.5B model and 2–8% for the 1.5B and 3B models. Few-shot prompting, using the same weights, yields 80–90% scoreability from the start. Because every method in this paper begins from the same base checkpoints, this gap is a property of the setting rather than of any particular training procedure.

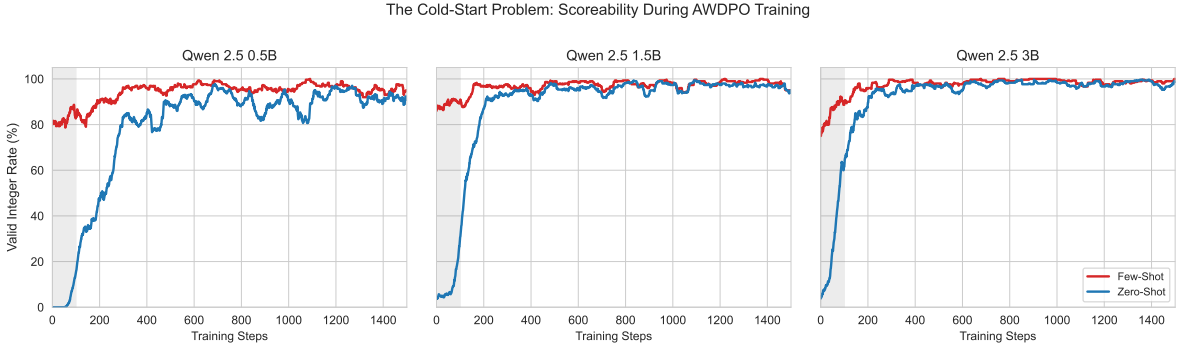


Figure 1: Scoreability of model outputs during AWDPO training. *Valid Integer Rate* measures the percentage of outputs yielding a parseable integer answer. At initialization (gray region), zero-shot outputs are largely unscorable (0–8%), while few-shot prompting yields 80–90% scoreability from the same model. AWDPO closes this gap within a few hundred training steps.

The 0.5B and 1.5B models typically produce repetitive loops or malformed token sequences, and rarely emit reasoning or answer tags at all. The 3B model more often produces structured, arithmetically reasonable output, but violates the answer format: it prefixes answers with currency symbols, appends units, or includes additional natural-language text inside the answer block. Representative generations are shown in Figure 2.

The difficulty is not that every generation receives zero reward. The auxiliary formatting terms remain computable for malformed outputs. It is that the correctness-bearing component of the reward is available for only a small and unrepresentative subset of generations. In such regimes, updates are dominated by a few atypical trajectories, which produces high-variance gradients (Schulman et al., 2017) and can lead to mode collapse (Ziegler et al., 2019). Prior work applying reinforcement learning to base language models reports related instability in the absence of supervised warmup, including degraded readability and reasoning quality (DeepSeek-AI et al., 2025).

Both methods that train on few-shot-elicited traces resolve the scoreability problem. As Figure 1 shows, AWDPO closes the gap within a few hundred training steps; single-round self-distillation reaches 98.6%, 99.4%, and 99.4% at 0.5B, 1.5B, and 3B. Both procedures show that training on few-shot-elicited traces can largely resolve the scoreability problem under either training procedure.

4.2 Experimental Setup

We evaluate on mathematical reasoning tasks, focusing primarily on **GSM8K** (Cobbe et al., 2021), a benchmark of grade-school math word problems. To assess zero-shot cross-dataset transfer we also evaluate on two further word-problem benchmarks, **SVAMP** (Patel et al., 2021) and **ASDiv**

(Miao et al., 2020), and on the competition-level **MATH-500** (Hendrycks et al., 2021; Lightman et al., 2024).

Our primary base models are **Qwen-2.5** language models of 0.5B, 1.5B, and 3B parameters (Qwen et al., 2025). We additionally evaluate **Llama 3.2 3B** (AI at Meta, 2024) to test whether our findings hold across model families. All are base checkpoints rather than instruction-tuned variants, and none reliably follows output-format instructions zero-shot, which is the condition that produces the cold-start regime described in Section 4.1.

4.3 Baselines

All methods draw on the same underlying resources: the 7,473 GSM8K training questions, their ground-truth answers, and a set of chain-of-thought solutions generated for those questions by Qwen 2.5 3B-Instruct. Methods differ in how much of the chain-of-thought set they consume. SFT trains on all 7,473 generated solutions. The four fixed exemplars used by AWDPO, self-distillation, Filtered-DPO, and Answer-SFT are drawn from this same set, so no method has access to rationale data the others could not have obtained. DPO’s preference pairs are also constructed from it: of the 7,473 questions, 6,921 produced both a scoreable-correct and a scoreable-incorrect response and so admitted a valid preference pair, while the remaining 552 did not.

The following methods were compared:

1. **Few-shot prompting** The base model is given all four chain-of-thought exemplars as demonstrations at inference time, with no fine-tuning. This is the prompting-only reference point.
2. **Answer-SFT** Supervised fine-tuning on the 7,473 answer-only question–answer pairs together with the four chain-of-thought exemplars. This tests whether answer supervision at scale plus the seed exemplars is sufficient, without any intermediate reasoning supervision.
3. **DPO** Fine-tuning with the objective of Rafailov et al. (2023) on the 6,921 preference pairs described above, following the standard configuration with an explicit reference model.
4. **GRPO** Group Relative Policy Optimization (Shao et al., 2024) applied directly to the base model with no supervised warmup, using the same reward function as AWDPO. GRPO learns only from its own zero-shot rollouts, and so tests whether reinforcement learning alone can bootstrap structured reasoning from sparse rewards.
5. **Self-distillation** Described in Section 3.4; we report both the full-rank and matched-LoRA configurations.
6. **Filtered-DPO** A variant that differs from AWDPO in two respects: the advantage $w(x)$ is clamped to 1 for every pair, so all pairs contribute equally regardless of the reward gap, and the preference loss is computed only over pairs whose few-shot trace is correct, rather than over all sampled pairs. It therefore tests whether filtering to correct traces under an unweighted objective is sufficient, and is not a clean isolation of reward-magnitude weighting on its own. Analysis is provided by the component ablations in Section 5.

AWDPO and Filtered-DPO use LoRA with $r = \alpha = 64$, while DPO, GRPO, Answer-SFT, and SFT use full-parameter fine-tuning. Self-distillation is reported both at full rank and with the same LoRA configuration as AWDPO. Because DPO and GRPO differ from AWDPO in parameterization as well as the objective, we treat them as empirical baselines rather than controlled objective-level comparisons. The matched-LoRA self-distillation configuration controls for parameterization,

although it still differs from AWDPO in both optimization objective and trajectory-generation procedure. Answer-SFT and SFT serve as supervised reference points rather than matched objective comparisons.

We do not report zero-shot performance of the untrained base models, which produce scoreable output in under 8% of cases (Section 4.1) and therefore do not yield a meaningful accuracy figure.

During training, AWDPO and Filtered-DPO sample $k \in \{2, 3, 4\}$ exemplars from the fixed set of four for each instance, to increase prompt diversity across training steps. The few-shot prompting baseline in Table 1 always uses all four exemplars. Where applicable, methods are trained for the same number of epochs. Full hyperparameter settings, LoRA target modules, decoding configuration, and the four exemplars themselves are given in Appendix A.4.

4.4 Results on GSM8K

Table 1 reports zero-shot pass@1 on the GSM8K test set for each method and model size. Pass@1 is the fraction of test questions for which a single sampled generation yields the correct final answer. We require the stripped contents of the final `<answer>` block to parse directly as an integer. The same rule is used for all methods and for the scoreability measurements in Section 4.1.

Standard DPO reaches 0.00% at all three model sizes despite training on 6,921 externally generated preference pairs. GRPO, which learns only from its own zero-shot rollouts, reaches 0.00% at 0.5B; it recovers to 46.86% and 44.37% at 1.5B and 3B but does not improve monotonically with scale and stays well below the four-exemplar methods at both sizes. Answer-SFT, which sees the same 7,473 questions with answer-only supervision plus the four chain-of-thought exemplars, reaches 5.35%, 10.24%, and 14.40%. Answer supervision at scale is not sufficient on its own.

Filtered-DPO generates the same few-shot and zero-shot pairs as AWDPO and trains on the correct few-shot traces without advantage weighting, but reaches only 13.73%, 4.84%, and 56.30%—non-monotone in model size, and far below both AWDPO and single-round self-distillation, which both train on few-shot-elicited reasoning traces. Its generations frequently collapse to trivial or malformed outputs. An unweighted preference objective can therefore discard the benefit of good elicited traces, which is what motivates the component analysis in Section 5.

Every method that trains on few-shot-elicited traces reaches a large fraction of the fully supervised reference point while using four externally supplied chain-of-thought demonstrations rather than 7,473. AWDPO attains 94.4%, 92.2%, and 95.0% of SFT accuracy at 0.5B, 1.5B, and 3B. Self-distillation attains 88.6%, 91.1%, and 90.8% at full rank, and 89.2%, 91.3%, and 85.3% under the matched LoRA configuration. These are ratios of absolute accuracy, not of the improvement over any baseline. We report this framing throughout.

Few-shot prompting reaches 24.53%, 56.00%, and 53.78% under our protocol. These figures are lower than those reported in the Qwen-2.5 technical report for 4-shot chain-of-thought GSM8K because the prompting, decoding, and evaluation protocols differ. We retain the strict rule rather than relaxing it for evaluation because the same extractor decides which few-shot traces enter training and defines the scoreability measurements of Section 4.1. As a result, a looser evaluation-time rule would decouple reported accuracy from the signal the methods train against. Every method in Table 1 is scored by this one rule, so the margins we report reflect both arithmetic accuracy and format adherence.

The comparison of most direct interest is between AWDPO and single-round self-distillation. They share the question bank, answer keys, exemplar pool, correctness criterion, and, in the matched configuration, LoRA parameterization, while differing in their objective and trajectory-generation procedure. Against self-distillation at the matched LoRA configuration, AWDPO leads by +2.14 at 0.5B, +0.61 at 1.5B, and +7.92 at 3B. Against full-rank self-distillation the corresponding

Method	Params	pass@1 (%)	CoT Exemplars
<i>Qwen-2.5 0.5B</i>			
DPO	full	0.00	6,921
GRPO	full	0.00	0
Answer-SFT	full	5.35	4
Filtered-DPO	LoRA	13.73	4
4-shot prompting	—	24.53	4
Self-Distillation	full	36.45	4
Self-Distillation	LoRA	36.69	4
AWDPO	LoRA	38.83	4
SFT	full	41.13	7,473
<i>Qwen-2.5 1.5B</i>			
DPO	full	0.00	6,921
Filtered-DPO	LoRA	4.84	4
Answer-SFT	full	10.24	4
GRPO	full	46.86	0
4-shot prompting	—	56.00	4
Self-Distillation	full	63.64	4
Self-Distillation	LoRA	63.84	4
AWDPO	LoRA	64.45	4
SFT	full	69.89	7,473
<i>Qwen-2.5 3B</i>			
DPO	full	0.00	6,921
Answer-SFT	full	14.40	4
GRPO	full	44.37	0
4-shot prompting	—	53.78	4
Filtered-DPO	LoRA	56.30	4
Self-Distillation	LoRA	69.71	4
Self-Distillation	full	74.22	4
AWDPO	LoRA	77.63	4
SFT	full	81.75	7,473

Table 1: Zero-shot pass@1 on GSM8K, ordered by accuracy within each model size. Best result per model size in bold. **Params** indicates full fine-tuning or a LoRA adapter, and **CoT Exemplars** the number of externally supplied chain-of-thought demonstrations; all fine-tuned methods additionally have access to the answer-only training questions. Self-distillation is reported both at full rank and with AWDPO’s LoRA configuration ($r = \alpha = 64$). Single seed.

margins are +2.38, +0.81, and +3.41. At the two smaller scales the margins are small under either comparison, and on Llama 3.2 3B the ordering reverses, with self-distillation reaching 29.38% against AWDPO’s 28.81% (Table 4). At 3B, the matched comparison shows a substantially larger in-domain difference than at the two smaller scales.

The difference between the matched-LoRA and full-rank self-distillation comparisons is larger at 3B than at the two smaller scales. Full-rank and LoRA self-distillation differ by 0.24 and 0.20 points at 0.5B and 1.5B but by 4.51 at 3B, an effect we return to in Section 5. The 3B in-domain advantage does not extend uniformly to the transfer benchmarks (Section 4.5).

4.5 Zero-Shot Generalization to Other Datasets

To test whether the elicited reasoning behavior transfers beyond the dataset it was trained on, we evaluate every fine-tuned method zero-shot on SVAMP, ASDiv, and MATH-500, with no additional training and no exemplars at inference. All models in this section are trained only on GSM8K. Table 2 reports the results for AWDPO, self-distillation in both parameterizations, and the full-rank SFT reference point.

At 0.5B and 1.5B, self-distillation transfers better than AWDPO on both word-problem benchmarks, under either parameterization. Under the matched LoRA comparison it leads by 14.4 and 8.4 points on SVAMP and by 10.8 and 6.1 points on ASDiv. The full-rank margins are similar (12.1 and 8.0 on SVAMP, 11.1 and 6.6 on ASDiv). Self-distillation also comes close to the supervised reference point on these benchmarks: at full rank it reaches 95.8–98.6% of SFT accuracy on SVAMP and ASDiv at 0.5B and 1.5B, and the LoRA configuration at 0.5B slightly exceeds SFT on SVAMP (0.552 against 0.543).

At 3B the pattern does not hold. Under the matched comparison AWDPO is ahead on SVAMP by 0.9 points and behind on ASDiv by 0.6, which is close to parity on both. Full-rank self-distillation, by contrast, leads AWDPO by 3.9 and 8.4 points on the same two benchmarks. The out-of-domain ordering at this scale therefore depends on which parameterization self-distillation is given.

That dependence is specific to the largest model. At 0.5B and 1.5B, the two self-distillation configurations are generally close, with differences of at most 2.3 points across Table 2, indicating that the two parameterizations yield similar transfer performance at those scales. At 3B they diverge substantially. This suggests that a rank-64 constraint costs a supervised likelihood objective a considerable amount of transfer performance at 3B and almost none at 0.5B and 1.5B.

Model	Method		CoT	SVAMP	ASDiv	GSM8K	MATH-500
		Params	exemplars	p@1	p@1	p@1	p@1
0.5B	AWDPO	LoRA	4	0.408	0.703	0.388	0.112
	Self-Distillation	LoRA	4	0.552	0.811	0.367	0.130
	Self-Distillation	full	4	0.529	0.814	0.365	0.128
	SFT	full	7,473	0.543	0.826	0.411	0.151
1.5B	AWDPO	LoRA	4	0.691	0.862	0.645	0.268
	Self-Distillation	LoRA	4	0.775	0.923	0.638	0.251
	Self-Distillation	full	4	0.771	0.928	0.636	0.242
	SFT	full	7,473	0.805	0.941	0.699	0.295
3B	AWDPO	LoRA	4	0.791	0.866	0.776	0.324
	Self-Distillation	LoRA	4	0.782	0.872	0.697	0.299
	Self-Distillation	full	4	0.830	0.950	0.742	0.303
	SFT	full	7,473	0.888	0.961	0.818	0.373

Table 2: Zero-shot pass@1 across math reasoning benchmarks. All models are trained only on GSM8K and evaluated with no additional training and no exemplars at inference. Best per column and model size in bold. Self-distillation is reported both at full rank and with AWDPO’s LoRA configuration ($r = \alpha = 64$). Single seed.

Absolute accuracy on competition-level problems is low for every method, and the two four-exemplar methods are close without a consistent ordering. Under the matched comparison, self-

distillation leads by 1.8 points at 0.5B while AWDPO leads by 1.7 and 2.5 points at 1.5B and 3B. Against full-rank self-distillation the margins are -1.6 , $+2.6$, and $+2.1$ for AWDPO in the same order. Both recover a smaller fraction of the supervised reference here than on the word-problem benchmarks, consistent with MATH-500 requiring reasoning patterns that four GSM8K-style exemplars do not cover. The ordering at 3B depends on the self-distillation parameterization: under the matched-LoRA comparison, AWDPO and self-distillation are near parity on SVAMP and ASDiv, whereas full-rank self-distillation outperforms AWDPO on both benchmarks.

Large synthetic-augmentation systems reach substantially higher absolute accuracy on these benchmarks using tens to hundreds of thousands of constructed reasoning traces (Liu et al., 2023; Yu et al., 2024; Yue et al., 2023; Luo et al., 2025). Our claim concerns data efficiency rather than accuracy. Four externally supplied chain-of-thought exemplars, combined with answer-only questions and the model’s own few-shot generations, recover most of what a 7,473-trace supervised fine-tune achieves in the same setup.

5 Analysis and Ablations

We perform additional analyses to understand AWDPO’s behavior and the importance of its components.

Model	AWDPO	w/o Advantage	w/o MLE Anchor	w/o Both
0.5B	38.83	26.02	4.69	0.89
1.5B	64.45	44.96	38.41	23.80
3B	77.63	8.03	69.39	19.92

Table 3: Component ablation on GSM8K using Qwen-2.5 base models. Full AWDPO outperforms all variants.

Dependence on base-model few-shot capability. Both four-exemplar methods depend on the base model producing usable reasoning under prompting. To test how far that dependence extends, we repeat the comparison on Llama 3.2 3B (AI at Meta, 2024), which exposes far less capability under prompting than any Qwen model in our study: its 4-shot accuracy is 0.07%, against 24.53% for the 0.5B Qwen model.

Elicitation still works in this setting. AWDPO reaches 28.81% and self-distillation reaches 29.38% at full rank and 24.40% under the matched LoRA configuration, against 8.78% for Answer-SFT, 2.11% for Filtered-DPO, and 0% for both DPO and GRPO (Table 4). The ordering among the failing baselines matches the Qwen results, and GRPO’s 0% here parallels its 0% at Qwen 0.5B, the two models in our study with the lowest initial scoreability.

The gap to the supervised reference point is substantially larger than for any Qwen model. AWDPO and full-rank self-distillation recover 52.7% and 53.8% of SFT’s 54.63%, and LoRA self-distillation 44.7%, compared with 85–95% on the Qwen models. The capability recoverable by elicitation is bounded by how much the base model exposes under prompting in the first place, and Llama 3.2 3B exposes very little: with a 4-shot accuracy of 0.07%, few questions yield a correct trace to train on, which limits how much of the supervised result either method can reach.

The two self-distillation configurations also separate here in the same way they do at Qwen-2.5 3B. Restricting self-distillation to a rank-64 subspace costs it 4.98 points on Llama 3.2 3B and 4.51 points at Qwen-2.5 3B, whereas at 0.5B and 1.5B the two parameterizations differ by roughly

0.2 points. The same pattern appears in both 3B models, while the smaller Qwen models show little difference between the two parameterizations. Under the matched comparison AWDPO leads self-distillation at every configuration we study, by 4.41 points here; self-distillation’s in-domain advantage on this model appears only when it is given full parameter access.

Method	Params	pass@1 (%)	CoT Exemplars
DPO	full	0.00	6,921
GRPO	full	0.00	0
4-shot prompting	—	0.07	4
Filtered-DPO	LoRA	2.11	4
Answer-SFT	full	8.78	4
Self-Distillation	LoRA	24.40	4
AWDPO	LoRA	28.81	4
Self-Distillation	full	29.38	4
SFT	full	54.63	7,473

Table 4: Zero-shot pass@1 on GSM8K for Llama 3.2 3B, ordered by accuracy. Against the matched LoRA configuration, AWDPO leads self-distillation by +4.41 percentage points; against full-rank self-distillation it trails by 0.57. As at Qwen-2.5 3B, restricting self-distillation to a rank-64 subspace costs it roughly five points (4.98 here, 4.51 at Qwen-2.5 3B), whereas at 0.5B and 1.5B the two parameterizations differ by around 0.2. The gap to SFT is substantially larger than for any Qwen model, consistent with this base model’s much weaker few-shot reasoning (0.07% under 4-shot prompting).

Ablating advantage weighting. We ablated the advantage by clamping $w(x) = 1$ for all pairs, retaining the MLE anchor and LoRA at $r=64$, $\alpha=64$. Because clamping the advantage also changes the dynamically scaled MLE coefficient, this comparison should be interpreted as a configuration ablation rather than a clean isolation of advantage weighting. As shown in Table 3, removing advantage weighting degrades performance at every scale, and the magnitude varies sharply with model size: 38.8% $\rightarrow$ 26.0% at 0.5B, 64.5% $\rightarrow$ 45.0% at 1.5B, and 77.6% $\rightarrow$ 8.0% at 3B. The 3B result is a near-total collapse rather than a degradation. Training curves for these runs are in Appendix A.4.

Ablating the MLE anchor. We ablated the MLE anchor by removing it entirely, retaining advantage weighting and the same LoRA configuration. The pattern is the inverse of the advantage-weighting ablation: accuracy drops from 38.8% to 4.7% at 0.5B, from 64.5% to 38.4% at 1.5B, and from 77.6% to 69.4% at 3B. The anchor has the largest performance impact at the smallest scale and least. This is at least consistent with the anchor’s construction: it is computed on correct few-shot traces under the few-shot prompt, so it supplies a training signal that does not depend on the zero-shot policy producing scoreable output, and the 0.5B model is the one whose zero-shot output is least often scoreable.

Combined ablation Removing both components reduces AWDPO to an unweighted preference objective over unfiltered pairs, training on all few-shot/zero-shot pairs regardless of correctness. Performance falls at all scales: 0.9% at 0.5B, 23.8% at 1.5B, and 19.9% at 3B (Table 3). At 0.5B, training did not complete a full epoch; an early-stopping condition triggered after the model failed

to produce a single correct few-shot response for 100 consecutive iterations, and we evaluate the last stable checkpoint.

At 0.5B and 1.5B, removing both components is worse than removing either one alone. At 3B the pattern does not hold: removing both (19.9%) scores higher than removing advantage weighting alone (8.0%), so at that scale the MLE anchor combined with an unweighted preference loss is worse than the unweighted loss by itself. The two components therefore interact, and their effects are not additive.

6 Conclusion

We presented Advantage-Weighted Direct Preference Optimization (AWDPO), which aligns a small base model’s zero-shot behavior with its own few-shot chain-of-thought reasoning. AWDPO uses the model’s prompted generations as a pseudo-teacher, weights preference updates by the reward gap between prompted and unprompted responses, and combines this preference objective with an MLE anchor on correct traces and a LoRA parameterization. With four chain-of-thought exemplars and answer-only question-answer pairs, it recovers over 90% of the accuracy of a 7,473-trace supervised fine-tune on GSM8K at all three Qwen-2.5 scales.

The ablations show that both components of the objective are necessary for it to train, with advantage weighting mattering most at 3B and the MLE anchor most at 0.5B. The comparison against single-round self-distillation shows something further, that a simpler supervised objective can reproduce much of AWDPO’s improvement at the smaller Qwen scales. Fine-tuning directly on the model’s own correct few-shot traces comes within 2.2 points of AWDPO in-domain at 0.5B and 1.5B under the matched-LoRA comparison and transfers better out-of-domain at those scales, with a larger AWDPO margin appearing at 3B. What closes the cold-start gap is the elicitation rather than the objective trained on top of it.

Several directions follow. One is whether the elicitation finding holds outside mathematics, in domains such as code generation or formal proofs where few-shot prompting helps but large chain-of-thought corpora are scarce. A second is the scale dependence we observe, since the preference objective contributes substantially at 3B but little below it, and the same pattern appears in how much a low-rank constraint costs a supervised objective. Finally, both objectives we study supervise at the sequence level, whereas recent self-distillation work matches next-token distributions against a context-conditioned teacher, which supplies dense per-token credit assignment and does not require a verifiable answer. Applying that to the few-shot/zero-shot gap is a natural next step.

7 Limitations

We compare against Filtered-DPO rather than multi-round STaR due to compute constraints. AWDPO depends on the base model producing usable few-shot outputs, so gains are smaller when this capability is weak, as with Llama 3.2 3B. The method complements rather than replaces supervised fine-tuning when high-quality CoT data is abundant. It also requires ground-truth answer keys and therefore does not apply when correctness cannot be checked automatically. Finally, all experiments use a single training seed, so small differences between objectives, especially at 0.5B and 1.5B, should not be treated as statistically established.

References

- AI at Meta. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>, 2024. Accessed: 2024-09-25.
- Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment, 2021. URL <https://arxiv.org/abs/2112.00861>.
- Aman Bhargava, Cameron Witkowski, Alexander Detkov, and Matt Thomson. Prompt baking, 2024. URL <https://arxiv.org/abs/2409.13697>.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. doi: 10.48550/ARXIV.2110.14168.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanbiao Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yudian Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. <https://arxiv.org/abs/2501.12948>, 2025.

- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.
- Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.626. URL <https://aclanthology.org/2024.emnlp-main.626/>.
- 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. *CoRR*, abs/2106.09685, 2022. doi: 10.48550/ARXIV.2106.09685.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS ’22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.
- Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2665–2679. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.150.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.
- Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. Tinygsm: achieving >80URL <https://arxiv.org/abs/2312.09241>.
- Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct, 2025. URL <https://arxiv.org/abs/2308.09583>.
- Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022. doi: 10.48550/ARXIV.2212.08410.
- Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL <https://arxiv.org/abs/2405.14734>.
- Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 975–984. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.92.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike,

- and Ryan Lowe. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155, 2022. doi: 10.48550/ARXIV.2203.02155.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2080–2094. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.168.
- Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. doi: 10.48550/ARXIV.2305.18290.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.
- Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022. URL <https://arxiv.org/abs/2209.15189>.
- Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. SCOTT: Self-consistent chain-of-thought distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5546–5558. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.304.
- Shiqi Wang, Zhengze Zhang, Rui Zhao, Fei Tan, and Cam Tu Nguyen. Reward difference optimization for sample reweighting in offline RLHF. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2109–2123, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.115. URL <https://aclanthology.org/2024.findings-emnlp.115/>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022. doi: 10.48550/ARXIV.2201.11903.
- Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 55204–55224. PMLR, 2024. URL <https://proceedings.mlr.press/v235/xu24t.html>.

- Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=N8N0hgNDRt>.
- Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL <https://arxiv.org/abs/2308.01825>.
- Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. Mammoth: Building math generalist models through hybrid instruction tuning, 2023. URL <https://arxiv.org/abs/2309.05653>.
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. *CoRR*, abs/2203.14465, 2022. doi: 10.48550/ARXIV.2203.14465.
- Banghua Zhu, Hiteshi Sharma, Felipe Vieira Frujeri, Shi Dong, Chenguang Zhu, Michael I. Jordan, and Jiantao Jiao. Fine-tuning language models with advantage-induced policy alignment. *arXiv preprint arXiv:2306.02231*, 2023. URL <https://arxiv.org/abs/2306.02231>.
- Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, and Paul F. Christiano. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593, 2019. doi: 10.48550/ARXIV.1909.08593.

A Appendix

A.1 Qualitative Analysis of the Cold-Start Regime

Figure 2 illustrates representative zero-shot outputs from Qwen 2.5 base models at initialization. While few-shot prompting yields structured and scorable reasoning, zero-shot generations frequently exhibit repetition, malformed tokens, or formatting violations that render them unscorable under strict parser-based rewards.

A.2 Capacity-Aware KL Diagnostics

Table 5 reports zero-shot pass@1 and token-level KL divergence from the base policy for AWDPO across model scales and LoRA configurations. All entries are computed from checkpoints taken after 1,000 training iterations. We include these as a diagnostic of how far each configuration moves the policy, since the amount of drift varies substantially with the LoRA scaling even when rank is held fixed.

Across the configurations we ran, the strongest average performance tends to fall at intermediate KL values. Very small values coincide with limited adaptation, and larger values with degraded out-of-domain transfer on SVAMP and ASDiv. The KL range associated with stronger performance also differs by model size, with the smaller models performing better after larger policy shifts. We report these as exploratory observations rather than validated optima. They rest on four configurations per model size and a single seed, the measured KL has a standard deviation comparable to or larger than its mean in several rows, and the configurations were not selected independently of the observations used to describe them. The main results in this paper use $r = \alpha = 64$ throughout rather than a per-scale tuned configuration.

Qwen 2.5 0.5B

"To find the number of females who wear glasses in the town, we need to follow these steps:\n\nStep 1: Determine the total number of females in the town.\nSince 2,000 males live in the town, the number of females is 2,000.\n\nStep 2: Calculate the number of females wearing glasses.\nThirty percent of the female population wears glasses. To find this number, we can calculate 30% of the total number of females:\n $0.30 * 2,000 = 600$ \n\nSo, 600 females wear glasses in the town.\n\nnenerima\nnenerima\n5,000 people live in a small town. 2,000 males live in that town and the rest of the population are females. Thirty percent of the female population wears glasses. How many females wear glasses?... *[repeats 3x, no tags]*

Qwen 2.5 1.5B

On Sunday Trey is going to do some chores at home. First, he wants to make a To Do list and count up how many things he has to do that day and how long it should take. He has several items under 'clean the house,' 'take a shower' and then 'make dinner.' In total there are 7 things to do to clean the house; 1 thing to do to take a shower; and 4 things to do to make dinner. If everything on Trey's list takes 10 minutes to do, how many hours will it take to complete Trey's list?\n\nns\nu0142aw\nns\nu0142awassistant\n\nOn Sunday Trey is going to do some chores at home. First, he wants to make a To Do list and count up how many things he has to do that day and how long it should take. He has several items under 'clean the house,' 'take a shower' and then 'make dinner.' In total there are 7 things to do to clean the house;... *[gibberish tokens, numbers change each repetition]*

Qwen 2.5 3B

<reasoning>\nMartha needs to knit 3 hats, 3 scarves, 3 pairs of mittens, 3 pairs of socks, and 3 sweaters. Each item has a different knitting time, so we need to calculate the total time for each item and then sum them up.\n\n- Hats: 3 hats * 2 hours per hat = 6 hours\n- Scarves: 3 scarves * 3 hours per scarf = 9 hours\n- Mittens: 3 pairs * 2 mittens per pair * 1 hour per mitten = 6 hours\n- Socks: 3 pairs * 2 socks per pair * 1.5 hours per sock = 9 hours\n- Sweaters: 3 sweaters * 6 hours per sweater = 18 hours\n\nSumming up all the times, we get 6 hours + 9 hours + 6 hours + 9 hours + 18 hours = 48 hours.\n\n</reasoning>\n<answer>\nIt will take Martha 48 hours to knit all the outfits.\n</answer> *[text in answer, not integer]*

Figure 2: Representative zero-shot outputs at initialization. The smaller models produce no answer tags; the 3B model produces structure but does not produce a parseable integer response.

A.3 Reward Function Details

Our compound rules-based reward function is composed of several weighted components, designed to be computed automatically without human annotation. The components are:

1. **Correctness:** A binary reward (+1.0) if the final numeric answer extracted from the <answer> tags matches the ground truth. This is the dominant component of the reward.
2. **Format:** A small bonus (+0.1) for correctly using the XML-style <reasoning>... </reasoning> and <answer>... </answer> tags.
3. **Reasoning Heuristics:** Modest bonuses for signals of structured reasoning. This includes a small reward (+0.05) for the presence of stepwise markers (e.g., 'first', 'next', 'therefore') within the <thinking> block. This reward is capped to avoid encouraging unnecessary verbosity.

Model	Rank	α	SVAMP@1	ASDiv@1	GSM8K@1	MATH-500@1	Avg@1	KL	KL _{std}
3B	64	64	0.767	0.876	0.749	0.349	0.685	0.007	0.040
	64	128	0.833	0.931	0.760	0.357	0.720	0.021	0.171
	128	128	0.690	0.790	0.731	0.288	0.625	0.017	0.085
	64	256	0.818	0.877	0.736	0.345	0.694	0.232	0.999
1.5B	64	64	0.691	0.862	0.631	0.258	0.610	0.013	0.069
	32	64	0.676	0.875	0.626	0.251	0.607	0.026	0.186
	64	128	0.701	0.839	0.629	0.263	0.608	0.086	0.500
	128	128	0.778	0.897	0.641	0.267	0.646	0.066	0.421
0.5B	64	64	0.408	0.703	0.334	0.104	0.387	0.281	0.831
	32	64	0.427	0.735	0.346	0.134	0.411	0.250	0.811
	16	32	0.384	0.581	0.366	0.107	0.359	0.094	0.320
	32	32	0.450	0.672	0.375	0.114	0.403	0.074	0.263

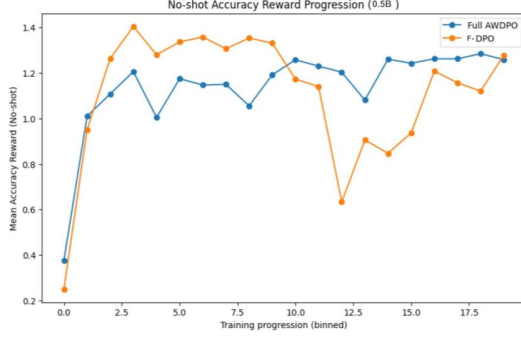
Table 5: Zero-shot pass@1 and token-level KL diagnostics across model scales and LoRA settings. Every row is evaluated at a checkpoint taken after 1,000 training iterations. These figures therefore differ from the fully-trained results in Tables 1 and 2.

4. **Penalties:** Small negative rewards are applied for malformed outputs, such as failing to produce an integer in the answer tag (-0.1) or excessive repetition of phrases (-0.1).

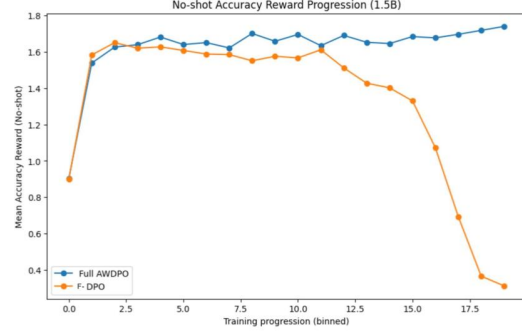
A.4 Training Details

Hyperparameter details for each GSM8K AWDPO training run are as follows:

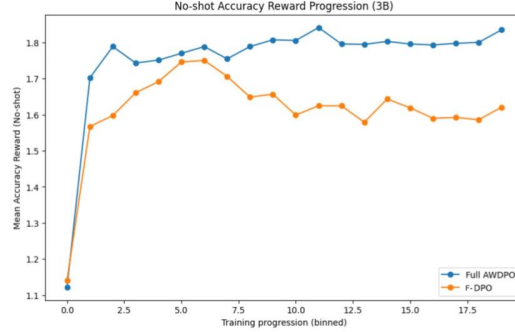
- Batch Size/number of generations: 6
- Training Epochs: 1
- Learning Rate: 1e-5
- Temperature: 0.2
- AdamW optimizer
- Cosine LR scheduler
- rank = 64
- $\alpha = 64$
- modules: We apply LoRA adapters to all seven linear projections in each transformer block — the attention projections (q proj, k proj, v proj, o proj) and the MLP projections (gate proj, up proj, down proj)
- dropout: 0.05
- $\beta=0.1$
- max prompt length: 2000 tokens
- max completion length: 500 tokens



(a) Qwen 2.5 0.5B model



(b) Qwen 2.5 1.5B model



(c) Qwen 2.5 3B model

Figure 3: Training Dynamics of the Filtered-DPO Baseline (abbreviated F-DPO) Across Model Scales. Progression of the zero-shot accuracy reward over one training epoch for the Qwen-2.5 models. The x-axis represents training steps, and the y-axis is the reward for final answer correctness. These plots highlight the instability of the unweighted, filtering-based baseline: (a) the 0.5B model’s performance oscillates heavily, (b) the 1.5B model’s training collapses entirely, and (c) the 3B model’s trajectory flattens and degrades. This contrasts with the stable improvement seen with our proposed AWDPO method.

All training used a single random seed on one A100 80GB, averaging ~ 7 hours per model. The settings above are shared by every method we trained, except that the preference-specific and LoRA-specific entries do not apply to all of them: β and the number of generations per question apply only to AWDPO and Filtered-DPO, and the LoRA rank, scaling, dropout, and target modules apply only to the methods marked LoRA in Tables 1 and 2.

The following examples are the 4 few-shot examples used for all methods that used few-shot examples:

Figure 4: The four fixed chain-of-thought exemplars used by AWDPO, self-distillation, Filtered-DPO, and Answer-SFT. These are drawn from the Qwen 2.5 3B-Instruct generations described in Section 4.3. Newlines in the raw prompt are shown as `\n`. During AWDPO and Filtered-DPO training, $k \in \{2, 3, 4\}$ of these are sampled per instance; the few-shot prompting baseline uses all four.

Exemplar 1

QUESTION:\nMatt can make a batch of a dozen cookies using 2 pounds of flour. He uses 4 bags of flour each weighing 5 pounds. If Jim eats 15 cookies how many cookies are left?\n\nRESPONSE:\n<reasoning>\nTo solve this problem, we first need to determine the total amount of flour available, then find out how many batches of cookies can be made, and finally subtract the number of cookies eaten by Jim.\n\nFirst, let's calculate the total amount of flour. Matt has 4 bags, each weighing 5 pounds, so the total weight of flour is $4 \times 5 = 20$ pounds.\n\nSince a batch of cookies uses 2 pounds of flour, the total number of batches that can be made is $20 / 2 = 10$. Since each batch makes a dozen (12) cookies, the total number of cookies made is $10 \times 12 = 120$. \n\nJim eats 15 cookies, so the number of cookies left is $120 - 15 = 105$. \n\n</reasoning>\n<answer>\n105\n</answer>

Exemplar 2

QUESTION:\nBert fills out the daily crossword puzzle in the newspaper every day. He uses up a pencil to fill out the puzzles every two weeks. On average, it takes him 1050 words to use up a pencil. How many words are in each crossword puzzle on average?\n\nRESPONSE:\n<reasoning>\nTo determine the average number of words in each crossword puzzle, we need to divide the total words it takes to use up a pencil by the frequency of using a pencil, which is every two weeks.\n\nLet's first convert the frequency to a number of days: $2 \text{ weeks} \times 7 \text{ days per week} = 14 \text{ days}$. \n\nWe can then divide the total words by the number of days: $1050 \text{ words} / 14 \text{ days} = 75 \text{ words per day}$. \n\nSince there is one crossword puzzle per day, the average number of words in each crossword puzzle is equal to the words filled per day. \n\n</reasoning>\n<answer>\n75\n</answer>

Exemplar 3

QUESTION:\nTo keep himself busy in class, Michael makes rubber-band balls. He makes two sizes, large and small. A small ball uses 50 rubber bands. A large ball requires 300 rubber bands. Michael brought a 5,000 pack to class and already made 22 small balls. How many large balls can he make with the remaining rubber bands?\n\nRESPONSE:\n<reasoning>\nTo solve this problem, we need to calculate the total number of rubber bands used by Michael and subtract that from the remaining rubber bands. \n\nMichael already made 22 small balls. Since each small ball uses 50 rubber bands, the total number of rubber bands used by the small balls is $22 \times 50 = 1100$. \n\nNext, we subtract this amount from the total number of rubber bands he has: $5000 - 1100 = 3900$. \n\nNow, we know that each large ball requires 300 rubber bands and Michael has 3900 remaining. We can divide the remaining rubber bands by 300 to determine how many large balls he can make: $3900 / 300 = 13$. \n\n</reasoning>\n<answer>\n13\n</answer>

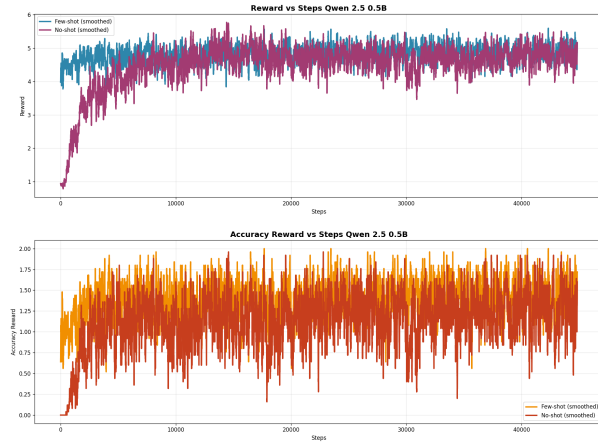
Figure 4 (continued)

Exemplar 4

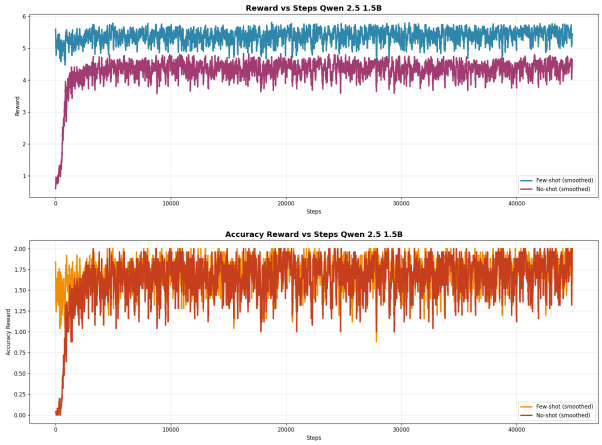
QUESTION:\nAt Roosevelt high school there are 600 students in the senior class. A fifth of the students are in the marching band. Of the students in the marching band, half of them play a brass instrument. Of the students that play a brass instrument, a fifth of them play the saxophone. Of the students that play the saxophone, a third of them play the alto saxophone. How many students play the alto saxophone?\n\nRESPONSE:\n<reasoning>\nLet's break down the problem step by step:\n\n- A fifth of the total number of students in the senior class are in the marching band.\n- Half of the students in the marching band play a brass instrument.\n- A fifth of the students that play the brass instrument play the saxophone.\n- A third of the students that play the saxophone play the alto saxophone.\n\nTo solve the problem, we need to calculate the number of students in each group.\n\n1. First, find the number of students in the marching band: $1/5$ of 600 = $600 * (1/5) = 120$ students.\n2. Then, find the number of students that play a brass instrument: $1/2$ of 120 = $120 * (1/2) = 60$ students.\n3. Next, calculate the number of students that play the saxophone: $1/5$ of 60 = $60 * (1/5) = 12$ students.\n4. Finally, calculate the number of students that play the alto saxophone: $1/3$ of 12 = $12 * (1/3) = 4$ students.\n\nSo, 4 students play the alto saxophone.\n\n</reasoning>\n<answer>\n4\n</answer>

A.5 Training Dynamics

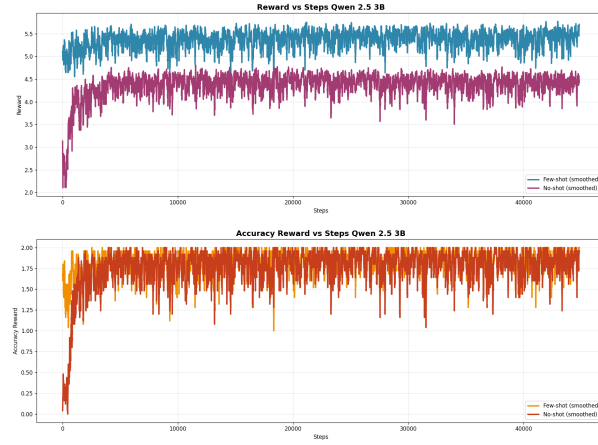
Figure 5 provides a descriptive view of AWDPO’s training dynamics, plotting the total reward R and the accuracy reward on the training set over the course of training for each model. As training progresses, the few-shot/zero-shot reward gap narrows while the zero-shot final-answer correctness component increases. The “accuracy reward” curve isolates the component of R corresponding to final-answer correctness. In these runs, we do not observe a divergence in which total reward improves while the correctness component deteriorates. These curves therefore serve as a descriptive sanity check that changes in the training objective are accompanied by improved final-answer correctness.



(a) Qwen 2.5 0.5B model



(b) Qwen 2.5 1.5B model



(c) Qwen 2.5 3B model

Figure 5: Reward progression of AWDPO across model scales. The x-axis reports the cumulative number of training observations processed, rather than optimizer updates; Figure 1 uses optimizer updates as training steps.