# LEARNING STRATIFIED POLICIES FOR INFERENCE-TIME OBJECTIVES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Inference-time scaling improves the performance of large language models by sampling multiple responses and evaluating their collective utility. However, many existing approaches obtain these samples by repeatedly sampling from a single policy. In this work, we study if the sample budget at inference time can instead be allocated across a population of policies. We introduce stratified policies, where each of $k$ learned policies contributes one response to a $k$-sample inference-time objective. We realize this policy population using separate LoRA adapters on a shared language model backbone and optimize it for pass@$k$. We theoretically motivate the approach by analyzing the linear setting; a population of LoRAs can achieve zero best-of-$k$ approximation error in settings where every single LoRA has strictly positive prediction error. Empirically, we demonstrate that stratified policies consistently improve pass@$k$ over a single-policy baseline across multiple reasoning tasks. Overall, our results highlight how distributing samples across complementary policies can be a more effective use of sample budget than repeated i.i.d. sampling from a single policy at inference time.

## 1 INTRODUCTION

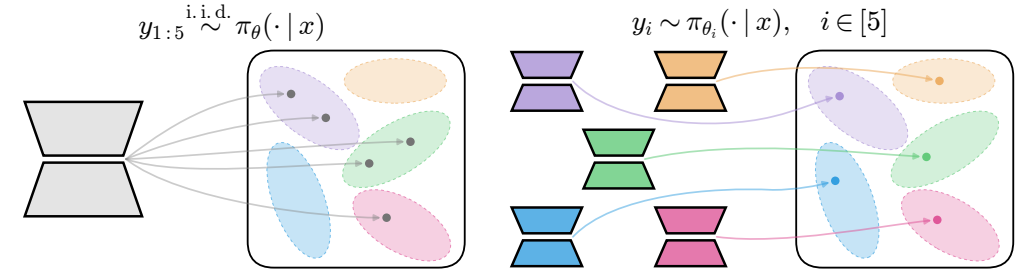
Inference-time computation is an effective way to improve the performance of large language models. Instead of relying on a single response, inference-time procedures sample multiple responses and evaluate their collective utility, such as best-of-$k$ (Huang et al., 2025), reranking (Cobbe et al., 2021), or majority voting (Wang et al., 2023). Inspired by this, recent works explicitly optimize language models for the inference-time procedure used at deployment (Tang et al., 2025; Chow et al., 2025; Walder & Karkhanis, 2025). This is different from standard post-training methods that optimize the quality of individual responses (Ouyang et al., 2022; Rafailov et al., 2023).

These approaches share a common sampling scheme: all $k$ responses are sampled independently from a single policy. Although this is natural, it imposes a particular allocation of the inference-time budget: every response must come from the same distribution. Interestingly, even when this distribution assigns probability mass to several useful solution modes, i.i.d. sampling can repeatedly sample from the same modes and miss others. For example, consider a policy whose response distribution places all its probability mass on five distinct solution modes. From the AM-GM inequality, five i.i.d. samples can cover all five modes with probability at most $5!/5^5 \approx 3.84\%$. This upper bound is achieved when the policy assigns equal mass to each mode. Therefore, even a perfectly uniform policy may fail to realize this diversity in a finite set of responses.

In this work, we formulate inference-time optimization through a population of policies: the sample budget at inference time is treated as an allocation across a policy population. More concretely, given a budget of $k$ responses, we jointly learn $k$ policies and sample exactly one response from each. We refer to this as stratified sampling, inspired by classical stratified sampling, which partitions a population into strata and samples separately from each stratum (Cochran, 1977). Unlike classical stratification, the learned policies need not partition the response space or have disjoint support. The standard single-policy formulation is recovered when all policies in the population are constrained to be identical. When this constraint is relaxed, the population can "hedge its bets" by learning policies that specialize in ways that improve the utility of the full set of responses. For pass@$k$, this can encourage policy complementarity without any explicit diversity objective (see Fig. 1).



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.




(a) i.i.d. sampling (b) stratified sampling

Figure 1: An illustrative example of stratification. With a fixed budget of five responses, i.i.d. sampling from a single policy may repeatedly sample the same solution modes (left). For five modes, the probability of covering all modes is at most $5!/5^5 \approx 3.84\%$, which is attained when the policy assigns equal probability to each mode. In contrast, stratified sampling allocates one response to each policy in a learned population, which allows specialized policies to produce complementary responses (right); under perfect specialization, all five modes are covered with probability one.

We realize this policy population efficiently using a shared (frozen) language model backbone with a separate low-rank adaptation (LoRA) (Hu et al., 2022) adapter for each policy. We focus on pass@$k$ as the utility function, where the objective is to maximize the probability that at least one response from the set is correct. Building on policy-gradient optimization for inference-time objectives, we train the population using Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

Theoretically, we show that this population structure has an expressivity advantage in the linear setting. Specifically, when the target requires applying different linear maps over different regions of the input space, a LoRA population can assign a separate adapter to each region, but a single linear map must approximate all regions simultaneously. With sufficient rank, the policy population can achieve zero best-of-$k$ approximation error, whereas every single linear map incurs strictly positive error whenever the maps in different regions differ.

We evaluate stratified policies on a broad range of reasoning tasks spanning valid arithmetic expression construction, logical deduction, constraint satisfaction, and mathematical word problems. Across all tasks, stratification improves pass@5 over a single-policy baseline under the same inference-time response budget. On SUDOKU, specifically, the pass@5 score increases from $74.2\%$ to $80.3\%$. Furthermore, when the rank of the single LoRA adapter is increased $5\times$ to match the population, performance remains essentially the same, at $74.2\%$, which suggests that the improvement is not explained by additional parameters alone. The learned population of policies also exhibits complementary behavior: for instance, on SUDOKU, only $6.0\%$ of examples are solved by all five adapters, whereas $44.0\%$ are solved by exactly one. Together, these results demonstrate that population size provides an orthogonal scaling axis for inference-time objectives: instead of adding parameters to a single policy and repeatedly sampling from it, parameters can be distributed across policies that contribute complementary responses.

To summarize, our work makes the following contributions:

I. We introduce stratified policies for inference-time objectives, which replace $k$ i.i.d. samples from a single policy with one sample from each of $k$ learned policies (Section 3.1). For pass@$k$, the objective recovers the binary-reward special case of Grinsztajn et al. (2023).

II. We provide a practical realization of this population using multiple LoRA adapters on a shared language model backbone, and train the resulting population using a group-mean pass@$k$ advantage within GRPO (Sections 3.2 and 3.3).

III. Theoretically, we prove that in a linear setting, stratification is strictly more expressive: with sufficient rank, a LoRA population can achieve zero best-of-$k$ approximation error while any single linear map incurs strictly positive error (Section 3.4).

IV. We empirically demonstrate consistent pass@5 improvements across multiple reasoning tasks, along with the emergence of complementary specialization (Section 4).

We emphasize that our goal is not to design a state-of-the-art method for pass@$k$. Instead, we show that allocating the sample budget across a learned population of policies, rather than repeatedly sampling a single policy, is a complementary axis to single-policy pass@$k$ training.

## 2 Problem Setup

**Language Models.** We view a language model as a policy $\pi_\theta$, parameterized by $\theta$, that generates a response $y$ conditioned on a prompt $x$. Let $\rho$ denote the prompt distribution and $r(x, y)$ the reward function used to evaluate the response. A standard post-training objective maximizes the expected reward of a single response:

$$\max_\theta \ \mathbb{E}_{x\sim\rho,\ y\sim\pi_\theta(\cdot|x)} \left[r(x, y)\right].$$

**Inference-Time Objectives.** At inference time, performance may be determined by the collective utility of a set of responses, instead of the quality of individual responses. Let $F(x, y_1, \ldots, y_k)$ denote the utility of $k$ responses under a given inference-time procedure. This is captured by the following training objective:

$$\max_\theta \ \mathbb{E}_{x\sim\rho,\ y_{1:k}\sim\pi_\theta(\cdot|x)^{\otimes k}} \left[F(x, y_1, \ldots, y_k)\right],$$

where the responses are sampled independently conditioned on $x$.

For instance, the utility function for best-of-$k$ is $F(x, y_1, \ldots, y_k) = \max_{i\in\{1,\ldots,k\}} r(x, y_i)$. When the reward is binary (0 for failure and 1 for success), the expected best-of-$k$ utility exactly equals pass@$k$, which is the probability that at least one of the $k$ responses is correct.

**Objective.** Given an inference-time utility $F$ and a sample budget $k$, our goal is to post-train a language model to maximize the expected utility of the set of responses. While our formulation captures general $k$-response utilities, we focus on pass@$k$ for the rest of the paper.

## 3 Learning Stratified Policies

In this section, we present a framework for learning a population of policies for inference-time objectives. First, we formulate inference-time optimization through stratified sampling from a population of policies (Section 3.1). We then describe a practical parameterization of this population using multiple LoRA adapters on a shared language model (Section 3.2). Finally, we instantiate our framework for pass@$k$ optimization (Section 3.3), and theoretically motivate it (Section 3.4).

### 3.1 Inference-Time Objectives through Stratified Sampling

As discussed in Section 2, the single-policy formulation allocates a budget of $k$ responses by repeatedly sampling from a single policy $\pi_\theta$. However, repeatedly sampling solutions from the same policy may produce similar solutions, which could potentially limit their collective utility.

With this lens, we introduce a population of policies $\Pi_\Theta = (\pi_{\theta_1}, \ldots, \pi_{\theta_k})$, where $\Theta = (\theta_1, \ldots, \theta_k)$. Instead of obtaining all $k$ responses from the same policy, we sample exactly one response from each policy. This gives us the following objective:

$$\max_\Theta \ \mathbb{E}_{x\sim\rho,\ y_{1:k}\sim\bigotimes_{i=1}^{k}\pi_{\theta_i}(\cdot|x)} \left[F(x, y_1, \ldots, y_k)\right], \tag{1}$$

where responses are sampled independently conditioned on $x$.

We refer to this (budget-)allocation of one sample per policy as stratified sampling. Note that the policies need not have disjoint support. Instead, each policy defines a response distribution that contributes one sample to the set. The single-policy objective can be recovered as the special case when $\theta_1 = \cdots = \theta_k$. Therefore, our formulation generalizes the single-policy setting.

Intuitively, optimizing a policy population allows each policy to contribute according to the utility of the full response set. When the objective rewards complementary responses (e.g., best-of-$k$), each policy can specialize in different subsets of prompts or different ways to solve them. Such specialization can emerge through the objective and learning dynamics, without a predefined partitioning of the prompt or response space. Although responses are sampled independently, the policies are trained jointly through the utility of the full response set. Therefore, each policy's learning signal depends on the responses generated by the other population members.

### 3.2 A Practical Implementation with Multiple LoRA Adapters

With the stratified objective defined, we now describe a practical parameterization of the policy population. A straightforward approach is to maintain $k$ independent language models, one for each policy. However, this requires storing and training a separate set of parameters for each policy in the population. Instead, we utilize a frozen language model backbone and parameterize each policy using a separate low-rank adaptation (LoRA) module (Hu et al., 2022).

Let $W_\ell$ denote a frozen weight matrix, where $\ell$ indexes the matrices to which LoRA is applied. For policy $i$, the corresponding adapted matrix is:

$$W_\ell + \Delta W_\ell^{(i)}, \qquad \operatorname{rank}\left(\Delta W_\ell^{(i)}\right) \le R, \tag{2}$$

where $R$ denotes the LoRA rank. In practice, the update is parameterized as:

$$\Delta W_\ell^{(i)} = \frac{\alpha}{R} B_\ell^{(i)} A_\ell^{(i)}, \tag{3}$$

where $A_\ell^{(i)}$ and $B_\ell^{(i)}$ are the LoRA factors and $\alpha$ is a scaling parameter (set to $R$ in our experiments).

All policies share the same frozen matrices $W_\ell$ but use separate LoRA adapters. Given a prompt, we generate one response per adapter and evaluate the resulting set according to utility $F$. This parameterization realizes a population without maintaining $k$ full model copies.

### 3.3 Instantiation for Pass@$k$

We now instantiate our framework for the best-of-$k$ utility: $F(x, y_1, \ldots, y_k) = \max_i r(x, y_i)$. For binary correctness rewards, the objective in Eq. 1 is equal to the probability that at least one policy in the population generates a correct response.

**Learning with Leave-One-Out Baseline.** We build on the inference-time policy gradient formulation of Tang et al. (2025). When using a leave-one-out (LOO), response $y_i$ is assigned an advantage equal to its marginal contribution to the maximum reward:

$$A_i^{\mathrm{LOO}} = \max_{j=1,\ldots,k} r_j - \max_{j \neq i} r_j, \tag{4}$$

where $r_i = r(x, y_i)$ and $k \ge 2$. The advantage is non-zero only for the best response, and is effectively equal to the difference between the highest and second-highest rewards.

Policy gradients for our population of policies can be constructed similarly since responses are sampled independently conditioned on the prompt. More concretely, for policy $\pi_{\theta_i}$, the (unclipped) policy gradient is given by:

$$\nabla_{\theta_i} J_{\max}(\Theta) = \mathbb{E}\left[A_i^{\mathrm{LOO}} \nabla_{\theta_i} \log \pi_{\theta_i}(y_i \mid x)\right], \tag{5}$$

where $J_{\max}$ denotes Eq. 1 with the maximum-reward utility. The baseline $\max_{j \neq i} r_j$ does not depend on $y_i$. Consequently, subtracting it does not change the policy gradient. In practice, we use this advantage construction within Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

**Denser Learning Signal via Group-Mean Baseline.** When rewards are binary, the LOO advantage is zero whenever multiple responses in the group are successful, even if other responses fail. Thus, these groups provide no learning signal even if there is room for improvement. Motivated by this, we instead use a group-mean baseline for a denser learning signal. More concretely, we compute group mean $\bar{r} = \frac{1}{k}\sum_{j=1}^{k} r_j$, and sample the winner $i^\star$ uniformly at random from $\arg\max_{i \in \{1,\ldots,k\}} r_i$ to randomly break ties. Response $y_i$ is then assigned advantage:

$$A_i = \begin{cases} r_i - \bar{r}, & \text{if } i = i^\star, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

For binary rewards, if $m$ of the $k$ responses are correct, the selected winner receives advantage $1 - m/k$. Thus, every group containing both correct and incorrect responses produces a non-zero update, whereas groups that are all-correct or all-incorrect produce no update. Randomly breaking ties prevents favoring a particular policy in the population. Appendix G shows that, on SUDOKU, group-mean improves population performance, and stratification helps under both estimators.

Unlike LOO, this modification does not give an unbiased estimate of the best-of-$k$ policy gradient.

### 3.4 Theoretical Analysis

We theoretically demonstrate that a stratified population of policies can be more expressive than a single policy for best-of-$K$ evaluation. More concretely, we consider a linear setting with a frozen base model $Wx$, where $W \in \mathbb{R}^{d\times d}$. Inputs lie in the unit ball $\mathbb{B}_2^d := \{x \in \mathbb{R}^d : \|x\|_2 \le 1\}$, which is partitioned into regions $S_1, \ldots, S_K$. The target function $f^\star$ is piecewise linear on the partition:

$$f^\star(x) = T_j x, \qquad \forall x \in S_j,$$

where $T_j \in \mathbb{R}^{d\times d}$ is the target map for region $S_j$.

Policy $i$ in the population updates the base model with a rank-$r$ LoRA adapter $\Delta_i$, which gives us the predictor $(W+\Delta_i)x$. Evaluation is best-of-$K$: the population error at an input $x$ is determined by the best prediction among its member policies.

Intuitively, a rank-$r$ LoRA population can assign each adapter to a regional correction $T_j - W$. With sufficient rank, this achieves zero error. In contrast, a single linear predictor has to approximate all maps simultaneously. Fig. 2 illustrates this gap in a one-dimensional example with $K = 5$.

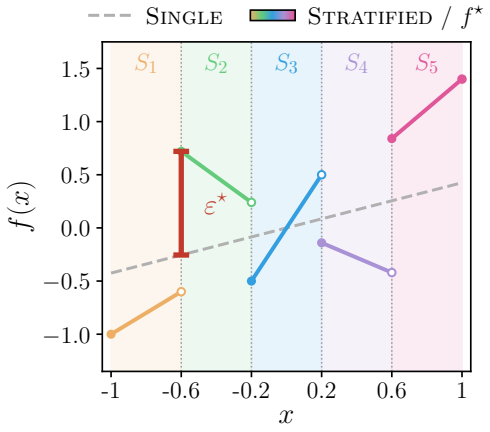



Figure 2: An illustration of Theorem 1. Colored segments represent the regional target maps $f^\star(x) = T_j x$. A rank-1 population fits every region exactly, whereas the best single linear map $Bx$ has an irreducible gap $\varepsilon^\star > 0$.

**Definition 1** (Prediction errors). *For a fixed population of adapters $\Delta_{1:K}$, we define the best-of-$K$ prediction error at input $x$ as*

$$\varepsilon_{\text{pop}}(x; \Delta_{1:K}) := \min_{i\in[K]} \|(W + \Delta_i)x - f^\star(x)\|_2.$$

*For a single linear predictor $B$, we define*

$$\varepsilon_{\text{single}}(x; B) := \|Bx - f^\star(x)\|_2.$$

While the population is fixed across inputs, the minimizing policy in $\varepsilon_{\text{pop}}$ may vary with $x$. Thus, to make accurate best-of-$K$ predictions, at least one policy in the population must predict well at each input. This does not require any individual policy to approximate the target over the entire domain $\mathbb{B}_2^d$. Note that the minimum represents best-of-$K$ evaluation; we are not learning a routing rule.

We use $\sigma_\ell(A)$ to denote the $\ell$-th largest singular value of a matrix $A$. When $\ell > \text{rank}(A)$, $\sigma_\ell(A)$ is defined as zero.

We build on the expressivity result in Zeng & Lee (2024) and formalize this separation below; the proof is provided in Appendix B.

**Theorem 1** (Expressive power of LoRA populations for best-of-$K$). *Suppose each region $S_j$ contains a closed Euclidean ball of radius $\rho > 0$. Then, we have the following minimax guarantee:*

$$\min_{\substack{\Delta_{1:K} \\ \text{rank}(\Delta_i)\le r,\ \forall i\in[K]}} \max_{x\in\mathbb{B}_2^d} \varepsilon_{\text{pop}}(x; \Delta_{1:K}) \le \max_{j\in[K]} \sigma_{r+1}(T_j - W),$$

*whereas*

$$\min_{B\in\mathbb{R}^{d\times d}} \max_{x\in\mathbb{B}_2^d} \varepsilon_{\text{single}}(x; B) \ge \frac{\rho}{2} \max_{i,j\in[K]} \|T_i - T_j\|_{\text{op}}.$$

*Consequently, if $r \ge \max_j \text{rank}(T_j - W)$ and the maps $\{T_j\}_{j=1}^K$ are not all identical, then a LoRA population can achieve zero error, while a single linear map always has strictly positive error.*

It is important to note that we cannot eliminate this separation by simply increasing the LoRA rank, since the lower bound on single-predictor error holds regardless of rank.

While this analysis does not provide any guarantees for nonlinear language models or for learned population of policies, it motivates our use of multiple adapters for pass@$k$.

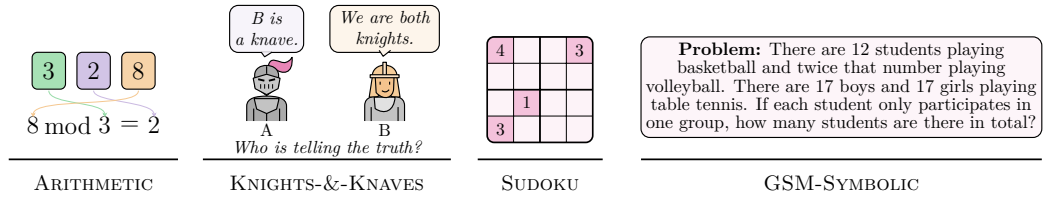



Figure 3: We evaluate our method on a diverse set of reasoning tasks. These span constructing valid arithmetic expressions, logical deduction, constraint satisfaction, and mathematical word problems.

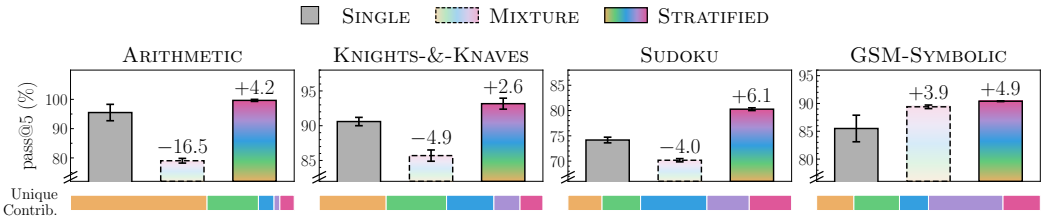



Figure 4: Pass@5 accuracy of a single policy (SINGLE) versus our stratified population of five specialized policies (STRATIFIED) across reasoning tasks. MIXTURE (dashed) samples from the same five policies i.i.d. instead of once from each. Error bars denote standard error across three seeds. Each panel's $y$-axis is truncated (kink mark) to make the gap visible. The (renormalized) horizontal bar below each panel shows how much each policy contributes to the population's unique-only solutions, i.e., problems solved by exactly one policy. STRATIFIED outperforms SINGLE on pass@5 across all four tasks, and each policy member contributes unique solutions to the population.

## 4 EXPERIMENTAL EVALUATION

In this section, we empirically evaluate stratified policies to address the following research questions: (i) Does sampling from a stratified policy population at inference time improve pass@$k$ over repeatedly sampling from a single policy? (ii) Do the learned policies develop complementary behavior, i.e., solve different subsets of problems? (iii) Can the gains from policy stratification be explained simply by additional trainable parameters, or does it depend on maintaining a policy population?

We begin by describing the benchmark tasks and experimental setup. Next, we present the main performance results, followed by analyses of policy complementarity and ablations.

### 4.1 EXPERIMENTAL SETUP

**Tasks.** We evaluate on four reasoning tasks (Fig. 3): ARITHMETIC, introduced in Shah et al. (2026), in which the model is given three unordered digits and must form a valid expression that relates them; KNIGHTS-&-KNAVES, a challenging logical deduction task used to evaluate LLM reasoning and generalization (Mondorf & Plank, 2024); SUDOKU, a constraint-satisfaction task on which LLMs continue to struggle (Seely et al., 2025); and GSM-SYMBOLIC, a procedurally generated variant of GSM8K designed to test mathematical reasoning more reliably, on which many models perform worse than on the original GSM8K problems (Mirzadeh et al., 2025). We use the REASONING GYM implementations (Stojanovski et al., 2026), which is a collection of reasoning environments built to standardize the training and evaluation of reasoning models. See Appendix C for more details.

**Training.** For ARITHMETIC, we train a `TinyGPT` causal transformer. For KNIGHTS-&-KNAVES and SUDOKU, we fine-tune Qwen2.5-1.5B-Instruct (Yang et al., 2024), while for GSM-SYMBOLIC, we fine-tune Llama-3.1-8B-Instruct (Grattafiori et al., 2024). We use rank-32 LoRA adapters and train with GRPO without a KL penalty, using the advantage defined in Eq. 6. The stratified population (STRATIFIED) maintains 5 policies and samples one response from each, whereas the single-policy baseline (SINGLE) trains one policy using the same five-sample budget and advantage, which is similar to Tang et al. (2025) without stratification. Both of these methods assign nonzero advantage to at most one response. So only one needs a gradient-tracked forward and backward pass. Therefore, both require the same order of training compute. See Appendix D for more details.

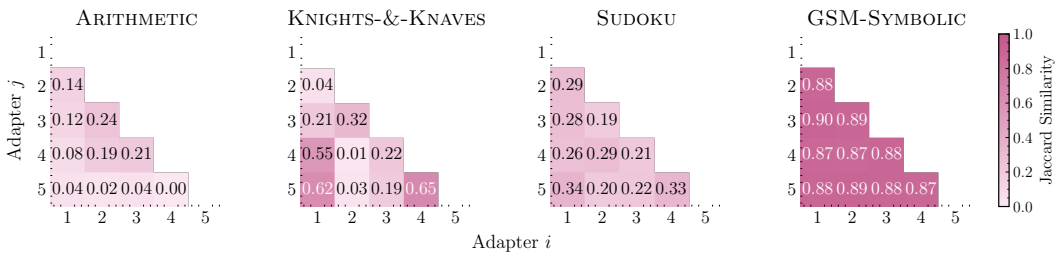



Figure 5: Pairwise Jaccard similarity between the sets of problems solved by each policy. Lower values indicate that a pair of policies solve mostly disjoint subsets of problems; a value of 1 indicates total overlap. ARITHMETIC, KNIGHTS-&-KNAVES, and SUDOKU show low overlap; the values for GSM-SYMBOLIC remain $\leq 0.9$, which still shows non-redundancy in this harder setting.

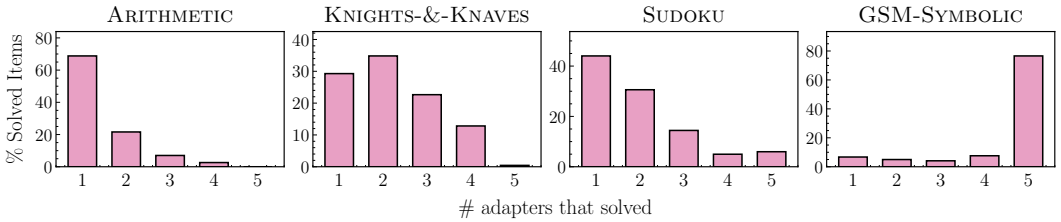



Figure 6: Distribution, over problems solved by at least one policy, of how many of the five policies solve each problem under greedy decoding. For ARITHMETIC, KNIGHTS-&-KNAVES, and SUDOKU, probability mass is concentrated at low counts; this indicates that most solved problems are solved by only one or two policies (not the full population). For GSM-SYMBOLIC, there is substantial mass on counts below 5; this shows that not all problems are solved by all 5 policies.

**Evaluation.** We report pass@5 under an inference-time budget of five samples. Results are averaged over three random seeds, and error bars denote standard error. To check whether sampling exactly one response from each adapter is essential, we also compare against MIXTURE. It uses the same trained adapters, but samples an adapter uniformly at random for each of the five responses, which misses several adapters with high probability (see Fig. 1). See Appendix E for more details.

### 4.2 RESULTS

**Stratification improves pass@5.** Across all four tasks, STRATIFIED improves pass@5 over SINGLE under the same inference-time response budget (see Fig. 4). If we view it as reduction in the gap to optimality (100% pass@5), STRATIFIED reduces the gap by 92% on ARITHMETIC, and between 24 and 34% on the other three tasks. MIXTURE is worse than STRATIFIED on every task and worse than SINGLE on three out of four, which suggests that all adapters are needed to improve pass@5. On pass@1 (see Appendix F), however, SINGLE outperforms STRATIFIED on most tasks. This trade-off between pass@1 and pass@5 is not surprising because we are optimizing for pass@5 directly (Barakat et al., 2026), not any individual policy's pass@1.

**Policies make unique contributions.** The gains in pass@5 from stratification suggest that the learned policies are not redundant copies of one another. Fig. 4 additionally shows the (normalized) share of each policy's unique-only contribution per task under greedy decoding. Across all four tasks, each policy in the population solves problems that cannot be solved by the rest of the population. Therefore, the entire population (and not a single policy) gives complementary coverage.

**Policies solve different problem sets.** We analyze if different policies solve different subsets of problems. Fig. 5 shows pairwise Jaccard similarity (defined for sets $A$ and $B$ as $|A \cap B|/|A \cup B|$) between the set of items solved by the learned policies under greedy decoding. The overlap is lowest on ARITHMETIC and KNIGHTS-&-KNAVES (several pairs with similarity below 0.1), fairly low on SUDOKU, and higher (though still $\leq 0.9$) on GSM-SYMBOLIC. Interestingly, KNIGHTS-&-KNAVES shows a bimodal pattern, where some pairs are almost identical and others complementary.

**Most solved problems are solved by only one or two policies.** The distribution over how many of the five policies solve a given problem (see Fig. 6) is concentrated at low counts for ARITHMETIC, KNIGHTS-&-KNAVES, and SUDOKU. This indicates that most solved problems are not solved by the entire population, but only by one or two policies. For GSM-SYMBOLIC instead, although there

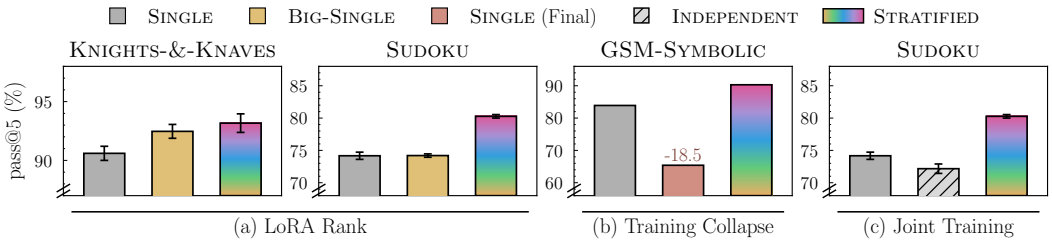



Figure 7: (a) Scaling the rank of SINGLE by 5× (BIG-SINGLE) to match STRATIFIED's total trainable parameters still underperforms it. (b) SINGLE is unstable on GSM-SYMBOLIC; the final checkpoint, SINGLE (Final), achieves a substantially lower score than the pre-collapse checkpoint used elsewhere. STRATIFIED does not exhibit this collapse on any seed. (c) An independently trained ensemble (INDEPENDENT) with matched total training compute underperforms STRATIFIED.

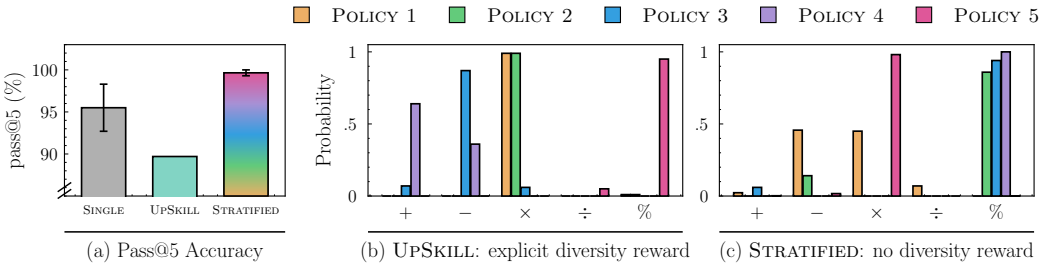



Figure 8: Comparison with UPSKILL on ARITHMETIC. (a) STRATIFIED outperforms UPSKILL, which uses an explicit diversity reward. (b, c) Operator distribution of each policy. Stratified policies achieve a comparable level of diversity to UPSKILL: different policies focus on different operators.

is substantial mass at count 5, a non-trivial fraction of problems are still solved by fewer than all five policies. This shows that part of the gain in pass@5 comes from complementary coverage.

**Stratified policies provide a new scaling axis.** To test whether the gains from STRATIFIED come from more trainable parameters, we compare it against BIG-SINGLE, which increases the LoRA rank of SINGLE by 5× to match the total number of trainable parameters in STRATIFIED. Fig. 7 (a) shows that BIG-SINGLE improves pass@5 from 90.6 to 92.5 on KNIGHTS-&-KNAVES, but achieves no gains on SUDOKU. In both cases, STRATIFIED still performs better. Therefore, the improvement from stratification cannot be explained by additional parameters alone. This shows that population size could be an orthogonal scaling axis to simply increasing the rank of a single adapter. An independently trained ensemble (INDEPENDENT) of five SINGLE runs with matched total training compute is also worse on SUDOKU (Fig. 7 (c)), which shows that the gains depend on joint training.

**Stratified policies can be robust to training collapse.** We found SINGLE to be unstable late in training on GSM-SYMBOLIC on 2 out of 3 seeds. For instance, the final checkpoint for one of the seeds achieves only 65.4 pass@5, while the pre-collapse checkpoint used elsewhere in the paper achieves 83.9 (Fig. 7 (b) shows one such seed). This corresponds to a drop of 18.5 points. In contrast, STRATIFIED was stable across all three seeds. This shows that stratified learning can be more stable than optimizing a single policy for pass@5.

**Stratified policies specialize without an explicit diversity reward.** UPSKILL (Shah et al., 2026) encourages diverse responses using an explicit mutual-information diversity reward. STRATIFIED instead learns to improve collective population performance using the pass@5 advantage defined in Eq. 6, with diversity useful insofar as it improves this objective. We additionally compare against UPSKILL on ARITHMETIC, the benchmark introduced in their paper. Interestingly, as can be seen in Fig. 8 (a), even SINGLE achieves higher pass@5 than UPSKILL on this task, despite using no explicit diversity reward. Importantly, not optimizing for diversity does not prevent it from emerging. Fig. 8 (b, c) shows that the policies learned by STRATIFIED develop different operator preferences despite having no explicit diversity reward. Although three of the five policies primarily use the modulo operator, they are still complementary: among correct responses using the % operator, their pairwise Jaccard similarity ranges from 0.12 to 0.23, and 32.8% of the full set of evaluation problems are solved using % by exactly one of these three policies.

## 5 Related Work

**Optimizing Inference-Time Objectives.** Sampling multiple responses can improve the performance of a language model by increasing the probability of finding a correct response (Brown et al., 2024). Methods such as verifiers and self-consistency then use these multiple responses to produce the final prediction (Cobbe et al., 2021; Wang et al., 2023). This motivates training for the utility of a response set instead of a single response. In line with this, recent work directly optimizes for the inference-time procedure used at deployment. Prior work in program synthesis already considered maximizing the expected reward of the best candidate program among multiple samples (Bunel et al., 2018). For language models, inference-aware fine-tuning has been studied for best-of-$N$ sampling (Chow et al., 2025). Tang et al. (2025) derive policy-gradient estimators for inference-time objectives such as pass@$k$ and majority voting. Walder & Karkhanis (2025) propose a training method that can utilize more training samples than the inference-time budget. Chen et al. (2025) also develop training advantages that are specific to pass@$k$ for reasoning models, but their group-relative estimator requires a training-time sample budget $N_{\text{rollout}}$ larger than the eventual pass@$k$ target $k$. This is different from the setting studied in our paper, where $N_{\text{rollout}} = k$, so we do not include this method as a baseline. Thrampoulidis et al. (2026) interpret pass@$k$ advantage shaping as optimization of surrogate reward objectives. Takashiro et al. (2026) study baselines for max@$k$ policy gradients and introduce a leave-two-out baseline for variance reduction. Bahlous-Boldi et al. (2026) optimize sets of responses using randomized scalarizations of vector-valued rewards. These approaches optimize a single policy from which a set of responses is sampled. In contrast, our work jointly optimizes multiple policies and allocates one response to each.

**Learning Complementary Populations.** Stochastic multiple choice learning trains deep ensembles through 'winner-take-gradient' updates (Lee et al., 2016). Letzelter et al. (2026) extend this to language modeling by training multiple LoRA adapters with a supervised winner-takes-all loss. In reinforcement learning, Grinsztajn et al. (2023) optimize the expected best reward from a population of policies, each parameterized by a shared encoder and separate decoders. Building on this, Hottung et al. (2025) use a single (shared) decoder and condition it on strategy vectors. Our formulation considers stratification of general $k$-sample inference-time objectives; its pass@$k$ instantiation recovers the binary-reward special case of the population objective in Grinsztajn et al. (2023).

**Structured Diversity in Language Models.** Shah et al. (2026) learn distinguishable skills through a mutual-information objective and sample one response per skill at inference time. Chen et al. (2026b) jointly optimize quality and semantic diversity using determinantal point processes over response embeddings. Chen et al. (2026a) reward responses based on their contribution to the group's semantic coverage, while Li et al. (2025) jointly reward response quality and semantic diversity. Several other methods explicitly encourage diversity through regularization, uniqueness rewards, diversity-aware preference optimization, or mode-conditioned reinforcement learning (Yao et al., 2025; Hu et al., 2026; Wan et al., 2026; Lanchantin et al., 2025; Yuan et al., 2026). These methods explicitly encourage diversity; we instead jointly train separate LoRA policies using the inference-time objective, without an explicit diversity signal.

**Diversity at Inference Time.** Diversity can also be encouraged during inference through diverse beam search, sampling without replacement, or prompt variation (Vijayakumar et al., 2018; Kool et al., 2019; Wang et al., 2026). Geuter & Kornhardt (2025) route test-time queries across existing language models. Wang et al. (2025) aggregate responses through successive layers of language models. These methods modify inference-time generation, allocation, or aggregation, whereas we jointly train policies for their collective utility, with one response sampled from each policy.

## 6 Concluding Discussion

In this work, we introduced a stratified formulation of inference-time objectives, where the response budget is allocated across a population of jointly learned policies. Using separate low-rank adapters on a shared language model backbone, we empirically demonstrated improvements in pass@$k$ over a single-policy baseline across four reasoning tasks under the same inference-time budget of responses. While our formulation captures general $k$-sample objectives, our experiments focus on pass@$k$. A promising direction for future work is to study other inference-time objectives, such as majority voting and verifier-based selection. Further, adapting the population size and sample allocation to the prompt and available inference budget is another interesting direction.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and checking the proof of our theory. We also used it for setting up and implementing parts of the codebase, and to edit parts of the manuscript for readability. We have reviewed all AI-assisted work. We checked all proofs and reviewed all AI-assisted code for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

Our work is algorithmic in nature and we do not see any direct societal implications.

### REPRODUCIBILITY STATEMENT

We provide our codebase as part of the submission. The appendices contain the training and evaluation details needed to reproduce our experiments, as well as the proof for our theoretical result.

## REFERENCES

Ryan Bahlous-Boldi, Isha Puri, Idan Shenfeld, Akarsh Kumar, Mehul Damani, Sebastian Risi, Omar Khattab, Zhang-Wei Hong, and Pulkit Agrawal. Vector Policy Optimization: Training for Diversity Improves Test-Time Search. *CoRR*, abs/2605.22817, 2026.

Anas Barakat, Souradip Chakraborty, Khushbu Pahwa, and Amrit Singh Bedi. Why Pass@k Optimization Can Degrade Pass@1: Prompt Interference in LLM Post-training. *CoRR*, 2602.21189, 2026.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. *CoRR*, abs/2407.21787, 2024.

Rudy Bunel, Matthew Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. Leveraging Grammar and Reinforcement Learning for Neural Program Synthesis. *ICLR*, 2018.

Haoxuan Chen, Tianming Liang, Wei-Shi Zheng, and Jian-Fang Hu. Breaking *Winner-Takes-All*: Cooperative Policy Optimization Improves Diverse LLM Reasoning. *CoRR*, 2605.11461, 2026a.

Yilei Chen, Souradip Chakraborty, Lorenz Wolf, Ioannis Paschalidis, and Aldo Pacchiano. Post-training Large Language Models for Diverse High-Quality Responses. *ICLR*, 2026b.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k Training for Adaptively Balancing Exploration and Exploitation of Large Reasoning Models. *CoRR*, abs/2508.10751, 2025.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Aviral Kumar, Rishabh Agarwal, Sridhar Thiagarajan, Craig Boutilier, and Aleksandra Faust. Inference-Aware Fine-Tuning for Best-of-N Sampling in Large Language Models. *ICLR*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training Verifiers to Solve Math Word Problems. *CoRR*, abs/2110.14168, 2021.

William G. Cochran. *Sampling Techniques*. John Wiley & Sons, 1977.

Jonathan Geuter and Gregor Kornhardt. RoBoN: Routed Online Best-of-n for Test-Time Scaling with Multiple LLMs. *CoRR*, abs/2512.05542, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *CoRR*, abs/2407.21783, 2024.

Nathan Grinsztajn, Daniel Furelos-Blanco, Shikha Surana, Clément Bonnet, and Thomas D Barrett. Winner Takes It All: Training Performant RL Populations for Combinatorial Optimization. *NeurIPS*, 2023.

André Hottung, Mridul Mahajan, and Kevin Tierney. PolyNet: Learning Diverse Solution Strategies for Neural Combinatorial Optimization. *ICLR*, 2025.

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. *ICLR*, 2022.

Zhiyuan Hu, Yucheng Wang, Yufei He, Jiaying Wu, Yilun Zhao, See-Kiong Ng, Cynthia Breazeal, Anh Tuan Luu, Hae Won Park, and Bryan Hooi. Rewarding the Rare: Uniqueness-Aware RL for Creative Problem Solving in LLMs. *ACL Findings*, 2026.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. Is Best-of-N the Best of Them? Coverage, Scaling, and Optimality in Inference-Time Alignment. *ICML*, 2025.

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic Beams and Where to Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement. *ICML*, 2019.

Jack Lanchantin, Angelica Chen, Shehzaad Dhuliawala, Ping Yu, Jason Weston, Sainbayar Sukhbaatar, and Ilia Kulikov. Diverse Preference Optimization. *CoRR*, abs/2501.18101, 2025.

Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. *NIPS*, 2016.

Victor Letzelter, Hugo Malard, Mathieu Fontaine, Gaël Richard, Slim Essid, Andrei Bursuc, and Patrick Perez. Multiple Choice Learning of Low-Rank Adapters for Language Modeling. *ICML*, 2026.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly Reinforcing Diversity and Quality in Language Model Generations. *CoRR*, 2509.02534, 2025.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. *ICLR*, 2025.

Philipp Mondorf and Barbara Plank. Liar, Liar, Logical Mire: A Benchmark for Suppositional Reasoning in Large Language Models. *EMNLP*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, 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. *NeurIPS*, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *NeurIPS*, 2023.

Jeffrey Seely, Yuki Imajuku, Tianyu Zhao, Edoardo Cetin, and Llion Jones. Sudoku-Bench: Evaluating creative reasoning with Sudoku variants. *CoRR*, abs/2505.16135, 2025.

Devan Shah, Owen Yang, Daniel Yang, Chongyi Zheng, and Benjamin Eysenbach. UpSkill: Mutual Information Skill Learning for Structured Response Diversity in LLMs. *CoRR*, abs/2602.22296, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *CoRR*, abs/2402.03300, 2024.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning Gym: Reasoning Environments for Reinforcement Learning with Verifiable Rewards. *NeurIPS Datasets and Benchmarks Track*, 2026.

Shota Takashiro, Soichiro Nishimori, Paavo Parmas, Yongmin Kim, Kohsei Matsutani, Gouki Minegishi, Yusuke Iwasawa, Takeshi Kojima, and Yutaka Matsuo. On Advantage Estimates for Max@K Policy Gradients. *CoRR*, abs/2606.06080, 2026.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Remi Munos. Optimizing Language Models for Inference Time Objectives using Reinforcement Learning. *ICML*, 2025.

Christos Thrampoulidis, Sadegh Mahdavi, and Wenlong Deng. Advantage Shaping as Surrogate Reward Maximization: Unifying Pass@K Policy Gradients. *TMLR*, 2026.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. *CoRR*, abs/1610.02424, 2018.

Christian Walder and Deep Tejas Karkhanis. Pass@K Policy Optimization: Solving Harder Reinforcement Learning Problems. *NeurIPS*, 2025.

Zhongwei Wan, Yun Shen, Zhihao Dou, Donghao Zhou, Yu Zhang, Xin Wang, Hui Shen, Jing Xiong, Chaofan Tao, Zixuan Zhong, Peizhou Huang, and Mi Zhang. DSDR: Dual-Scale Diversity Regularization for Exploration in LLM Reasoning. *CoRR*, abs/2602.19895, 2026.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-Agents Enhances Large Language Model Capabilities. *ICLR*, 2025.

Tianchun Wang, Zichuan Liu, Yuanzhou Chen, Jonathan Light, Weiyang Liu, Haifeng Chen, Xiang Zhang, and Wei Cheng. On the Effect of Sampling Diversity in Scaling LLM Inference. *CoRR*, abs/2502.11027, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *ICLR*, 2023.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen2.5 Technical Report. *CoRR*, abs/2412.15115, 2024.

Jian Yao, Ran Cheng, Xingyu Wu, Jibin Wu, and KC Tan. Diversity-Aware Policy Optimization for Large Language Model Reasoning. *NeurIPS*, 2025.

Jiayi Yuan, Hangoo Kang, James Jihao Liu, Yejin Choi, Vikram Iyer, Liwei Jiang, and Natasha Jaques. Forty Shades of Blue: Quality-Diversity Alignment via Mode-Conditioned Reinforcement Learning. *CoRR*, abs/2609.14896, 2026.

Yuchen Zeng and Kangwook Lee. The Expressive Power of Low-Rank Adaptation. *ICLR*, 2024.

# A TABLE OF CONTENTS

In this section, we provide an outline of the contents provided in the paper's appendices.

- Appendix B contains the proof for Theorem 1.
- Appendix C describes the tasks used in the experiments.
- Appendix D provides additional training details for the experiments.
- Appendix E provides additional evaluation details for the experiments.
- Appendix F provides the full results for the main experiments, including pass@1 scores.
- Appendix G empirically justifies the choice of the group-mean baseline over leave-one-out.

# B PROOFS

In this section, we provide the proof of Theorem 1.

For each region $S_j$, the population can adapt the base model parameters $W$ to the target map $T_j$. Lemma 1 with $L = 1$ in Zeng & Lee (2024) tells us that there exists a rank-$r$ adapter $\Delta_j$ such that:

$$\|W + \Delta_j - T_j\|_{\text{op}} = \sigma_{r+1}(T_j - W).$$

Therefore, for all $x \in S_j$, the following holds:

$$\begin{aligned}\min_{i\in[K]} \|(W + \Delta_i)x - f^\star(x)\|_2 &\le \|(W + \Delta_j)x - T_j x\|_2 \\ &\le \|W + \Delta_j - T_j\|_{\text{op}}\|x\|_2 \\ &\le \sigma_{r+1}(T_j - W).\end{aligned}$$

The last inequality follows from $\|x\|_2 \le 1$.

Since the unit ball $\mathbb{B}_2^d$ is partitioned into $S_1, \dots, S_K$, taking the maximum over $x$ gives us:

$$\min_{\text{rank}(\Delta_i)\le r} \max_{\|x\|_2\le 1} \min_{i\in[K]} \|(W + \Delta_i)x - f^\star(x)\|_2 \le \max_{j\in[K]} \sigma_{r+1}(T_j - W).$$

We now derive a lower bound for a single linear predictor. Fix matrix $B$ and define:

$$\varepsilon := \max_{\|x\|_2\le 1} \|Bx - f^\star(x)\|_2.$$

Let $c_j$ denote the center of a closed Euclidean ball of radius $\rho$ contained in $S_j$. Then, for every unit vector $u$, $c_j + \rho u, c_j - \rho u \in S_j$. Since $f^\star(x) = T_j x$ on $S_j$,

$$\|(B - T_j)(c_j + \rho u)\|_2 \le \varepsilon, \qquad \|(B - T_j)(c_j - \rho u)\|_2 \le \varepsilon.$$

Hence, by the triangle inequality, we have:

$$\begin{aligned}\|(B - T_j)u\|_2 &= \frac{1}{2\rho} \big\|(B - T_j)(c_j + \rho u) - (B - T_j)(c_j - \rho u)\big\|_2 \\ &\le \frac{\varepsilon}{\rho}.\end{aligned}$$

Taking the maximum over all unit vectors $u$ gives us:

$$\|B - T_j\|_{\text{op}} \le \frac{\varepsilon}{\rho}.$$

Therefore, for any $i, j \in [K]$,

$$\|T_i - T_j\|_{\text{op}} \le \|T_i - B\|_{\text{op}} + \|B - T_j\|_{\text{op}} \le \frac{2\varepsilon}{\rho}.$$

Consequently, the following holds:

$$\varepsilon \ge \frac{\rho}{2} \max_{i,j\in[K]} \|T_i - T_j\|_{\text{op}}.$$

Taking the minimum over $B$ gives us:

$$\min_B \max_{\|x\|_2\le 1} \|Bx - f^\star(x)\|_2 \ge \frac{\rho}{2} \max_{i,j\in[K]} \|T_i - T_j\|_{\text{op}}.$$

This completes the proof.

# C TASK DETAILS

## C.1 ARITHMETIC

This environment was introduced by Shah et al. (2026) and requires the model to combine three given single-digit numbers into a valid arithmetic expression that relates them.

An instance of the task is an unordered triple of digits $(n_1, n_2, n_3) \in \{0, \ldots, 9\}^3$, which is constructed such that some pair of the three, when combined via one of the operators $\{+, -, \times, \div, \%\}$, gives the third. For example, the triple $\{2, 3, 6\}$ can be used to form the expression $2 \times 3 = 6$. The three digits are presented to the model in random order. Therefore, a valid order is not revealed by their position. Task variability comes from the sampled digits, the operator used to construct the triple, and the order in which the digits are shown.

The model needs to respond with an expression of the form $L$ op $R$, where $L$ and $R$ are two of the three given digits and op is one of the operators. A task is considered solved iff $L$ op $R$ gives the remaining digit.

## C.2 KNIGHTS-&-KNAVES

This is a logical-deduction task in which a model is given a set of self-referential statements and must determine which of several named individuals are telling the truth and which are lying (Mondorf & Plank, 2024). It is a challenging task that tests both logical reasoning and generalization to novel statement structures; we use the REASONING GYM (Stojanovski et al., 2026) implementation.

An instance of the task describes an island inhabited only by two named individuals. Each of them is either a knight, who always tells the truth, or a knave, who always lies. Different instances use different equivalent terminologies: e.g., knight/knave or hero/villain. Each individual makes one statement about the types of the individuals on the island; statements can be simple assertions or nested combinations of negation, conjunction, disjunction, implication, and biconditional over other individuals' statements. Every instance is guaranteed to have a unique assignment of types. Task variability comes from the sampled statements, the names used, and the terminology.

The model needs to respond with a type for each of the two individuals. A task is considered solved iff the assigned types match the unique solution consistent with the statements.

## C.3 SUDOKU

This is the $4 \times 4$ variant of Sudoku, which is a constraint-satisfaction task on which language models continue to struggle even when the board size is reduced (Seely et al., 2025). We use the REASONING GYM (Stojanovski et al., 2026) implementation at the hardest difficulty setting.

An instance of the task is a $4 \times 4$ grid, which is partially filled with digits between 1 to 4, such that there is a unique completion that satisfies the following constraints: each of the four rows, each of the four columns, and each of the four non-overlapping $2 \times 2$ subgrids must contain every digit from 1 to 4 exactly once. At the difficulty setting that we use, 12 of the 16 cells are empty initially, which is the maximum number of empty cells for which a unique solution can be guaranteed on a $4 \times 4$ board. This means that that only 4 clues are given. Task variability comes from the underlying solved grid and from which 12 cells are removed to form the starting state.

The model needs to respond with a completed $4 \times 4$ grid. A task is considered solved iff the completed grid exactly matches the (unique) solution.

## C.4 GSM-SYMBOLIC

This is a procedurally generated variant of GSM8K (Cobbe et al., 2021) that is designed to test mathematical reasoning more reliably than the original benchmark (Mirzadeh et al., 2025): models frequently perform worse on GSM-SYMBOLIC than on the corresponding original GSM8K problems, which suggests that some of their performance on GSM8K is due to memorization. We use the REASONING GYM (Stojanovski et al., 2026) implementation.

An instance of the task is generated by taking one of a fixed set of GSM8K-style word-problem templates and resampling its numeric quantities and entity names. Task variability comes from both the choice of template and the resampled quantities and names.

The model needs to respond with a single (final) numeric answer to the word problem. A task is considered solved iff this value exactly matches the oracle answer for the sampled instance.

# D Training Details

## D.1 Arithmetic

For this task, we use a small causal transformer: `TinyGPT` with 2 layers and 4 attention heads (more details in (Shah et al., 2026)). SINGLE uses a single such network, and STRATIFIED instantiates $k = 5$ separate TinyGPT networks. We train both methods from a random initialization, with supervised teacher-forcing warmup phase for 100 steps and using RL for pass@5 for 2000 steps.

## D.2 Knights-&-Knaves

Both SINGLE and STRATIFIED fine-tune Qwen2.5-1.5B-Instruct (Yang et al., 2024) with a learning rate of $5 \times 10^{-6}$. We first train a single LoRA adapter for 2000 steps, and then use it to initialize the adapters for these methods, which are trained for a further 2000 steps.

## D.3 Sudoku

Both SINGLE and STRATIFIED fine-tune Qwen2.5-1.5B-Instruct with a learning rate of $1 \times 10^{-6}$. We first train a single LoRA adapter on an easy tier of $4 \times 4$ SUDOKU (4 to 8 empty cells) for 20,000 steps, then continue training it on a harder tier (8 to 10 empty cells) for a further 20,000 steps. We use this adapter to initialize SINGLE and STRATIFIED, which are trained for 10,000 steps at the hardest difficulty level (12 out of 16 cells empty).

**Independently trained ensemble baseline.** INDEPENDENT (Fig. 7 (c)) is a population of five independently trained SINGLE policies. Each is initialized from the same warm-start adapter and uses the same configuration as SINGLE, but is trained for only 2,000 steps to ensure that the five runs match the compute of one STRATIFIED run ($5 \times 2{,}000 = 10{,}000$ steps). We train 15 policies in three populations of five and report the mean and standard error over the three populations. For pass@5, we take one sample from each policy in a population, similar to STRATIFIED.

## D.4 GSM-Symbolic

Both SINGLE and STRATIFIED fine-tune Llama-3.1-8B-Instruct (Grattafiori et al., 2024) with a learning rate of $1 \times 10^{-6}$. They are trained without any warm-start for 2000 steps.

# E Evaluation Details

**Evaluation Dataset.** For KNIGHTS-&-KNAVES, SUDOKU, and GSM-SYMBOLIC, we evaluate on the same 1000 randomly sampled instances for all methods and seeds, and on 300 randomly sampled problems for ARITHMETIC.

**Decoding.** For the LLM tasks, responses are sampled with vLLM at the same sampling parameters as the training rollouts: temperature 1.0, top-$p = 1.0$, no top-$k$ truncation, and a maximum of 768 new tokens. For ARITHMETIC, responses are sampled at temperature 0.9 (see (Shah et al., 2026)).

**Checkpoint.** We evaluate the last checkpoint of each run for ARITHMETIC, KNIGHTS-&-KNAVES, and SUDOKU. For GSM-SYMBOLIC, SINGLE collapses late in training (see Fig. 7b), so we report the last checkpoint before the collapse based on training rewards.

**Diversity Analyses.** The overlap analyses (Figs. 5 and 6) use greedy decoding, on 1000 instances for SUDOKU and GSM-SYMBOLIC and 500 for KNIGHTS-&-KNAVES and ARITHMETIC.

Table 1: GRPO training hyperparameters.

| **Hyperparameter** | **Value** |
|---|---|
| LoRA Rank | 32 (160 for BIG-SINGLE) |
| Population Size $k$ (`num_generations`) | 5 |
| Training Batch Size | 5 (one prompt per training step) |
| Training Epochs | 1 |
| Optimizer | AdamW (8-bit) |
| Adam $\beta_1$, $\beta_2$ | 0.9, 0.99 |
| Weight Decay | 0.0 |
| Gradient Accumulation Steps | 1 |
| Max Gradient Norm | 1.0 |
| Learning-rate Schedule | constant |
| KL Coefficient ($\beta$) | 0.0 |
| Sampling Temperature | 1.0 |
| Max Prompt Length | 512 tokens |
| Max Completion Length | 768 tokens |
| Max Sequence Length | 1024 tokens |
| Precision | bf16 |

# F FULL RESULTS

Table 2: Full results in % with mean (standard error) over three seeds.

| | ARITHMETIC | | KNIGHTS-&-KNAVES | | SUDOKU | | GSM-SYMBOLIC | |
|---|---|---|---|---|---|---|---|---|
| **Method** | Pass@1 | Pass@5 | Pass@1 | Pass@5 | Pass@1 | Pass@5 | Pass@1 | Pass@5 |
| SINGLE | 93.4 (3.9) | 95.5 (2.8) | 56.8 (0.8) | 90.6 (0.6) | 40.9 (0.3) | 74.2 (0.6) | 52.1 (4.0) | 85.5 (2.4) |
| STRATIFIED, adapter 1 | 62.9 (3.7) | – | 36.6 (6.4) | – | 34.5 (0.8) | – | 56.2 (0.5) | – |
| STRATIFIED, adapter 2 | 40.2 (0.4) | – | 32.5 (3.2) | – | 34.8 (1.7) | – | 54.9 (1.1) | – |
| STRATIFIED, adapter 3 | 23.5 (4.1) | – | 41.4 (0.9) | – | 35.5 (0.5) | – | 56.9 (0.7) | – |
| STRATIFIED, adapter 4 | 20.1 (1.1) | – | 45.0 (0.2) | – | 34.0 (0.7) | – | 56.9 (0.6) | – |
| STRATIFIED, adapter 5 | 15.8 (10.4) | – | 43.2 (2.8) | – | 34.4 (1.1) | – | 58.1 (1.2) | – |
| STRATIFIED | 32.5 (1.7) | 99.7 (0.4) | 39.8 (0.7) | 93.2 (0.8) | 34.6 (0.5) | 80.3 (0.3) | 56.6 (0.2) | 90.4 (0.1) |

We provide the full results for our main experiments in Table 2. Pass@1 is defined as the expected accuracy of a single response; for SINGLE, it is the mean accuracy over its samples, and for STRATIFIED, it is the mean accuracy over its five adapters (which is equivalent to the accuracy of one response sampled from an adapter chosen uniformly at random).

# G CHOICE OF ADVANTAGE BASELINE

Table 3: Comparison between group-mean (Eq. 6) and leave-one-out (Eq. 4) advantage on SUDOKU.

| | **Group-Mean** | | | | **Leave-One-Out** | | | |
|---|---|---|---|---|---|---|---|---|
| **Method** | Seed 42 | Seed 123 | Seed 456 | Mean | Seed 42 | Seed 123 | Seed 456 | Mean |
| SINGLE | 75.0 | 73.1 | 74.4 | 74.2 (0.6) | 72.5 | 75.3 | 72.4 | 73.4 (1.0) |
| STRATIFIED | 80.0 | 80.0 | 80.8 | 80.3 (0.3) | 74.2 | 76.1 | 77.5 | 75.9 (1.0) |
| Gap | +5.0 | +6.9 | +6.4 | +6.1 (0.6) | +1.7 | +0.8 | +5.1 | +2.5 (1.3) |

Table 3 compares the group-mean baseline with the leave-one-out baseline on SUDOKU. The results show two things. First, the group-mean baseline improves STRATIFIED on every seed (+4.3 on average), while its effect on SINGLE is marginal and not consistent across seeds. Second, STRATIFIED outperforms SINGLE on all seeds under both baselines, but the gap is smaller under leave-one-out.