Co-evolution: A "One-to-many" LLM Fine-Tuning Paradigm

Dapeng Jiang[1,3*] **Dianqiao Lei**[1*] **Haichuan Tan**[1,4,5*]
Wenxuan Song[1] Shenqi Zong[1] Yanyan Lan[1,2†]
[1]Institute for AI Industry Research (AIR), Tsinghua University, Beijing, China
[2]Beijing Frontier Research Center for Biological Structure, Tsinghua University, Beijing, China
[3] College of AI, Tsinghua University, Beijing, China
[4] Department of Computer Science and Technology, Tsinghua University
[5] AIPher Technology

Abstract

Fine-tuning has become a central mechanism for adapting large language models (LLMs) to downstream tasks, yet most existing methods follow a one-to-one paradigm: a single pretrained model is optimized into a single improved policy. This paradigm optimizes average single-policy performance, leaving no mechanism to distinguish redundant successes from genuinely complementary contributions. In this paper, we propose Co-Evolve, a one-to-many LLM fine-tuning framework that transforms a single base model into a cooperative population of persistent, specialized descendants and explicitly optimizes their complementarity. To reduce redundancy among descendants, we introduce a marginal-coverage objective that rewards each model for paying attention to examples under-covered by its peers. We instantiate this objective with evolution strategies, enabling derivative-free population-level optimization. We further introduce a seed-replay checkpoint mechanism that reduces the disk storage required to maintain multiple descendants by avoiding storing multiple full checkpoints during deployment. Experiments across multiple domains show that Co-Evolve consistently improves ensemble-level performance over strong fine-tuning and ensemble baselines. Our results suggest that optimizing for complementary model-level specialization provides a scalable alternative to single-policy fine-tuning. Our code is available at https://github.com/jdp22/Co-Evolution-Core.

1 Introduction

Large language models (LLMs) are commonly adapted to downstream tasks through post-training methods such as reinforcement learning from human feedback (RLHF) with PPO, reinforcement learning with verifiable rewards such as GRPO, direct preference optimization, and derivative-free evolution strategies [20, 25, 26, 24, 22]. These methods have improved instruction following, reasoning, and preference alignment. However, their standard formulation remains *one-to-one*: a single pretrained model is optimized into a single downstream policy.

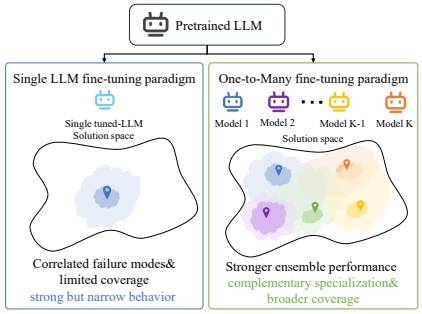

Figure 1: Conventional fine-tuning follows a one-to-one paradigm; we instead adopt a one-to-many paradigm, encouraging complementary coverage of the solution space.

*Core contributors
†Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

This formulation is effective for improving average single-model performance, but it is not designed to create complementary behaviors across multiple models. Complex reasoning tasks often contain heterogeneous instances that admit different solution paths, domain-specific heuristics, or alternative intermediate representations. Optimizing one policy for average reward can therefore concentrate probability mass around a narrow set of high-reward behaviors, reducing entropy and behavioral diversity [13, 18, 6, 4]. As a result, a tuned model may become stronger yet narrower: it can solve more common cases while preserving correlated failures on examples that require different reasoning strategies. Standard fine-tuning also provides no explicit mechanism to distinguish a redundant success on an already-covered input from a contribution that expands the collective coverage of a model group.

Existing approaches only partially address this issue. Test-time scaling methods, including chain-of-thought sampling and self-consistency, obtain multiple reasoning attempts from a single model, but all attempts are still drawn from the same underlying policy [32, 31]. Diversity-aware and set-level training objectives further reward marginal contributions among sampled outputs, improving sample-level coverage or pass@K utility [29, 2, 15]. However, these methods mainly shape transient responses from one policy rather than persistent parameterized specialists. LLM ensembles, routing methods, and multi-agent post-training systems demonstrate the value of model plurality, but they typically rely on heterogeneous off-the-shelf models, inference-time orchestration, or collaboration-specific objectives [5, 30, 19, 21, 16]. They therefore leave open the question of how to transform a single base LLM into a compact population of descendants that are explicitly trained to be less redundant.

We propose Co-Evolve, a *one-to-many* LLM fine-tuning paradigm that evolves one base model into multiple persistent descendants. As shown in Fig. 1, conventional fine-tuning produces one tuned successor, whereas Co-Evolve optimizes a population whose members are encouraged to cover different regions of the solution space. The key idea is cooperative specialization: each descendant receives larger credit on examples that are under-covered by its peers and lower credit on examples already solved by the population. This marginal-coverage objective turns fine-tuning from independent reward maximization into population-level credit assignment, encouraging the group to expand its collective coverage rather than repeatedly optimizing the same successes.

Our contributions can be summarized as follows: (1) *New Paradigm.* Co-Evolve is a new one-to-many LLM fine-tuning paradigm that explicitly optimizes a single base model into a population of persistent, specialized descendants with complementary strengths. (2) *Novel Methodology.* Co-Evolve is optimized by a cooperative specialization objective using evolution strategies. Furthermore, we introduce a storage-efficient seed-replay mechanism that represents each descendant by a shared base checkpoint and compact update logs. (3) *Comprehensive Experiments.* Finally, we conduct experiments across multiple tasks on models across different modalities, showing that Co-Evolve consistently improves ensemble-level performance over strong fine-tuning and test-time ensemble baselines.

2 Related Work

Single-policy LLM post-training. Post-training is the dominant mechanism for adapting pretrained LLMs to downstream tasks and human preferences. Representative methods include RLHF with PPO-style policy optimization [20, 25], GRPO for reasoning-oriented RL fine-tuning [26], preference optimization methods such as DPO [24], and recent evolution-strategy approaches for full-parameter LLM fine-tuning [22]. Despite their different optimization mechanisms, these methods usually optimize one base model into one improved policy. This single-policy objective is strong for average performance, but it does not allocate credit according to whether a solved example is redundant with other models. Co-Evolve keeps the reward-driven adaptation goal, but changes the unit of optimization from a single policy to a cooperative population.

Output- and set-level diversity. A long line of work shows that reasoning performance can improve when multiple candidate solutions are considered. Chain-of-thought prompting and self-consistency sample multiple reasoning paths from one model and aggregate the resulting answers [32, 31]. Recent training-time methods further optimize group- or set-level utilities, including pass@K objectives, group-aware reinforcement learning, and diversity-preserving set-level policy optimization [29, 2, 15, 1]. These methods are closely related to our motivation because they recognize that independent

single-response reward is insufficient. The main difference is that their diversity is defined over sampled outputs from one policy, whereas Co-Evolve moves diversity optimization to the parameter level by training multiple persistent descendants.

Model plurality, routing, and multi-agent post-training. LLM ensemble and routing methods combine multiple models through voting, ranking, dynamic selection, or collaborative generation [5, 11, 10, 30, 19]. More recent multi-agent post-training frameworks explicitly fine-tune multiple LLM agents for collaboration, discussion, or system-level coordination [21, 16, 7]. These works are complementary to Co-Evolve but target a different problem. These methods mainly study how to use existing models at inference time. In contrast, Co-Evolve focuses on training-time population construction. Therefore, the performance gain can be attributed to learned model-level specialization rather than to complex inference-time orchestration.

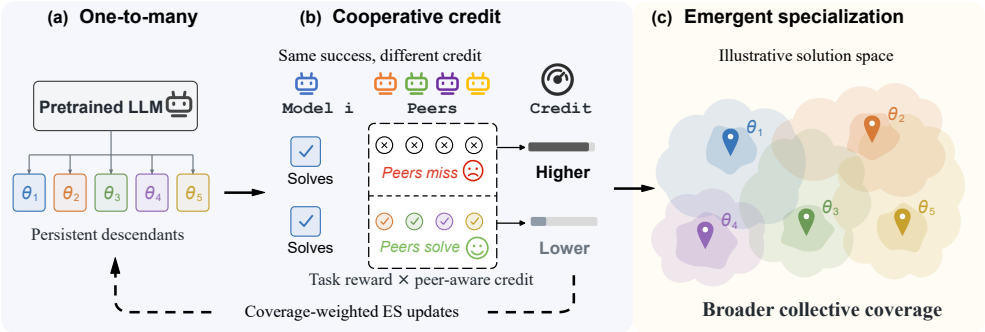

Figure 2: An overview of our proposed Co-Evolve.

3 Method

As shown in Fig. 2, we propose Co-Evolve as a one-to-many fine-tuning framework that adapts a single pretrained LLM into a population of persistent descendants. In Sec. 3.1, we briefly review evolutionary fine-tuning for LLMs. In Sec. 3.2, we introduce our cooperative specialization objective, which encourages each LLM to cover regions of the solution space that are still not well covered by other LLMs. We show the training algorithm in Sec 3.3. Unlike other multi-LLM methods, we show the storage-efficient replay mechanism in Sec. 3.4.

3.1 Preliminaries: Evolutionary Fine-Tuning of LLMs

Evolutionary algorithms provide a derivative-free alternative to gradient-based post-training of language models. Instead of backpropagating through the token-level likelihood or a differentiable surrogate loss, these methods maintain a population of candidate solutions, evaluate each candidate with a scalar fitness, and use mutation–selection–recombination to construct the next generation. In continuous parameter spaces, Evolution Strategies (ES) can be viewed as a simple and scalable instance of this principle: given a pretrained LLM with parameters θ and a response-level reward $R(x, \theta)$, ES samples a population of Gaussian-mutated models

$$\theta^{(n)} = \theta + \sigma\epsilon^{(n)}, \qquad \epsilon^{(n)} \sim \mathcal{N}(0, I), \qquad n = 1, \ldots, N, \tag{1}$$

evaluates each perturbation on a mini-batch from training dataset $\mathcal{D}$, $\{x_s\}_{s=1}^{M} \sim \mathcal{D}$ with fitness

$$\mathcal{F}^{(n)} = \frac{1}{M} \sum_{s=1}^{M} R(x_s, \theta^{(n)}), \tag{2}$$

and updates the center model by recombining perturbation directions according to their normalized fitness:

$$\theta \leftarrow \theta + \frac{\alpha}{N\sigma} \sum_{n=1}^{N} \tilde{\mathcal{F}}^{(n)} \epsilon^{(n)}. \tag{3}$$

ES is particularly suitable for generated text objectives where rewards are sparse, delayed, binary, or otherwise non-differentiable [22]. Our method builds on this single-model ES view, but replaces the independent fitness $\mathcal{F}^{(n)}$ with a model-specific credit-weighted fitness so that a population of LLMs specializes cooperatively rather than redundantly solving the same inputs.

3.2 Cooperative Specialization Objective

We train K language models $\{\theta_i\}_{i=1}^{K}$ as a group in parallel from a shared pretrained initialization. Our deployment objective is to improve the ensemble performance of the model group. Independent training does not directly optimize this objective: with identical initialization, reward, and optimization procedure, different models have no explicit incentive to develop complementary behaviors and may waste capacity on already-solved inputs. We close this gap with a per-model marginal coverage credit.

The objective of each model θ_i is formalized as

$$\mathcal{J}_i(\theta_i;\theta_{-i}) \;=\; \mathbb{E}_{x\sim\mathcal{D}}\big[R(x,\theta_i)\,c_i(x;\theta_{-i})\big] \;=\; \mathbb{E}_{x\sim p_i}\big[R(x,\theta_i)\big], \tag{4}$$

where $p_i(x) = c_i(x;\theta_{-i})p(x)$ is model i's specialized training distribution. Intuitively, each model still maximizes the standard task reward, but under a reweighted data distribution that assigns larger mass to inputs under-covered by the rest of the ensemble.

Let $R(x,\theta_i) \in [0,1]$ denote the reward of model θ_i on input x. For model θ_i, we define the marginal coverage weight

$$w_i(x;\theta_{-i}) \;=\; 1 - \frac{1}{K}\sum_{j\neq i} R(x,\theta_j) \;\in\; \left[\frac{1}{K},1\right]. \tag{5}$$

This weight is largest when no peer model solves the input and smallest when the input is already covered by all peer models. Therefore, a model receives more training credit on examples where it can expand the collective coverage of the population, and less credit on examples already solved by the rest of the ensemble. Theoretical analysis can be found at Appendix A.

For implementation, we use the normalized credit

$$c_i(x;\theta_{-i}) \;=\; \frac{w_i(x;\theta_{-i})}{\bar{w}_i}, \qquad \bar{w}_i \;=\; \mathbb{E}_{x\sim\mathcal{D}}[w_i(x;\theta_{-i})]\,. \tag{6}$$

The normalization ensures $\mathbb{E}_{x\sim\mathcal{D}}[c_i(x;\theta_{-i})] = 1$, so the credit only redistributes weight across samples without changing the global learning-rate scale.

Current-round peer coverage. In each ES round, all perturbations of all models are evaluated on the same mini-batch before the ES recombination step. We therefore compute the peer coverage weights directly from the current-round reward table, without relying on stale rewards from previous rounds. Specifically, for each model j at round t, we define its current-round population-mean reward on example x_s as

$$\bar{R}_j^{(t)}(x_s) \;=\; \frac{1}{N}\sum_{n=1}^{N} R\Big(x_s,\theta_j^{(n,t)}\Big)\,, \tag{7}$$

where $\theta_j^{(n,t)} = \theta_j^{(t)} + \sigma\epsilon_j^{(n,t)}$ is the n-th perturbation of model j in round t.

3.3 Training: Optimization via Evolution Strategies

To optimize the non-differentiable objective $\mathcal{J}_i$, each model i follows the ES framework described in Sec. 3.1. In each round, it samples N isotropic perturbations $\epsilon^{(n)} \sim \mathcal{N}(0,I)$ and evaluates the credit-weighted fitness:

$$\mathcal{F}_i^{(n)} \;=\; \frac{1}{M}\sum_{s=1}^{M} R\Big(x_s,\theta_i^{(n)}\Big)\,c_i(x_s;\theta_{-i}), \tag{8}$$

which provides an estimate of $\mathcal{J}_i(\theta_i^{(n)};\theta_{-i})$. Substituting the definition of c_i from (6), this fitness is equivalent to the accuracy on the coverage gap normalized by the batch-mean coverage weight:

$$\mathcal{F}_i^{(n)} \;=\; \frac{\sum_{s=1}^{M} R\Big(x_s,\theta_i^{(n)}\Big)\,w_i(x_s;\theta_{-i})}{\sum_{s=1}^{M} w_i(x_s;\theta_{-i})}. \tag{9}$$

Algorithm 1 Training Procedure of Co-Evolve: fine-tuning with current-round marginal-coverage credit

Require: initial models $\{\theta_i\}_{i=1}^{K}$ (K: number of models); training set $\mathcal{D}$ with mini-batch size M; number of rounds T; perturbations per model per round N; perturbation scale σ; ES learning rate α;

for $t = 1, \ldots, T$ **do**
  Sample shared mini-batch $\{x_s\}_{s=1}^{M} \sim \mathcal{D}$
  for $i = 1, \ldots, K$ **in parallel do**
    for $n = 1, \ldots, N$ **do**
      Sample $\epsilon_i^{(n,t)} \sim \mathcal{N}(0, I)$
      $\theta_i^{(n,t)} \leftarrow \theta_i^{(t)} + \sigma \epsilon_i^{(n,t)}$
      Evaluate $R(x_s, \theta_i^{(n,t)})$ for all $s = 1, \ldots, M$
    end for
  end for
  Barrier: collect the current-round reward table
  Compute $\bar{R}_i^{(t)}(x_s) \leftarrow \frac{1}{N} \sum_{n=1}^{N} R(x_s, \theta_i^{(n,t)})$ for all i, s
  for $i = 1, \ldots, K$ **in parallel do**
    $w_i^{(t)}(x_s) \leftarrow 1 - \frac{1}{K} \sum_{j \neq i} \bar{R}_j^{(t)}(x_s)$ for all s
    for $n = 1, \ldots, N$ **do**
      $\mathcal{F}_i^{(n)} \leftarrow \frac{\sum_s R(x_s, \theta_i^{(n,t)})\, w_i^{(t)}(x_s)}{\sum_s w_i^{(t)}(x_s)}$ ▷ Eq. (9)
    end for
    $\tilde{\mathcal{F}}_i^{(n)} \leftarrow (\mathcal{F}_i^{(n)} - \mu_{\mathcal{F}})/\sigma_{\mathcal{F}}$ for $n = 1, \ldots, N$ ▷ z-score over n
    $\theta_i^{(t+1)} \leftarrow \theta_i^{(t)} + \frac{\alpha}{N\sigma} \sum_{n=1}^{N} \tilde{\mathcal{F}}_i^{(n)} \epsilon_i^{(n,t)}$ ▷ Eq. (3)
  end for
end for
return $\{\theta_i^{(T+1)}\}_{i=1}^{K}$

The center model is then updated according to the recombination rule in (3), using the z-scored fitness $\tilde{\mathcal{F}}_i^{(n)}$ calculated across the N perturbations.

Algorithm summary. The full procedure is detailed in Algorithm 1. In each round, the paradigm processes K models in parallel. Each model independently computes its marginal coverage weights w_i based on the current-round population-mean reward table. Parameters are then updated via Natural Evolution Strategies using the credit-weighted fitness $\mathcal{F}_i^{(n)}$, which prioritizes solving inputs currently missed by the rest of the group.

3.4 Inference: Storage-Efficient Replay

At inference time, the K trained models $\{\theta_i\}_{i=1}^{K}$ run independently on the same input x and their outputs are combined by an ensemble strategy. The key challenge is storage: a naive implementation would save a separate full checkpoint of size $O(d)$ per model. We avoid this entirely through seed-recoverable replay.

Because every perturbation $\epsilon^{(n)}$ is drawn from a seeded PRNG, the entire trajectory of model i is determined by its base model $\theta_i^{(0)}$ and the sequence of round records $\{(s_t^{(n)}, \tilde{c}_t^{(n)})_{n=1}^{N}\}_{t \geq 1}$, where $s_t^{(n)}$ is the integer seed and $\tilde{c}_t^{(n)} = \frac{\alpha}{N\sigma} \tilde{\mathcal{F}}_i^{(n)}$ is the update coefficient for perturbation n at round t. The full weight state at any round t is therefore

$$\theta_i^{(t)} \;=\; \theta_i^{(0)} \;+\; \sum_{\tau=1}^{t} \sum_{n=1}^{N} \tilde{c}_\tau^{(n)}\, \epsilon\Big(s_\tau^{(n)}\Big)\,, \tag{10}$$

where $\epsilon(s) = \mathrm{randn}(d;\ \mathit{seed} = s)$ is regenerated on the fly without storage.

Naive deployment of a K-model ensemble requires saving K independent full checkpoints, costing $O(K \cdot d)$ storage. The replay approach reduces this to one shared base-model checkpoint of size

Algorithm 2 Inference procedure of Co-Evolve: Seed-Replay Checkpoint Materialization

Require: base model $\theta^{(0)}$; replay log $\mathcal{L}_i = \{(s_t^{(n)}, \tilde{c}_t^{(n)})_{n=1}^{N}\}_{t=1}^{t^*}$ for model i; target round t^*

$\theta \leftarrow \theta^{(0)}$ ▷ load base model once; $O(d)$
for $t = 1, \ldots, t^*$ **do**
  for $n = 1, \ldots, N$ **do**
    $\epsilon \leftarrow \mathrm{randn}(d;\ seed{=}s_t^{(n)})$ ▷ regenerate on the fly
    $\theta \leftarrow \theta + \tilde{c}_t^{(n)}\,\epsilon$ ▷ apply update; $O(d)$ in-place
  end for
end for
return θ ▷ exact replica of $\theta_i^{(t^*)}$

$O(d)$, plus K compact replay logs whose combined size is negligible. The dominant storage cost therefore drops from $O(K \cdot d)$ to $O(d)$, a K-fold reduction, while every intermediate checkpoint remains exactly recoverable on demand.

Replay algorithm. Algorithm 2 formalizes the inference procedure. Any intermediate checkpoint at round t^* is recovered by sequentially applying the accumulated seed-noise updates from round 1 to t^*, starting from the base model.

Method	Countdown	Olympiad	ROCStories	USPTO
`Qwen2.5-3B-Instruct`				
Base	12.80	29.32	54.55	14.19
PPO	42.30$_{+29.50}$	30.05$_{+0.73}$	66.95$_{+12.40}$	32.17$_{+17.98}$
GRPO	40.40$_{+27.60}$	29.96$_{+0.64}$	67.17$_{+12.62}$	50.25$_{+36.06}$
ES	55.45$_{+42.65}$	32.70$_{+3.38}$	<u>68.87</u>$_{+14.32}$	<u>58.94</u>$_{+44.75}$
`+test-time majority voting`				
Base+TT-MV	34.75$_{+21.95}$	35.23$_{+5.91}$	54.90$_{+0.35}$	13.59$_{-0.60}$
PPO+TT-MV	60.15$_{+47.35}$	35.74$_{+6.42}$	66.83$_{+12.28}$	32.07$_{+17.88}$
GRPO+TT-MV	50.45$_{+37.65}$	36.08$_{+6.76}$	67.13$_{+12.58}$	50.45$_{+36.26}$
ES+TT-MV	<u>65.15</u>$_{+52.35}$	<u>36.92</u>$_{+7.60}$	68.54$_{+13.99}$	56.84$_{+42.65}$
Co-Evolve	**69.85**$_{+57.05}$	**39.66**$_{+10.34}$	**70.83**$_{+16.28}$	**61.64**$_{+47.45}$
`Qwen2.5-0.5B-Instruct`				
Base	0.10	6.12	31.62	2.00
ES	14.70$_{+14.60}$	8.65$_{+2.53}$	<u>33.34</u>$_{+1.72}$	<u>31.67</u>$_{+29.67}$
ES+TT-MV	<u>15.30</u>$_{+15.20}$	<u>9.28</u>$_{+3.16}$	32.76$_{+1.14}$	30.45$_{+28.45}$
Co-Evolve	**17.30**$_{+17.20}$	**11.60**$_{+5.48}$	**34.72**$_{+3.10}$	**32.17**$_{+30.17}$

Table 1: Performance comparison across four tasks. Co-Evolve consistently achieves the best performance across tasks. The small subscript denotes the absolute improvement over the corresponding base model within the same model group. Best results are shown in bold, and second-best results are underlined within each group. We further compare against an independent ES ensemble in Sec. 5.2.

4 Experiments

4.1 Experiment settings

Baselines We compare our method against representative fine-tuning approaches: PPO, GRPO, and ES. The PPO and GRPO implementations are based on the VERL [27] framework, while ES uses the official implementation from the original paper. For a fair comparison, all methods are trained with comparable computing budget as stated in Appendix. J. To match inference cost, we further include a test-time majority voting baseline (TT-MV), where the number of voting candidates equals the number of models K. For a fair comparison across algorithms, all methods are run under the compute-matched budget in Appendix J. We also report TT-MV with the same number of descendants as Co-Evolve.

Datasets For the language-only LLMs, we evaluate on four challenging tasks: Countdown [8], OlympiadBench [9], ROCStories [17] and USPTO [28], which cover the math solving, physics

understanding, creative writing, knowledge QA and scientific reasoning. For the Multi-modal LLM, we evaluate on the visual question answering benchmark GQA [12]. More details about the datasets and corresponding reward design can be found in the Appendix D.

Implementation Details We evaluate Co-Evolve on LLMs with different scales and training regimes. We evaluate on `Qwen2.5-0.5B-Instruct`, `Qwen2.5-3B-Instruct`, `Qwen2.5-Math-1.5B-Instruct` and `Qwen2.5-VL-3B-Instruct` [33, 23]. For all of the tasks, we follow the official prompt and we set the `max_prompt_length = 2048` to guarantee the inclusiveness of necessary information of the question. For the max generation length, we set `max_generation_length = 1024` for countdown, `max_generation_length = 1024` for OlympiadBench and `max_generation_length = 64` for ROCStories and USPTO. We set the `temperature = 0.0` for ES and Co-Evolve. For GRPO and PPO, we set the `temperature = 0.7`. All of our experiments are conducted on A100 GPUs. More details can be found in Appendix J.

4.2 Main results

Table 1 summarizes the results on four language-only benchmarks. Across both backbones, Co-Evolve achieves the best performance on every task. On `Qwen2.5-3B-Instruct`, Co-Evolve improves over the strongest single-model baseline, by 14.40 points on Countdown and 2.70 points on USPTO, while also yielding consistent gains on OlympiadBench and ROCStories. The improvements are largest on tasks with heterogeneous solution patterns, suggesting that marginal-coverage credit helps allocate descendants toward regions that remain under-covered by a single policy.

Method	**Countdown**	**Olympiad**
`Qwen2.5-Math-1.5B-Instruct`		
Base	30.70	40.93
PPO	$43.85_{+13.15}$	$41.98_{+1.05}$
GRPO	$36.95_{+6.25}$	$41.35_{+0.42}$
ES	$44.70_{+14.00}$	$42.83_{+1.90}$
`+TT-MV`		
Base+TT-MV	$49.90_{+19.20}$	$44.73_{+3.80}$
PPO+TT-MV	$53.45_{+22.75}$	$44.91_{+3.98}$
GRPO+TT-MV	$50.80_{+20.10}$	$45.57_{+4.64}$
ES+TT-MV	$\underline{53.50}_{+22.80}$	$\underline{46.23}_{+5.30}$
Co-Evolve	$\mathbf{56.90}_{+26.20}$	$\mathbf{47.47}_{+6.54}$

Table 2: Performance comparison on math reasoning tasks using `Qwen2.5-Math-1.5B-Instruct`. The small subscript denotes the absolute improvement over the corresponding base model. Best results are shown in bold, and second-best results are underlined.

The gains are not explained by using more inference candidates alone. Under the same test-time budget, Co-Evolve outperforms ES+TT-MV on all four tasks, including a 4.80 point gain on USPTO. This indicates that co-evolved descendants acquire persistent complementary behaviors, rather than merely producing repeated samples from one fine-tuned model. The same trend holds for the `Qwen2.5-0.5B-Instruct`, where Co-Evolve consistently improves over both ES and ES+TT-MV, suggesting that cooperative specialization does not rely solely on large model capacity.

Table 2 further shows that the benefit persists on the math-specialized `Qwen2.5-Math-1.5B-Instruct` model, where Co-Evolve surpasses ES+TT-MV by 3.40 points on Countdown and 1.24 points on OlympiadBench despite the stronger domain-specific initialization. Overall, these results support our central claim: optimizing persistent descendants for marginal coverage reduces redundant failures and expands ensemble-level solution coverage beyond both single-policy fine-tuning and test-time voting.

4.3 Co-Evolve on Vision-Language Model

We extend our evaluation to the vision-language setting. We use `Qwen2.5-VL-3B-Instruct` as the multimodal backbone and evaluate on GQA, a visual question answering benchmark that requires grounded understanding of objects, attributes, spatial relations, and compositional visual reasoning.

Method	**GQA**
`Qwen2.5-VL-3B-Instruct`	
Base+TT-MV	$\underline{48.70}$
Co-Evolve	$\mathbf{61.26}$

Table 3: Performance comparison on the GQA visual question answering benchmark.

As shown in Table 3, Co-Evolve substantially outperforms the Base+TT-MV baseline, which obtains 48.70. The result suggests that training multiple descendants with the cooperative specialization objective can produce complementary multimodal reasoning behaviors. Different descendants may become better at different visual reasoning patterns, thereby improving the performance.

These results indicate that even under a vision-language backbone with additional visual encoders and multimodal alignment, Co-Evolve remains effective, suggesting that cooperative model-level specialization is a general mechanism that can transfer across modality regimes.

5 Further Analysis

5.1 Scalability of Co-Evolve

We further investigate the scalability of Co-Evolve with respect to the number of descendants. Specifically, we vary the population size $K \in \{1, 3, 5, 8\}$ and evaluate `Qwen2.5-3B-Instruct` on the Countdown task. We use the same training and inference settings for each descendant for fair comparison.

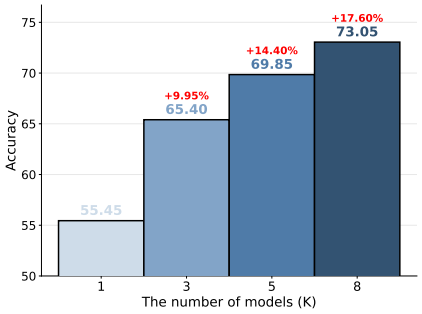

Figure 3: Scalability of Co-Evolve with respect to the number of co-evolved descendants K on the Countdown task.

As shown in Figure 3, increasing K consistently improves the final Countdown accuracy. The single-model setting with $K = 1$ achieves an accuracy of 55.45%. Increasing the population size to $K = 3$ raises the accuracy to 65.40%, yielding an absolute improvement of 9.95%. Further scaling to $K = 5$ and $K = 8$ improves the accuracy to 69.85% and 73.05%, respectively.

These results show that a larger population expands the collective coverage of the model group and improves ensemble-level reasoning performance, showing the scalability of Co-Evolve.

5.2 Ablation study of Cooperative Specialization Objective: K independent models baseline

We ablate the cooperation in our Objective. In particular, we instantiate the training credit as $c_i^{(\lambda)} = (1-\lambda) + \lambda c_i$, where c_i is the normalized marginal-coverage credit defined in Sec. 3.2. Therefore, $\lambda = 0$ removes the cooperative specialization signal and reduces training to standard parallel reward-driven evolutionary fine-tuning, while $\lambda = 1$ corresponds to the full Co-Evolve objective. The intermediate setting $\lambda = 0.5$ tests whether a weaker specialization pressure is sufficient.

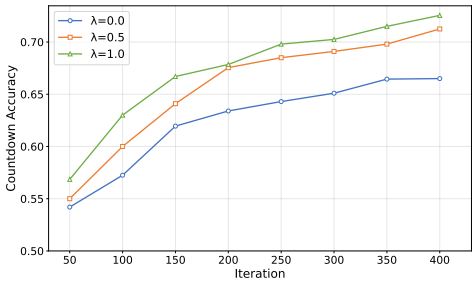

Figure 4: Ablation of Cooperative Specialization Objective.

As shown in Figure 4, the specialization coefficient has a clear impact on both convergence speed and final Countdown accuracy. Without the cooperative credit, the $\lambda = 0$ curve improves slowly and plateaus at roughly 0.66 accuracy after 400 iterations. Introducing the marginal-coverage credit substantially improves the learning trajectory: $\lambda = 0.5$ reaches around 0.71, while $\lambda = 1.0$ achieves the best final performance, around 0.72. These results confirm that the Cooperative Specialization Objective is a key driver of collective improvement. By reweighting each model toward under-covered solution space, the objective reduces redundant optimization and encourages the model group to occupy complementary solution regions.

5.3 Coverage and Specialization Analysis

To further examine whether Co-Evolve learns complementary descendants rather than merely benefiting from repeated sampling, we analyze the coverage pattern of five co-evolved models on the Countdown test set. We compare three conditions: five stochastic samples from the base model, five stochastic samples from the best single co-evolved model, and five greedy descendants trained by Co-Evolve.

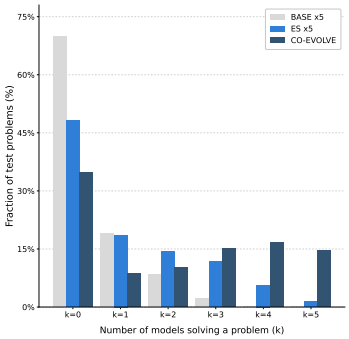

Figure 5: k-coverage distribution on Countdown, where k denotes the number of models that solve a test problem.

Figure 5 visualizes the k-coverage distribution on Countdown, where k denotes the number of models that solve each test instance. Compared with repeated sampling from the single model optimized by ES, Co-Evolve substantially reduces the fraction of unsolved instances ($k = 0$) from 48.1% to 34.8%, while shifting more mass toward high-coverage regions ($k = 3, 4, 5$). This indicates that Co-Evolve does not simply maximize disagreement among descendants. Instead, it raises the competence of individual models while preserving sufficient complementarity to expand the population-level coverage. This distinction is important: the desired outcome is not maximum diversity, but high collective coverage with less redundant failure.

6 Discussion and Future Work

Co-Evolve should be understood not simply as an ensemble method, but as a training-time mechanism for constructing a less redundant population of models. By moving diversity optimization from test-time sampling to the parameter level, Co-Evolve enables descendants to develop persistent and complementary strengths. This suggests a broader view of fine-tuning: a pretrained LLM can be adapted not only into a single improved policy, but also into a cooperative population. This view also distinguishes Co-Evolve from test-time diversity methods: rather than repeatedly sampling from one policy, it stores complementary behaviors in the parameters of different descendants, making specialization persistent and reproducible across inputs.

A key limitation is that ensemble deployment still incurs additional inference-time computation, since multiple descendants may need to be queried. Although the seed-replay mechanism reduces the storage cost of maintaining multiple descendants, the computational cost scales with the number of active models. Future work could therefore explore efficiency-aware deployment, such as lightweight routers that select only the most relevant descendants for each input. Such routers could be trained from coverage statistics collected during co-evolution, activating only a few descendants for easy inputs while reserving full-population inference for uncertain or under-covered cases.

Another important direction is to better characterize and control specialization. While our results show improved performance, reduced overlap, and qualitative evidence of complementary behavior, more analyses could reveal what each descendant learns. Future analysis could examine whether descendants specialize by domain, difficulty level, reasoning template, or error pattern, and whether such specialization remains stable across longer training horizons. Co-Evolve may also be extended to open-ended generation, code synthesis, scientific reasoning, and multi-modal understanding, where diverse solution strategies are especially valuable. Future work should further study safety-aware Co-Evolve to ensure that increased diversity improves capability without amplifying unreliable or misaligned behavior. Since a more diverse population may also diversify failure modes, disagreement-based auditing and conservative aggregation may be useful safeguards. Furthermore, because both the baseline methods and Co-Evolve require substantial computational resources, we report single-run results in this paper. While prior work suggests that these baselines have relatively modest run-to-run variance under comparable settings, a more comprehensive multi-seed evaluation of Co-Evolve remains an important direction for future work.

7 Conclusion

In this paper, we introduced Co-Evolve, a one-to-many LLM fine-tuning paradigm that transforms a single pretrained model into a cooperative population of specialized descendants. Unlike conventional one-to-one fine-tuning, which optimizes a single policy for average performance, Co-Evolve promotes model-level complementarity through a marginal-coverage objective, assigning greater credit to examples under-covered by peer models and thereby reducing redundant optimization.

We instantiate this paradigm with evolution strategies, enabling derivative-free optimization for sparse and non-differentiable rewards, and introduce a seed-replay mechanism to reduce the storage cost of maintaining multiple descendants. Experiments across language-only benchmarks, math-specialized models, and a vision-language setting show that Co-Evolve consistently improves ensemble-level performance over strong fine-tuning and test-time voting baselines. These results indicate that the gains come from learned and persistent specialization, rather than merely from using more inference candidates.

Overall, our findings suggest that fine-tuning a pretrained LLM into a cooperative population offers a scalable alternative to optimizing a single monolithic policy, with potential extensions to population-level post-training, efficient routing, and safety-aware co-evolution.

Acknowledgments and Disclosure of Funding

This work is supported by Innovative Drug Research and Development–National Science and Technology Major Project (No. 2025ZD1802501) and Beijing Frontier Research Center for Biological Structure Fundings. Dapeng Jiang is supported by Xingjian College (College of Future Technology) Future Technology Challenge Project.

References

[1] Rui Ai, Yu Pan, David Simchi-Levi, and Chonghuan Wang. Shape-grpo: Shapley-enhanced reward allocation for multi-candidate llm training, 2026. URL https://arxiv.org/abs/2603.29871.

[2] Oron Anschel, Alon Shoshan, Adam Botach, Shunit Haviv Hakimi, Asaf Gendler, Emanuel Ben Baruch, Nadav Bhonker, Igor Kviatkovsky, Manoj Aggarwal, and Gerard Medioni. Group-aware reinforcement learning for output diversity in large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32394–32415, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1649. URL https://aclanthology.org/2025.emnlp-main.1649/.

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

[4] Yilei Chen, Souradip Chakraborty, Lorenz Wolf, Yannis Paschalidis, and Aldo Pacchiano. Post-training large language models for diverse high-quality responses, 2026. URL https://arxiv.org/abs/2509.04784.

[5] Zhijun Chen, Xiaodong Lu, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Ming Li, Likang Xiao, Dingqi Yang, Xiao Huang, Yikun Ban, Hailong Sun, and Philip S. Yu. Harnessing multiple large language models: A survey on llm ensemble, 2026. URL https://arxiv.org/abs/2502.18036.

[6] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models, 2025. URL https://arxiv.org/abs/2505.22617.

[7] Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. MAS: Stable reinforcement learning for multi-agent llm systems. *arXiv preprint arXiv:2602.08847*, 2026. URL https://arxiv.org/abs/2602.08847.

[8] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (sos): Learning to search in language, 2024. URL https://arxiv.org/abs/2404.03683.

[9] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

[10] Jinwu Hu, Yufeng Wang, Shuhai Zhang, Kai Zhou, Guohao Chen, Yu Hu, Bin Xiao, and Mingkui Tan. Efficient dynamic ensembling for multiple llm experts, 2025. URL https://arxiv.org/abs/2412.07448.

[11] Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. Ensemble learning for heterogeneous large language models with deep parallel collaboration, 2024. URL https://arxiv.org/abs/2404.12715.

[12] Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering, 2019. URL https://arxiv.org/abs/1902.09506.

[13] Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on llm generalisation and diversity, 2024. URL https://arxiv.org/abs/2310.06452.

[14] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[15] Chenyi Li, Yuan Zhang, Bo Wang, Guoqing Ma, Wei Tang, Haoyang Huang, and Nan Duan. Setpo: Set-level policy optimization for diversity-preserving llm reasoning, 2026. URL https://arxiv.org/abs/2602.01062.

[16] Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. MARFT: Multi-agent reinforcement fine-tuning. *arXiv preprint arXiv:2504.16129*, 2025. URL https://arxiv.org/abs/2504.16129.

[17] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and evaluation framework for deeper understanding of commonsense stories, 2016. URL https://arxiv.org/abs/1604.01696.

[18] Sonia Krishna Murthy, Tomer Ullman, and Jennifer Hu. One fish, two fish, but not the whole sea: Alignment reduces language models' conceptual diversity. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, page 11241–11258. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.561. URL http://dx.doi.org/10.18653/v1/2025.naacl-long.561.

[19] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024. URL https://arxiv.org/abs/2406.18665.

[20] 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, 2022. URL https://arxiv.org/abs/2203.02155.

[21] Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://aclanthology.org/2025.acl-long.1459/.

[22] Xin Qiu, Yulu Gan, Conor F. Hayes, Qiyao Liang, Yinggan Xu, Roberto Dailey, Elliot Meyerson, Babak Hodjat, and Risto Miikkulainen. Evolution strategies at scale: Llm fine-tuning beyond reinforcement learning, 2026. URL https://arxiv.org/abs/2509.24372.

[23] 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, Jiaxi 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, 2025. URL https://arxiv.org/abs/2412.15115.

[24] 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, 2024. URL https://arxiv.org/abs/2305.18290.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[26] 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.

[27] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[28] Rik van der Lingen. Reaction SMILES USPTO year 2023, December 2023. URL https://figshare.com/articles/dataset/Reaction_SMILES_USPTO_year_2023/24921555. Dataset.

[29] Christian Walder and Deep Karkhanis. Pass@k policy optimization: Solving harder reinforcement learning problems, 2025. URL https://arxiv.org/abs/2505.15201.

[30] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.04692.

[31] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

[32] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

[33] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

A Derivation of the Marginal-Coverage Weight

In this section, we show that the marginal-coverage weight used in Co-Evolve can be interpreted as the marginal gain of a concave diminishing-return coverage utility. This derivation clarifies that Co-Evolve does not directly optimize the hard majority-voting objective; instead, it optimizes a smooth coverage-oriented surrogate that provides denser training signals.

Consider a population of K descendants. For an input x, let

$$R_i(x) = R(x, \theta_i) \in [0, 1]$$

denote the reward of descendant i, and let

$$S(x) = \sum_{i=1}^{K} R_i(x)$$

denote the population coverage score. When rewards are binary, $S(x)$ is the number of descendants that solve the input.

A hard majority-voting utility can be written as

$$U_{\text{maj}}(S) = \mathbf{1}[S \geq \tau], \qquad \tau = \left\lceil \frac{K+1}{2} \right\rceil.$$

The marginal contribution of descendant i under this objective is nonzero only when the peer coverage is exactly pivotal:

$$\Delta_i^{\mathrm{maj}}(x) = \mathbf{1}\left[\sum_{j\neq i} R_j(x) = \tau - 1\right].$$

This credit is sparse, because most examples are not exactly one correct vote away from changing the majority decision. Therefore, directly optimizing hard majority voting can lead to unstable training.

Co-Evolve instead uses a concave coverage surrogate. Define

$$U_{\mathrm{cov}}(S) = S - \frac{S(S-1)}{2K}.$$

This utility has diminishing returns: the first descendant that solves an example receives the largest marginal value, and later descendants solving the same example still contribute positive value but with decreasing marginal gain.

Let

$$S_{-i}(x) = \sum_{j\neq i} R_j(x)$$

be the peer coverage excluding descendant i. For binary rewards, if descendant i changes from incorrect to correct, the population coverage increases from $S_{-i}(x)$ to $S_{-i}(x)+1$. The marginal gain under U_{cov} is therefore

$$\Delta_i^{\mathrm{cov}}(x) = U_{\mathrm{cov}}(S_{-i}(x)+1) - U_{\mathrm{cov}}(S_{-i}(x)).$$

Let $s = S_{-i}(x)$. Then

$$U_{\mathrm{cov}}(s+1) = (s+1) - \frac{(s+1)s}{2K},$$

and

$$U_{\mathrm{cov}}(s) = s - \frac{s(s-1)}{2K}.$$

Thus,

$$\begin{aligned} U_{\mathrm{cov}}(s+1) - U_{\mathrm{cov}}(s) &= \left[(s+1) - \frac{(s+1)s}{2K}\right] - \left[s - \frac{s(s-1)}{2K}\right] \\ &= 1 - \frac{(s+1)s - s(s-1)}{2K} \\ &= 1 - \frac{2s}{2K} \\ &= 1 - \frac{s}{K}. \end{aligned}$$

Substituting back $s = S_{-i}(x)$ gives

$$\Delta_i^{\mathrm{cov}}(x) = 1 - \frac{S_{-i}(x)}{K} = 1 - \frac{1}{K}\sum_{j\neq i} R_j(x).$$

This is exactly the marginal-coverage weight used in Eq. 5.

B Case Study: Complementary Reasoning Paths in Co-Evolve

We present a qualitative case study on the Countdown task to illustrate how Co-Evolve improves population-level coverage through complementary reasoning paths. Consider the instance with numbers $[50, 17, 30, 5]$ and target 68. The base model is highly unstable on this example: among five sampled responses, only one produces a valid solution, while the remaining attempts either fail to return an answer or converge to incorrect expressions. In contrast, the co-evolved population retains two distinct valid solution paths. One descendant discovers an additive solution, $50 - 17 + 30 + 5 = 68$, while another discovers a multiplicative/division solution, $(50 - 30) \times 17/5 = 68$. These two expressions are not simple permutation variants; they correspond to different arithmetic structures, respectively relying on sign-pattern enumeration and factor decomposition. Meanwhile,

other descendants fail in different ways, such as returning no answer or producing an arithmetic hallucination. Under execution-based answer-level plurality voting, the two structurally different correct expressions are grouped into the same valid-answer class because both use the given numbers exactly once and evaluate to the target. As a result, the ensemble aggregation recovers the correct answer even though only a subset of descendants solve the instance. This example highlights that the gain of Co-Evolve does not merely come from repeated sampling of a single model, but from maintaining a population of descendants with complementary reasoning behaviors and less correlated failure modes.

C Model Backbones

We evaluate Co-Evolve on a set of models that vary in parameter scale, domain specialization, and modality. This design allows us to study the consistency of Co-Evolve across compact general-purpose LLMs, a math-specialized LLM, and a vision-language model to show the generality of our method. The evaluated models include `Qwen2.5-0.5B-Instruct`, `Qwen2.5-3B-Instruct`, `Qwen2.5-7B-Instruct`, `Qwen2.5-Math-1.5B-Instruct`, and `Qwen2.5-VL-3B-Instruct` [23, 3]. Qwen2.5 models are dense decoder-only language models released in multiple sizes, and the Qwen2.5 series was trained on a substantially larger pre-training dataset and with improved post-training compared with previous Qwen releases. The official Qwen2.5 report states that the pre-training corpus was scaled to 18 trillion tokens and that post-training includes supervised fine-tuning and multi-stage reinforcement learning. The release also emphasizes improvements in instruction following, long-text generation, structured-data understanding, JSON-style structured output, coding, mathematics, and multilinguality.

Qwen2.5-0.5B-Instruct. `Qwen2.5-0.5B-Instruct` is the smallest general-purpose instruction-tuned model in our evaluation suite. It is a causal language model that has undergone both pre-training and post-training, and it follows the Qwen2.5 Transformer architecture with RoPE positional encoding, SwiGLU feed-forward layers, RMSNorm, attention QKV bias, grouped-query attention, and tied word embeddings. The model contains 0.49B total parameters, of which 0.36B are non-embedding parameters, and uses 24 Transformer layers with 14 query attention heads and 2 key-value heads. Its model card specifies a full context length of 32,768 tokens and a maximum generation length of 8,192 tokens.

This model is included to test Co-Evolve under a highly resource-constrained yet instruction-following setting. Its much lower representational capacity compared with larger models makes it useful for measuring whether Co-Evolve depends on strong latent reasoning ability or large-scale memorization, while its membership in the same Qwen2.5 instruction-tuned family as the larger text-only models provides a clean low-scale anchor for analyzing scaling trends.

Qwen2.5-3B-Instruct. `Qwen2.5-3B-Instruct` is a compact general-purpose instruction-tuned LLM positioned between the 0.5B and 7B variants. Like the other Qwen2.5 text-only Instruct models, it is a causal language model trained through pre-training and post-training. The model card lists 3.09B total parameters, 2.77B non-embedding parameters, 36 Transformer layers, and grouped-query attention with 16 query heads and 2 key-value heads. Its architecture uses RoPE positional encoding, SwiGLU feed-forward layers, RMSNorm, attention QKV bias, and tied word embeddings. The official context length is 32,768 tokens, with a maximum generation length of 8,192 tokens.

We use this model as a medium-scale text-only backbone. It is large enough to exhibit substantially stronger language understanding, instruction following, and reasoning behavior than the 0.5B model, while still being computationally efficient. In the evaluation suite, `Qwen2.5-3B-Instruct` helps distinguish effects caused by model scale from those caused by domain-specific or multimodal training.

Qwen2.5-7B-Instruct. `Qwen2.5-7B-Instruct` is the largest general-purpose text-only model used in our experiments. It is an instruction-tuned causal language model from the Qwen2.5 family, sharing the same broad design principles as the smaller Qwen2.5 variants: Transformer decoder architecture, RoPE positional encoding, SwiGLU feed-forward layers, RMSNorm, attention QKV bias, and grouped-query attention. The model contains 7.61B total parameters and 6.53B non-embedding parameters. It uses 28 Transformer layers with 28 query attention heads and 4 key-value

heads. Unlike the 0.5B and 3B variants, the model card specifies a longer full context length of 131,072 tokens while maintaining the same maximum generation length of 8,192 tokens.

This model serves as the strongest general-purpose text-only baseline in our evaluation. Its larger parameter count and longer context window enable testing whether Co-Evolve benefits from stronger instruction following, richer world knowledge, and improved long-context processing. Comparing the 0.5B, 3B, and 7B models allows us to analyze the scale sensitivity of Co-Evolve within the same model family.

Qwen2.5-Math-1.5B-Instruct. `Qwen2.5-Math-1.5B-Instruct` is a math-specialized instruction-tuned model from the Qwen2.5-Math series. Unlike the general-purpose Qwen2.5 Instruct models, Qwen2.5-Math is designed primarily for mathematical problem solving. The official model card states that Qwen2.5-Math mainly supports English and Chinese math problems through Chain-of-Thought and Tool-Integrated Reasoning, and explicitly does not recommend using this series for general non-mathematical tasks. The Qwen2.5-Math series includes 1.5B, 7B, and 72B base and instruction-tuned models, as well as a math reward model.

The training pipeline of Qwen2.5-Math differs from that of the general Qwen2.5 models. According to the Qwen2.5-Math technical report and blog, the series was improved via self-improvement techniques: synthetic mathematical data generated by stronger Qwen2-Math models was added during pre-training, a math-specific reward model was trained for rejection sampling, and reinforcement learning was applied after supervised fine-tuning. The math corpus was expanded from roughly 700B tokens in the previous version to over 1T tokens, and Qwen2.5-Math-Instruct further introduced both Chinese and English SFT data as well as Tool-Integrated Reasoning data.

We include this model to isolate the effect of domain-specialized training. Although its nominal scale is smaller than the 3B and 7B general-purpose models, its post-training is explicitly optimized for mathematical reasoning. Therefore, `Qwen2.5-Math-1.5B-Instruct` allows us to test whether Co-Evolve interacts differently with a model whose reasoning behavior has been shaped by math-specific data, reward modeling, and tool-integrated reasoning objectives.

Qwen2.5-VL-3B-Instruct. `Qwen2.5-VL-3B-Instruct` is a 3B-class instruction-tuned vision-language model from the Qwen2.5-VL series. In contrast to the text-only Qwen2.5 models, it is designed to process multimodal inputs such as images, videos, and text. The official model card describes several key capabilities: recognition of common visual objects; understanding of text, charts, icons, graphics, and layouts in images; visual-agent behavior for computer and phone use; long-video understanding; event localization; object localization via bounding boxes or points; and structured output generation for documents such as invoices, forms, and tables.

Architecturally, Qwen2.5-VL introduces multimodal-specific modifications beyond the text-only Qwen2.5 models. The model extends dynamic resolution processing to video via dynamic frame-rate sampling and updates the multimodal RoPE with temporal IDs and absolute-time alignment to better represent temporal order and event timing. Its vision encoder is based on a ViT design optimized with window attention, SwiGLU, and RMSNorm, improving training and inference efficiency while aligning the visual encoder more closely with the Qwen2.5 LLM architecture. The Qwen2.5-VL family is released in 3B, 7B, and 72B sizes, and the 3B model is intended as a compact multimodal variant.

We include `Qwen2.5-VL-3B-Instruct` to evaluate Co-Evolve under a different training and modality regime. Compared with `Qwen2.5-3B-Instruct`, it has a similar nominal parameter count but has undergone vision-language alignment and multimodal training, making it useful for testing whether Co-Evolve is specific to text-only LLMs or also transfers to a multimodal instruction-tuned backbone.

D Datasets and Test Rewards

We use the first 200 samples of the original training set as the train set. We use the complete test set.

We evaluate language-only LLMs on four tasks, namely Countdown, OlympiadBench, ROCStories, and USPTO, and evaluate multimodal LLMs on GQA. These benchmarks cover symbolic mathematical reasoning, scientific problem solving, narrative reasoning, chemistry-oriented reasoning, and

grounded visual question answering. For each dataset, we define a task-specific test reward that directly measures whether the model output satisfies the required correctness criterion.

Countdown. [8] Countdown is a controlled arithmetic reasoning task derived from the numbers game. Given a target number and a small set of input numbers, the model is required to construct an arithmetic expression that evaluates exactly to the target. Although the allowed operations are elementary, the task is challenging because solving an instance often involves combinatorial search, planning over intermediate values, and avoiding locally plausible but globally invalid calculation paths. We use Countdown to test whether a model can perform precise symbolic manipulation and generate verifiable reasoning steps rather than merely producing fluent mathematical text.

The test reward for Countdown is answer-based. Specifically, we extract the expression from the `<answer>` field and evaluate whether it uses all given numbers exactly once and whether the resulting value equals the target. The final reward is

$$r \in \{0, 1\},$$

where $r = 1$ only if both the number-usage constraint and the target-value constraint are satisfied.

OlympiadBench. [9] OlympiadBench is a competition-level benchmark composed of mathematics and physics problems collected from Olympiad-style examinations. Compared with standard school-level word problems, its questions usually require deeper scientific reasoning: the model must identify relevant theorems or physical principles, decompose the problem into intermediate steps, and carry out multi-step derivations. This makes OlympiadBench a suitable benchmark for evaluating advanced problem-solving ability in language models, especially their capacity to combine domain knowledge, formal reasoning, and long-horizon inference.

For OlympiadBench, we use a binary test reward:

$$r \in \{0, 1\}.$$

Following the MATH-500 evaluation protocol, we first attempt strict extraction from `\boxed{}`. If such an answer cannot be reliably extracted, we fall back to a flexible last-number extraction rule. The extracted prediction is then compared with the ground-truth answer after multi-level LaTeX normalization and numeric comparison. The reward is 1 if the normalized prediction matches the reference answer, and 0 otherwise.

ROCStories. [17] ROCStories is a corpus of short five-sentence stories centered on everyday events. While the language in the stories is generally simple, the task requires understanding temporal order, causal relations, character intentions, and commonsense expectations. In our evaluation, ROCStories is used to probe narrative reasoning and sentence ordering ability: a strong model should not only produce grammatical text, but also maintain a coherent situation, respect the preceding context, and arrange events in a way that is consistent with human commonsense.

The test reward for ROCStories is a soft reward in $[0, 1]$ that combines absolute sentence-position accuracy and local adjacency accuracy:

$$r = 0.6 \cdot \text{PositionAcc} + 0.4 \cdot \text{AdjacentAcc}.$$

Here, PositionAcc measures the fraction of sentences placed in the correct absolute position, while AdjacentAcc measures the fraction of consecutive gold sentence pairs that appear in the correct relative order. This reward encourages both global ordering correctness and local narrative coherence.

USPTO. [28] USPTO is a large-scale chemical reaction dataset extracted from United States Patent and Trademark Office documents. Reactions are commonly represented in reaction-SMILES form, which encodes reactants, reagents, and products as structured textual sequences. Unlike general natural-language tasks, USPTO requires the model to reason over domain-specific symbolic representations and to capture regularities of chemical transformations. We use this benchmark to evaluate scientific reasoning in chemistry, including the ability to infer plausible reaction classes while preserving chemically meaningful molecular relationships.

For USPTO-50K, the test reward is binary:

$$r \in \{0, 1\}.$$

We extract an integer prediction only from the <answer>X</answer> field, without applying any fallback rule. The extracted integer is then compared against the ground-truth reaction class label in the range 1–10. The reward is 1 if the predicted class exactly matches the reference class, and 0 otherwise.

GQA. [12] GQA is a visual question answering benchmark designed to evaluate compositional reasoning over real-world images. Each image is associated with structured scene information, including objects, attributes, and relations, and questions often require multiple reasoning steps such as object recognition, spatial reasoning, comparison, and logical composition. Compared with simpler VQA settings, GQA places stronger emphasis on grounding answers in visual evidence and reducing reliance on language priors. We therefore use GQA to assess whether multimodal LLMs can connect natural-language questions with image content and perform structured visual reasoning.

For GQA, we use a binary fuzzy-matching test reward:

$$r \in \{0, 1\}.$$

We first extract a short predicted phrase from \boxed{} when available, and then apply pattern-based heuristics. Matching is performed in three stages: first, we check exact match, singular-form match, and synonym match using a hand-coded synonym map; second, we allow substring containment; third, we scan the whole response for the ground-truth word, including singular and synonym forms. No LLM judge is used. The reward is 1 if any of these matching criteria succeeds, and 0 otherwise.

E Dataset Prompt Templates

We list below the exact system and user prompt templates used for each benchmark in our experiments. Placeholder tokens enclosed in braces (e.g. {numbers}) are filled with per-instance values at inference time.

E.1 Countdown

Standard variant (<answer> tags). Used in most cases.

System

You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer.

User

Using the numbers {numbers}, create an equation that equals {target}. You can use basic arithmetic operations (+, −, *, /) and each number can only be used once. Show your work in <think> </think> tags. And return the final answer in <answer> </answer> tags, for example <answer> (1 + 2) / 3 </answer>.

Math-Boxed variant (\boxed{} format). Used for models in the Qwen2.5-Math series that natively emit \boxed{} expressions.

System

Please reason step by step, and put your final arithmetic expression inside \boxed{}. The expression must use each of the given numbers exactly once with +, −, *, / and parentheses only.

User

Using the numbers {numbers}, create an arithmetic expression that equals {target}. Each number must be used exactly once. Use only +, −, *, / and parentheses.

E.2 OlympiadBench

OlympiadBench has a single prompt variant that appends the reasoning instruction directly to the problem statement; no separate system message is used.

System

(none)

User

`{problem}` Please reason step by step, and put your final answer within `\boxed{}`.

E.3 ROCStories

System

You are a helpful assistant that excels at story comprehension and logical reasoning. Given shuffled sentences from a story, you carefully analyze the narrative flow and temporal cues to determine the correct chronological order.

User

Below are 5 sentences from a story, but they are in the wrong order.
Please arrange them in the correct chronological order.

Title: `{title}`

Sentence A: `{sentence_0}`
Sentence B: `{sentence_1}`
Sentence C: `{sentence_2}`
Sentence D: `{sentence_3}`
Sentence E: `{sentence_4}`

Output the correct order as comma-separated letters (e.g., B,A,D,E,C).
Only output the letters, nothing else.

E.4 USPTO-50K

System

You are an expert organic chemist. Your task is to classify chemical reactions into one of 10 standard reaction categories based on the transformation type.

Reaction Classes:
1: Heteroatom alkylation/arylation — N, O, S attacking C (e.g., S_N2, ether formation)
2: Acylation — Forming C = O bonds with N, O, S (e.g., amide, ester formation)
3: C–C bond formation — New C–C bonds (e.g., Suzuki, Heck, Grignard)
4: Heterocycle formation — Creating rings with N, O, S
5: Protections — Adding protecting groups (Boc, Bn, TBS, etc.)
6: Deprotections — Removing protecting groups
7: Reductions — Adding H, removing O (e.g., ketone→alcohol, nitro→amine)
8: Oxidations — Adding O, removing H (e.g., alcohol→ketone)
9: Functional group interconversion — Changing one FG to another
10: Functional group addition — Adding new FG to molecule (e.g., halogenation)

User

Classify this reaction:

Reactants » Product:
`{rxn_smiles}`

Analyze the key transformation and output the class number (1–10) in `<answer>X</answer>` tags.

E.5 GQA

GQA is a visual question answering task. Each sample is a multimodal message consisting of one image and the following text prompt; no separate system message is used.

System

(none)

User (multimodal: image + text)

[image]

Look at the image and answer the question.

Question: `{question}`

Please reason step by step, and put your final answer within `\boxed{}`.

F Ensemble Aggregation Details

For Co-Evolve and TT-MV, aggregation is performed after per-model answer extraction. The K extracted answers are combined by plurality vote: the most frequent non-`None` answer is selected as the final prediction. Task-specific extraction and correctness rules are given below.

Countdown. Each model's response is parsed to find an arithmetic expression. The expression is *validated*: it must use exactly the provided set of numbers (each consumed exactly once) and evaluate to the target integer. Only valid, evaluable expressions are admitted to the vote; invalid or unparseable responses are discarded. The vote key is the *numeric result* (i.e. two expressions that evaluate to the same integer count as the same answer, even if syntactically different).

OlympiadBench. Answers are extracted from `\boxed{...}` markup using both strict and flexible parsers. The vote key is the normalized answer string returned by the extractor. Correctness is checked by the underlying math equivalence checker (the same checker used during training).

ROCStories. Each island's response is parsed for a sentence ordering (labels A–E). The ordering is normalized to a compact string without separators (e.g. `"BCEDA"`). Responses that do not yield exactly five unique labels are discarded. The vote key is this normalized order string; an answer is correct if and only if it exactly matches the ground-truth order.

USPTO-50K. Each island emits a reaction-class integer (1–10) extracted from `<answer>...</answer>` tags. The vote key is the class label string; correctness requires exact integer equality with the ground truth.

GQA. Each model's response is parsed for a short answer string, which is lower-cased and stripped of leading/trailing whitespace before voting. The vote key is this normalized string; correctness requires exact string match with the normalized ground-truth answer.

Tie-breaking. When two or more answers share the plurality count, ties are broken by preferring the answer whose supporting islands achieved the highest training-round reward in the most recent

ES iteration. If rewards are equal or unavailable, the winner is chosen uniformly at random (seeded by the global random state).

G Additional Experiment

To further examine whether the proposed one-to-many fine-tuning paradigm remains effective with a stronger general-purpose backbone, we conduct an additional experiment on USPTO using `Qwen2.5-7B-Instruct`. USPTO is a chemistry-oriented benchmark that requires the model to infer reaction categories from reaction-SMILES inputs, making it a useful testbed for evaluating domain-specific symbolic reasoning beyond general natural-language understanding. We compare Co-Evolve with single-model fine-tuning baselines and their test-time majority-voting variants under the same evaluation protocol.

Method	**USPTO**
`Qwen2.5-7B-Instruct`	
Base	23.28
GRPO	$54.85_{+31.57}$
ES	$\underline{60.44}_{+37.16}$
`+TT-MV`	
Base+TT-MV	$22.58_{-0.70}$
GRPO+TT-MV	$54.95_{+31.67}$
ES+TT-MV	$60.13_{+36.85}$
Co-Evolve	$\mathbf{66.33}_{+43.05}$

Table 4: Performance comparison on USPTO using `Qwen2.5-7B-Instruct`. The small subscript denotes the absolute improvement over the corresponding base model. Best result is shown in bold, and second-best result is underlined.

As shown in Table 4, Co-Evolve achieves the best performance, reaching 66.33 accuracy and improving over the base model by 43.05 points. It also outperforms the strongest single-model baseline, ES, by 5.89 points, and surpasses ES+TT-MV by 6.20 points. Notably, test-time majority voting alone does not consistently improve performance on this task: Base+TT-MV slightly decreases from 23.28 to 22.58, and ES+TT-MV is slightly lower than ES. These results further support that the gain of Co-Evolve comes from learned complementary specialization among persistent descendants, rather than merely from aggregating multiple inference candidates.

H Scaling with Training-Set Size

We study how the amount of training data affects population-level performance. We use `Qwen2.5-0.5B-Instruct` on Countdown with $K = 5$, $N = 30$, $T = 250$, and $\lambda = 1$, while keeping the 2,000-example test set, model initialization, and random seeds fixed. The training sets are nested prefixes/extensions of the same split. Increasing M from 100 to 3,000 examples raises greedy cross-model MV@5 from 14.25% to 19.40%.

Because increasing M also increases the number of perturbation-example evaluations, this is an end-to-end data-and-compute scaling study rather than a fixed-FLOP data-only ablation. The gain is not monotonic for every decoding metric at larger training-set sizes; we therefore interpret this experiment as evidence that additional training data can improve collective coverage in this regime, rather than as a claim of monotonic scaling for arbitrary M.

The scaling runs use the same Qwen2.5-0.5B base model, $K = 5$, $N = 30$, $T = 250$, $\lambda = 1$, island seeds 42–46, and the fixed 2,000-example test split. The audited greedy MV@5 results for the scaling endpoints are 14.25%, 17.45%, 18.20%, and 19.40% for $M \in \{100, 400, 800, 3000\}$, respectively. The $M = 3000$ endpoint is a separately audited continuation run and is included to show the longer-range trend.

The $M = 100$ run uses packed 2/2/1 execution on three GPUs with CUDA graphs, whereas the $M \geq 400$ runs use one island per GPU with eager execution. A step-0 comparability audit preserved perturbation seeds, update order, and coefficient signs, but was not bitwise identical. These results

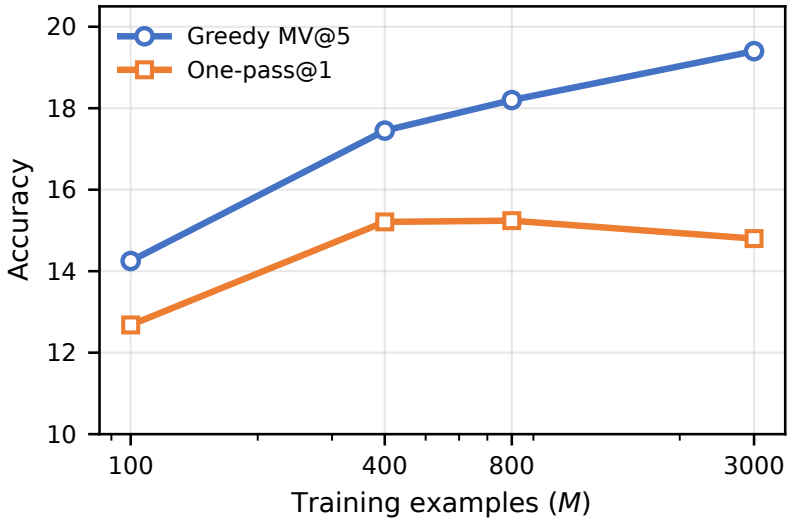

Figure 6: Training-data scaling on Countdown with Qwen2.5-0.5B-Instruct.

should therefore be read as a verified training-data-and-compute scaling comparison, not as a fixed-compute causal estimate of training-set size.

The $M = 3000$ continuation uses the same K, N, T, λ, seeds, and test set, but its audit records use the legacy timeout-safe scorer. We report it as a separately audited endpoint rather than treating it as a bitwise matched extension of the $M = 100$–800 comparison. The corresponding replay logs, data manifests, test-set hash, and endpoint audits are retained with the release artifacts under `baseline/scaling_countdown_05b/results/`.

I Details on Omitted 0.5B Policy-Gradient Baselines

Table 1 omits PPO/GRPO results for Qwen2.5-0.5B-Instruct because these runs did not yield stable converged checkpoints under our comparable training budget. The 0.5B model is used as a low-capacity stress test, where the initial task success rate is very low and sparse verifiable rewards provide weak policy-gradient signals. In our diagnostic GRPO runs with the same VERL-based implementation used for larger models, Countdown showed no accuracy improvement, OlympiadBench only oscillated within a narrow range, and USPTO-50K either terminated early or exhibited entropy collapse rather than stable convergence.

We therefore report only methods that produced stable checkpoints in the 0.5B setting: Base, ES, ES+TT-MV, and Co-Evolve. This omission is not intended to claim that policy-gradient RL cannot work for 0.5B models, but rather to avoid comparing non-converged or collapsed PPO/GRPO runs against converged baselines. A dedicated hyperparameter sweep for low-capacity sparse-reward RL is orthogonal to our main claim that Co-Evolve improves ensemble-level performance over stable ES and test-time voting baselines on the same backbone.

J Experimental Details

The main goal of this section is to explain how we control the training compute of different baselines. Since the evaluated methods have different optimization structures, we match them by leading-order aggregate training FLOPs rather than by the raw number of training iterations. One can find more details of calculation in original ES paper [22].

J.1 Compute-Matched Baseline Control

Let P denote the number of model parameters and let ℓ denote the average effective sequence length of a prompt-response trajectory. We use the standard dense-Transformer approximation

$$C_{\text{fwd}} \approx 2P\ell, \qquad C_{\text{bwd}} \approx 4P\ell. \tag{11}$$

This accounting is used only to match leading-order model computation across methods. It does not attempt to predict exact wall-clock time, which depends on implementation details such as GPU utilization, communication, vLLM scheduling [14], reward checking, and optimizer overhead.

Reference budget. We take the aggregate training cost of Co-Evolve as the reference budget. In our main experiments, Co-Evolve uses $K = 5$ models, $N = 30$ perturbations per model, mini-batch size $M = 200$, and $T = 250$ ES rounds. Since each perturbation-example evaluation requires one forward pass, the leading-order training cost is

$$\begin{aligned} C_0 = C_{Co-Evolve} &\approx 2TKNMP\ell \\ &= 2 \times 250 \times 5 \times 30 \times 200\, P\ell \\ &= 15{,}000{,}000P\ell. \end{aligned} \tag{12}$$

Equivalently, Co-Evolve performs 7.5×10^6 forward-equivalent model evaluations. All trainable baselines are configured to match this budget up to rounding.

ES baselines. For ES-based methods, the leading-order training cost is

$$C_{\text{ES}} \approx 2TKNMP\ell. \tag{13}$$

The independent-models ES baseline, ES-Ind, uses the same configuration as Co-Evolve but removes the cooperative specialization signal by setting $\lambda = 0$. Therefore, ES-Ind uses $K = 5, N = 30, M = 200, T = 250$ and has the same training cost C_0.

The single-model ES baseline, ES-1, sets $K = 1$. To match the same aggregate compute, we increase its number of ES rounds:

$$T_{\text{ES-1}} = \frac{15{,}000{,}000}{2 \times 30 \times 200} = 1250. \tag{14}$$

Thus ES-1 receives the same total ES compute as the full five-models Co-Evolve population, but it does not have model-level specialization.

Gradient-based RL baselines. For GRPO and PPO, let U denote the number of rollout/update steps, B the number of prompts per update, and G the number of sampled responses per prompt. GRPO requires one policy forward pass, one reference-model forward pass, and one policy backward pass per sampled response, giving

$$C_{\text{GRPO}} \approx U_{\text{grpo}}BG(2P\ell + 2P\ell + 4P\ell) = 8U_{\text{grpo}}BGP\ell. \tag{15}$$

With $B = 200$ and $G = 8$, matching C_0 gives

$$U_{\text{grpo}} = \text{round}\left(\frac{15{,}000{,}000}{8 \times 200 \times 8}\right) = 1172. \tag{16}$$

PPO additionally trains a critic/value model. Assuming the critic has the same parameter scale as the policy, each sampled response further requires a critic forward and backward pass, giving

$$C_{\text{PPO}} \approx U_{\text{ppo}}BG(2P\ell + 2P\ell + 4P\ell + 2P\ell + 4P\ell) = 14U_{\text{ppo}}BGP\ell. \tag{17}$$

With $B = 200$ and $G = 4$, matching C_0 gives

$$U_{\text{ppo}} = \text{round}\left(\frac{15{,}000{,}000}{14 \times 200 \times 4}\right) = 1339. \tag{18}$$

This protocol controls for training compute in three ways. First, ES-Ind uses exactly the same number of model evaluations as Co-Evolve, isolating the effect of marginal-coverage credit. Second, ES-1 is given five times more ES rounds than each individual Co-Evolve model, so it matches the aggregate compute of the full population. Third, GRPO and PPO are run for the number of rollout/update steps implied by the same reference budget C_0. Therefore, the performance difference is not attributable simply to using more training FLOPs.

Table 5: Compute-matched training schedules. All trainable baselines are matched to the Co-Evolve reference budget $C_0 = 15{,}000{,}000 P\ell$ up to rounding.

Method	Configuration	Rounds / updates	Training FLOPs
Co-Evolve	$K = 5,\ N = 30,\ M = 200,\ \lambda = 1$	$T = 250$	$15.0000\mathrm{M}P\ell$
ES-Ind	$K = 5,\ N = 30,\ M = 200,\ \lambda = 0$	$T = 250$	$15.0000\mathrm{M}P\ell$
ES-1	$K = 1,\ N = 30,\ M = 200$	$T = 1250$	$15.0000\mathrm{M}P\ell$
GRPO	$B = 200,\ G = 8$	$U = 1172$	$15.0016\mathrm{M}P\ell$
PPO	$B = 200,\ G = 4$	$U = 1339$	$14.9968\mathrm{M}P\ell$

J.2 Inference Settings

Greedy decoding and self-consistency majority voting are inference-only baselines and therefore incur no additional training FLOPs. A single greedy decode costs approximately $2P\ell$ per query. A majority-voting baseline with q candidates costs approximately

$$C^{\text{infer}}_{\text{MV@}q} \approx 2qP\ell. \tag{19}$$

At inference time, Co-Evolve queries K persistent descendants, so its inference cost is

$$C^{\text{infer}}_{Co-Evolve} \approx 2KP\ell. \tag{20}$$

When comparing Co-Evolve with majority voting, we set $q = K$ to match the number of test-time model queries. Thus the inference budget is controlled, while the source of diversity differs: majority voting samples multiple outputs from one policy, whereas Co-Evolve uses persistent descendants trained with marginal-coverage credit.

For decoding, greedy evaluation uses temperature 0 and one sampled output. Majority voting uses temperature 0.7, top-$p = 0.95$, and majority-vote aggregation. All methods use bfloat16 precision and the vLLM backend when applicable.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide such information in Introduction and Abstract.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the Sec. 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.

- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: See Sec. 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We release the core training implementation at `https://github.com/jdp22/Co-Evolution-Core`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: You can find the core training implementation at https://github.com/jdp22/Co-Evolution-Core. All of the datasets can be downloaded from the Internet.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: You can find the details of datasets in Appendix. D and details of experiment settings in Appendix. J.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Because both the baseline methods and Co-Evolve require substantial computational resources, we report single-run results in this paper. While prior work suggests that these baselines have relatively modest run-to-run variance under comparable settings, a more comprehensive multi-seed evaluation of Co-Evolve remains an important direction for future work.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We use NVIDIA A100-SXM4-80GB GPUs. The computational resources needed can be found at Appendix. J.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have include the societal impacts in Sec. 6.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:we do not release pretrained generative models or other assets with a high risk of misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The related paper is properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide a anonymized URL.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.