ESSAM: A Novel Competitive Evolution Strategies Approach to Reinforcement Learning for Memory Efficient LLMs Fine-Tuning

Zhishen Sun[1,2*] **Sizhe Dang**[1,2*] **Guang Dai**[1] **Haishan Ye**[1,2†]
[1]SGIT AI Lab, State Grid Shaanxi Electric Power Company Limited
[2]Xi'an Jiaotong University
{szs777,darknight1118}@stu.xjtu.edu.cn
guang.gdai@gmail.com
yehaishan@xjtu.edu.cn

Abstract

Reinforcement learning (RL) has become a key training step for improving mathematical reasoning in large language models (LLMs), but it often has high GPU memory usage, which makes it hard to use in settings with limited resources. To reduce these issues, we propose Evolution Strategies with Sharpness-Aware Maximization (ESSAM), a full parameter fine-tuning framework that tightly combines the zeroth-order search in parameter space from Evolution Strategies (ES) with the Sharpness-Aware Maximization (SAM) to improve generalization. We conduct fine-tuning experiments on the mainstream mathematical reasoning task GSM8K. The results show that ESSAM achieves an average accuracy of 78.27% across all models and its overall performance is comparable to RL methods. It surpasses classic RL algorithm PPO with an accuracy of 77.72% and is comparable to GRPO with an accuracy of 78.34%, and even surpassing them on some models. Further generalization experiments show that the models trained with ESSAM exhibit stronger generalization ability. Their average performance achieves the best results on 5 out of 6 datasets, indicating that ESSAM can effectively improve the generalization performance of fine-tuned models. In terms of GPU memory usage, ESSAM reduces the average GPU memory usage by $18\times$ compared to PPO and by $10\times$ compared to GRPO, achieving an extremely low GPU memory usage. In addition, we design an accelerated variant of ESSAM, which achieves nearly a twofold speedup while maintaining the same GPU memory usage as ESSAM, and attains an average accuracy of 78.02% across all models, outperforming PPO. Code:https://github.com/szs777/ESSAM.

1 Introduction

Recent breakthroughs in the mathematical reasoning ability of large language models (LLMs) have led to their deep integration with many scientific fields such as mathematics and physics, making them an indispensable part of modern life [7, 18, 11, 3]. These gains in mathematical reasoning are largely driven by the rise of Reinforcement learning (RL) as a fine-tuning method [14, 15, 20, 21]. However, despite its effectiveness, applying RL to fine-tune LLMs requires extremely high GPU memory resources [4], such as fine-tuning an 8B model with PPO, 314.44 GiB of GPU memory is

*Equal contribution. This work was completed during the internship at SGIT AI Lab, State Grid Shaanxi Electric Power Company Limited.

†Corresponding author.

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

required (Appendix H), which is not friendly to resource constrained open-source communities, and limits the further development of using RL to fine-tune LLMs.

To mitigate the issue that RL requires extremely high GPU memory resources, [13] explore using Evolution Strategies (ES) to fine-tune LLMs. ES is a population based zeroth-order method, which does not rely on gradients and updates model parameters using only forward generation and reward evaluation. Therefore, ES can significantly reduce GPU memory usage and training cost. However, on mainstream mathematical reasoning tasks such as GSM8K [2], the overall performance of ES is often limited and difficult to match the performance of RL methods. The reason lies in the fact that ES only explores within the model's parameter space and then directly updates the parameters. This update approach tends to drive the model toward sharp minima, resulting in poor generalization performance. In addition, in their ES experiments, [13] use only 200 examples for training and 2,000 examples for evaluation. This split is not a standard setup, and during training the 200 examples are used as a whole without the common practice of shuffling the training data and iterating with small mini batches. This can introduce more randomness and differs from standard LLMs fine-tuning pipelines. So the practical performance of ES still needs further study.

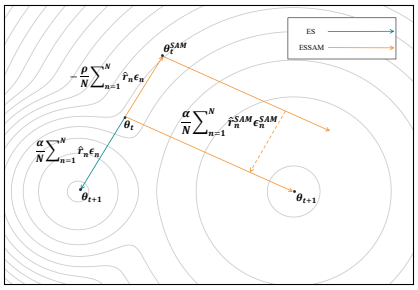


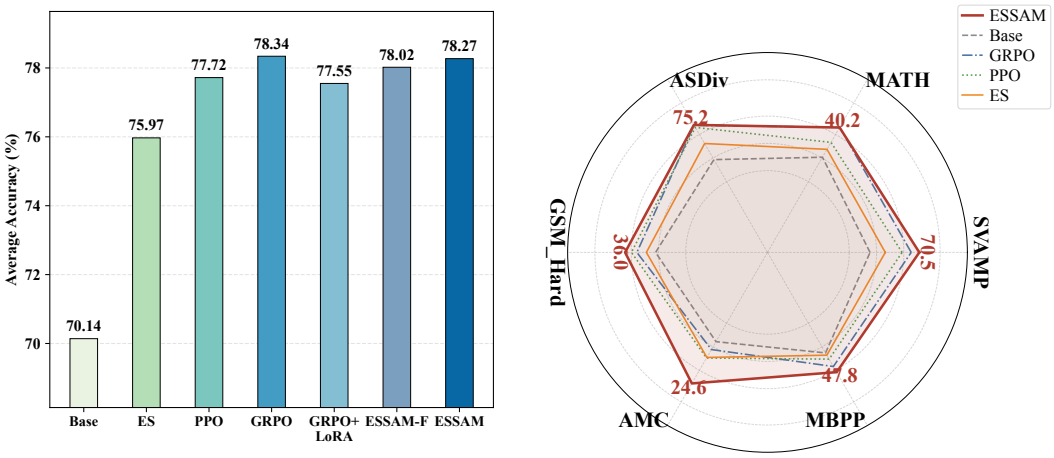
Figure 1: An illustration of the ESSAM parameter update.

Recent work on Sharpness-Aware Minimization (Maximization) has shown strong gains in generalization [5]. Its key idea is to reduce both the loss value and the loss sharpness, so that training moves toward flat minima where the loss stays low in a whole neighborhood of the parameters. This can reduce overfitting and make the model more robust. We introduce the idea of SAM into zeroth-order search and propose a new zeroth-order fine-tuning method called Evolution Strategies with Sharpness-Aware Maximization (ESSAM). As shown in Figure 1, ESSAM tightly integrates the sharpness-aware perturbation mechanism of SAM with ES, it uses the update direction that would otherwise move the original parameters toward sharp regions to perform a reverse update, obtains a new update direction at the perturbed parameters, and then uses this new direction to update the original parameters, so that the solution of the model is guided to converge to flatter regions, enhancing the model's robustness and generalization ability. In addition, we further propose an accelerated variant of ESSAM. This accelerated algorithm adopts a SAM mechanism based on Frobenius norm normalization during the reverse update and uses a smaller SAM population to improve training efficiency while maintaining low GPU memory usage.

We follow a more standard training procedure. We use a standard split between training data and evaluation data, shuffle the training data, and perform multi step updates with small mini batches, rather than training once on a small set of samples as a whole. This reduces randomness from small sample training and better averages the noise induced variation across iterations, and the training signal is smoother and the optimization process is more stable.

Experimental results show that ESSAM extends ES to the mathematical reasoning task GSM8K and significantly outperforms standard ES while maintaining low GPU memory usage. As shown in Figure 2a and Table 1, ESSAM achieves an average accuracy of 78.27% across all models, surpassing PPO with 77.72% and GRPO+LoRA with 77.55%, while remaining close to GRPO with 78.34%, and it even outperforms RL methods on some models. In addition, we evaluate the fine-tuned models on five more challenging out-of-distribution mathematical reasoning benchmarks and one verifiable code benchmark. The models fine-tuned with ESSAM achieve the best average performance on 5 out of these 6 benchmarks (Figure 2b), demonstrating stronger generalization ability. We also evaluate the accelerated variant of ESSAM, namely ESSAM-F. As shown in Table 2, ESSAM-F achieves nearly a twofold speedup while maintaining the same GPU memory usage as ESSAM, and still attains an average accuracy of 78.02% across all models, outperforming PPO overall. We summarize our contributions as follows:

- We propose a new zeroth-order full-parameter fine-tuning framework, ESSAM. It combines ES with the Sharpness-Aware Maximization mechanism, introduces neighborhood probing and a two-stage evaluation update in parameter space, and guides optimization toward flatter and more robust solutions.



(a) GSM8K average accuracy

(b) Generalization performance

Figure 2: (a) The average accuracy of each algorithm across all models on the GSM8K task (%). (b) The average performance of each algorithm across datasets over all models.

- We extend zeroth-order full parameter fine-tuning to the mainstream mathematical reasoning task GSM8K. We also adopt a more standard data usage and training procedure, including a standard split of training and evaluation sets, shuffling the training data, and performing multi-step updates with small mini-batches, which improves training stability and reproducibility.
- Experimental results show that ESSAM achieves performance comparable to mainstream RL methods on GSM8K and significantly outperforms standard ES. Further out-of-distribution generalization experiments show that ESSAM effectively improves the generalization ability of fine-tuned models.
- We systematically evaluate GPU memory usage. As shown in Figure 4, ESSAM maintains the same inference-level GPU memory usage as ES, and on average reduces GPU memory usage by 18× compared with PPO, by 10× compared with GRPO, and by approximately 4× compared with GRPO+LoRA, greatly reducing the demand for computational resources.
- We further propose an accelerated variant of ESSAM, namely ESSAM-F. In the first-stage reverse update, it adopts a SAM mechanism based on Frobenius norm normalization and uses a smaller SAM population, achieving nearly a twofold speedup while maintaining the same GPU memory usage.

2 Related Work

Evolution Strategies. Evolution Strategies (ES) are population based zeroth-order optimization methods that update parameters using fitness scores from randomly perturbed candidates. Classic methods such as CMA-ES [8] and Natural Evolution Strategies [17] laid the foundation for applying evolutionary search to neural network optimization. In LLM settings, early studies mainly focused on small scale or parameter-efficient subspaces, such as fine-tuning the last layer of mT5 [16] or optimizing LoRA parameters [9]. Recent work [13] scales ES to full parameter fine-tuning of billion-parameter LLMs through engineering and parallelization advances, but its generalization on mainstream mathematical reasoning tasks remains limited. Therefore, we introduce a sharpness-aware mechanism into ES to improve its generalization performance.

Reinforcement learning. RL fine-tuning has become an important approach for improving the performance of LLMs [1, 23, 12], especially their mathematical reasoning ability [7]. PPO [14] and GRPO [15] are two widely used RL methods for LLM training. PPO stabilizes policy updates with a clipped surrogate objective and usually relies on value function estimation, while GRPO improves efficiency by removing the value model and using group-based rewards as a baseline. Although these methods can effectively improve LLM performance, they still require substantial GPU memory due to backpropagation, policy optimization, and additional training states.

Sharpness-Aware Minimization (Maximization). SAM [5] improves generalization by guiding models toward flatter regions. In language model settings, prior work shows that sharpness-aware training can reduce overfitting and improve downstream performance. Several studies further improve SAM from the perspectives of stability and efficiency, such as selecting perturbed parameters based on Fisher information [22] or reducing the extra training cost by modifying the perturbation and update procedure [10]. When gradients are unavailable or only forward passes can be used, sharpness-aware ideas have also been extended to black-box and zeroth-order optimization [19, 6]. Inspired by these works, we combine SAM with ES and propose a zeroth-order sharpness-aware method for improving the mathematical reasoning ability of LLMs.

3 Method

In this section, we detail the ESSAM algorithm flow (Sec. 3.1), and present the efficient memory-saving mechanism used in the algorithm (Sec. 3.2). Finally, we explain why our proposed algorithm can be viewed as performing Sharpness-Aware Maximization (Sec. 3.3).

3.1 Evolution Strategies with Sharpness-Aware Maximization

ES generates a population of perturbation models with Gaussian noise and updates parameters in a reward-weighted manner, exploring and updating the parameter space only once in each iteration (Appendix A). Different from standard ES, our ESSAM method introduces the SAM mechanism. It first uses the current reward weighted aggregation to move the model parameters in the opposite direction of reward increase to a nearby neighborhood point, and then uses the perturbations and rewards computed at this point to update the parameters. This process helps avoid sharp solutions and drives the model toward flatter and more robust regions, thereby improving generalization.

Specifically, given a pretrained LLM with parameters θ_0 and a reward function $R(\theta)$, we aim to maximize the expected reward by iteratively updating model parameters. At iteration t, we sample N i.i.d. Gaussian noises $\epsilon_n \sim \mathcal{N}(0, I)$ for $n = 1, \ldots, N$, and construct perturbed models $\theta_t^{(n)} = \theta_t + \sigma\epsilon_n$. We evaluate each perturbed model to obtain reward scores $r_n = R\Big(\theta_t^{(n)}\Big)$ for $n = 1, \ldots, N$. We then apply z-score normalization [13] to keep the reward scale consistent:

$$\mu_r = \frac{1}{N}\sum_{n=1}^{N} r_n, \qquad s_r = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(r_n - \mu_r)^2}, \qquad \hat{r}_n = \frac{r_n - \mu_r}{s_r + \delta}, \quad n = 1, \ldots, N. \tag{1}$$

where δ is a small constant for numerical stability.

Next, different from standard ES, which directly updates parameters using $\hat{r}_n$, we use $\hat{r}_n$ to compute a weighted aggregation of the corresponding noise directions. This forms a combined perturbation for neighborhood probing, and we control its magnitude using the hyperparameter ρ, which perturbs the parameters from θ_t to a neighborhood point θ_t^{SAM}. The perturbation is given by:

$$\theta_{t+1}^{\mathrm{SAM}} = \theta_t - \rho \cdot \frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N} \hat{r}_n\, \epsilon_n. \tag{2}$$

Above operations are shown in the first stage of ESSAM shown in Algorithm 1.

After obtaining $\theta_{t+1}^{\mathrm{SAM}}$, we sample another N i.i.d. Gaussian noises $\epsilon_n^{\mathrm{SAM}} \sim \mathcal{N}(0, I)$ at $\theta_{t+1}^{\mathrm{SAM}}$ and repeat the same steps of perturbation, evaluation, and z-score normalization, producing normalized rewards $\hat{r}_n^{\mathrm{SAM}}$ at $\theta_{t+1}^{\mathrm{SAM}}$. Finally, we restore the parameters back to the original θ_t, and use $\hat{r}_n^{\mathrm{SAM}}$ together with their corresponding noise to form a weighted aggregation for the final update with learning rate α, where we have absorbed $1/\sigma$ into α:

$$\theta_{t+1} = \theta_t + \alpha \cdot \frac{1}{N}\sum_{n=1}^{N} \hat{r}_n^{\mathrm{SAM}}\, \epsilon_n^{\mathrm{SAM}}. \tag{3}$$

Above operations are shown in the second stage of ESSAM shown in Algorithm 1.

Algorithm 1 Evolution Strategies with Sharpness-Aware Maximization (ESSAM)

Input: initial parameters θ_0, reward function R, population size N, noise scale σ, hyperparameter ρ, learning rate α, iterations T, number of parallel processes P
Output: fine-tuned parameters θ_T
Create P processes, each instantiates a model with the same initial parameters θ_0, with one process as the main process
for $t = 0$ **to** $T - 1$ **do**
  (Stage 1) Build a SAM neighborhood point θ_t^{SAM}
  Sample N random seeds $s_1, \ldots, s_N$
  $\{r_n\}_{n=1}^N \leftarrow SRE(\theta_t, R, \sigma, \{s_n\}_{n=1}^N, N, P)$
  Compute mean μ_r, std s_r
  Normalize $\hat{r}_n = \frac{r_n - \mu_r}{s_r + \delta}$
  $\theta_t^{\mathrm{SAM}} \leftarrow DIPU(\theta_t, \{s_n\}_{n=1}^N, \{\hat{r}_n\}_{n=1}^N, -\rho, N)$
  Broadcast parameters of all processes to θ_t^{SAM}
  (Stage 2) Evaluate around θ_t^{SAM} **and update** θ_t
  Sample N random seeds $s_1^{\mathrm{SAM}}, \ldots, s_N^{\mathrm{SAM}}$
  $\{r_n^{\mathrm{SAM}}\}_{n=1}^N \leftarrow SRE(\theta_t^{\mathrm{SAM}}, R, \sigma, \{s_n^{\mathrm{SAM}}\}_{n=1}^N, N, P)$
  Compute mean μ_r^{SAM}, std s_r^{SAM}
  Normalize $\hat{r}_n^{\mathrm{SAM}} = \frac{r_n^{\mathrm{SAM}} - \mu_r^{\mathrm{SAM}}}{s_r^{\mathrm{SAM}} + \delta}$
  $\theta_t \leftarrow DIPU(\theta_t^{\mathrm{SAM}}, \{s_n\}_{n=1}^N, \{\hat{r}_n\}_{n=1}^N, \rho, N)$
  $\theta_{t+1} \leftarrow DIPU(\theta_t^{\mathrm{SAM}}, \{s_n^{\mathrm{SAM}}\}_{n=1}^N, \{\hat{r}_n^{\mathrm{SAM}}\}_{n=1}^N, \alpha, N)$
  Broadcast parameters of all processes to θ_{t+1}
end for

In addition, we propose an accelerated variant of ESSAM, called ESSAM-F (Algorithm 3). The difference between ESSAM-F and ESSAM lies in how the SAM neighborhood point is constructed in the first stage. ESSAM-F introduces a SAM mechanism based on Frobenius norm normalization. It first uses a SAM population N_{SAM}, which is much smaller than N, and applies a centered reward weighted aggregation of the sampled noise directions to move the model parameters to a nearby neighborhood point. At this point, the perturbation magnitude is normalized by the Frobenius norm of the aggregated direction. It then uses the perturbations and rewards computed at this neighborhood point to update the original parameters. The Appendix B presents the detailed algorithmic procedure of ESSAM-F.

3.2 Memory Saving Mechanism

To make ESSAM practical for fine-tuning LLMs, we adopt the memory saving techniques proposed by [13] to reduce GPU memory usage during training. The detailed implementation is shown in Algorithms 4, 5 and 6 in Appendix C. Both stages of ESSAM and its accelerated variant use these memory saving techniques. Together with the zeroth-order update that relies on forward generation and reward evaluation rather than backpropagation, they greatly reduce GPU memory usage and avoid the high memory cost seen in RL methods.

Algorithm 1 and Algorithm 3 show the detailed procedure of ESSAM and ESSAM-F.

3.3 Understanding the algorithm design of ESSAM

In the SAM setting, when constructing the SAM neighborhood point in the first stage, its structural form is as follows [5]: $\theta_{t+1}^{\mathrm{SAM}} = \theta_t - \rho \cdot \frac{g}{\|g\|}$, Where g is a gradient direction.

In our setting, the gradient estimate is computed as:

$$g_t = \frac{1}{\sigma} \frac{1}{N} \sum_{n=1}^{N} (r_n - \mu_r)\, \epsilon_n, \tag{4}$$

The estimated variance s_r^2 is computed as:

$$s_r^2 = \frac{1}{N-1}\sum_{n=1}^{N}(r_n - \mu_r)^2, \tag{5}$$

In our algorithm, the SAM neighborhood point is constructed as follows: $\theta_{t+1}^{\mathrm{SAM}} = \theta_t - \rho \cdot \frac{g_t}{s_r}$.

Proposition 1 shows why the first stage update of ESSAM is essentially performing SAM.

Proposition 1. *Let the stochastic gradient estimation g_t defined in Eq.* (4) *and the variance s_r^2 defined in Eq.* (5)*. Then it holds that*

$$\mathbb{E}\big[g_t\big] = \frac{N-1}{N}\nabla R(\theta) + \tau, \qquad \tau = O(\sigma) \tag{6}$$

$$\mathbb{E}\big[\|g_t\|^2\big] = \frac{N^3 + (d-2)N^2 + (3-2d)N + d - 2}{N^3} \cdot \big\|\nabla R(\theta)\big\|^2 + \gamma, \quad \gamma = O(\sigma). \tag{7}$$

$$\mathbb{E}\big[s_r^2\big] = \sigma^2 \big\|\nabla R(\theta)\big\|^2 + \zeta, \qquad \zeta = O(\sigma^3). \tag{8}$$

$$\mathbb{E}\big[\|g_t\|^2\big] \approx \frac{N^3 + (d-2)N^2 + (3-2d)N + d - 2}{N^3} \cdot \sigma^{-2} \cdot \mathbb{E}\big[s_r^2\big].$$

where d denotes the number of model parameters.

Since $\frac{N^3+(d-2)N^2+(3-2d)N+d-2}{N^3}\cdot\sigma^{-2}$ is a constant, $\frac{g_t}{s_r}$ and $\frac{g_t}{\|g_t\|}$ can be regarded as equivalent, scaled only by a constant factor. Therefore, the first stage update of our proposed algorithm is essentially performing Sharpness-Aware Maximization. Detailed proof of Proposition 1 is in Appendix D.

4 Experiment

We conduct experiments on a mainstream mathematical reasoning task GSM8K. Section 4.1 presents the detailed results and the generalization results. Section 4.2 provides an analysis of GPU memory usage and computation. Section 4.3 presents the ablation study and analyzes the sensitivity to hyperparameters. Section 4.4 experimentally evaluates the performance, runtime, and GPU memory usage of ESSAM-F. The Appendix E presents the detailed experimental setup.

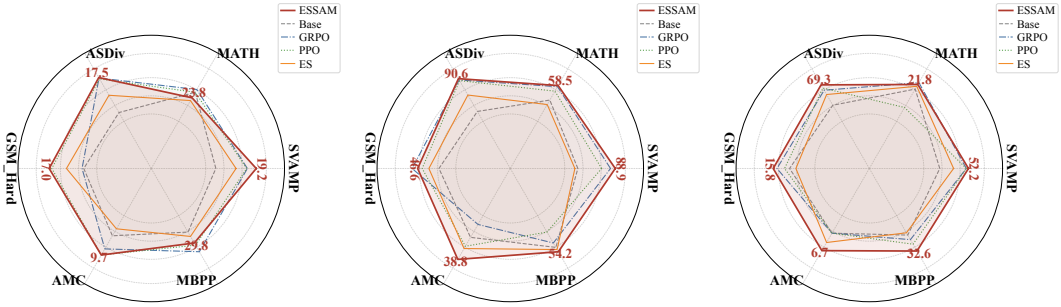
4.1 Main Results

On the mainstream mathematical reasoning task GSM8K, ESSAM significantly improves the performance of all tested models and achieves performance on par with mainstream RL methods. As shown in Table 1, we systematically compare the fine-tuning performance of ESSAM with ES, PPO, GRPO, and GRPO+LoRA across different model families and model sizes. Overall, ESSAM brings stable improvements on all tested models and significantly outperforms standard ES. The average accuracy of ESSAM reaches 78.27%, showing a clear improvement over the original models with 70.14% and standard ES with 75.97%. This shows that, after introducing the sharpness-aware mechanism, ESSAM can effectively alleviate the performance limitation of standard ES on mathematical reasoning tasks and improve the overall effectiveness of zeroth-order fine-tuning.

Compared with mainstream RL methods, ESSAM also shows strong competitiveness. Its average accuracy of 78.27% exceeds the 77.72% achieved by PPO and the 77.55% achieved by GRPO+LoRA, while remaining close to the 78.34% achieved by GRPO. ESSAM outperforms GRPO+LoRA on six of the seven models, further demonstrating its effectiveness compared with parameter-efficient RL methods. Specifically, on the Qwen2.5 family, ESSAM achieves the best results on the 1.5B and 3B models, reaching 78.92% and 87.71%, respectively. On the 0.5B and 7B models, ESSAM also outperforms PPO and remains close to GRPO. On the LLaMA family, ESSAM brings stable

Table 1: Accuracy of different algorithms on different models and average accuracy for the GSM8K task (%). Best results are in bold, and second-best results are underlined. Additional multi-seed results are provided in Appendix F.

METHOD	QWEN2.5-INSTRUCT				LLAMA-INSTRUCT			AVERAGE
	0.5B	1.5B	3B	7B	3.2-1B	3.2-3B	3.1-8B	
ORIGINAL	46.47	73.77	84.84	90.52	39.04	74.60	81.73	70.14
PPO	53.90	<u>78.85</u>	87.19	91.28	61.49	83.02	88.29	77.72
GRPO	**54.24**	78.39	<u>87.34</u>	**93.02**	**62.32**	**84.00**	**89.08**	**78.34**
GRPO+LORA	53.30	77.94	86.13	<u>92.57</u>	61.41	83.32	88.17	77.55
ES	52.46	75.97	86.66	90.98	57.24	80.97	87.54	75.97
ESSAM(OURS)	<u>54.06</u>	**78.92**	**87.71**	<u>92.57</u>	<u>61.79</u>	<u>83.93</u>	<u>88.93</u>	<u>78.27</u>
ESSAM-F(OURS)	<u>54.06</u>	78.70	87.19	92.27	61.41	83.85	88.63	78.02



(a) Qwen2.5-0.5B-Instruct (b) Qwen2.5-3B-Instruct (c) LLaMA3.2-1B-Instruct

Figure 3: Performance of models fine-tuned with different algorithms on out-of-distribution benchmarks. The generalization results for the other models are presented in Appendix G.

improvements on the 1B, 3B, and 8B models, and its results are all higher than PPO and close to GRPO. The Appendix K presents example responses generated by models fine-tuned with ESSAM.

ESSAM can improve the generalization ability of the model. We further evaluate the fine-tuned models on five more challenging out-of-distribution mathematical reasoning benchmarks and one verifiable code generation benchmark. The results show that ESSAM exhibits stronger generalization ability on these downstream tasks. After averaging the results over all models, ESSAM achieves the best average performance on 5 out of the 6 benchmarks, and its overall average performance across all models and all datasets is also higher than ES, PPO, and GRPO (Figure 2b). More detailed results are shown in Figure 3, on Qwen2.5-0.5B-Instruct, ESSAM performs best on SVAMP, ASDiv, and GSM-Hard, and is tied for the best result on AMC. On Qwen2.5-3B-Instruct, ESSAM achieves the best results on 5 out of the 6 benchmarks. On LLaMA3.2-1B-Instruct, ESSAM also achieves the best results on 5 out of the 6 benchmarks. These results show that the advantage of ESSAM is not limited to the training task GSM8K, but can generalize well to more challenging out-of-distribution tasks, demonstrating stronger cross-task transfer ability and generalization ability.

4.2 GPU Memory Usage and Time Efficiency Analysis

ESSAM can fine-tune LLMs using only inference-level GPU memory, greatly reducing GPU memory usage. We measure and compare the GPU memory usage of base models with different sizes and architectures during training. The results in Figure 4 show a highly consistent and clear trend. Across all models, ESSAM maintains the same extremely low, inference-level GPU memory usage as standard ES. This indicates that, with the help of memory-saving techniques, introducing the SAM mechanism, neighborhood probing, and the two-stage evaluation update into the ES framework does not introduce additional GPU memory overhead. In contrast, the GPU memory usage of PPO and GRPO increases rapidly as the model size grows, often reaching hundreds of GiB on larger models. This is mainly due to their reliance on backpropagation and optimizer states, with PPO additionally requiring a value model. Compared with PPO, ESSAM achieves about $15\times$–$21\times$ GPU

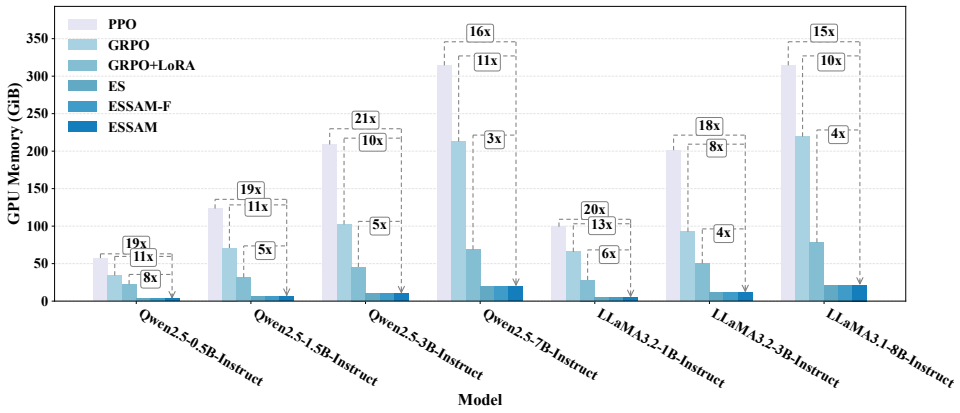
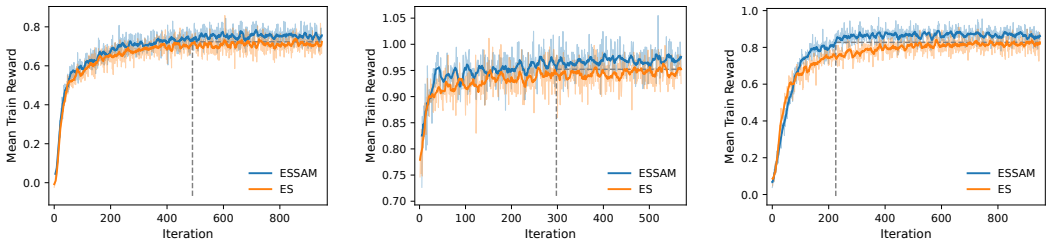
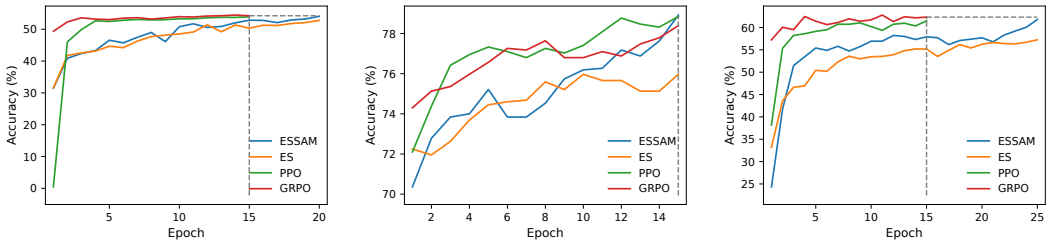


Figure 4: GPU memory usage when fine-tuning different LLMs with different algorithms. The annotated ratios indicate the GPU memory usage of each baseline relative to ESSAM. More details can be found in Appendix H.



(a) Qwen2.5-0.5B-Instruct (b) Qwen2.5-1.5B-Instruct (c) LLaMA3.2-1B-Instruct

Figure 5: The training mean reward curves of ESSAM and ES. These curves show that ESSAM has better training trend and converges earlier than ES, leading to better computational efficiency. More results are presented in the Appendix I.



(a) Qwen2.5-0.5B-Instruct (b) Qwen2.5-1.5B-Instruct (c) LLaMA3.2-1B-Instruct

Figure 6: Test accuracy curves of models fine-tuned with different algorithms during training. More results are shown in the Appendix J.

memory savings across different models; compared with GRPO, it achieves about $8\times$–$13\times$ savings. On average, ESSAM reduces GPU memory usage by $18\times$ compared with PPO and by $10\times$ compared with GRPO. Although GRPO+LoRA reduces the GPU memory requirements of GRPO, its average GPU memory usage remains approximately $4\times$ that of ESSAM. This substantially reduces the demand for computational resources while maintaining inference-level GPU memory usage, making stable LLM reasoning fine-tuning feasible in resource constrained settings.

ESSAM has good time efficiency and shows a clear advantage in iteration efficiency compared with standard ES. As shown in Figure 5, ESSAM converges faster than ES and reaches, at an early stage of training, the reward level that ES only approaches near the end of training. This indicates that, under the same target reward, ESSAM needs fewer iterations to achieve a similar training result. Figure 6 further shows how model accuracy changes with training progress. From the accuracy perspective, ESSAM can achieve fine-tuning performance comparable to RL methods with no more than twice the training progress, or even with the same training progress, while ES struggles to reach this level under the same training progress as ESSAM. Although ESSAM introduces an additional

Table 2: Training time in hours when fine-tuning different LLMs with different algorithms.

Base Model	GRPO	GRPO+LoRA	ES	ESSAM	ESSAM-F
Qwen-2.5-0.5B-It	13.5	44	39	45	27
Qwen-2.5-1.5B-It	26.5	77	19	34	20
Qwen-2.5-3B-It	48	126	40	91	41
Qwen-2.5-7B-It	106	159.5	71	107	67
LLaMA-3.2-1B-It	19	74	18	33	20
LLaMA-3.2-3B-It	49	127.5	55	116	52
LLaMA-3.1-8B-It	105	173	117	233	113

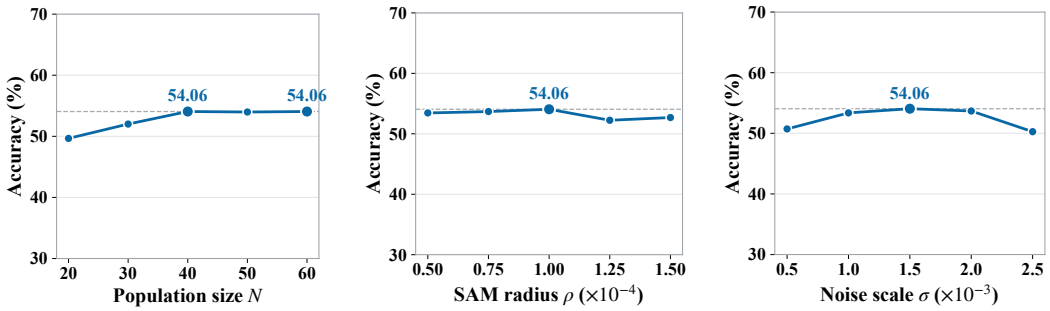


Figure 7: Ablation study of the population size N, noise scale σ, and SAM radius ρ on Qwen2.5-0.5B-Instruct. The results show that ESSAM is sensitive to these hyperparameters but remains relatively stable within a reasonable range.

round of neighborhood probing and reward evaluation in each iteration, which slightly increases the per-iteration cost, its faster convergence leads to better overall training efficiency. In other words, the extra cost in each iteration is partly offset by the reduced number of iterations needed to reach a strong training result. This makes ESSAM more efficient than standard ES when the goal is to reach a target reward or target accuracy. This also suggests that the SAM mechanism helps the optimization reach flatter and more robust regions faster, thereby improving time efficiency in practice.

Table 2 reports the runtime of different algorithms when fine-tuning different LLMs. In Table 2, although ESSAM takes longer than GRPO and ES, the runtime overhead varies across models. Except for Qwen2.5-0.5B-Instruct, where ESSAM takes about 3.3 times the runtime of GRPO, the runtime of ESSAM on most other models is approximately 2 times that of GRPO, while the two methods have nearly identical runtimes on Qwen2.5-7B-Instruct. Compared with GRPO+LoRA, ESSAM requires less training time on five of the seven models. This extra runtime is acceptable because ESSAM can achieve GRPO-level fine-tuning performance while using only inference-level GPU memory.

4.3 Ablation Study

We conduct an ablation study on Qwen2.5-0.5B-Instruct to investigate the sensitivity of ESSAM to different hyperparameters. The experimental results are shown in Figure 7. Overall, ESSAM is sensitive to the choices of the population size N, noise scale σ, and SAM radius ρ, but remains relatively stable within a reasonable range. For N, performance improves as the population size increases from 20 to 40, but the gain becomes saturated when N is further increased to 50 and 60, suggesting that a moderate population size can provide effective search directions while balancing performance and efficiency. For σ, overly small values lead to insufficient exploration, while overly large values introduce excessive perturbations, and both degrade performance. For ρ, ESSAM is relatively stable when ρ is small, but performance starts to fluctuate and drop when ρ exceeds a certain value. The best performance is achieved at $\rho = 1.0 \times 10^{-4}$, indicating that a proper neighborhood perturbation strength helps guide the model toward flatter and more robust regions.

4.4 ESSAM-F

We evaluate the accelerated variant of ESSAM. As shown in Table 1, although ESSAM-F suffers a slight performance drop compared with the original ESSAM, its overall performance remains highly competitive. In terms of average accuracy across all models, ESSAM-F achieves 78.02%, which is only 0.25 percentage points lower than ESSAM's 78.27%, while still outperforming PPO's 77.72% and remaining close to GRPO's 78.34%. Looking at the results on individual models, ESSAM-F matches ESSAM on Qwen2.5-0.5B-Instruct, matches PPO on Qwen2.5-3B-Instruct, and still outperforms PPO on Qwen2.5-7B-Instruct, LLaMA3.2-3B-Instruct, and LLaMA3.1-8B-Instruct, indicating that it can preserve performance well while reducing computational cost. At the same time, ESSAM-F has a clear advantage in runtime efficiency. Since it uses a smaller SAM population in the reverse update of the first stage, it achieves nearly a twofold speedup (Table 2) while maintaining the same GPU memory usage as ESSAM (Figure 4). Across all seven models, ESSAM-F also requires less total training time than GRPO, GRPO+LoRA, and standard ES. These results show that ESSAM-F further improves the practical training efficiency of the method with almost no sacrifice in performance, and achieves a better balance among performance, runtime, and GPU memory usage.

5 Conclusion

In this paper, we propose ESSAM, a zeroth-order full parameter fine-tuning method that combines Evolution Strategies with Sharpness-Aware Maximization to improve the mathematical reasoning ability of LLMs under limited computational resources. ESSAM relies only on forward generation and rule-based reward evaluation, maintaining inference-level GPU memory usage that is much lower than PPO, GRPO, and GRPO+LoRA. Experiments show that ESSAM consistently improves all tested models, outperforms standard ES, achieves higher average accuracy than GRPO+LoRA, and maintains performance comparable to PPO and GRPO. Further generalization experiments show that ESSAM fine-tuned models perform strongly on out-of-distribution benchmarks. We also propose an accelerated variant, ESSAM-F, which achieves nearly a twofold speedup with the same GPU memory usage, requires less total training time than GRPO across all seven models, and still outperforms PPO overall. Therefore, our work offers a practical and efficient option for resource-constrained LLM math reasoning fine-tuning.

6 Limitations

Although ESSAM substantially reduces GPU memory usage, its additional neighborhood evaluations introduce some training time overhead, which is partially mitigated by ESSAM-F. Given the available computational resources, we conducted multi-seed experiments on three representative models, and extending these evaluations to more settings would provide a more comprehensive assessment of the method's stability. Furthermore, the performance of ESSAM at larger model scales remains worth exploring. Future work will also investigate its applicability to a broader range of tasks and model architectures, including code-oriented training, agentic reinforcement learning, and multimodal models, to further evaluate its generalization and versatility.

Acknowledgments and Disclosure of Funding

This work was supported in part by the fund of State Grid Shaanxi Electric Power Company Limited under Grant 5226PX250001. We declare that we have no competing interests related to this work.

References

[1] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[3] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[4] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf, 2024.

[5] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[6] Yao Fu, Yihang Jin, Chunxia Zhang, Junmin Liu, and Haishan Ye. Zero-order sharpness-aware minimization. *arXiv preprint arXiv:2511.09156*, 2025.

[7] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[8] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[9] Feihu Jin, Yifan Liu, and Ying Tan. Derivative-free optimization for low-rank adaptation in large language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[10] Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12360–12370, 2022.

[11] OpenAI. O3 and o4 mini system card, 2025.

[12] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[13] Xin Qiu, Yulu Gan, Conor F Hayes, Qiyao Liang, Elliot Meyerson, Babak Hodjat, and Risto Miikkulainen. Evolution strategies at scale: Llm fine-tuning beyond reinforcement learning. *arXiv preprint arXiv:2509.24372*, 2025.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[15] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[16] Orlando Grabiel Toledano-López, Julio Madera, Hector González, Alfredo Simón-Cuevas, Thomas Demeester, and Erik Mannens. Fine-tuning mt5-based transformer via cma-es for sentiment analysis. In *IberLEF@ SEPLN*, 2022.

[17] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[18] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[19] Feiyang Ye, Yueming Lyu, Xuehao Wang, Masashi Sugiyama, Yu Zhang, and Ivor Tsang. Sharpness-aware black-box optimization. *arXiv preprint arXiv:2410.12457*, 2024.

[20] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[21] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[22] Qihuang Zhong, Liang Ding, Li Shen, Peng Mi, Juhua Liu, Bo Du, and Dacheng Tao. Improving sharpness-aware minimization with fisher mask for better generalization on language models. *arXiv preprint arXiv:2210.05497*, 2022.

[23] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Evolution Strategies

Algorithm 2 Evolution Strategies (ES)

Input: initial parameters θ_0, reward function R, population size N, noise scale σ, learning rate α, iterations T
Output: model parameters θ_T
for $t = 0$ **to** $T - 1$ **do**
  for $n = 1$ **to** N **do**
    Sample Gaussian noise $\epsilon_n \sim \mathcal{N}(0, I)$
    Construct perturbed parameters $\theta_t^{(n)} = \theta_t + \sigma\epsilon_n$
    Evaluate reward $r_n = R(\theta_t^{(n)})$
  end for
  Compute mean $\mu_r = \frac{1}{N}\sum_{n=1}^{N} r_n$
  Compute std $s_r = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(r_n - \mu)^2}$
  for $n = 1$ **to** N **do**
    Normalize reward $\hat{r}_n = \frac{r_n - \mu_r}{s_r + \varepsilon}$
  end for
  Update parameters $\theta_{t+1} = \theta_t + \alpha \cdot \frac{1}{N}\sum_{n=1}^{N} \hat{r}_n\, \epsilon_n$
end for

Algorithm 2 presents the basic framework of using ES to fine-tune LLMs.

Specifically, Given a pretrained LLM with parameters θ_0 and a reward function $R(\theta)$, we aim to maximize the expected reward by iteratively updating model parameters. At iteration t, we sample N i.i.d. Gaussian noises $\epsilon_n \sim \mathcal{N}(0, I)$ for $n = 1, \ldots, N$, and construct perturbed models:

$$\theta_t^{(n)} = \theta_t + \sigma\epsilon_n, \quad n = 1, \ldots, N. \tag{9}$$

We evaluate each perturbed model to obtain reward scores:

$$r_n = R\left(\theta_t^{(n)}\right), \quad n = 1, \ldots, N. \tag{10}$$

We then apply z-score normalization to keep the reward scale consistent:

$$\mu_r = \frac{1}{N}\sum_{n=1}^{N} r_n, \qquad s_r = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(r_n - \mu_r)^2}, \tag{11}$$

$$\hat{r}_n = \frac{r_n - \mu_r}{s_r + \varepsilon}, \quad n = 1, \ldots, N, \tag{12}$$

where ε is a small constant for numerical stability. Finally, we update the parameters by aggregating perturbations weighted by normalized rewards:

$$\theta_{t+1} = \theta_t + \alpha \cdot \frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N} \hat{r}_n\, \epsilon_n. \tag{13}$$

To reduce tuning complexity, we absorb $1/\sigma$ into the learning rate α. The update rule becomes:

$$\theta_{t+1} = \theta_t + \alpha \cdot \frac{1}{N}\sum_{n=1}^{N} \hat{r}_n\, \epsilon_n. \tag{14}$$

B ESSAM with Frobenius-Normalized SAM Perturbation

Algorithm 3 ESSAM with Frobenius-Normalized SAM Perturbation (ESSAM-F)

Input: initial parameters θ_0, reward function R, ES population size N, SAM population size N_{SAM}, noise scale σ, SAM radius ρ, learning rate α, iterations T, number of parallel processes P
Output: fine-tuned parameters θ_T
Create P processes, each instantiates a model with the same initial parameters θ_0, with one process as the main process
for $t = 0$ **to** $T-1$ **do**
  (Stage 1) Build a Frobenius-normalized SAM neighborhood point θ_t^{SAM}
  Sample N_{SAM} random seeds $s_1, \ldots, s_{N_{\mathrm{SAM}}}$
  $\{r_n\}_{n=1}^{N_{\mathrm{SAM}}} \leftarrow SRE(\theta_t, R, \sigma, \{s_n\}_{n=1}^{N_{\mathrm{SAM}}}, N, P)$
  Compute mean μ_r
  $g_t^{\mathrm{norm}} \leftarrow \textsc{WeightedNoiseFrobNorm}(\{s_n\}_{n=1}^{N_{\mathrm{SAM}}}, \{r_n - \mu_r\}_{n=1}^{N_{\mathrm{SAM}}}, N)$
  $\theta_t^{\mathrm{SAM}} \leftarrow DIPU\Big(\theta_t, \{s_n\}_{n=1}^{N_{\mathrm{SAM}}}, \{r_n - \mu_r\}_{n=1}^{N_{\mathrm{SAM}}}, -\frac{\rho}{g_t^{\mathrm{norm}}+\epsilon}, N\Big)$
  Broadcast parameters of all processes to θ_t^{SAM}
  (Stage 2) Evaluate around θ_t^{SAM} **and update** θ_t
  Sample N random seeds $s_1^{\mathrm{SAM}}, \ldots, s_N^{\mathrm{SAM}}$
  $\{r_n^{\mathrm{SAM}}\}_{n=1}^{N} \leftarrow SRE(\theta_t^{\mathrm{SAM}}, R, \sigma, \{s_n^{\mathrm{SAM}}\}_{n=1}^{N}, N, P)$
  Compute mean μ_r^{SAM}, std s_r^{SAM}
  Normalize $\hat{r}_n^{\mathrm{SAM}} = \frac{r_n^{\mathrm{SAM}} - \mu_r^{\mathrm{SAM}}}{s_r^{\mathrm{SAM}}+\delta}$
  $\theta_t \leftarrow DIPU\Big(\theta_t^{\mathrm{SAM}}, \{s_n\}_{n=1}^{N_{\mathrm{SAM}}}, \{r_n - \mu_r\}_{n=1}^{N_{\mathrm{SAM}}}, \frac{\rho}{g_t^{\mathrm{norm}}+\epsilon}, N\Big)$
  $\theta_{t+1} \leftarrow DIPU(\theta_t, \{s_n^{\mathrm{SAM}}\}_{n=1}^{N}, \{\hat{r}_n^{\mathrm{SAM}}\}_{n=1}^{N}, \alpha, N)$
  Broadcast parameters of all processes to θ_{t+1}
end for

Algorithm 3 presents the detailed algorithmic procedure of ESSAM-F, the accelerated variant of ESSAM. The difference between ESSAM-F and ESSAM lies in the way the SAM neighborhood point is constructed in the first stage, it introduces a SAM mechanism based on Frobenius norm normalization. It first uses a smaller SAM population and a centered reward weighted aggregation of the sampled noise directions to move the model parameters to a nearby neighborhood point, where the perturbation magnitude is normalized by the Frobenius norm of the aggregated direction. It then uses the perturbations and rewards computed at this neighborhood point to update the original parameters.

Specifically, given a pretrained LLM with parameters θ_0 and a reward function $R(\theta)$, we aim to maximize the expected reward by iteratively updating model parameters. At iteration t, we first sample $N_{\mathrm{SAM}} (N_{\mathrm{SAM}} \ll N)$ i.i.d. Gaussian noises $\epsilon_n \sim \mathcal{N}(0, I)$ for $n = 1, \ldots, N_{\mathrm{SAM}}$, and construct perturbed models:

$$\theta_t^{(n)} = \theta_t + \sigma\epsilon_n, \qquad n = 1, \ldots, N_{\mathrm{SAM}}. \tag{15}$$

We evaluate each perturbed model to obtain reward scores:

$$r_n = R\Big(\theta_t^{(n)}\Big), \qquad n = 1, \ldots, N_{\mathrm{SAM}}. \tag{16}$$

We compute the reward mean:

$$\mu_r = \frac{1}{N_{\mathrm{SAM}}} \sum_{n=1}^{N_{\mathrm{SAM}}} r_n. \tag{17}$$

Then, instead of applying z-score normalization, we directly use centered rewards to compute a weighted aggregation of the sampled noise directions:

$$g_t = \frac{1}{N_{\mathrm{SAM}}} \sum_{n=1}^{N_{\mathrm{SAM}}} (r_n - \mu_r)\epsilon_n. \tag{18}$$

To normalize its scale, we compute the Frobenius norm

$$g_t^{\mathrm{norm}} = \|g_t\|_F, \tag{19}$$

and construct the SAM neighborhood point as

$$\theta_t^{\mathrm{SAM}} = \theta_t - \rho \cdot \frac{g_t}{g_t^{\mathrm{norm}} + \epsilon}, \tag{20}$$

where ρ is the SAM radius and ϵ is a small constant for numerical stability.

After obtaining θ_t^{SAM}, The procedure of the second stage is exactly the same as that of the second stage in ESSAM, and the population size is N. The detailed procedure of ESSAM-F is shown in Algorithm 3.

C Memory-Saving Mechanism

Algorithm 4 Seed Replay Eval (SRE)

Input: parameters θ, reward function R, noise scale σ, seeds $\{s_n\}_{n=1}^N$, population size N, number of parallel processes P
Output: rewards $\{r_n\}_{n=1}^N$
Assign seeds $\{s_n\}_{n=1}^N$ to P processes for fully-parallel evaluations
for $n = 1$ **to** N **do**
  For the process handling s_n, reset its random number generator using seed s_n
  for each LLM layer **do**
    Using seed s_n, sample noise $\epsilon_{n,l} \sim \mathcal{N}(0, I)$
    Perturb in-place: $\theta_l \leftarrow \theta_l + \sigma\epsilon_{n,l}$
  end for
  Evaluate reward $r_n = R(\theta)$ using greedy decoding
  For the process handling s_n, reset its random number generator using seed s_n
  for each LLM layer **do**
    Using seed s_n, sample noise $\epsilon_{n,l} \sim \mathcal{N}(0, I)$
    Restore in-place: $\theta_l \leftarrow \theta_l - \sigma\epsilon_{n,l}$
  end for
end for

Algorithm 5 Decomposed In-place Update (DIPU)

Input: parameters θ, seeds $\{s_n\}_{n=1}^N$, weights $\{w_n\}_{n=1}^N$, step size η, population size N
Output: updated parameters θ
for $n = 1$ **to** N **do**
  For the process handling s_n, reset its random number generator using seed s_n
  for each LLM layer **do**
    Using seed s_n, sample noise $\epsilon_{n,l} \sim \mathcal{N}(0, I)$
    Update in-place: $\theta_l \leftarrow \theta_l + \eta \cdot \frac{1}{N}\, w_n\, \epsilon_{n,l}$
  end for
end for

Algorithm 6 Weighted Noise Frobenius Norm

Input: seeds $\{s_n\}_{n=1}^N$, coefficients $\{c_n\}_{n=1}^N$, population size N
Output: Frobenius norm $g^{\text{norm}} = \left\| \sum_{n=1}^N c_n \epsilon_n \right\|_F$
$\text{total_sq} \leftarrow 0$
for each LLM layer l **do**
  $u_l \leftarrow 0$
  for $n = 1$ **to** N **do**
    Reset random number generator using seed s_n
    Using seed s_n, sample noise $\epsilon_{n,l} \sim \mathcal{N}(0, I)$
    $u_l \leftarrow u_l + c_n \epsilon_{n,l}$
  end for
  $\text{total_sq} \leftarrow \text{total_sq} + \|u_l\|_F^2$
end for
$g^{\text{norm}} \leftarrow \sqrt{\text{total_sq}}$

D Proof

By the Taylor's expansion, we have

$$R(\theta+\sigma\epsilon_n) = R(\theta) + \sigma\left\langle \nabla R(\theta), \epsilon_n\right\rangle + \alpha(\theta, \sigma\epsilon_n), \tag{21}$$

where

$$\alpha(\theta, \sigma\epsilon_n) \triangleq R(\theta+\sigma\epsilon_n) - \Big(R(\theta) + \sigma\left\langle \nabla R(\theta), \epsilon_n\right\rangle\Big). \tag{22}$$

By the $\mathcal{L}$-smoothness, we can obtain that

$$\mathbb{E}[|\alpha(\theta, \sigma\epsilon_n)|] \leq \mathbb{E}\left[\frac{\mathcal{L}\sigma^2}{2}\left\|\epsilon_n\right\|^2\right] \leq \frac{(d+2)\mathcal{L}\sigma^2}{2}, \tag{23}$$

Lemma 1. *Let the estimated variance s_r be defined in Eq.* (5) *Then it holds that*

$$\mathbb{E}\big[s_r^2\big] = \sigma^2\left\|\nabla R(\theta)\right\|^2 + \zeta, \tag{24}$$

with

$$\begin{aligned}\zeta = \mathbb{E}&\left[\frac{1}{N-1}\sum_{i=1}^N\left(\alpha(\theta,\sigma\epsilon_i) - \frac{1}{N}\sum_{j=1}^N\alpha(\theta,\sigma\epsilon_j)\right)^2\right]\\ &+\mathbb{E}\left[\frac{2\sigma}{N-1}\sum_{i=1}^N\left\langle\nabla R(\theta),\ \epsilon_i - \frac{1}{N}\sum_{j=1}^N\epsilon_j\right\rangle\left(\alpha(\theta,\sigma\epsilon_i) - \frac{1}{N}\sum_{j=1}^N\alpha(\theta,\sigma\epsilon_j)\right)\right].\end{aligned}$$

Moreover,

$$|\zeta| \leq \frac{(d+2)^2\mathcal{L}^2\sigma^4}{4} + \frac{4N}{N-1}(d+1)^{\frac{1}{2}}(d+2)\mathcal{L}\sigma^3\left\|\nabla R(\theta)\right\|. \tag{25}$$

Proof. First, we have

$$\begin{aligned}&R(\theta+\sigma\epsilon_i) - \frac{1}{N}\sum_{j=1}^N R(\theta+\sigma\epsilon_j)\\ &\overset{\text{Eq. (21)}}{=} R(\theta) + \sigma\left\langle\nabla R(\theta),\epsilon_i\right\rangle + \alpha(\theta,\sigma\epsilon_i) - \frac{1}{N}\sum_{j=1}^N\Big(R(\theta) + \sigma\left\langle\nabla R(\theta),\epsilon_j\right\rangle + \alpha(\theta,\sigma\epsilon_j)\Big)\\ &= \sigma\left\langle\nabla R(\theta),\ \epsilon_i - \frac{1}{N}\sum_{j=1}^N\epsilon_j\right\rangle + \alpha(\theta,\sigma\epsilon_i) - \frac{1}{N}\sum_{j=1}^N\alpha(\theta,\sigma\epsilon_j).\end{aligned} \tag{26}$$

Define

$$\beta_i \triangleq \alpha(\theta,\sigma\epsilon_i) - \frac{1}{N}\sum_{j=1}^N\alpha(\theta,\sigma\epsilon_j).$$

Then

$$\begin{aligned}\sum_{i=1}^N\left(R(\theta+\sigma\epsilon_i) - \frac{1}{N}\sum_{j=1}^N R(\theta+\sigma\epsilon_j)\right)^2 &= \sum_{i=1}^N\left(\sigma\left\langle\nabla R(\theta),\ \epsilon_i - \frac{1}{N}\sum_{j=1}^N\epsilon_j\right\rangle + \beta_i\right)^2\\ &= \sum_{i=1}^N\Bigg(\sigma^2\left\langle\nabla R(\theta),\ \epsilon_i - \frac{1}{N}\sum_{j=1}^N\epsilon_j\right\rangle^2 + \beta_i^2\\ &\quad + 2\sigma\left\langle\nabla R(\theta),\ \epsilon_i - \frac{1}{N}\sum_{j=1}^N\epsilon_j\right\rangle\beta_i\Bigg).\end{aligned} \tag{27}$$

Furthermore, it holds that

$$\mathbb{E}\left[\sum_{i=1}^{N}\beta_i^2\right] = \mathbb{E}\left[\left(1-\frac{1}{N}\right)\sum_{i=1}^{N}\alpha^2(\theta,\sigma\epsilon_i)\right] \leq (N-1)\,\frac{\mathcal{L}^2\sigma^4(d+2)^2}{4}, \tag{28}$$

and

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{N}\left|\left\langle \nabla R(\theta),\ \epsilon_i - \frac{1}{N}\sum_{j=1}^{N}\epsilon_j\right\rangle \beta_i\right|\right] &\leq \mathbb{E}\left[N\left\|\left\langle \nabla R(\theta), \epsilon_i - \frac{1}{N}\sum_{j=1}^{N}\epsilon_j\right\rangle\right\| \|\beta_i\|\right] \\ &\overset{\text{Eq. (23)}}{\leq} 2N(d+1)^{\frac{1}{2}}(d+2)\mathcal{L}\sigma^2\left\|\nabla R(\theta)\right\|. \end{aligned} \tag{29}$$

Therefore, we can obtain that

$$\begin{aligned}\mathbb{E}\big[s_r^2\big] &= \mathbb{E}\Bigg[\frac{1}{N-1}\sum_{i=1}^{N}\Bigg(\sigma^2\left\langle \nabla R(\theta), \epsilon_i - \frac{1}{N}\sum_{j=1}^{N}\epsilon_j\right\rangle^2 \\ &\qquad + \beta_i^2 + 2\sigma\left\langle \nabla R(\theta), \epsilon_i - \frac{1}{N}\sum_{j=1}^{N}\epsilon_j\right\rangle \beta_i\Bigg)\Bigg] \\ &= \sigma^2\left\|\nabla R(\theta)\right\|^2 + \zeta. \end{aligned} \tag{30}$$

Finally,

$$\begin{aligned}|\zeta| &\leq \mathbb{E}\left[\frac{1}{N-1}\sum_{i=1}^{N}\beta_i^2 + \frac{2\sigma}{N-1}\sum_{i=1}^{N}\left|\left\langle \nabla R(\theta),\ \epsilon_i - \frac{1}{N}\sum_{j=1}^{N}\epsilon_j\right\rangle \beta_i\right|\right] \\ &\leq \frac{(d+2)^2\mathcal{L}^2\sigma^4}{4} + \frac{4N}{N-1}(d+1)^{\frac{1}{2}}(d+2)\mathcal{L}\sigma^3\left\|\nabla R(\theta)\right\|. \end{aligned} \tag{31}$$

□

Lemma 2. *Let the stochastic gradient estimation g_t be defined in Eq.* (4) *Then its norm satisfies*

$$\mathbb{E}\big[\|g_t\|^2\big] = \frac{N^3 + (d-2)N^2 + (3-2d)N + d - 2}{N^3}\left\|\nabla R(\theta)\right\|^2 + \gamma, \tag{32}$$

with

$$\begin{aligned}\gamma = \mathbb{E}\Bigg[&\frac{1}{\sigma^2}\frac{1}{N^2}\left(\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n\right)^2 + \frac{1}{\sigma^2}\frac{1}{N^4}\left(\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\right)^2 \\ &- 2\frac{1}{\sigma^2}\frac{1}{N^3}\left(\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n\right)^{\top}\left(\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\right) \\ &+ 2\Bigg\langle\frac{1}{N}\sum_{n=1}^{N}\epsilon_n\epsilon_n^{\top}\nabla R(\theta) - \frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\big\langle\nabla R(\theta),\epsilon_i\big\rangle\,\epsilon_n, \\ &\frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n - \frac{1}{\sigma}\frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\Bigg\rangle\Bigg]. \end{aligned} \tag{33}$$

Furthermore, it holds that

$$|\gamma| \leq \frac{(d+2)^2\mathcal{L}^2\sigma^2}{N} + \frac{4(d+2)\mathcal{L}\,\sigma}{N}\left\|\nabla R(\theta)\right\|. \tag{34}$$

Proof. First, we have

$$g_t = \frac{1}{N}\sum_{n=1}^{N}\epsilon_n\epsilon_n^\top\nabla R(\theta) \;-\; \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\big\langle\nabla R(\theta),\,\epsilon_j\big\rangle\,\epsilon_i \\ +\;\frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n - \frac{1}{\sigma}\frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n \tag{35}$$

Thus,

$$\begin{aligned}\mathbb{E}\big[g_t\big] &= \mathbb{E}\big[\frac{1}{N}\sum_{n=1}^{N}\epsilon_n\epsilon_n^\top\nabla R(\theta) \;-\; \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\big\langle\nabla R(\theta),\,\epsilon_j\big\rangle\,\epsilon_i\big] \\ &\quad+\;\mathbb{E}\big[\frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n - \frac{1}{\sigma}\frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\big] \\ &= \frac{N-1}{N}\nabla R(\theta)+\tau\end{aligned} \tag{36}$$

$$|\tau| = \mathbb{E}\big[\frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n - \frac{1}{\sigma}\frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\big] \le \mathcal{L}\sigma(d+2) \tag{37}$$

Define

$$A \triangleq \frac{1}{N}\sum_{n=1}^{N}\epsilon_n\epsilon_n^\top\nabla R(\theta) \;-\; \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\big\langle\nabla R(\theta),\,\epsilon_j\big\rangle\,\epsilon_i, \tag{38}$$

$$B \triangleq \frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n - \frac{1}{\sigma}\frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n. \tag{39}$$

$$\begin{aligned}\|g_t\|^2 &= \left\|\frac{1}{N}\sum_{n=1}^{N}\epsilon_n\epsilon_n^\top\nabla R(\theta) \;-\; \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\big\langle\nabla R(\theta),\,\epsilon_j\big\rangle\,\epsilon_i\right\|^2 \\ &\quad+\;\left\|\frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n - \frac{1}{\sigma}\frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\right\|^2 \\ &\quad+2\Bigg\langle\frac{1}{N}\sum_{n=1}^{N}\epsilon_n\epsilon_n^\top\nabla R(\theta) - \frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\big\langle\nabla R(\theta),\epsilon_i\big\rangle\,\epsilon_n, \\ &\qquad\frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n - \frac{1}{\sigma}\frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\Bigg\rangle.\end{aligned} \tag{40}$$

Then

$$\|A\|^2 = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\Big\langle \epsilon_i\epsilon_i^\top \nabla R(\theta),\ \epsilon_j\epsilon_j^\top \nabla R(\theta)\Big\rangle + \frac{1}{N^4}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\Big\langle \langle \nabla R(\theta), \epsilon_i\rangle\, \epsilon_j,\ \langle \nabla R(\theta), \epsilon_k\rangle\, \epsilon_l\Big\rangle - \frac{2}{N^3}\left\langle \sum_{n=1}^{N}\epsilon_n\epsilon_n^\top \nabla R(\theta),\ \sum_{i=1}^{N}\sum_{j=1}^{N}\langle \nabla R(\theta), \epsilon_i\rangle\, \epsilon_j \right\rangle. \tag{41}$$

$$\mathbb{E}\left[\|A\|^2\right] = \frac{N+d}{N}\big\|\nabla R(\theta)\big\|^2 \ + \ \frac{3(N-1)+d+1}{N^3}\big\|\nabla R(\theta)\big\|^2 \ - \ \frac{2(N+d)}{N^2}\,\big\|\,\nabla R(\theta)\big\|^2 = \frac{N^3+(d-2)N^2+(3-2d)N+d-2}{N^3}\big\|\,\nabla R(\theta)\big\|^2 \tag{42}$$

$$\|B\|^2 = \frac{1}{\sigma^2}\frac{1}{N^2}\left\|\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n\right\|^2 \ + \ \frac{1}{\sigma^2}\frac{1}{N^4}\left\|\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\right\|^2 - \ 2\,\frac{1}{\sigma^2}\frac{1}{N^3}\left(\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n\right)^\top\left(\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\right). \tag{43}$$

$$\left\langle A, B\right\rangle = \left\langle \frac{1}{N}\sum_{n=1}^{N}\epsilon_n\epsilon_n^\top\nabla R(\theta) - \frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\big\langle \nabla R(\theta),\epsilon_i\big\rangle\,\epsilon_n,\ \frac{1}{\sigma}\frac{1}{N}\sum_{n=1}^{N}\alpha(\theta,\sigma\epsilon_n)\,\epsilon_n - \frac{1}{\sigma}\frac{1}{N^2}\sum_{n=1}^{N}\sum_{i=1}^{N}\alpha(\theta,\sigma\epsilon_i)\,\epsilon_n\right\rangle. \tag{44}$$

$$\gamma = \mathbb{E}\left[\|B\|^2\right] + 2\mathbb{E}\left[\left\langle A, B\right\rangle\right] \tag{45}$$

Thus,

$$\mathbb{E}\big[\|g_t\|^2\big] = \mathbb{E}\left[\|A\|^2\right] + \mathbb{E}\left[\|B\|^2\right] + 2\mathbb{E}\left[\left\langle A, B\right\rangle\right] = \frac{N^3+(d-2)N^2+(3-2d)N+d-2}{N^3}\big\|\,\nabla R(\theta)\big\|^2 + \gamma \tag{46}$$

There are also,

$$\mathbb{E}\left[\|B\|^2\right] \le \frac{\mathcal{L}^2\sigma^2(d+2)^2}{4N} + \frac{\mathcal{L}^2\sigma^2(d+2)^2}{4N} + \frac{\mathcal{L}^2\sigma^2(d+2)^2}{2N} = \frac{\mathcal{L}^2\sigma^2(d+2)^2}{N} \tag{47}$$

$$\mathbb{E}\Big[\big\langle A, B\big\rangle\Big] \le \frac{\mathcal{L}\sigma(d+2)}{2N}\big\|\nabla R(\theta)\big\| + \frac{\mathcal{L}\sigma(d+2)}{2N}\big\|\nabla R(\theta)\big\| + \frac{\mathcal{L}\sigma(d+2)}{2N}\big\|\nabla R(\theta)\big\| + \frac{\mathcal{L}\sigma(d+2)}{2N}\big\|\nabla R(\theta)\big\| = \frac{2\mathcal{L}\sigma(d+2)}{N}\big\|\nabla R(\theta)\big\|. \tag{48}$$

Finally,

$$|\gamma| \leq \frac{(d+2)^2\mathcal{L}^2\sigma^2}{N} + \frac{4(d+2)\mathcal{L}\,\sigma}{N}\big\|\nabla R(\theta)\big\|. \tag{49}$$

□

D.1 Proof of Proposition 1

Proof. From Lemma 1 and Lemma 2, we know

$$\mathbb{E}\big[\|g_t\|^2\big] = \frac{N^3+(d-2)N^2+(3-2d)N+d-2}{N^3}\big\|\,\nabla R(\theta)\big\|^2 + \gamma, \qquad \gamma = O(\sigma), \tag{50}$$

$$\mathbb{E}\big[s_r^2\big] = \sigma^2\,\big\|\nabla R(\theta)\big\|^2 + \zeta, \qquad \zeta = O(\sigma^3). \tag{51}$$

Comparing Eq. 50 and Eq. 51, and ignoring the perturbation terms ζ and γ (which are higher-order terms with respect to σ), we can obtain that:

$$\mathbb{E}\big[\|g_t\|^2\big] = \frac{N^3+(d-2)N^2+(3-2d)N+d-2}{N^3}\cdot\sigma^{-2}\cdot\mathbb{E}\big[s_r^2\big]\,. \tag{52}$$

□

E Hyperparameter Settings

Models. In our experiments, we train the following models.

- **Qwen:** Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct;
- **Llama:** LLaMA3.2-1B-Instruct, LLaMA3.2-3B-Instruct, LLaMA3.1-8B-Instruct.

Datasets. In our experiments, we use the GSM8K dataset to train and evaluate LLMs. The problem and prompt template are shown in Figure 8. This task mainly tests text understanding, multi step reasoning, and arithmetic computation.

Problem: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

Template: You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer.
Please solve the following problem: {question}
Show your work in `<think>` `</think>` tags. And put the final numerical answer after `####`.

Figure 8: An example GSM8K problem and the prompt template.

Reward function. For verifiable math reasoning tasks such as GSM8K, we use a rule based reward function. The rules are as follows, we split the reward into an outcome accuracy reward and an output format reward:

$$R_{\text{accuracy}}(\hat{y}, y) = \begin{cases} 1, & \text{is_equivalent}(\hat{y}, y), \\ 0, & \text{otherwise.} \end{cases} \tag{53}$$

$$R_{\text{format}}(a) = \begin{cases} 1.0, & \text{the output follows the full format;} \\ 0.6, & \text{it contains both a } \textit{thinking marker} \\ & \text{and an } \textit{answer marker}; \\ 0.1, & \text{it contains a } \textit{thinking marker} \text{ only;} \\ 0.5, & \text{it contains an } \textit{answer marker} \text{ only;} \\ 0.0, & \text{otherwise.} \end{cases} \tag{54}$$

The final reward we use is a weighted sum of the accuracy reward and the format reward:

$$R(a, y) = R_{\text{accuracy}}(\hat{y}, y) + 0.1\, R_{\text{format}}(a), \tag{55}$$

where y is the ground truth answer, $\hat{y}$ is the model predicted answer, and a is the full model response to the question.

Hyperparameter setting. For all ES and ESSAM experiments, we set the hyperparameter $\rho = 10^{-4}$ and the learning rate $\alpha = 2.5 \times 10^{-4}$. For the ESSAM-F algorithm, all hyperparameters are kept the same as those in ESSAM except for N_{SAM}. Specifically, we set N_{SAM} to $\frac{1}{6}N$ (when $N = 60$) or $\frac{1}{8}N$ (when $N = 40$). The population size N and the noise scale σ are chosen as shown in Table 3. For the PPO and GRPO experiments, we set the learning rate α and the KL-divergence penalty coefficient β as shown in Table 4. In addition, for the GRPO experiments, we set the group size to $N = 8$.

Compute resources. Our experiments are conducted using four NVIDIA A100 80GB PCIe GPUs.

Table 3: Hyperparameter settings for ES and ESSAM across different models. Each pair $(\cdot, \cdot)$ denotes (population size N, noise scale σ); the symbol ✓ indicates the hyperparameter setting for each model-method combination.

METHOD	MODEL	$(40, 1.5 \times 10^{-3})$	$(40, 2 \times 10^{-3})$	$(60, 1.5 \times 10^{-3})$	$(60, 2 \times 10^{-3})$
ES	QWEN-0.5B-IT	✓			
	QWEN-1.5B-IT			✓	
	QWEN-3B-IT		✓		
	QWEN-7B-IT				✓
	LLAMA-1B-IT	✓			
	LLAMA-3B-IT				✓
	LLAMA-8B-IT				✓
ESSAM	QWEN-0.5B-IT	✓			
	QWEN-1.5B-IT			✓	
	QWEN-3B-IT		✓		
	QWEN-7B-IT				✓
	LLAMA-1B-IT	✓			
	LLAMA-3B-IT				✓
	LLAMA-8B-IT				✓

Table 4: Hyperparameter settings for PPO and GRPO across different models. Each pair $(\cdot, \cdot)$ denotes (learning rate α, KL-divergence penalty coefficient β); the symbol ✓ indicates the hyperparameter setting for each model-method combination.

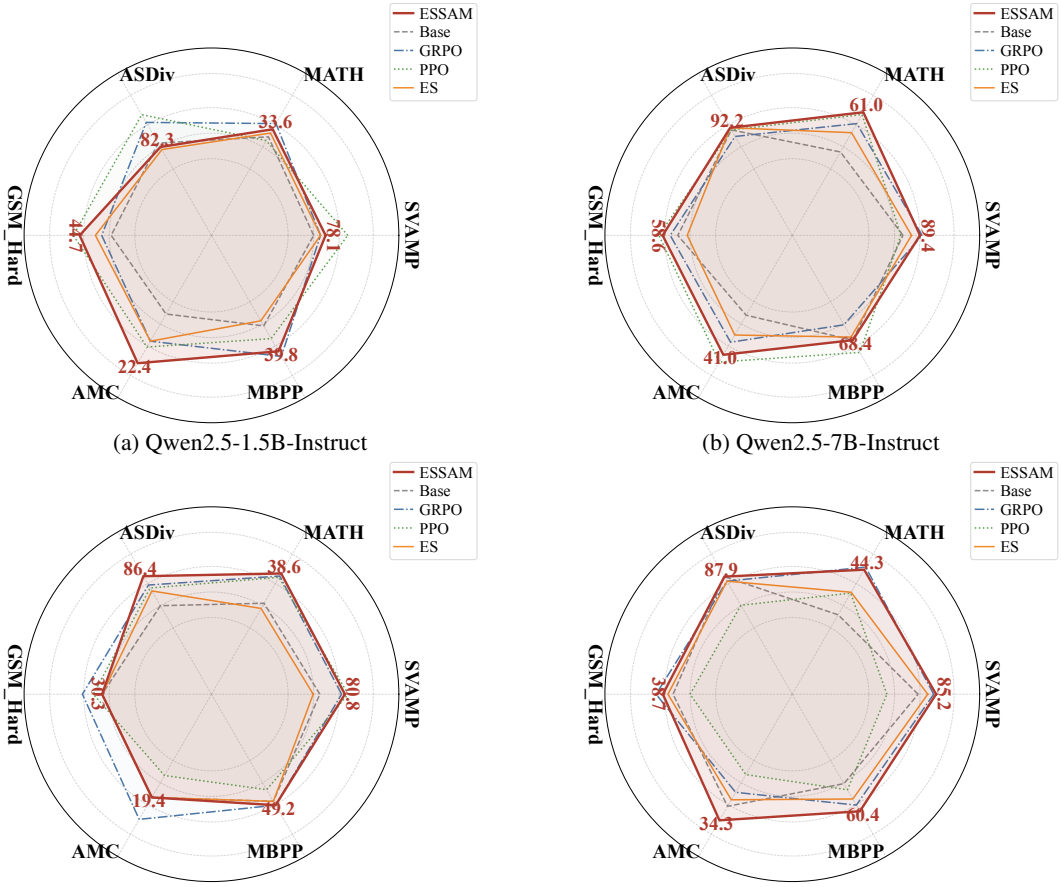
METHOD	MODEL	$(10^{-6}, 10^{-3})$	$(10^{-5}, 10^{-3})$	$(10^{-6}, 5 \times 10^{-3})$	$(10^{-5}, 5 \times 10^{-3})$
PPO	QWEN-0.5B-IT	✓			
	QWEN-1.5B-IT	✓			
	QWEN-3B-IT	✓			
	QWEN-7B-IT		✓		
	LLAMA-1B-IT	✓			
	LLAMA-3B-IT				✓
	LLAMA-8B-IT			✓	
GRPO	QWEN-0.5B-IT	✓			
	QWEN-1.5B-IT	✓			
	QWEN-3B-IT	✓			
	QWEN-7B-IT	✓			
	LLAMA-1B-IT				✓
	LLAMA-3B-IT		✓		
	LLAMA-8B-IT	✓			

F Multi-Seed Evaluation

Table 5 presents the results of multi-seed GSM8K fine-tuning experiments with ESSAM, PPO, and GRPO on Qwen2.5-0.5B-Instruct, LLaMA3.2-1B-Instruct, and Qwen2.5-7B-Instruct. We repeat the experiments for each combination of model and method using three different random seeds.

Table 5: GSM8K test accuracy (%) across multiple random seeds for different models and methods. Results are reported as mean $\pm$ standard deviation.

Base Model	ESSAM	PPO	GRPO
Qwen2.5-0.5B-Instruct	54.01 ± 0.46	53.88 ± 0.72	54.17 ± 0.84
LLaMA3.2-1B-Instruct	61.71 ± 0.13	61.13 ± 0.43	62.19 ± 0.58
Qwen2.5-7B-Instruct	92.64 ± 0.20	91.23 ± 0.53	93.00 ± 0.65



(a) Qwen2.5-1.5B-Instruct

(b) Qwen2.5-7B-Instruct

(c) Llama-3.2-3B-Instruct

(d) Llama-3.1-8B-Instruct

Figure 9: More generalization experiments.

G Generalization Experiments

As shown in Figure 9, we further present the generalization performance of more fine-tuned models.

H Detailed GPU Memory Usage

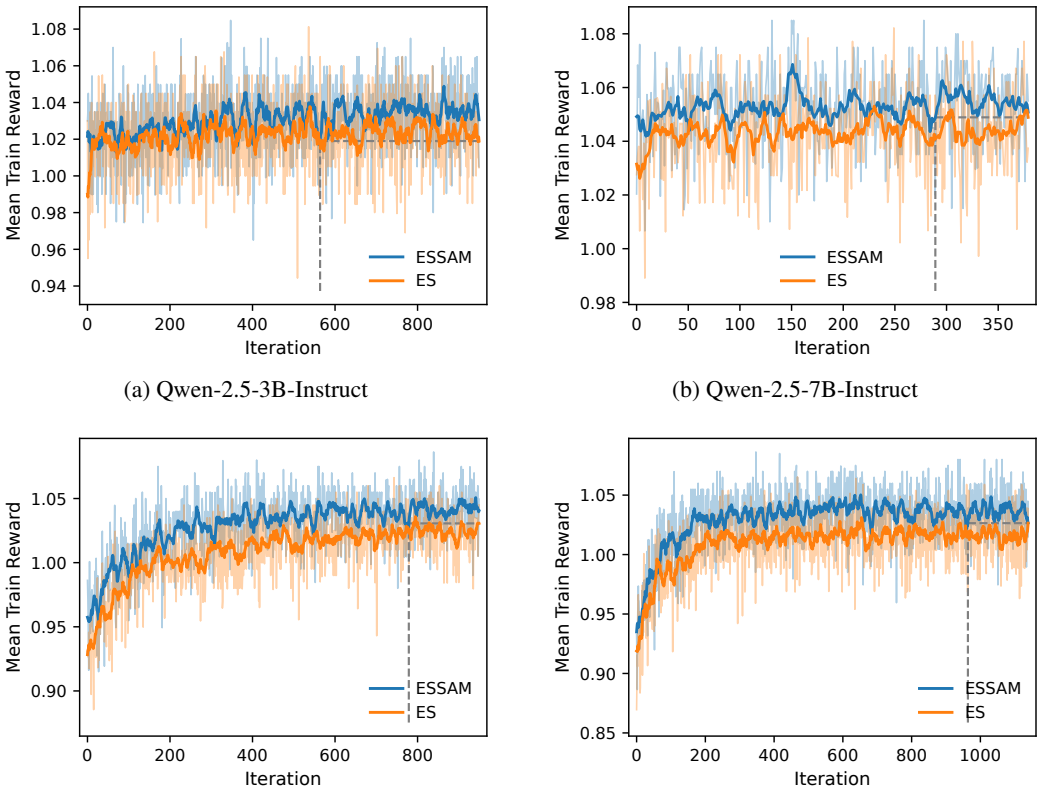
Table 6 shows the detailed GPU memory usage when fine-tuning different LLMs with different algorithms.

Table 6: The detailed GPU memory usage.

Base Model	RL			ES	ESSAM
	PPO	GRPO	GRPO+LoRA		
Qwen-2.5-0.5B-Instruct	57.32	34.20	22.43	2.99	2.99
Qwen-2.5-1.5B-Instruct	123.40	70.80	31.61	6.57	6.57
Qwen-2.5-3B-Instruct	208.96	102.24	45.57	9.96	9.96
Qwen-2.5-7B-Instruct	314.40	212.88	68.88	19.85	19.85
LLaMA-3.2-1B-Instruct	99.20	65.68	28.05	4.90	4.90
LLaMA-3.2-3B-Instruct	201.24	92.60	50.19	11.49	11.49
LLaMA-3.1-8B-Instruct	314.44	219.48	78.21	21.45	21.45

I Training Reward Curve

As shown in Figure 10, we present the curves of the evolution of the average training reward for more models fine-tuned with ES and ESSAM.



(a) Qwen-2.5-3B-Instruct

(b) Qwen-2.5-7B-Instruct

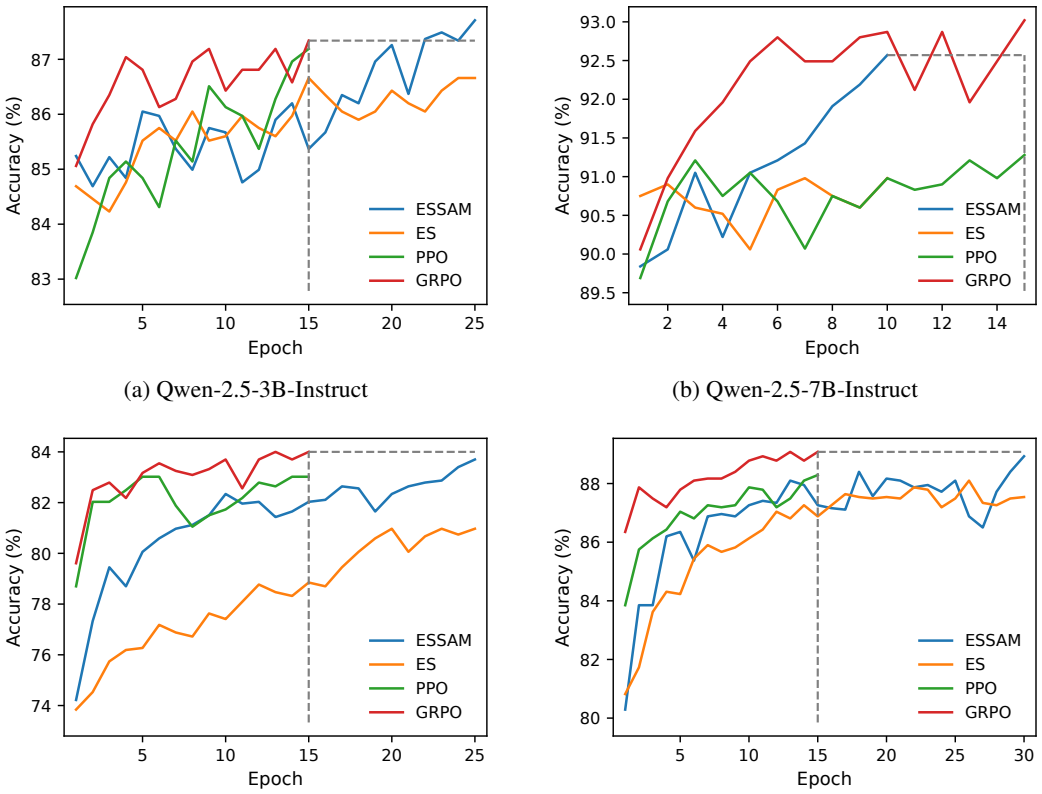
(c) LLaMA-3.2-3B-Instruct

(d) LLaMA-3.1-8B-Instruct

Figure 10: The training mean reward curves of ESSAM and ES.

J Test Set Accuracy

As shown in Figure 11, we present additional test accuracy curves for models fine-tuned with different algorithms.



(a) Qwen-2.5-3B-Instruct

(b) Qwen-2.5-7B-Instruct

(c) LLaMA-3.2-3B-Instruct

(d) LLaMA-3.1-8B-Instruct

Figure 11: Test accuracy curves of models fine-tuned with different algorithms during training.

K Example

In this section, we present example responses from models fine-tuned with ESSAM.

Qwen2.5-0.5b-Instruct

Prompt: You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer. Please solve the following problem: A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? Show your work in `<think>` `</think>` tags. And put the final numerical answer after `####`.

Model Response: Let me solve this step by step. `<think>` First, we know that the robe takes 2 bolts of blue fiber. According to the problem, the white fiber is half the amount of blue fiber. So, the white fiber is $\frac{2}{2} = 1$ bolt. Therefore, the total number of bolts needed for the robe is the sum of the blue and white fibers, which is 2 + 1 = 3 bolts. `</think>` `####` 3

Qwen2.5-1.5b-Instruct

Prompt: You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer. Please solve the following problem: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? Show your work in <think> </think> tags. And put the final numerical answer after ####.

Model Response: Let me solve this step by step. <think> First, let's determine the total number of eggs laid by Janet's ducks per day. Janet's ducks lay 16 eggs per day. Janet eats 3 eggs for breakfast every morning. So, the number of eggs left after breakfast is 16 - 3 = 13 eggs. Next, she bakes muffins for her friends with 4 eggs. So, the number of eggs left after baking muffins is 13 - 4 = 9 eggs. These 9 eggs are sold at the farmers' market. Each egg is sold for $2. Therefore, the total amount of money she makes from selling the eggs is $9 \times 2 = 18$ dollars. </think> #### 18

Qwen2.5-3b-Instruct

Prompt: You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer. Please solve the following problem: Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi's flock is 20 chickens? Show your work in <think> </think> tags. And put the final numerical answer after ####.

Model Response: Let me solve this step by step. <think> First, we need to determine the total amount of feed Wendi gives to her chickens in a day. We know that each chicken gets 3 cups of feed per day. Wendi has a flock of 20 chickens.
So, the total feed given to the chickens in a day is:

$$20 \text{ chickens} \times 3 \text{ cups/chicken} = 60 \text{ cups}$$

Next, we need to account for the feed given in the morning and the afternoon. Wendi gives 15 cups in the morning and 25 cups in the afternoon.
So, the total feed given in the morning and afternoon is:

$$15 \text{ cups} + 25 \text{ cups} = 40 \text{ cups}$$

To find out how much feed is needed for the final meal, we subtract the feed given in the morning and afternoon from the total daily feed:

$$60 \text{ cups} - 40 \text{ cups} = 20 \text{ cups}$$

Therefore, the amount of feed needed for the final meal is 20 cups. </think> #### 20

Qwen2.5-7b-Instruct

Prompt: You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer. Please solve the following problem: Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?

Show your work in <think> </think> tags. And put the final numerical answer after ####.

Model Response: Let me solve this step by step. <think> First, we need to calculate the new value of the house after the repairs. The original value of the house was $80,000, and the repairs increased the value by 150%.
The increase in value due to repairs is:

$$80,000 \times 1.50 = 120,000$$

So, the new value of the house is:

$$80,000 + 120,000 = 200,000$$

Next, we need to calculate the total cost Josh incurred. This includes the purchase price of the house and the repair costs:

$$80,000 + 50,000 = 130,000$$

Now, we can find the profit by subtracting the total cost from the new value of the house:

$$200,000 - 130,000 = 70,000$$

So, the profit Josh made is $70,000. </think> #### 70000

LLaMA-3.2-1B-Instruct

Prompt: You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer. Please solve the following problem: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week? Show your work in <think> </think> tags. And put the final numerical answer after ####.

Model Response: Let me solve this step by step. <think> Step 1: Calculate the total number of sprints per week
James decides to run 3 sprints 3 times a week, so the total number of sprints per week is:

$$3 \text{ sprints} \times 3 \text{ times} = 9 \text{ sprints}$$

Step 2: Calculate the total meters run per sprint
Each sprint is 60 meters.
Step 3: Calculate the total meters run per week
To find the total meters run per week, multiply the total number of sprints per week by the meters run per sprint:

$$9 \text{ sprints} \times 60 \text{ meters/sprint} = 540 \text{ meters}$$

</think> #### 540

LLaMA-3.2-3B-Instruct

Prompt: You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer. Please solve the following problem: Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have together if Seattle has 20 sheep? Show your work in <think> </think> tags. And put the final numerical answer after ####.

Model Response: Let me solve this step by step. <think> First, let's determine the number of sheep Charleston has. Since Charleston has 4 times as many sheep as Seattle, and Seattle has 20 sheep.
So, the number of sheep Charleston has is:

$$4 \times 20 = 80 \text{ sheep}$$

Next, let's determine the number of sheep Toulouse has. Since Toulouse has twice as many sheep as Charleston, and Charleston has 80 sheep.
So, the number of sheep Toulouse has is:

$$2 \times 80 = 160 \text{ sheep}$$

Now, let's add up the number of sheep all three cities have together. Seattle has 20 sheep, Charleston has 80 sheep, and Toulouse has 160 sheep.
So, the total number of sheep is:

$$20 + 80 + 160 = 260 \text{ sheep}$$

</think> #### 260

LLaMA-3.1-8B-Instruct

Prompt: You are a helpful assistant. You first think about the reasoning process in your mind and then provide the user with the answer. Please solve the following problem: John drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home. He tries to get home in 4 hours but spends the first 2 hours in standstill traffic. He spends the next half-hour driving at a speed of 30mph, before being able to drive the remaining time of the 4 hours going at 80 mph. How far is he from home at the end of those 4 hours? Show your work in <think> </think> tags. And put the final numerical answer after ####.

Model Response: Let me solve this step by step. <think> Step 1: Calculate the distance John drives for the first 3 hours at 60 mph. Distance = Speed $\times$ Time, so the distance is:

$$60 \text{ mph} \times 3 \text{ hours} = 180 \text{ miles}$$

So, John is 180 miles away from home after driving for 3 hours.
Step 2: Since John turns around and tries to get home in 4 hours, we need to calculate the distance he covers in those 4 hours. However, he spends the first 2 hours in standstill traffic, which means he doesn't move at all during those 2 hours. So, for the first 2 hours, the distance covered is:

$$0 \text{ miles}$$

Step 3: Then, he spends the next half-hour (0.5 hours) driving at a speed of 30 mph. Distance = Speed $\times$ Time, so the distance covered is:

$$30 \text{ mph} \times 0.5 \text{ hours} = 15 \text{ miles}$$

So, in the next half-hour, he covers 15 miles.
Step 4: After the half-hour of driving at 30 mph, he has 1.5 hours left to reach his goal of getting home in 4 hours. During these remaining 1.5 hours, he drives at a speed of 80 mph. Distance = Speed $\times$ Time, so the distance covered is:

$$80 \text{ mph} \times 1.5 \text{ hours} = 120 \text{ miles}$$

So, in the last 1.5 hours, he covers 120 miles.
Step 5: Now, let's calculate the total distance he covers in the 4 hours to get back home. Distance covered in the first 2 hours of standstill traffic = 0 miles
Distance covered in the next half-hour at 30 mph = 15 miles

Distance covered in the last 1.5 hours at 80 mph = 120 miles
Total distance covered is:

$$0 + 15 + 120 = 135 \text{ miles}$$

Since John was initially 180 miles away from home and he covers a total distance of 135 miles to get back home, we need to calculate the remaining distance:

$$180 \text{ miles} - 135 \text{ miles} = 45 \text{ miles}$$

Therefore, John is 45 miles away from home at the end of those 4 hours. `</think>` `####` 45.

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: See abstract and introduction, we present the paper's contributions and scope.

   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: We discuss the limitations of our work in 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: The complete set of assumptions and the full and correct proof for the theory in Section 3.3 are provided in the Appendix D.

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 provide the code link in the abstract and describe the experimental setup in detail in the Appendix E.

   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: Our code and reproduction instructions are available in the repository linked in the abstract.

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: We describe the experimental setup in detail in the Appendix E. We also provide the code link in the abstract, and the code includes more experimental details.

   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: [Yes]

   Justification: We report the mean and standard deviation across multiple random seeds for selected experiments in Appendix F.

   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 provide information about the computational resources used for our experiments in the Appendix E.

   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 this paper conforms, 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: [N/A]

    Justification: The work has no societal impact. This paper only studies a machine learning algorithm and has no societal impact.

    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: The paper poses no such risks.

    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 models and datasets used in the paper are properly credited by citing their original papers.

    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: [N/A]

    Justification: The paper does not release new assets.

    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 or 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 or 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: LLMs are only used for writing, editing, or formatting purposes and do not affect the core methodology, scientific rigor, or originality of the research.

    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.