EA4LLM: A Gradient-Free Approach to Large Language Model Optimization via Evolutionary Algorithms

WenTao Liu [1 2] **Siyu Song** [3] **Hao Hao** [1] **Aimin Zhou** [1 2]

Abstract

In recent years, large language models (LLMs) have made remarkable progress, with model optimization primarily relying on gradient-based optimizers such as Adam. However, these gradient-based methods impose stringent hardware requirements, demanding high-concurrency, high-memory GPUs. Moreover, they require all neural network operations to be differentiable, thereby excluding many promising non-differentiable architectures from practical use.

To address these limitations, we propose EA4LLM, an evolutionary algorithm for optimizing LLMs, and, for the first time, empirically verify full-parameter optimization from the pretraining stage across model sizes ranging from 0.5B to 32B. We conduct extensive experiments and provide key insights into how evolutionary algorithms can effectively optimize neural networks.

Our work challenges the prevailing assumption that gradient-based optimization is the only viable approach for training neural networks. It also holds significant potential to reduce the computational cost of training large language models, thereby enabling groups with limited computational resources to participate in deep learning research.

1. Introduction

Over the past few decades, gradient-based optimization has been the dominant—and often the main approach for training neural networks. A series of improved gradient-based algorithms have been successively proposed, including Stochastic Gradient Descent (SGD) (Robbins & Monro, 1951), Adaptive Moment Estimation (Adam) (Kingma & Ba, 2015), and more recently, EvoLved Sign Momentum (Lion) (Chen et al., 2023).

However, gradient-based optimization suffers from two major limitations. First, it requires computing and storing gradients during training, leading to substantial memory and computational overhead. In practice, training typically incurs 3–8$\times$ the model parameter size in GPU memory (for parameters, gradients, and optimizer states) and 2–3$\times$ the computational cost of a forward pass (due to forward propagation, backpropagation, and gradient updates). Second, gradient-based methods require the model to be differentiable, which excludes many promising non-differentiable architectures. For example, in attention network design, using FAISS (Johnson et al., 2019) for approximate nearest neighbor hashing to retrieve context from the key-value (KV) cache assumes differentiability, which prevents the use of this approach to implement efficient sparse attention mechanisms.

Igel et al. (Igel, 2003) and Tim Salimans et al. (Salimans et al., 2017) explore using Evolution Strategies (ES) as an alternative to Markov Decision Process (MDP)-based methods for optimizing neural networks in reinforcement learning. In recent years, some methods have utilized EAs to optimize large models' prompts (Sun et al., 2022a;b; Zhao et al., 2023; Guo et al., 2023) and small parameter spaces (Jin et al., 2024; Carmona et al., 2024; Huang et al., 2025), validating the ability of EAs to optimize large model parameters in smaller-scale settings. More recently, Xin Qiu et al. (Qiu et al., 2025) adapted the reward computation from GRPO (Guo et al., 2025) to design the fitness evaluation and population update mechanism in their ES algorithm, successfully extending ES to full-parameter fine-tuning of LLMs in the reinforcement learning stage for the first time. However, these works only apply ES within the reinforcement learning context, and their effectiveness heavily depends on the design of the RL reward function.

We further investigate the capability of Evolution Strategies for direct neural network optimization beyond reinforcement learning. We propose EA4LLM, a method that leverages evolutionary algorithms to optimize LLMs, and

[1]Shanghai Institute of Artifical Intelligence Education, East China Normal University, Shanghai, China [2]Shanghai Innovation Institute, Shanghai, China [3]School of Computer Science and Technology, East China Normal University, Shanghai, China. Correspondence to: Aimin Zhou <amzhou@cs.ecnu.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

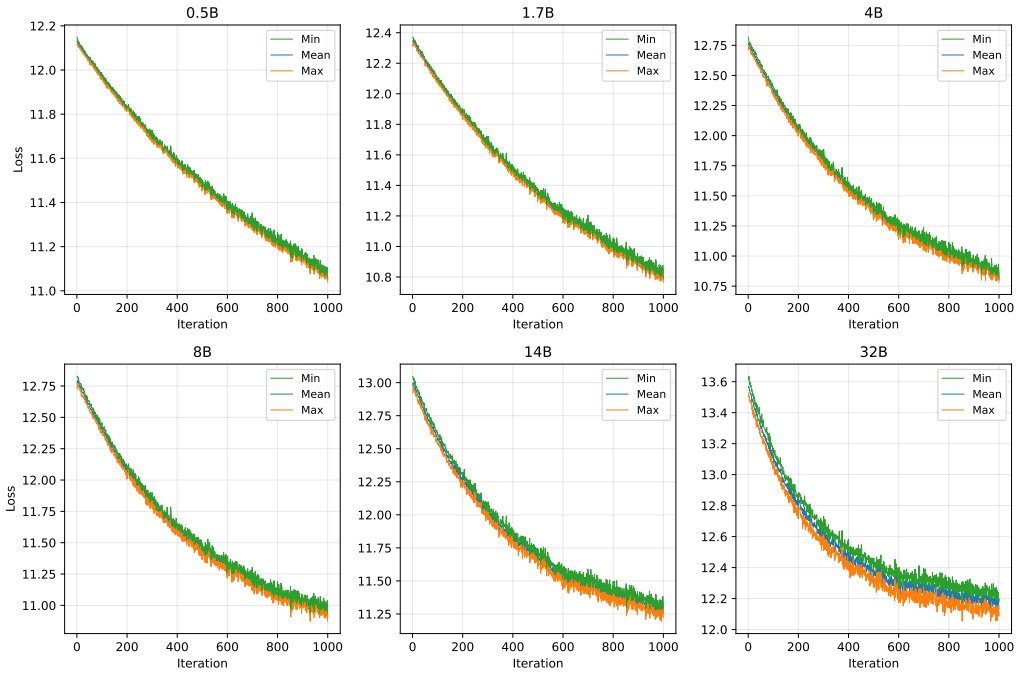

Figure 1. Loss Curve of EAGPT Optimized from Pretraining Using Evolutionary Algorithms.

demonstrate for the first time that ES can effectively perform full-parameter optimization of LLMs starting from pretraining—validated across model sizes from 0.5B to 32B—as illustrated in 1. In our implementation, we introduce a simple yet effective approach that links the model's output logits to the fitness computation in ES. This implies that virtually any neural network with a probabilistic training objective P can, in principle, be optimized using ES. Different from prior ES in supervised learning that typically treats mini-batch loss or performance as a black-box objective, we explicitly aggregate token-level log-probabilities (log-softmax) into the ES fitness and carry out full-parameter optimization for LLMs in pretraining; combined with antithetic sampling, rank shaping, grouped/layered controls, and cross-process synchronized subsampled evaluation, this substantially improves the practicality and efficiency of zeroth-order optimization in this setting. Building upon standard ES (Salimans et al., 2017), we conduct extensive exploratory experiments and provide key insights—as well as identified shortcomings—regarding the direct application of ES to neural network optimization.

Our work challenges the assumption that gradient-based methods are the sole viable option for training neural networks. It also holds significant potential to reduce the computational cost of training large language models, thereby enabling research groups with limited computational resources—even those without access to GPUs—to conduct meaningful deep learning research.

2. Related Work

Evolutionary Algorithms (EAs) form a vast and diverse family of optimization techniques inspired by the process of natural selection (Back, 1996; Zhou et al., 2011; Zhang et al., 2008). Among EAs, Evolution Strategies (ES) (Rechenberg, 1973), which we utilize in our work, constitute merely a small fraction of this extensive domain. ES methods focus on optimizing parameters through random perturbations, crossover, mutation, and other mechanisms to sample a set of solutions. These solutions (individuals) are then evaluated based on their fitness, and the population is updated accordingly. This process repeats until a termination condition is met, such as reaching a maximum number of iterations. Since ES does not require gradient information, they are particularly suitable for non-differentiable problems or scenarios where gradient computation is impractical (Hansen, 2003).

Despite the significant potential of using EAs for neural network optimization, research in this area has been relatively limited compared to the widespread adoption of gradient-based methods. Early efforts by Igel et al. (Igel, 2003) and Tim Salimans et al. (Salimans et al., 2017) demonstrated that ES could serve as an effective alternative to traditional reinforcement learning approaches, paving the way for subsequent studies (Lorenc & Neruda, 2025; Schulman et al., 2017). In recent years, some methods have utilized EAs to optimize large models' prompts (Sun et al., 2022a;b; Zhao et al., 2023; Guo et al., 2023) and small parameter spaces (Jin et al., 2024; Carmona et al., 2024; Huang et al., 2025), validating the ability of EAs to optimize large model

parameters in smaller-scale settings. More recently, Xin Qiu et al. (Qiu et al., 2025) extended the application of ES to fine-tuning large language models (LLMs) in reinforcement learning, highlighting the algorithm's capability to handle complex, high-dimensional optimization landscapes.

However, these pioneering works primarily focused on specific contexts such as reinforcement learning, leaving ample room for exploring the broader applicability of ES in direct neural network optimization. Our work builds upon this foundation, aiming to unlock the full potential of evolutionary strategies for training neural networks beyond the confines of reinforcement learning. By doing so, we not only challenge the prevailing paradigm of gradient-based optimization but also open new avenues for efficient, resource-conscious deep learning research.

3. Method

3.1. Linking Logits to ES Fitness

We optimize the model by connecting its output logits to an Evolution Strategies (ES) fitness function that measures next-token predictive quality on a text corpus.

Let a tokenized sequence be $x = (x_1, x_2, \ldots, x_T)$ with vocabulary $\mathcal{V}$. Given parameters θ, the model produces logits $z_t \in \mathbb{R}^{|\mathcal{V}|}$ at position t. We convert logits to probabilities via softmax:

$$p_\theta(x_{t+1} \mid x_{\le t}) = \mathrm{softmax}(z_t)[x_{t+1}] = \frac{\exp\left(z_{t,x_{t+1}}\right)}{\sum_{v\in\mathcal{V}} \exp\left(z_{t,v}\right)}. \tag{1}$$

To handle padding and truncation, we introduce a binary mask $m_t \in \{0,1\}$ that excludes pad positions. The per-sample reward (fitness contribution) is the mean log-probability of the true next token:

$$R(x;\theta) = \frac{1}{\sum_{t=1}^{T-1} m_t} \sum_{t=1}^{T-1} m_t \, \log p_\theta(x_{t+1} \mid x_{\le t}). \tag{2}$$

Given a dataset $\mathcal{D}$, we define the overall fitness as the average across samples:

$$F(\theta) = \frac{1}{|\mathcal{D}|} \sum_{x\in\mathcal{D}} R(x;\theta). \tag{3}$$

In ES, a perturbation ϵ is sampled and applied to parameters as $\theta' = \theta + \sigma\,\epsilon$, yielding a perturbed fitness

$$r(\epsilon) \triangleq F(\theta + \sigma\,\epsilon) = \frac{1}{|\mathcal{D}|} \sum_{x\in\mathcal{D}} R\big(x;\theta + \sigma\,\epsilon\big), \tag{4}$$

which directly ties the model's logits—through p_θ—to the ES objective. In our implementation, we evaluate $R(x;\theta)$ via log-softmax of shifted logits and label-gathering, and average rewards across the dataset, consistent with next-token causal language modeling.

3.2. Evolution Strategies and Our Extensions

Baseline ES Operator. We adopt a standard ES estimator (Salimans et al., 2017). Let $\epsilon_j \sim \mathcal{N}(0, I)$ for $j = 1, \ldots, N$ denote i.i.d. Gaussian noise samples. We evaluate perturbed fitness values

$$r_j = F\big(\theta + \sigma\,\epsilon_j\big). \tag{5}$$

We build normalized weights w_j from r_j to form a gradient-like update. We support either z-score normalization

$$\begin{aligned} w_j^{\mathrm{z}} &= \frac{r_j - \mu_r}{\sigma_r + 10^{-8}}, \\ \mu_r &= \frac{1}{N}\sum_{k=1}^{N} r_k, \\ \sigma_r^2 &= \frac{1}{N}\sum_{k=1}^{N} (r_k - \mu_r)^2 \end{aligned} \tag{6}$$

or rank-based shaping

$$w_j^{\mathrm{rank}} = 2 \cdot \frac{\mathrm{rank}(r_j)}{N-1} - 1 \in [-1, 1], \tag{7}$$

where $\mathrm{rank}(\cdot)$ assigns ranks $0, \ldots, N-1$ after sorting.

The ES update then aggregates noise weighted by w_j:

$$\hat{g} = \frac{1}{N}\sum_{j=1}^{N} w_j\,\epsilon_j. \tag{8}$$

We apply either the standard scaling or the NES-style $1/\sigma$ scaling:

$$\theta \leftarrow \theta + \alpha\,\hat{g} \quad \text{(standard)}, \qquad \theta \leftarrow \theta + \frac{\alpha}{\sigma}\,\hat{g} \quad \text{(NES with } 1/\sigma\text{)}. \tag{9}$$

Subsampled Evaluation. To reduce per-iteration computation, we evaluate fitness on a random subset of the dataset instead of all samples. At iteration t, let $\mathcal{D}_t \subset \mathcal{D}$ be a uniformly sampled subset with size M (without replacement). We define the mini-batch fitness

$$F_M(\theta) = \frac{1}{M}\sum_{x\in\mathcal{D}_t} R(x;\theta), \qquad M \ll |\mathcal{D}|. \tag{10}$$

Perturbed evaluations use the same subset:

$$r_j^{(t)} = F_M\big(\theta + \sigma\,\epsilon_j\big), \qquad j = 1, \ldots, N. \tag{11}$$

This yields a stochastic ES estimator that is unbiased under uniform sampling, i.e., $\mathbb{E}_{\mathcal{D}_t}[F_M(\theta)] = F(\theta)$, while substantially reducing wall-clock time and memory. In our

Algorithm 1 EA4LLM: Zeroth-Order Pretraining via ES

Input: Pretrained LLM parameters θ_0; token-level fitness $R(x;\theta)$ via log-softmax; iterations T; population size N; noise scale σ; learning rate α; subset size M; normalization (z-score or rank shaping).
for $t = 1$ **to** T **do**
  Sample a uniform subset $\mathcal{D}_t \subset \mathcal{D}$ with $|\mathcal{D}_t| = M$ and broadcast indices across processes.
  Draw $\{\epsilon_j\}_{j=1}^{N}$ with $\epsilon_j \sim \mathcal{N}(0, I)$.
  for $j = 1$ **to** N **do**
    Form temporary parameters: $\theta_{\text{tmp}} = \theta + \sigma\,\epsilon_j$.
    Evaluate reward on the subset: $r_j = \frac{1}{M}\sum_{x\in\mathcal{D}_t} R\big(x;\theta_{\text{tmp}}\big)$.
  end for
  Compute weights w_j from $\{r_j\}$ by z-score normalization or rank shaping.
  Aggregate noise: $\hat{g} = \frac{1}{N}\sum_{j=1}^{N} w_j\,\epsilon_j$.
  Update parameters: $\theta \leftarrow \theta + \alpha\,\hat{g}$.
end for
Return: θ_T.

implementation, the subset indices are broadcast across processes so all workers evaluate the same $\mathcal{D}_t$, and antithetic sampling plus rank shaping further mitigates the variance introduced by sub-sampling.

The pseudocode of our final algorithm is shown in Algorithm 1.

4. Experiment

4.1. Experimental Setups

Global Hyperparameters and Setup. Unless otherwise stated, we adopt the following configuration drawn from our experiment runner and training script:

- **Population**: $P = 30$; **noise scale**: $\sigma = 10^{-3}$; **step size**: $\alpha = 5 \times 10^{-4}$.
- **Iterations**: 1000 with 5000 evaluation samples per iteration; seeds are synchronized across processes.
- **Micro-batch**: 16; **max sequence length**: 256;
- **Models**: Qwen3 configurations spanning 0.5B–32B (see Appendix A.4).
- **Data sampling**: up to 10,000 lines from a novel corpus.
- **Baseline toggles**: antithetic off, $1/\sigma$ scaling off, rank shaping off, grouped multipliers off; scheduler decays disabled ($\gamma_\alpha = \gamma_\sigma = 1.0$, $S_\alpha = S_\sigma = 0$).

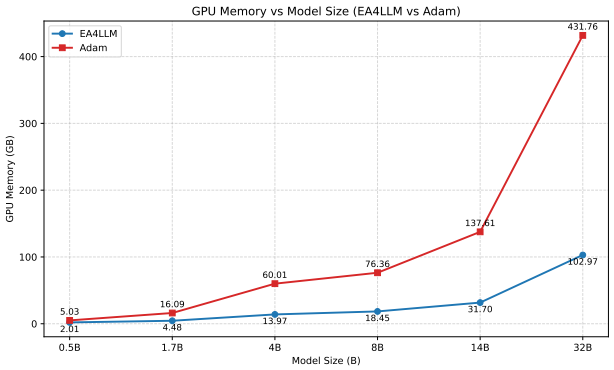

Figure 2. Minimal GPU memory usage across model sizes (0.5B–32B), comparing ES with gradient-based optimization. ES consistently uses less memory, with a larger margin as the model size increases.

4.2. Experimental Results

Cross-Scale Training Validation (0.5B–32B). We pretrain Qwen3-style models from 0.5B to 32B with full-parameter updates and consistently observe stable loss reduction; the trend matches the loss curve shown earlier in Figure 1, indicating ES effectively optimizes across model scales.

Memory Footprint Comparison: Evolutionary vs Gradient Optimization. We compare the minimal GPU memory usage when training 0.5B–32B models using ES versus gradient-based pipelines. By avoiding backpropagation and the storage of gradients/optimizer states, ES exhibits lower memory usage across all scales; the relative advantage increases with model size. Moreover, as model parameters increase, the memory advantage becomes more pronounced; for models of 4B and above, ES-based pretraining reduces GPU memory usage by more than $4\times$ compared to gradient-based methods. See Figure 2.

Ablations: Evaluation Budget and Population Size. On the 1.7B model, we study the effect of evaluation samples per iteration (1000/3000/5000/7000) and population size (20/30/40). Results match expectations: larger evaluation budgets and larger populations provide more accurate optimization signals, leading to more pronounced loss reduction. Considering time and performance constraints, in other experiments we primarily adopt evaluation budget 5000 and population size 30. See Figure 3 and Figure 4.

5. Conclusion

The Significance of Our Work Our approach provides a practical, architecture-agnostic zeroth-order pretraining path

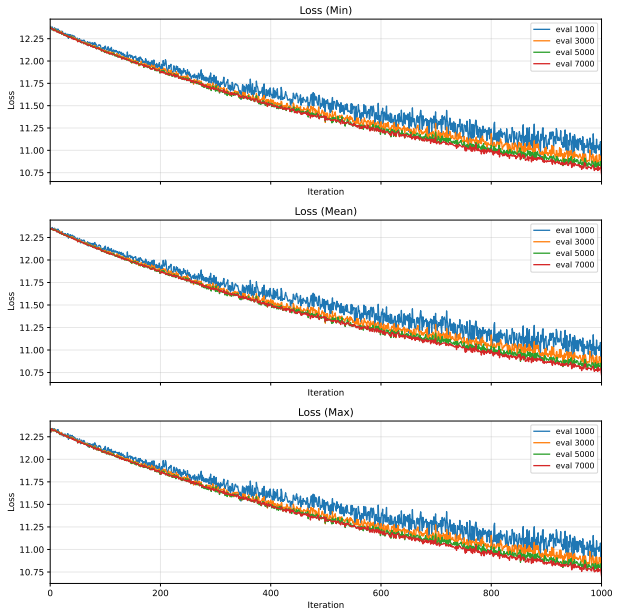

Figure 3. Different evaluation budgets (1.7B). Larger budgets yield more accurate optimization signals and stronger loss reduction.

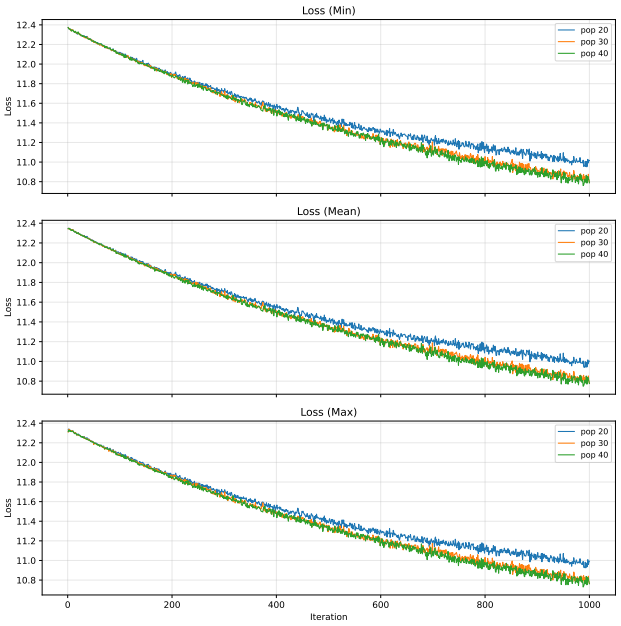

Figure 4. Different population sizes (eval=5000). Larger populations yield more accurate optimization signals and stronger loss reduction.

for large language models by tying token-level log-softmax to an ES fitness (Section 3). We employ subsampled evaluation to reduce per-iteration cost while maintaining an unbiased estimator under uniform sampling, and aggregate perturbations via weights from either z-score or rank shaping. Experimentally (Section 4), ES delivers stable loss reduction from 0.5B to 32B with full-parameter updates, and achieves substantial memory advantages over gradient-based training: for models of 4B and above, ES-based pretraining reduces GPU memory usage by more than $4\times$. Ablations on a 1.7B model confirm expected trends: larger evaluation budgets and populations provide more accurate optimization signals and stronger loss reduction; in practice, we primarily adopt an evaluation budget of 5000 and a population of 30.

Limitations and Future Work While promising, zeroth-order ES pretraining has limitations. Variance can be higher than first-order gradients and performance is sensitive to population size N, noise scale σ, learning rate α, and normalization choice; subsampling, though unbiased, introduces additional stochasticity. Sample efficiency may trail gradient-based methods because each iteration requires multiple forward evaluations. Scaling to extremely large models depends on efficient parameter perturbation and IO. In our experiments, certain variance-reduction and control mechanisms (e.g., antithetic sampling, NES-style $1/\sigma$ scaling, grouped multipliers) were disabled; exploring these and hybrid pipelines (alternating ES and gradient steps) is a natural direction. Future work includes adaptive schedules for α and σ, improved variance reduction, and task-aware fitness shaping beyond next-token log-likelihood to better align with downstream objectives.

References

Back, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

Carmona, V. I. S., Jiang, S., and Dong, B. How well can a genetic algorithm fine-tune transformer encoders? a first approach. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pp. 25–33, 2024.

Chen, X., Liu, B., Yuan, R., Song, R., He, L., Wang, J., Wang, H., He, J., and Liu, T.-Y. Symbolic discovery of optimization algorithms. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 5288–5305. PMLR, 2023.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Guo, Q., Wang, R., Guo, J., Li, B., Song, K., Tan, X., Liu, G., Bian, J., and Yang, Y. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*, 2023.

Hansen, N. Reducing the building blocks of an EA. In *Proceedings of the 2003 conference on Genetic and evolutionary computation*, pp. 1420–1431. Springer-Verlag, 2003.

Huang, B., Jiang, Y., Chen, M., Wang, Y., Chen, H., and Wang, W. When evolution strategy meets language models tuning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 5333–5344, 2025.

Igel, C. Neuroevolution for reinforcement learning using evolution strategies. In *Proceedings of the 2003 Congress on Evolutionary Computation*, pp. 1978–1985. IEEE, 2003.

Jin, F., Liu, Y., and Tan, Y. Derivative-free optimization for low-rank adaptation in large language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Lorenc, M. and Neruda, R. Utilizing novelty-based evolution strategies to train transformers in reinforcement learning, 2025. URL https://arxiv.org/abs/2502.06301.

Qiu, X., Gan, Y., Hayes, C. F., Liang, Q., Meyerson, E., Hodjat, B., and Miikkulainen, R. Evolution strategies at scale: Llm fine-tuning beyond reinforcement learning. *arXiv preprint arXiv:2509.24372*, 2025.

Rechenberg, I. Evolutionsstrategie : Optimierung technischer systeme nach prinzipien der biologischen evolution. 1973.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Sun, T., He, Z., Qian, H., Zhou, Y., Huang, X., and Qiu, X. Bbtv2: Towards a gradient-free future with large language models. *arXiv preprint arXiv:2205.11200*, 2022a.

Sun, T., Shao, Y., Qian, H., Huang, X., and Qiu, X. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855. PMLR, 2022b.

Zhang, Q., Zhou, A., and Jin, Y. Rm-meda: A regularity model-based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63, 2008.

Zhao, J., Wang, Z., and Yang, F. Genetic prompt search via exploiting language model probabilities. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 5296–5305, 2023.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and evolutionary computation*, 1(1):32–49, 2011.

Appendix

A. Comparative Analysis: Differences and Significance

A.1. Differences from ES in Supervised Learning

Salimans et al. (2017) studied, on small supervised datasets (e.g., MNIST), the correlation between ES gradient estimates g_t^{ES} obtained via random perturbations and the true gradients g_t^{True} from backpropagation, using the mini-batch performance/loss as a black-box objective $f(\theta + \sigma\epsilon)$. While that objective is computed from logits, it treats the loss as an opaque function and does not offer a full-parameter ES optimization of large-scale LLMs for pretraining.

Our approach explicitly reformulates the maximum-likelihood language modeling objective as ES fitness. Let the model outputs (logits) be z_t. We define

$$R(x;\theta) = \frac{1}{T}\sum_{t=1}^{T} m_t \cdot \log \operatorname{softmax}\big(z_t(\theta)\big)_{y_t}, \quad F(\theta) = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} R(x;\theta), \tag{12}$$

where m_t can be used for masking or weighting, and $F(\theta)$ aligns with the standard log-likelihood objective in language modeling. We use a simple population ES with i.i.d. Gaussian perturbations $\epsilon_j \sim \mathcal{N}(0, I)$; for each iteration we evaluate $F(\theta + \sigma\epsilon_j)$, apply either z-score normalization or rank-based shaping to the rewards, estimate the ES direction, and update parameters with the core ES step $\theta \leftarrow \theta + \alpha\,\hat{g}$. This design removes antithetic sampling, grouped/layered operators, and step-size multipliers, matching the actual algorithm we employ for large-scale pretraining while remaining effective empirically.

A.2. Differences from ES in the RL Stage

RL-stage ES and GRPO-style methods focus on task-specific rewards for fine-tuning and depend on reward design and sampling environments. In contrast, we apply ES *from pretraining* to full-parameter optimization with fitness defined by the aggregation of token-level log-probabilities consistent with maximum likelihood, independent of external environments or rewards. This directly targets the pretraining objective and avoids reliance on RL-specific reward shaping.

A.3. Differences from MeZO and Other Zeroth-Order Optimizers

MeZO is a memory-efficient zeroth-order optimizer that typically uses two function evaluations $f(\theta \pm \delta u)$ to approximate directional derivatives and then updates parameters in a first-order style, with the objective treated as a black-box mini-batch loss. In contrast, our method:

- uses *population-based* ES with i.i.d. Gaussian perturbations, without antithetic pairs or grouped/layered operators;
- integrates the *structured token-level log-softmax* objective into ES fitness, rather than treating the loss as opaque;
- stabilizes updates via *z-score* or *rank-based* reward normalization to reduce scale sensitivity and outlier effects;
- demonstrates *cross-scale effectiveness* (0.5B–32B) and a *memory footprint reduction* of over $4\times$ for models $\geq$4B during pretraining; in other experiments we primarily adopt evaluation budget 5000 and population size 30.

Therefore, our method directly optimizes the structured probabilistic objective of language modeling with a simple ES update and practical normalization strategies, making zeroth-order pretraining viable at LLM scale.

A.4. Model Configuration Details

We instantiate a series of Qwen3-style causal language models using the Transformers implementation (`Qwen3Config`) spanning 0.5B–32B parameters in our experiments. The key configuration settings for each size are summarized below.

Size	Layers	Hidden	Heads	KV Heads	Head Dim	Intermediate	Max Pos	Tie Emb	Use Cache	Vocab
0.5B	24	1024	8	4	128	4096	32768	true	false	151936
1.7B	28	2048	16	8	128	6144	32768	true	true	151936
4B	36	4096	32	8	128	9728	32768	true	true	151936
8B	36	4096	32	8	128	12288	32768	false	true	151936
14B	40	5120	40	8	128	17408	32768	false	true	151936
32B	64	8192	64	8	128	25600	40960	false	true	151936

Table 1. Qwen3-style model configurations used for ES pretraining experiments across 0.5B–32B.