ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2.
Available from http://www.i6doc.com/en/.

LLaMA Tunes CMA-ES

Oliver Kramer

Department of Computing Science
Carl von Ossietzky Universität Oldenburg
26111 Oldenburg, Germany
oliver.kramer@uni-oldenburg.de

Abstract. This paper introduces LLaMA-ES, an approach for tuning the hyperparameters of Evolution Strategies (ES), specifically the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), by leveraging a Large Language Model (LLM). The proposed method uses the LLM to iteratively suggest parameter adjustments based on the optimization history, enabling dynamic fine-tuning of the algorithm. We validate our approach through experiments on numerical benchmark optimization problems, employing the LLaMA3 model with 70 billion parameters. The results demonstrate that LLaMA-ES significantly enhances the performance of CMA-ES, achieving competitive results in parameter tuning and demonstrating the potential of LLMs in optimization tasks.

1 Introduction

Large Language Models (LLMs) are advanced models designed for a wide array of tasks. With a transformer-based architecture with billions of parameters, LLMs deliver impressive capabilities. LLMs can autonomously generate and execute programming and tuning tasks with the potential for self-improving algorithms.

This paper proposes LLaMA-ES, an iterative approach that uses LLaMA by Meta AI [8] to tune the parameters of an Evolution Strategy (ES) (for a comprehensive introduction to ES, see [1]), specifically the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [5]. By automating the process of refining and tuning algorithms' hyperparameters, LLMs facilitate the development of more sophisticated systems capable of autonomous learning and adaptation. In the experimental section, the advanced LLaMA3 70B model drives the tuning process on a small set of numerical benchmark problems.

The paper is structured as follows. Section 2 provides an overview of related work. Section 3 introduces the LLaMA-ES as an iterative LLM-based parameter tuning approach for CMA-ES. The experimental analysis is presented in Section 4 based on benchmark functions defined in Appendix A. Conclusions are drawn in Section 5.

2 Related Work

Tuning evolutionary algorithm parameters has a long history with random search, evolutionary algorithms, and Bayesian optimization. Recent advancements make use of LLMs for hyperparameter optimization and algorithm adaptation. Zhang et al. [9] demonstrated the efficacy of LLMs in hyperparameter optimization,

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

treating model code as a hyperparameter. Similarly, Zheng et al. [11] showcased GPT-4's utility in Neural Architecture Search (NAS). With OptFormer, DeepMind [3] introduced a transformer-based framework for hyperparameter optimization. The LLM-driven EA (LMEA) by Liu et al. [7] demonstrated competitive performance on combinatorial problems. AutoML-GPT by Zhang et al. [10] automated the training pipeline using LLMs. Guo et al. [4] directly applied LLMs to optimization tasks. The Language-Model-Based Evolutionary Optimizer (LEO) by Brahmachary et al. [2] applied LLMs to numerical optimization tasks. Liu et al. [6] combined LLM and evolution for automatic heuristic design. However, to date, no explicit approach has focused on tuning ES parameters.

3 LLaMA-ES

ES are powerful evolutionary algorithms designed for numerical optimization problems. Tuning the CMA-ES with an LLM involves using LLaMA to iteratively suggest adjustments to the parameters of the ES for an improvement of its performance on optimization tasks. The CMA-ES relies on parameters like 'CMA_rankmu' and 'CMA_rankone' to control the adaptation of its covariance matrix and the mean of the population distribution. These parameters significantly influence the algorithm's exploration and exploitation capabilities.

By integrating LLaMA (see Figure 1), the tuning process becomes more intelligent. After each CMA-ES run, the performance metrics and parameter settings are fed into LLaMA, which analyzes the historical data to generate new, potentially improved hyperparameters. This iterative feedback loop allows for continuous refinement of the algorithm's parameters without manual intervention.

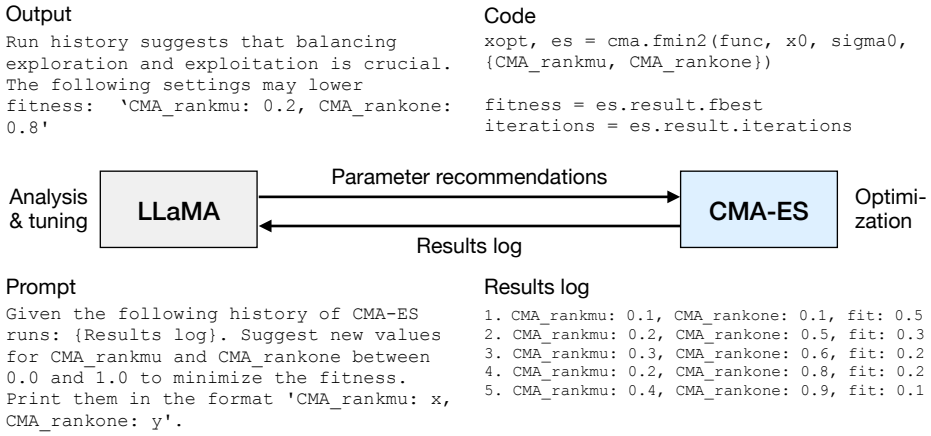

Fig. 1: Overview of LLaMA-ES showing LLaMA and CMA-ES interaction with LLM prompt, output, code, and results log

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

3.1 CMA-ES

The CMA-ES is a stochastic optimization algorithm designed for solving non-linear, non-convex optimization problems in continuous domains. It works by iteratively updating a multivariate normal distribution used to sample candidate solutions. The algorithm adapts the covariance matrix of the distribution based on the performance of the sampled solutions, enabling it to learn and exploit the characteristics of the local solution space during the optimization process.

Two tunable parameters in CMA-ES are 'CMA_rankmu' and 'CMA_rankone'. The 'CMA_rankmu' parameter controls the weighting scheme for the update of the covariance matrix, specifically influencing the step size adaptation by considering the mean of the best solutions. Higher values of 'CMA_rankmu' increase the influence of the top-performing individuals for controlling exploitation. Lower values promote exploration by giving more weight to a broader set of solutions.

The 'CMA_rankone' parameter affects the rank-one update of the covariance matrix, which adjusts the covariance matrix based on the differences between the best solutions and the current mean. This parameter determines how much emphasis is placed on the direction of successful mutations. Higher values intensify the focus on the direction of the best solutions, potentially speeding up convergence in promising regions. However, setting this parameter too high can lead to premature convergence to local optima.

3.2 Prompting

LLaMA-ES iteratively tunes the parameters of the CMA-ES algorithm by analyzing past optimization performance and suggesting new parameter values. The process is streamlined into a single step, where LLaMA is supplied with the history of CMA-ES runs through the following prompt:

```
1 Given the following history of CMA-ES runs: {Results log}.
2 Suggest new values for CMA_rankmu and CMA_rankone between 0.0 and 1.0 to
      ↪ minimize the fitness.
3 Print them in the format 'CMA_rankmu: x, CMA_rankone: y'.
```

Here, 'results log' is a dynamically generated string, a log of results, that includes the generation number, the values of 'CMA_rankmu' and 'CMA_rankone', and the corresponding fitness values for each past run. LLaMA uses this historical data to understand the relationship between hyperparameter settings and optimization results, proposing new values for 'CMA_rankmu' and 'CMA_rankone' within the specified interval constraints to further minimize fitness.

3.3 Parameter Parsing

LLaMA responds with suggested values for both parameters, formatted as specified in the prompt. These values are extracted by parsing the generated text for the last mentions of 'CMA_rankmu' and 'CMA_rankone'. The extracted parameters are then used to update the CMA-ES algorithm settings for the next iteration, which is executed to continue the optimization process.

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

4 Experimental Setup and Results

4.1 Setting

The experimental setting is designed to evaluate the effectiveness of the LLaMA3 70B model provided by Meta AI in tuning the parameters of the CMA-ES. The optimization is performed using the `pycma` library, a Python implementation of CMA-ES. The complete implementation of the LLaMA-ES can be found in this GitHub repository.

LLaMA-ES performs the hyperparameter optimization process in ten iterations. The LLaMA3 model is used to suggest new values for 'CMA_rankmu' and 'CMA_rankone' after each generation. A prompt containing the history of CMA-ES runs, including generation number, parameter values, and fitness, is sent to LLaMA, which analyzes this history and proposes new parameter values formatted as 'CMA_rankmu: x, CMA_rankone: y'. The suggested parameters are extracted and used to update the CMA-ES settings for the next iteration. The CMA-ES runs for a maximum of 1000 generations per experiment with strict tolerance settings ('tolfun', 'tolfunhist', and 'tolx' all set to '1e-100').

The experiments are conducted using four benchmark optimization problems: Sphere with $N = 100$ dimensions, Hyper-Ellipsoid, Cigar, and Rosenbrock with $N = 10$ dimensions (see Appendix A). Each of the four tuning processes is run 30 times to ensure robustness and to capture the variability in performance. For each optimization run, the starting point is $x^0 = (2, \ldots, 2)$, and the initial step size ('sigma0') is set to 1.0. The CMA-ES parameters are initialized with 'CMA_rankmu' and 'CMA_rankone' both set to 0.1.

4.2 Results

The optimization results of LLaMA-ES for the benchmark problems, as shown in Table 1, indicate successful convergence for all functions. Each row of the table shows the mean fitness of all first runs with 'CMA_rankmu' and 'CMA_rankone' initially set to 0.1, as well as the mean and best fitness achieved over 30 tuning runs for different benchmark functions.

For the Sphere function with $N = 100$, the mean fitness and best fitness values ($2.27 \cdot 10^{-14}$ and $1.61 \cdot 10^{-14}$, respectively) achieved by LLaMA-ES are not significantly better than the CMA-ES runs without tuning ($6.63 \cdot 10^{-14}$).

Function	N	Untuned	LLaMA-ES	LLaMA-ES*
Sphere	100	$6.63 \cdot 10^{-14}$	$2.27 \cdot 10^{-14}$	$1.61 \cdot 10^{-14}$
Hyper-Ellipsoid	10	$1.75 \cdot 10^{-56}$	$3.03 \cdot 10^{-60}$	$7.98 \cdot 10^{-65}$
Cigar	10	$2.54 \cdot 10^{-33}$	$1.60 \cdot 10^{-42}$	$6.78 \cdot 10^{-52}$
Rosenbrock	10	$2.66 \cdot 10^{-1}$	$2.47 \cdot 10^{-27}$	$6.15 \cdot 10^{-28}$

Table 1: Mean of untuned vs. mean (LLaMA-ES) and best results (LLaMA-ES*) of 30 tuning processes

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

The Hyper-Ellipsoid function with $N = 10$ shows a significant improvement, with mean and best fitness values reaching $3.03 \cdot 10^{-60}$ and $7.98 \cdot 10^{-65}$, respectively, compared to the untuned setting. The Cigar function with $N = 10$ achieves very low fitness values ($1.60 \cdot 10^{-42}$ mean and $6.78 \cdot 10^{-52}$ best), reflecting strong convergence and an improvement over the initial setting.

On Rosenbrock with $N = 10$, the approach also shows good optimization performance with a mean fitness of $2.47 \cdot 10^{-27}$ and a best fitness of $6.15 \cdot 10^{-28}$, indicating a significant improvement.

An exemplary output of the LLaMA3 70B model for analyzing and suggesting new parameter values is the following:

```
Run history suggests that balancing exploration and exploitation is crucial.
The following settings may lower fitness: 'CMA_rankmu: 0.2, CMA_rankone: 0.8'
```

The LLM's output emphasizes the importance of balancing exploration and exploitation by suggesting parameter adjustments that could further reduce fitness values. These suggestions are easily interpretable by humans, enhancing understanding of the optimization process.

5 Conclusions

The LLM's ability to understand and generate text allows it to interpret the optimization history in detail, enabling it to suggest parameter adjustments. When tuned by LLaMA, the CMA-ES can efficiently adapt to the characteristics of the optimization problem, enhancing its performance and effectiveness.

This synergy not only improves the performance of the CMA-ES but also demonstrates the potential of combining AI-driven insights with established optimization techniques to solve complex problems. Furthermore, the parameter suggestions are human-readable, enhancing the explainability of AI automation.

Potential extensions include adding more tunable parameters, testing various ES variants, and implementing real-time operator design. Additionally, incorporating hybrid optimization techniques, leveraging parallel computing, and applying the approach to a broader range of benchmark problems could further enhance the approach.

A Benchmark Functions

The experiments are based on numerical benchmark problems. The Sphere function is defined as $f(\mathbf{x}) = \sum_{i=1}^{N} x_i^2$, the Hyper-Ellipsoid as $f(\mathbf{x}) = \sum_{i=1}^{N} i \cdot x_i^2$, and the Cigar function as $f(\mathbf{x}) = x_1^2 + 10^6 \cdot \sum_{i=2}^{N} x_i^2$. All three employ a global minimum at $\mathbf{x}^* = (0, 0, \ldots, 0)$ with $f(\mathbf{x}^*) = 0$. The Rosenbrock function is defined as $f(\mathbf{x}) = \sum_{i=1}^{N-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$ with a global minimum at $\mathbf{x}^* = (1, 1, \ldots, 1)$ and $f(\mathbf{x}^*) = 0$.

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

References

[1] H.-G. Beyer and H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[2] S. Brahmachary, S. M. Joshi, A. Panda, K. Koneripalli, A. K. Sagotra, H. Patel, A. Sharma, A. D. Jagtap, and K. Kalyanaraman. Large language model-based evolutionary optimizer: Reasoning with elitism. *arXiv preprint arXiv:2403.02054*, 2024.

[3] Y. Chen, X. Song, C. Lee, Z. Wang, R. Zhang, D. Dohan, K. Kawakami, G. Kochanski, A. Doucet, M. Ranzato, S. Perel, and N. de Freitas. Towards learning universal hyperparameter optimizers with transformers. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[4] P. Guo, Y. Chen, Y. Tsai, and S. Lin. Towards optimizing with large language models. *arXiv preprint arXiv:2310.05204*, 2023.

[5] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. In *IEEE Congress on Evolutionary Computation, (CEC)*, pages p. 1–8. IEEE, 2001.

[6] F. Liu, X. Tong, M. Yuan, X. Lin, F. Luo, Z. Wang, Z. Lu, and Q. Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *International Conference on Machine Learning (ICML)*, 2024.

[7] S. Liu, Z. Lin, S. Yu, R. Lee, T. Ling, D. Pathak, and D. Ramanan. Language models as black-box optimizers for vision-language models. *arXiv preprint arXiv:2309.05950*, 2024.

[8] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[9] M. Zhang, N. Desai, J. Bae, J. Lorraine, and J. Ba. Using large language models for hyperparameter optimization. In *Conference on Neural Information Processing Systems (NeurIPS) Workshop - Foundation Models for Decision Making*, 2023.

[10] S. Zhang, C. Gong, L. Wu, X. Liu, and M. Zhou. AutoML-GPT: Automatic machine learning with GPT. *arXiv preprint arXiv:2305.02499*, 2023.

[11] M. Zheng, X. Su, S. You, F. Wang, C. Qian, C. Xu, and S. Albanie. Can GPT-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970*, 2023.