MATH 467 Project Report: Large Language Models as Hyperparameter Optimizers

Stephanie Zhou Jeena Mahajan Dredheza Gashi Baraliu

December 2025

Stephanie Zhou — zhoustep@usc.edu
Jeena Mahajan — jeenamah@usc.edu
Dredheza Gashi Baraliu — gashibar@usc.edu

Abstract

Hyperparameter tuning is an important component of optimization algorithms, yet traditional methods still fail to be computationally efficient. Methods like grid and random search are often very computationally expensive, unstable or inaccurate, especially as functions become increasingly complex. This experiment investigates whether large language models (LLMs) provide an effective alternative to traditional search methods for hyperparameter tuning. We evaluate LLM-guided hyperparameter selection for gradient descent with momentum on the Rosenbrock and Rastrigin functions as our benchmarks. Performance of the LLM models is then compared against the grid search and random search under the same iteration budgets. Using models GPT-4o and GPT-4o-mini, the LLMs are prompted with description of the functions and the hyperparameter ranges, and are then given the task of choosing the best learning rate and momentum values. Results show that LLMs consistently generated stable configurations and reliably delivered convergence behavior across runs. Our results found that methods outperformed the LLM's in hyperparameter tuning, however the findings demonstrated the promise of LLM guided hyperparameter tuning suggesting better performance for higher dimensional or resource constrained optimization.

1 Introduction

Hyperparameter tuning has long been a key component in numerical optimization and machine learning, as poor hyperparameter choices lead to poor performance including slow or no convergence and instability. Classic approaches provide systematic ways to tune hyperparameters, but not without their downsides (Bergstra & Bengio, 2012). They often require many evaluations and perform poorly as dimensionality increases (Snoek et al., 2012). Recent literature has proposed LLMs for hyperparameter tuning (Zhang et al., 2023). Prior experimentation has found success using LLMs to iteratively propose and refine hyperparameter configurations, outperforming random and Bayesian methods in some cases, and even extending the search space to include model code itself (Kochnev et al., 2025). Agent-based and sequential framework research has further shown that LLMs can plan experiments, adapt strategies and choose optimal hyperparameters all with very minimal human intervention (Custode et al., 2024). Inspired by these advances and results, this project examines a simplified but controlled setting of gradient descent with momentum on benchmark functions, to verify whether LLMs can meaningfully suggest hyperparameters.

1.1 Motivation

Selection of optimal hyperparameters plays a significant role in the performance of optimization algorithms. Poorly chosen learning rates or momentum values can lead to slow convergence, wasted iterations, or even outright divergence of the algorithm. Traditional approaches to hyperparameter tuning such as grid and random search, while effective and widely used, are often computationally expensive and inefficient, especially with increasing dimensionality of the hyperparameter space. LLMs however have been trained on vast datasets that include mathematical optimization procedures. This presents the opportunity to use LLMs as optimizers that can propose hyperparameter configurations with fewer evaluations, in hope of reducing computational cost but maintaining efficiency and accuracy of optimization performance.

1.2 AI Models Used

The LLM models were deliberately chosen using freely accessible openAI models in order to prioritize reproducibility and accessibility. Specifically we had chosen OpenAI GPT- 4.0 and OpenAI GPT- 4.0 mini to compare performance between the two. We included GPT- 4.0 as a higher capacity model which tends to produce more efficient and more accurate results of all freely accessible OpenAI GPT models. GPT- 4.0 mini was included in order to serve as a lighter, potentially faster alternative. The intention was to compare the performance of each model and see if the quicker more robust model would provide more accurate and optimal results for hyperparameter suggestions when compared to the mini model.

1.3 Contributions of this project

We aim to answer the following question:

> *Under a fixed, small evaluation budget, can LLM-guided hyperparameter proposals for gradient descent with momentum be as good as or more reliable than grid or random search?*

Specifically, our contributions are:

- We implement a controlled comparison of grid search, random search, and LLM-guided selection for (α, β).
- We evaluate on Rosenbrock (ill-conditioned valley) and Rastrigin (highly multimodal) to test qualitatively different failure modes.
- We report all experimental outcomes, including negative results (bad seeds / poor convergence basins), and analyze stability vs. performance tradeoffs.

2 Main Ideas and Algorithms

This section describes the key algorithms/theoretical ideas used.

2.1 Gradient descent

2.1.1 Gradient Descent Algorithm

Gradient descent is the optimization algorithm used in this work to minimize the benchmark function. The primary goal of the algorithm is to find the input x that minimizes the objective

function f(x). This is achieved by leveraging the gradient of the function, which represents the direction of steepest increase. To reduce the function value, gradient descent updates the current solution by moving in the opposite direction of the gradient. The optimization proceeds iteratively, with each update taking a small step downhill that is scaled by a step-size parameter α. Intuitively, this process can be understood as repeatedly taking small downhill steps on the function's surface until reaching a valley corresponding to a minimum.

$$x_{\{k+1\}} = x_k - \alpha \nabla f(x_k)$$

Figure 1: Gradient Descent Iterative Step

2.1.2 Gradient Descent Hyperparameters

Two key hyperparameters are tuned in gradient descent: the learning rate and momentum. The learning rate α controls the size of each update step and therefore determines how quickly the algorithm converges. If α is too small, the optimization progresses very slowly; if it is too large, the algorithm may overshoot the minimum or diverge altogether. An appropriately chosen learning rate results in fast and stable convergence and can be interpreted as how "big" of a step is taken downhill at each iteration. Momentum is incorporated through a velocity vector $\mathbf{v}$, which introduces memory of past gradients into the update rule. By accumulating gradient information over time, momentum smooths the optimization trajectory, helps the algorithm move through small bumps or noisy regions, and reduces the likelihood of getting stuck in shallow valleys.

$$v_{t+1} = \beta\, v_t - \alpha \nabla f(x_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

Figure 2: Gradient Descent with Momentum Iterative Step

2.2 Hyperparameter selection procedures

We compare three ways to choose (α, β).

Grid search. Evaluate all combinations from a predefined discrete set:

$$\alpha \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}, \quad \beta \in \{0.0, 0.5, 0.9\}.$$

This yields 12 trials.

Random search. Sample (α, β) uniformly at random from a continuous domain (the exact ranges used are reported in the experiment tables).

LLM-guided proposals. We provide the LLM with:

- the objective (Rastrigin or Rosenbrock),
- allowed ranges for α and β,
- and a request for stable, effective parameters.

The LLM outputs candidate (α, β) pairs. We then run the optimizer and record results.

3 Numerical Results / Practical Examples

3.1 Experimental setup (shared across experiments)

Initialization and iteration budget. Unless stated otherwise, runs start from

$$x_0 = (-1.5,\, 2),$$

with a maximum of $T = 5000$ iterations. We record:

- final objective value $f(x_T)$ (or at stopping time),
- iterations used until stopping (if early stopping occurs),
- the chosen (α, β).

Repository. Our code and experiment scripts are available at:
GitHub: https://github.com/stephaniezhou05/math-467-project

3.2 Benchmark functions

3.2.1 Rastrigin (multimodal)

The 2D Rastrigin function is:

$$f(x, y) = 20 + \big(x^2 - 10\cos(2\pi x)\big) + \big(y^2 - 10\cos(2\pi y)\big),$$

with global minimizer $(0, 0)$ and minimum value 0. It contains many local minima, so optimization can quickly get trapped depending on (α, β).

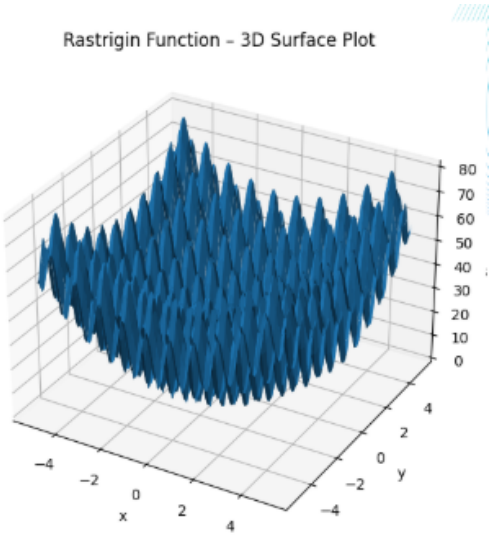

Figure 3: Rastrigin function.

3.2.2 Rosenbrock (ill-conditioned valley)

The Rosenbrock function is:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2,$$

with global minimizer $(1, 1)$ and minimum value 0. It has a narrow curved valley; convergence is sensitive to step sizes and momentum.

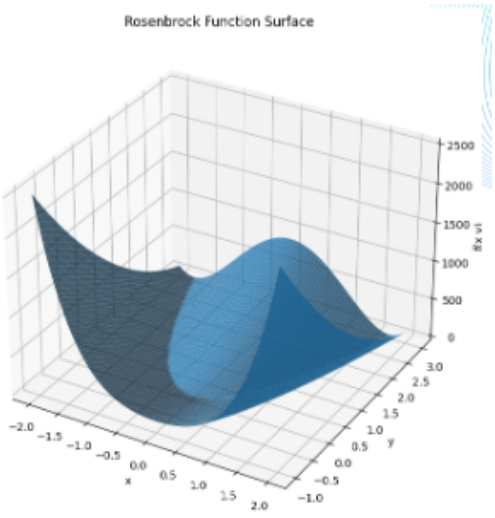

Figure 4: Rosenbrock function.

3.3 Results overview

We include:

- Grid search results

- Random search results across seeds
- LLM-guided results for GPT-4o and GPT-4o-mini
- A discussion of consistency (variance across seeds) vs. best-case performance

3.4 Grid Search on Rosenbrock

Performance Summary: The initial grid search identified the hyperparameter combination as a learning rate of α=0.001 and momentum m=0.9, achieving the lowest function value (2.39×109) in 2,020 iterations.

Table 1: Summary of Gradient Descent Performance Across Hyperparameters (Non-Refined Search)

Learning Rate	Momentum	Final f(x)	Iterations	Status
0.0001	0.0	4.484731	5000	Slow
0.0001	0.5	2.781895	5000	Slow
0.0001	0.9	0.008634	5000	Slow
0.001	0.0	0.006514	5000	Slow
0.001	0.5	0.000059	5000	Good
0.001	0.9	1.158889e-08	2020	Best
0.01	0.0	nan	5000	Diverged
0.01	0.5	nan	5000	Diverged
0.01	0.9	nan	5000	Diverged
0.1	0.0	nan	5000	Diverged
0.1	0.5	nan	5000	Diverged
0.1	0.9	nan	5000	Diverged

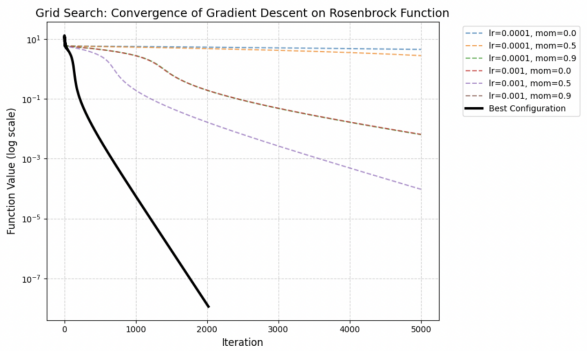

Figure 5: Graphical Representation of Gradient Descent (Non-Refined)

To further optimize performance, a refined grid search was conducted around this combination, testing learning rates slightly above and below the initial best value (0.8*, 1.2*, and 1.4*) and momentum values slightly adjusted around the best momentum (0.03, no change,+0.03). This refined search produced an improved combination of α=0.0014 and m=0.93, achieving the same near-zero function value in only 925 iterations.

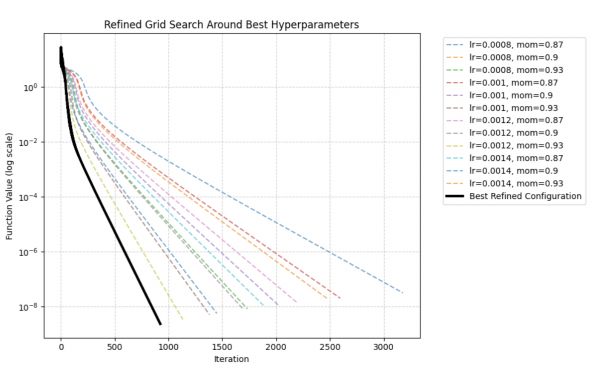

Figure 6: Graphical Representation of Gradient Descent (Refined)

3.5 Grid Search on Rastrigin

Performance Summary: For the Rastrigin function, the initial grid search identified the "best" hyperparameter combination as a learning rate of α=0.001 and momentum m=0, which reached a local minimum of 4.9748 in 36 iterations.

Learning Rate	Momentum	Final f(x)	Iterations	Status
0.0001	0.0	4.974790	332	Slow
0.0001	0.5	4.974790	167	Decent
0.0001	0.9	4.974790	183	Slower due to oscillations
0.001	0.0	4.974790	36	Best
0.001	0.5	4.974790	41	Good
0.001	0.9	4.974790	181	Slower due to oscillations
0.01	0.0	4.974790e+01	5000	Poor
0.01	0.5	1.945905e+01	5000	Poor
0.01	0.9	5.569644e+01	5000	Poor
0.1	0.0	3.791113e+01	5000	Poor
0.1	0.5	5.908400e+02	5000	Poor
0.1	0.9	2.819786e+03	5000	Poor

Figure 7: Summary of Gradient Descent Performance (Non-Refined Search)

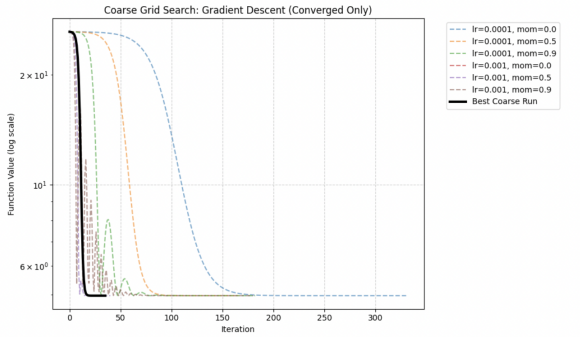

Figure 8: Graphical Representation of Gradient Descent (Non-Refined)

To further optimize performance, a refined grid search was conducted around this combination, testing learning rates slightly above and below the initial best value (0.8*, 1.2*, and 1.4*) and momentum values slightly adjusted around the best momentum (0.03, no change,+0.03). Slightly increasing the learning rate to α=0.0014 and adding a small amount of momentum (m=0.03) reduced the number of iterations to 24, while still avoiding instability.

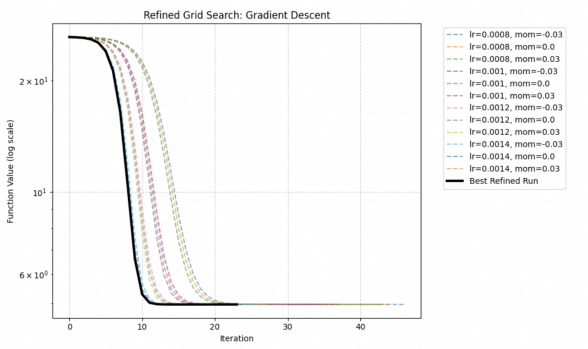

Figure 9: Graphical Representation of Gradient Descent (Refined)

3.6 Random Search on Rosenbrock

We evaluated random search under three random seeds. For each seed, we drew four (α, β) pairs uniformly from the allowed ranges, ran momentum GD, and recorded the best-performing configuration.

Performance summary. Random search showed inconsistent behavior for seeds, revealing that the search is sensitive to the hyperparameters chosen. Only seed 0 was able to find a solution that is close to the optimal solution. Comparison of the other seeds shows that they were not as effective since their sampled values could not ensure convergence. All experiments utilized the maximum number of iterations of 2000.

Seed	Best LR	Momentum	Final Value	Iterations
0	3.6001×10^{-4}	0.861	0.001826	2000
1	2.5639×10^{-5}	0.869	0.815882	2000
2	5.1045×10^{-4}	0.489	0.021873	2000

Table 2: Random search on Rosenbrock across three seeds (best of 4 samples per seed).

Notes. Seed 1 is an example of a negative outcome: a very small learning rate produced nearly stagnant progress. Seed 2 shows that mismatched momentum can lead to slow or unstable valley traversal.

3.7 Random Search on Rastrigin

We also applied random search to Rastrigin across three seeds (four samples per seed).

Performance summary. Random search showed strong variability in final values even though iteration counts were low at 44-51, which is indicative of fast convergence into different basins. Seed 0 found a near-global solution, while seeds 1 and 2 converged to higher local minima.

Seed	Best LR	Momentum	Final Value	Iterations
0	1.6052×10^{-3}	0.539	1.6052×10^{-3}	51
1	6.2163×10^{-4}	0.533	0.994959	51
2	1.2570×10^{-3}	0.431	3.97983	44

Table 3: Random search on Rastrigin across three seeds (best of 4 samples per seed).

Notes. Seeds 1 and 2 are negative outcomes relative to seed 0: the optimizer converged quickly, but to poorer local minima. This highlights that "fast convergence" does not imply "good convergence" for multimodal landscapes.

3.8 LLM-Guided Optimization on Rosenbrock (GPT-4o vs GPT-4o-mini)

We tested two LLMs (GPT-4o and GPT-4o-mini). For each seed, the model generated one candidate (α, β); we then ran momentum GD up to 2000 iterations.

Performance summary. Across all seeds and both models, the LLMs recommended nearly identical hyperparameters:

$$\alpha = 5 \times 10^{-4}, \qquad \beta = 0.80.$$

Final values ranged from 0.00141 to 0.00227 with extremely small variance. All runs used the full 2000-iteration budget, suggesting the suggested α favors stability.

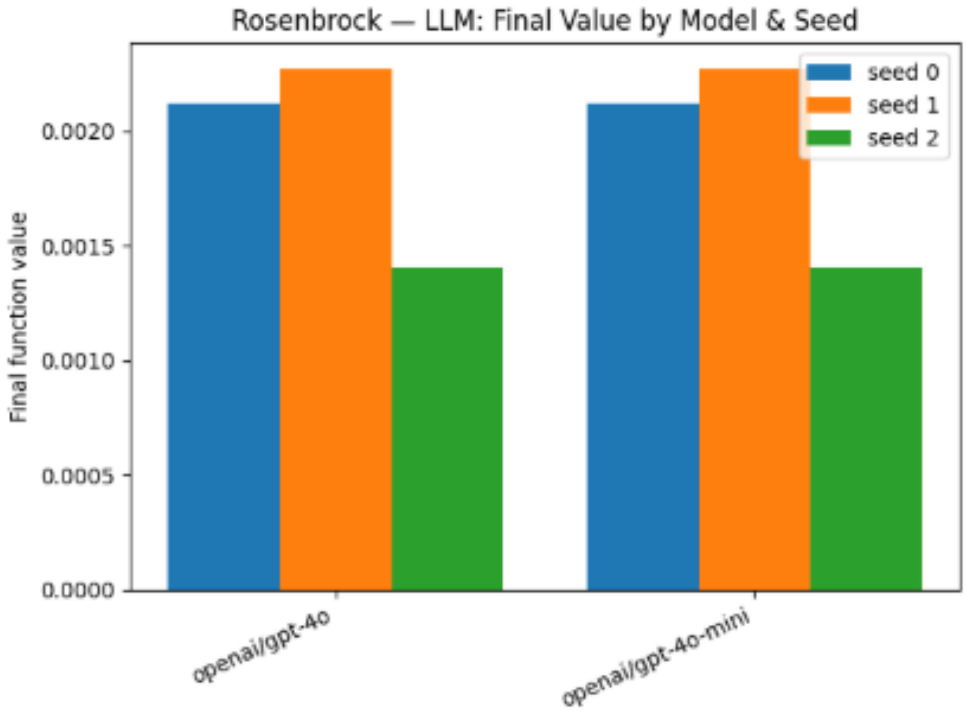

Figure 10: Final objective values for GPT-4o and GPT-4o-mini across three seeds on Rosenbrock.

Seed	Model	LR	Momentum	Final Value	Iterations
0	GPT-4o	5.0×10^{-4}	0.800	0.002119	2000
0	GPT-4o-mini	5.0×10^{-4}	0.800	0.002119	2000
1	GPT-4o	5.0×10^{-4}	0.800	0.002272	2000
1	GPT-4o-mini	5.0×10^{-4}	0.800	0.002272	2000
2	GPT-4o	5.0×10^{-4}	0.800	0.001407	2000
2	GPT-4o-mini	5.0×10^{-4}	0.800	0.001407	2000

Table 4: LLM-guided hyperparameters and outcomes on Rosenbrock.

3.9 LLM-Guided Optimization on Rastrigin (GPT-4o vs GPT-4o-mini)

We next tested LLM-guided selection on Rastrigin.

Performance summary. Both models converged to about 0.994959 for seeds 0 and 1, but to a worse basin (≈ 3.97983) for seed 2. GPT-4o tended to propose a larger learning rate and higher momentum than GPT-4o-mini, but both ultimately landed in similar basins.

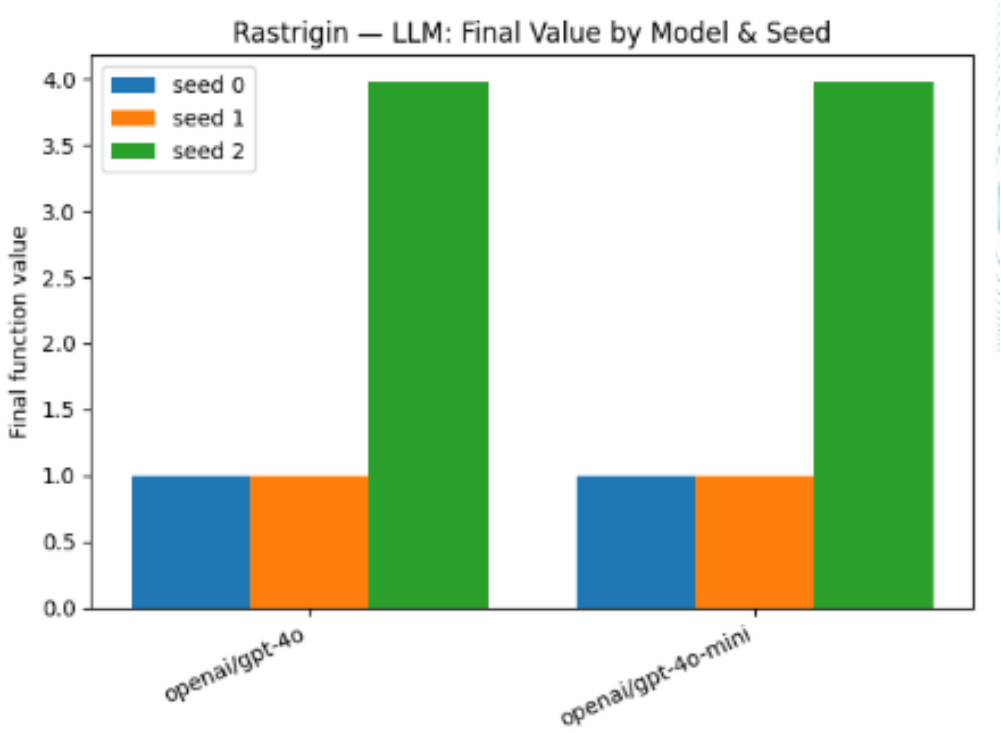

Figure 11: Final objective values for GPT-4o and GPT-4o-mini across seeds on Rastrigin.

Seed	Model	LR	Momentum	Final Value	Iterations
0	GPT-4o	8.0×10^{-4}	0.850	0.994959	151
0	GPT-4o-mini	1.0×10^{-4}	0.700	0.994959	98
1	GPT-4o	5.0×10^{-4}	0.800	0.994959	101
1	GPT-4o-mini	1.0×10^{-4}	0.700	0.994959	101
2	GPT-4o	8.0×10^{-4}	0.850	3.97983	151
2	GPT-4o-mini	1.0×10^{-4}	0.700	3.97983	100

Table 5: LLM-generated hyperparameters and outcomes on Rastrigin.

Negative result. Seed 2 is a clear failure case: both LLMs produced settings that converged to a high local minimum. This supports the idea that LLM-guided selection can be consistent, but it still misses the globally best basins in strongly multimodal landscapes.

4 Discussion

4.1 What did the numerical results show? Expected vs. Surprising Observations

Rosenbrock.

- In the case of Grid Search, learning rates that were too small caused the search process to converge slowly, requiring the entire 5,000 iterations to converge but not necessarily to the lowest point, whereas too-high learning rates caused the search process to go out of bounds because of overshooting in the valley. Having a high average learning rate with high momentum caused the search process to gain sufficient velocity to traverse the curved valley of the Rosenbrock function efficiently without overshooting. This was confirmed by the results of the refined grid search, where the slight boost in learning rates caused quicker convergence, whereas the additional momentum helped to smoothen the path through the curved valley, minimizing zig-zagging between points of convergence. This confirms that the refinement process is more concerned with improving convergence rates than the objective itself.

- LLM-guided tuning produced remarkably consistent results across seeds and across models, suggesting the LLM settled on a stable heuristic (moderate α, high β).

- Note that random search had high variance: one random initial seed discovered the good solution, while other seeds resulted in stagnation or slow progress. This is consistent with Rosenbrock's robustness to step sizes being trapped in a small valley.

Rastrigin.

- For Grid Search, very small learning rates yielded convergence with painfully slow gradients, and often took all 5,000 iterations without reaching the lowest value, whereas high learning rates led to divergence by overshooting the numerous small basins in this rugged landscape; in smoother functions momentum was harmful in this case: gradient descent without momentum converged cau- cautiously through the landscape, avoiding oscillations and navigating the small basins efficiently. With momentum, it was often overshooting the basins and getting stuck for longer times, while slowing convergence. The refined grid search agreed with these observa- tions: increasing the learning rate slightly and adding a small amount of momentum reduced It increases the number of iterations to 24 while still preventing instability. The small momentum helped slightly accelerate convergence w/o overshooting, which demonstrates that careful tun- ing is particularly important in rugged, multi-modal landscapes such as Rastrigin.

- Basin sensitivity was shown by both random search and LLM-guided selection. That is, most runs converged: quickly but not necessarily to good minima. A notable observation is that the LLMs were consistent across seeds for two seeds, but both failed together on seed 2, indicating shared vulnerability due to certain initial conditions or basin geometries.

4.2 Possible explanations for the Three Methods' Performance

- The performance of grid search differs between the Rosenbrock and Rastrigin functions because of how systematically it explores the hyperparameter space. For the Rosenbrock function, which has a smooth and well-structured optimization landscape, grid search is effective because performance varies predictably with learning rate and momentum. This allows the grid to reliably capture a near-optimal combination. In contrast, the Rastrigin function has a highly non-convex, multi-modal landscape where even small changes in hyperparameters can lead to very different outcomes. In this setting, the fixed and structured nature of grid search can limit its effectiveness, as it may repeatedly sample hyperparameters that lead to similar local minima, which makes it harder to escape poor regions. This explains why grid search performs well for Rosenbrock but is less effective for Rastrigin.

- A plausible explanation is that LLMs encode a general "safe default" policy for momentum GD (e.g., $\alpha \approx 10^{-4}$–10^{-3} and $\beta \approx 0.7$–0.9), which works well on smooth/ill-conditioned problems (Rosenbrock) but does not explicitly reason about basin hopping in multimodal landscapes (Rastrigin).
- Random search can sometimes outperform by luck, but requires more samples to be reliable.

4.3 Limitations

This project has several limitations that should be considered when interpreting the results. First, limited access to computational resources constrained the scale of the experiments, restricting the number of hyperparameter configurations, iterations, and repeated trials that could be evaluated. As a result, only a relatively small sample of experiments was conducted, which may limit the generalizability of the findings. Finally, more advanced optimization techniques and theoretical analyses—such as adaptive learning-rate methods or second-order methods —were outside the scope of experimentation conducted. As a result, the project prioritized conceptual understanding and empirical exploration over exhaustive theoretical or large-scale experimental validation.

4.4 Future Experimentation

Future work will extend the current experiment to analyze the performance of LLMs on different sets of benchmark objective functions with varying and possibly more complex geometries, where traditional methods are known to be inefficient, including non-convex and higher dimensional problems. This will allow for a stronger comparison between the strengths of LLMs vs. commonly used methods that traditionally pose issues when considering more complex functions. Another direction of future experimentation is to evaluate alternative LLMs that have greater computational power and accuracy and are potentially not publicly accessible yet. This can also extend into prompt engineering to analyze whether the efficacy of the LLMs in hyperparameter tuning is attributed to the variance of prompts as opposed to the model itself.

5 Conclusion

In this project, we investigated the performance of gradient descent on benchmark functions, focusing on the effects of learning rate and momentum through both coarse and refined hyperparameter searches. For the Rosenbrock function, a structured grid search identified the optimal hyperparameters, allowing the optimizer to efficiently navigate the narrow, curved valley and reach the global minimum. In contrast, for the multi-modal Rastrigin function, random search outperformed grid search, as its stochastic nature allowed it to "luck" into hyperparameters that effectively escaped local minima in the rugged landscape. Our experiments also demonstrated that LLMs can provide reasonable, conservative suggestions for learning rates and momentum values, which may be especially useful in higher-dimensional optimization problems where exhaustive searches are computationally expensive. On these benchmarks, however, simple baseline methods already achieved strong performance, with GPT-4o-mini often producing competitive recommendations. Overall, these results provide a foundation for applying gradient-based optimization and LLM guidance to more complex, realistic optimization tasks, highlighting both the importance of hyperparameter tuning and the potential of model-assisted approaches.

References

[1] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[2] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012.

[3] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2017.

[4] Zhang et al. Using Large Language Models for Hyperparameter Optimization. *Cornell University*, 2023.

[5] Gao et al. Large Language Model Agent for Hyper-Parameter Optimization (AgentHPO) *OpenReview preprint*.

[6] Custode et al. An investigation on the use of Large Language Models for hyperparameter tuning in Evolutionary Algorithms *GECCO Companion*, 2024.

[7] Kochnev et al. Optuna vs Code Llama: Are LLMs a New Paradigm for Hyperparameter Tuning? *Cornell University*, 2025.