A Pipeline for Automated Supervised Tuning of LLMs

Diego D. Fernandes[1], Thalyson G. N. Silva [1,3], Rafael S. Santos[1], Felipe G. Marajo[1], Cleilton L. Rocha[2], Ana Luiza B. P. Barros[1], Gustavo A. L. Campos[1]

[1]State University of Ceara (UECE)
Fortaleza, Ceará, Brazil

[2]Atlantic Institute
Fortaleza, Ceará, Brazil

[3]Federal Institute of Education, Science, and Technology of Ceará (IFCE)
Fortaleza, Ceará, Brazil

{diego.dias, thalyson.nepomuceno, raf.silva, felipe.marajo}@aluno.uece.br,
cleilton_rocha@atlantico.com.br, {analuiza.barros, gustavo.campos}@uece.br

Abstract. *The increasing complexity of language models and the growing diversity of Natural Language Processing (NLP) tasks have emphasized the need for automated and efficient tuning processes. But, while advances in Automated Machine Learning (AutoML) and parameter-efficient fine-tuning (PEFT) techniques have been made, there is still a lack of structured workflows adapted to the specific challenges of Large Language Models (LLM). This work proposes a conceptual, modular Automated Large Language Model (AutoLLM) pipeline that integrates both full and lightweight supervised fine-tuning strategies under a unified optimization framework. The pipeline leverages simulation-based search methods, particularly bioinspired algorithms such as Genetic Algorithms (GA), to automate hyperparameter tuning in supervised LLM adaptation. Preliminary experiments were conducted using Quantized Low-Rank Adaptation (QLoRA) for lightweight fine-tuning on the summarization task. The results illustrate the adaptability and efficiency of the approach in optimizing LLMs for one downstream application while maintaining computational feasibility. The proposed pipeline offers a structured foundation for advancing AutoML in the context of language model tuning.*

1. Introduction

Recent progress in NLP has led to the development of increasingly powerful language models that are essential for performing various relevant AI tasks, such as text classification, sentiment analysis, named entity recognition, text generation, automatic question answering, among others [Hasan et al. 2024]. Traditionally, these tasks were approached using sequential models such as Recurrent Neural Networks (RNNs) [Elman 1990], especially Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997] and Gated Recurrent Unit (GRU) [Cho et al. 2014] networks, which were designed to capture long-term dependencies in training data. More recently, the introduction of the Transformer architecture [Vaswani et al. 2017] has eliminated the need for recurrence and improved efficiency in modeling long-range dependencies.

It is worth highlighting the emergence of LLMs, such as Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al. 2019], Generative Pre-trained Transformer (GPT) [Achiam et al. 2023], T5 [Raffel et al. 2020], and Large Language

Model Meta AI (LLaMA) [Touvron et al. 2023]. By combining massive pre-training with supervised fine-tuning for specific tasks, these models have come to dominate benchmarks across a wide range of NLP tasks. Despite their success, the development of language models within the LLM context remains a challenge, requiring multiple interdependent decisions for construction, adaptation, and reuse. The complexity of fine-tuning LLMs and their associated computational demands stem from the large number of parameters and the wide range of possible hyperparameter configurations.

Similarly, designing Transformer models from scratch for NLP tasks involves a large set of variables in the architectural design, made complex by the pursuit of model robustness, interpretability [Doshi-Velez and Kim 2017], and personalization [Lu et al. 2021]. Strategies for selecting architecture and hyperparameters typically rely on manual expert intervention using heuristics. Other common strategies, useful in smaller search spaces or with limited computational resources, include grid search and random search. However, these approaches are labor-intensive, prone to human bias, and not easily scalable

In this context, several AutoML adaptations have focused on the design, selection, and tuning of language models, aiming to optimize both structural aspects and components related to training and fine-tuning [Zöller and Huber 2019, Hossain and Timmer 2021]. AutoML pipelines face significant difficulties and limitations when applied to LLM fine-tuning due to the complex, multi-stage nature of the LLM lifecycle [Tornede et al. 2023]. This includes pipelines addressing fine-tuning strategies like full supervised fine-tuning and lightweight fine-tuning of LLMs [Houlsby et al. 2019]. Beyond traditional classification or regression metrics, the diversity of NLP tasks necessitates specific evaluation metrics such as BLEU [Papineni et al. 2002], ROUGE [Lin 2004], perplexity [Jelinek 1998], and execution accuracy [Wang et al. 2019].

Considering these limitations, this work proposes a conceptual and modular AutoLLM pipeline, adapted to language models, that integrates automated strategies for both full and lightweight supervised fine-tuning of LLMs. The pipeline supports hyperparameter optimization of instruction tuning in upstream models, leveraging techniques such as Low-Rank Adaptation (LoRA) and QLoRA. To illustrate its application, we present preliminary experiments focused on lightweight fine-tuning for NLP tasks, specifically Summarization. Although the experimental illustration is limited to LLM lightweight fine-tuning, the pipeline was designed to support future extensions and guide decisions across diverse tasks and optimization strategies.

This article is structured into five sections: Section 2 reviews related works, covering hyperparameter optimization strategies, AutoML adaptations for language models, and recent advances in instruction tuning and PEFT. Section 3 introduces the conceptual AutoLLM pipeline, designed as a modular and extensible framework for supervised tuning of language models. Section 4 presents the application of the pipeline using LoRA and QLoRA for lightweight fine-tuning, along with experimental results across a summarization task. Section 5 concludes the study and discusses directions for future work.

2. Related Works

The research on hyperparameter optimization and fine-tuning of large language models has evolved rapidly alongside the development of transformer-based architectures. This section categorizes and discusses related works according to three major dimensions: (i)

general-purpose hyperparameter optimization strategies; (ii) AutoML adaptations for language model tuning; and (iii) recent advances in instruction tuning and parameter-efficient fine-tuning (PEFT) for LLMs.

Regarding general-purpose hyperparameter optimization, [Bergstra and Bengio 2012] revealed the limitations of exhaustive methods like grid search in high-dimensional spaces. Random search emerged as a more efficient alternative for finding good hyperparameter combinations under computational constraints. Its efficiency comes from effectively sampling across all dimensions of the search space, rather than systematic evaluation. Building on this, Bayesian optimization approaches, exemplified by [Snoek et al. 2012], utilize acquisition functions to guide the search process through the hyperparameter space. These functions are mathematical models that balance exploration and exploitation to select the most promising configurations for evaluation.

In parallel, multi-fidelity optimization techniques [Mulakala et al. 2024] were developed to further enhance efficiency. Multi-fidelity optimization refers to strategies that start by evaluating candidate configurations using low-cost approximations, e.g., fewer training epochs, smaller subsets of data, and progressively allocate higher computational resources to the most promising candidates. This approach significantly reduces overall search costs while maintaining the ability to identify high-quality solutions. It is particularly attractive for expensive training scenarios such as LLM fine-tuning.

More recently, the focus has shifted towards the tuning and adaptation of large language models for downstream applications. In this context, downstream tasks refer to specific goal-oriented tasks that models are fine-tuned to perform after their general pretraining phase. Examples include text classification, question answering, summarization, text-to-SQL generation, and semantic similarity analysis. Improvements in downstream task performance directly reflect how effectively fine-tuning strategies enable LLMs to specialize for these practical applications. Several recent works have addressed the challenges of optimizing LLMs for such downstream tasks [Mulakala et al. 2024, Tribes et al. 2023, Halfon et al. 2024].

[Tribes et al. 2023] investigated hyperparameter optimization specifically for instruction tuning of LLMs, addressing the difficulty of navigating large search spaces under computational constraints. They demonstrated that combining LoRA with multi-fidelity optimization significantly improves performance on downstream tasks. Similarly, [Halfon et al. 2024] proposed the Coverage-Based Search (CBS) technique to explore the hyperparameter space more strategically during instruction tuning, comparing lightweight LoRA adaptations with full fine-tuning and highlighting the cost-efficiency of lightweight methods. Additionally, Mulakala et al. [Mulakala et al. 2024] introduced an adaptive multi-fidelity optimization strategy tailored to LLMs, confirming that early low-cost evaluations can substantially accelerate the search process without sacrificing model performance.

Regarding broader AutoML perspectives, foundational surveys such as [Zöller and Huber 2019] , [Hossain and Timmer 2021], and [Yao et al. 2018] have outlined how automated processes can address the challenges of model design, selection, and tuning. In the context of language models specifically, [Houlsby et al. 2019] provided a comprehensive survey on parameter-efficient transfer learning techniques, rein-

forcing the importance of modular and scalable fine-tuning approaches for large models. Additionally, studies on Neural Architecture Search (NAS) [Elsken et al. 2019] and automated variance control [Bouthillier et al. 2021] have further advanced the capabilities of automated workflows in machine learning.

Despite these advances, most existing works either focus on generic hyperparameter search frameworks or address specific aspects of LLM tuning. In contrast, the approach proposed in this work introduces a conceptual, modular AutoLLM pipeline that systematically integrates lightweight and full supervised fine-tuning strategies under a unified optimization framework. By leveraging bioinspired algorithms, such as Genetic Algorithms, and explicitly addressing multi-objective optimization across task-specific NLP metrics, our pipeline provides a principled and extensible structure for the automated tuning of LLMs. Furthermore, the experimental validation across a summarization task, illustrates the adaptability of the framework to heterogeneous input profiles and evaluation needs.

3. AutoML Pipeline for LLM Tuning

This work introduces a conceptual pipeline that aims to systematize one core strategy for model adaptation in Natural Language Processing (NLP): the automated fine-tuning of pretrained large language models (LLMs). The pipeline serves as a high-level representation that supports the extension of AutoML-based workflows tailored to NLP tasks (AutoLLM). It offers a modular view of how LLM tuning techniques can be integrated into a unified process guided by simulation-based optimization. The goal is to support experimentation and design decisions in a principled and reusable way.

3.1. Overview of the Stages of the AutoLLM Conceptual Pipeline

The adoption of automated approaches for the tuning and development of language models arises from the increasing complexity of these models and the diversity of tasks in Natural Language Processing (NLP). The conceptual AutoML pipeline proposed in this section systematically organizes the key components involved in the process of tuning and developing language models. The pipeline is intended to support the design and analysis of AutoML workflows specifically tailored to language models. Figure 1 provides an overview of the main stages covered by the pipeline.

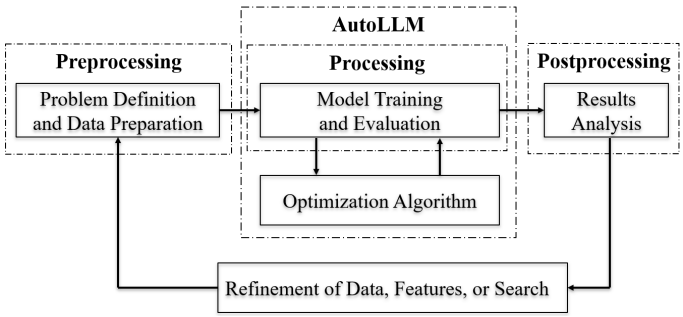

Figure 1. Overview of the Stages of the Conceptual AutoML Pipeline.

The first stage of the pipeline, referred to as Preprocessing, is concerned with problem definition and data preparation for the subsequent stage. The second stage, referred to as Processing, focuses on the training and evaluation of models. The third stage,

referred to as Postprocessing, is dedicated to the analysis of the results obtained. Since the pipeline operates in cycles, the outcomes from the Postprocessing stage can feed back into the earlier stages, enabling iterative adjustments throughout the modeling process. The integration of the second stage with an optimization algorithm automates the development of machine learning models.

In the AutoLLM Preprocessing phase, problem definition is dictated by the NLP task (e.g., text classification, summarization, QA, code generation). This stage handles textual, sequential, and contextual input data, performing initial analysis of vocabulary size, token distribution, text length, special characters, and class balance for supervised tasks. Text processing operations, conforming to the chosen pretrained model's tokenizer (e.g., BERT, GPT, T5), may include cleaning, tokenization, normalization, truncation, and padding

The Processing phase in a Machine Learning pipeline finds the best predictive model and hyperparameter configurations. This involves using simulation-based optimization to test, evaluate, and select promising combinations, even in complex search spaces. For language models, this phase faces challenges due to high data dimensionality, substantial LLM training costs, and the complexity of adapting models to textual tasks. Algorithmic efficiency and computational feasibility are critical requirements for the AutoLLM process during this phase.

The Postprocessing phase analyzes results after model training and validation to select the best-performing model. Selection is typically based on metrics like accuracy, F1-score, and AUC-ROC. For Transformer-based or LLM language models, analysis also includes task-specific NLP evaluation metrics: BLEU, ROUGE, and METEOR for generation tasks; cosine similarity and Spearman/Pearson correlation for similarity tasks; and execution accuracy and exact match for structured prediction tasks like text-to-SQL. While Preprocessing and Postprocessing are important, this work focuses on the Processing phase, which handles automated search space exploration

3.2. AutoLLM Optimization Framework via LLM Training and Testing

The Processing phase represents the core of the AutoLLM approach, where performance-driven optimization takes place through successive cycles of training and testing large language models. Figure 2 illustrates a conceptual framework entitled AutoLLM Optimization Framework via LLM Model Training and Testing, which outlines the iterative flow of automated experimentation conducted on LLMs using different supervised fine-tuning configurations.

The pipeline starts by defining optimization objectives, setting the evaluation metrics that guide the process. For AutoLLM, these objective functions combine classic supervised learning metrics (accuracy, F1-score, loss, execution time) with task-specific NLP metrics (BLEU, ROUGE, perplexity, or execution accuracy). The selection or weighting of these metrics can prioritize robustness, semantic performance, or computational efficiency, and multi-objective optimization can be employed.

Following objective definition, the search space for system exploration is defined. This initial configuration specifies adjustable hyperparameters like learning rate, number of epochs, batch size, optimizer, learning rate scheduler, and layers to freeze or fine-tune. It also includes lightweight fine-tuning parameters, such as the additional vector size of LoRA, the choice of injection layer, and adapter configurations. Adapters are small train-

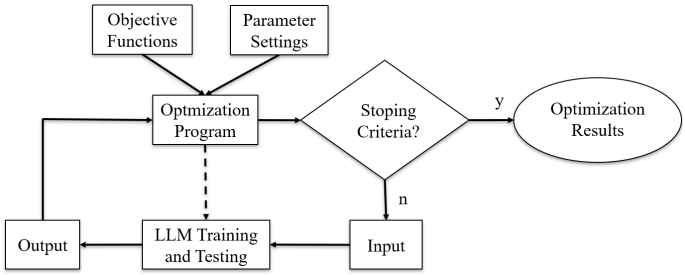

Figure 2. Overview of the AutoLLM Optimization Framework.

able modules inserted between Transformer layers for task-specific adaptation without modifying pretrained weights. Defining this search space establishes the boundaries and range of candidate solutions (models and hyperparameter configurations) for optimization.

With defined objectives and parameters, the pipeline executes an optimization program, generating new hyperparameter combinations for testing based on previous results. Algorithms such as random search, Bayesian optimization, and bioinspired methods (e.g. evolutionary algorithms) can be employed. For AutoLLM, the chosen algorithm must consider language model complexity, computational cost, and efficient convergence in high-dimensional search spaces. The suggested configurations are then directly applied to the language models' training and testing component. Each LLM undergoes supervised fine-tuning using these configurations, which can involve full fine-tuning or lightweight PEFT techniques such as LoRA, QLoRA, Prefix Tuning, or Adapters. After training, the model is evaluated with predefined metrics, and results regarding execution logs, processing time, memory consumption, and per-class performance are recorded.

The results are fed back into the optimization program, which uses this information to update its estimates of promising search space regions. This continuous feedback loop, a core AutoLLM characteristic, allows the optimization mechanism to learn from model behavior under different adjustments and progressively refine its search strategy. This iterative adaptation is crucial for complex language tasks where small adjustments significantly impact performance, which strongly depends on the data domain. The optimization cycle repeats until a stopping criterion is met, which can be metric stabilization, a time limit, maximum iterations, or satisfactory performance. Upon completion, consolidated results are presented, including the best fine-tuning configuration, trained models, execution logs, and evaluation metric values.

This Processing structure, based on the continuous interaction between empirical evaluation and automated search, forms the core of the AutoLLM approach. It enables the systematic application of supervised learning to language models and can be adapted to different tasks. The next subsection further explores this workflow by analyzing in greater detail the central component of the pipeline: the LLM Training and Testing block, which begins to operationalize both full and lightweight supervised fine-tuning strategies for optimization.

3.3. Supervised Fine-Tuning Strategies in AutoLLM

This subsection details the LLM Training and Testing block's internal structure, focusing on supervised fine-tuning logic. Figure 3 presents a hierarchical view of AutoLLM's proposed supervised fine-tuning strategies, organizing the main approaches integrated into

the optimization pipeline. The figure highlights two main paths: full fine-tuning, which adjusts all model weights, and light fine-tuning, which involves PEFT. In AutoLLM, the model training and testing block applies these strategies based on optimization algorithm configurations. Full supervised fine-tuning adjusts all base LLM parameters using a labeled dataset for specific tasks (e.g., classification, QA, text generation, sentiment analysis). This computationally expensive approach offers the most complete model specialization. AutoLLM explores hyperparameter combinations (e.g., learning rate, epochs, batch size, optimizer) for dynamic adjustment based on optimization cycle results.

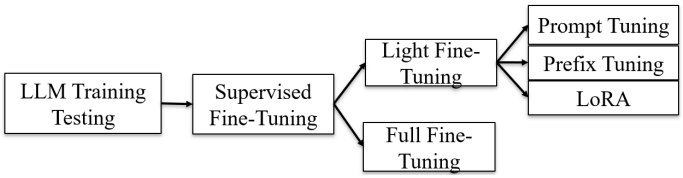

Figure 3. Hierarchical View of Supervised Fine-tuning Strategies.

The second path, lightweight supervised fine-tuning—preserves the base model's original weights by injecting trainable vectors or small modules, making it ideal under constraints of computation, time, or storage. In AutoLLM, this involves automated selection of techniques (e.g., Prompt Tuning, Prefix Tuning, or LoRA), target layers, and vector dimensions. Injected vectors include trainable embeddings or prefixes added to input or intermediate layers, while small modules like LoRA layers or Adapters integrate into the Transformer without altering its original weights. Although this approach still requires labeled data and backpropagation, it operates with significantly fewer trainable parameters.

The two fine-tuning strategies serve different purposes depending on task complexity, data availability, and performance requirements. Specialized tasks, such as legal document analysis, may require full fine-tuning, while generalist or multitask scenarios benefit from PEFT techniques for their modularity and efficiency. In such cases, a single model can handle tasks like classification, summarization, and question answering without full retraining. AutoLLM leverages this modularity to choose between fine-tuning strategies based on task demands and available resources. The following subsection demonstrates how AutoLLM is configured for lightweight supervised tuning on tasks such as summarization, text-to-SQL generation, and semantic similarity analysis, highlighting its adaptability and efficiency across various NLP applications.

4. Evaluation of AutoLLM with QLoRA

This section demonstrates the AutoLLM pipeline's feasibility using QLoRA, a parameter-efficient supervised fine-tuning technique. Subsection 4.1 describes the lightweight fine-tuning pipeline's structure and operation, detailing how genetic algorithms optimize LoRA-specific hyperparameters across NLP tasks. Subsection 4.2 then presents experimental results from applying the pipeline to the summarization task.

4.1. Lightweight Fine-Tuning Pipeline with QLoRA

Figure 4 illustrates AutoLLM's processing phase, depicting optimization dynamics via language model training and testing. This phase executes lightweight supervised fine-tuning using LoRA and QLoRA, guided by a bioinspired optimization algorithm. The ar-

chitecture comprises interconnected blocks forming an iterative model evaluation and refinement cycle. Each component of the figure is subsequently described per the pipeline's flow.

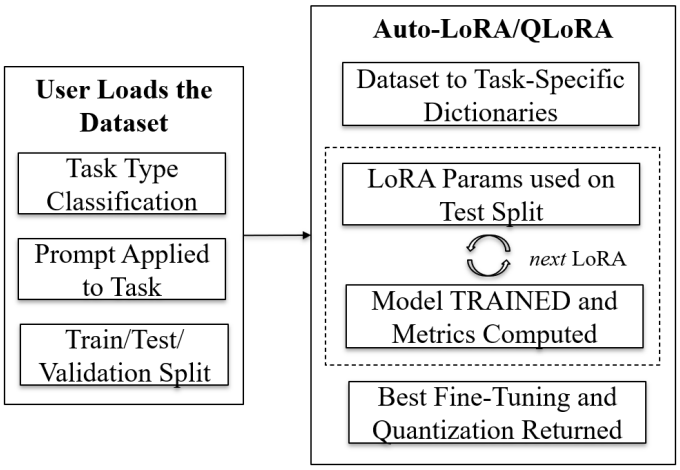

Figure 4. Auto-QLoRA Workflow for Dataset-Driven Fine-tuning.

The process begins by defining objective functions, which guide the optimization algorithm using task-specific performance metrics. For language models, these include ROUGE for summarization; ROUGE and BLEU for information retrieval; and Execution Accuracy for text-to-SQL generation. Incorporating these metrics enables multi-criteria optimization across various evaluation criteria.

Following the definition of the search space, Table 1 details the specific hyperparameters considered for optimization in the context of QLoRA adaptations. The rank (r) determines the dimensionality reduction applied during the low-rank decomposition, influencing the expressiveness versus efficiency trade-off. The scaling factor (lora_alpha) controls the magnitude of the LoRA update, affecting the model's ability to adjust without destabilizing the pretrained weights. The dropout probability (lora_dropout) introduces regularization during fine-tuning to prevent overfitting, especially in low-data regimes.

Table 1. LoRA Hyperparameters Used for Chromosome Encoding

Hyperparameter	Description	Options
r	LoRA decomposition rank	8, 12, 16
lora_alpha	LoRA scaling factor	16, 32, 48, 64
lora_dropout	Dropout probability in LoRA	None, 0.1, 0.2, 0.3
q	Apply LoRA to query module	True, False
k	Apply LoRA to key module	True, False
v	Apply LoRA to value module	True, False
o	Apply LoRA to output module	True, False
gate_proj	Apply LoRA to gate projection	True, False

Five Boolean flags (q, k, v, o, and gate_proj) control the application of LoRA adaptations to specific transformer modules, allowing fine-grained control over the adaptation scope. This hyperparameter space supports both aggressive strategies—with broader

adaptation, higher ranks, and lower dropout—and conservative ones, which limit changes to preserve pretrained knowledge. The latter is especially effective in low-data or overfitting-prone scenarios, as observed in the present experiment.

Once the search space is defined, the optimization program, implemented as a Genetic Algorithm, explores QLoRA configurations encoded as chromosomes. In each generation, individuals are evaluated using predefined metrics, selected based on performance, and modified through crossover and mutation. The fitness function combines multiple metrics to guide the search toward better solutions. For the experiments, the GA was configured with 10 generations, a population size of 10, Binary Tournament selection, one-point crossover, and a mutation rate of 10%.

Each generated configuration is applied to a pretrained LLM for lightweight supervised fine-tuning, after which the model is evaluated on a validation subset. The resulting performance metrics feed back into the GA's evolutionary cycle. This process is repeated iteratively until a stopping condition is met, such as reaching the maximum number of generations, exceeding a time limit, or achieving convergence of evaluation metrics. Once the criterion is satisfied, the pipeline consolidates the results and returns the best QLoRA configuration found, along with its corresponding hyperparameters and validation performance.

This modular structure enables AutoLLM to be applied across different types of NLP tasks. In the present study, the pipeline was tested on a summarization scenario. The next section details the results obtained for this task using the 3.0.0 version of the CNN DailyMail Dataset, highlighting the efficiency of the approach in performing lightweight adaptation of LLMs across different input profiles and evaluation criteria.

4.2. Experimental Results on NLP Tasks Using AutoLLM

The results indicate that the proposed AutoLLM pipeline is effective for supervised fine-tuning of language models. Compared to the base model, consistent improvements in prediction quality were achieved by applying lightweight QLoRA-based adaptations optimized through a bioinspired algorithm. Initial experiments showed gains in both general and task-specific metrics, such as loss and ROUGE for summarization. As shown in Figure 5, the best ROUGE scores improved across generations of the GA, with the linear fit (red line) highlighting a progressive improvement in output quality despite occasional fluctuations.

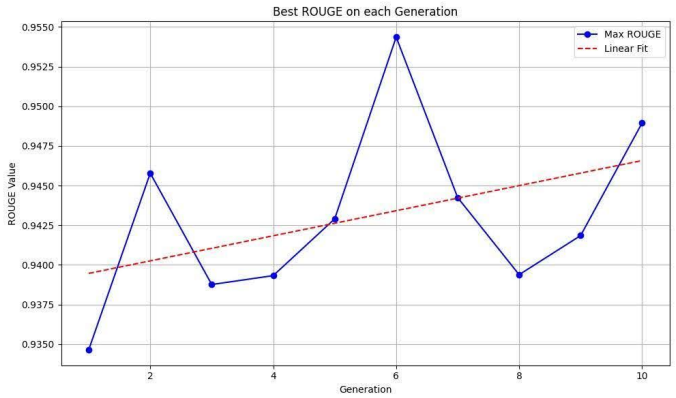

Figure 5. Best ROUGE value on each generation.

Figure 6 shows that, even with a maximum of 5 fine-tuning steps, the AutoLLM pipeline improved ROUGE from 0.9242 to 0.9543, outperforming the base model. This demonstrates the pipeline's ability to find effective parameter combinations under computational constraints. Although running on a high-performance NVIDIA H100 GPU, the setup was limited to 16GB of VRAM, restricting model size and batch configuration. Despite these limitations, the genetic algorithm completed in about 450 seconds, yielding strong results through efficient and constrained optimization.

Finally, the average ROUGE score across 10 independent GA runs was 0.9503, with a low standard deviation of 0.0027. This consistency indicates that the pipeline's performance is not dependent on a single run but reflects a stable improvement trend. The low variability highlights the robustness of the AutoML-based approach, suggesting that combining QLoRA strategies with bioinspired optimization yields replicable and generalizable results.

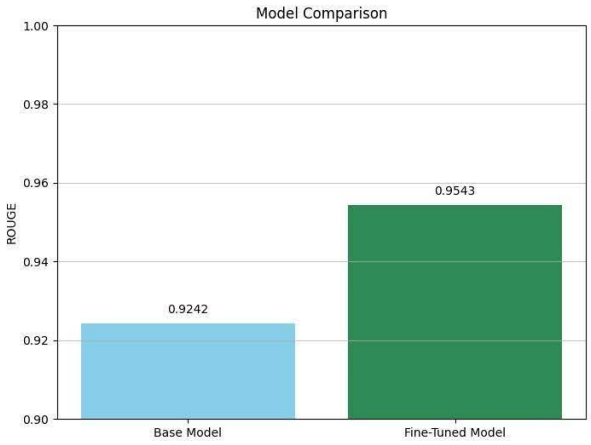

Figure 6. Comparison of the base model with the adjusted model.

5. Conclusions and Future Work

Although this represents an initial implementation, the results and the proposed framework allow us to draw preliminary conclusions and highlight important directions for future developments. The AutoLLM pipeline has shown promise in integrating supervised fine-tuning strategies with automated optimization methods, allowing consistent gains in language models with less manual effort and more efficient use of computational resources. Moreover, the modularity of the approach facilitates its adaptation to different tasks and contexts, and the use of lightweight fine-tuning, such as QLoRA, proved both viable and efficient, reinforcing the potential of the proposed method in scenarios with time and resource constraints.

For future work, we intend to extend the experiments along two distinct directions. The first involves SQL query generation, evaluating the generalization and robustness of the model across different data domains. The second line focuses on incorporating information retrieval tasks, which will require adaptations to the pipeline to handle documents and open-domain queries.

Additionally, we plan to conduct a broader quantitative evaluation by comparing AutoLLM with other automated approaches, such as grid search and frameworks like Optuna. This includes a comparative analysis of computational costs, including processing time and computational resource usage, to better contextualize the practical advantages of AutoLLM. We hope that these efforts will establish AutoLLM as a versatile and efficient tool for the supervised adaptation of LLMs in different real scenarios.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *The journal of machine learning research*, 13(1):281–305.

Bouthillier, X., Delaunay, P., Bronzi, M., Trofimov, A., Nichyporuk, B., Szeto, J., Mohammadi Sepahvand, N., Raff, E., Madan, K., Voleti, V., et al. (2021). Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3:747–769.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21.

Halfon, A., Gretz, S., Arviv, O., Spector, A., Toledo-Ronen, O., Katz, Y., Ein-Dor, L., Shmueli-Scheuer, M., and Slonim, N. (2024). Stay tuned: An empirical study of the impact of hyperparameters on llm tuning in real-world applications. *arXiv preprint arXiv:2407.18990*.

Hasan, A., Ehsan, M. A., Shahnoor, K. B., and Tasneem, S. S. (2024). *Automatic question & answer generation using generative Large Language Model (LLM)*. PhD thesis, Brac University.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hossain, M. R. and Timmer, D. (2021). Machine learning model optimization with hyper parameter tuning approach. *Glob. J. Comput. Sci. Technol. D Neural Artif. Intell*, 21(2):31.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Jelinek, F. (1998). *Statistical methods for speech recognition*. MIT press.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Lu, Y., Huang, C., Zhan, H., and Zhuang, Y. (2021). Federated natural language generation for personalized dialogue system. *arXiv preprint arXiv:2110.06419*.

Mulakala, B., Saini, M. L., Singh, A., Bhukya, V., and Mukhopadhyay, A. (2024). Adaptive multi-fidelity hyperparameter optimization in large language models. In *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–5. IEEE.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Tornede, A., Deng, D., Eimer, T., Giovanelli, J., Mohan, A., Ruhkopf, T., Segel, S., Theodorakopoulos, D., Tornede, T., Wachsmuth, H., et al. (2023). Automl in the age of large language models: Current challenges, future opportunities and risks. *arXiv preprint arXiv:2306.08107*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tribes, C., Benarroch-Lelong, S., Lu, P., and Kobyzev, I. (2023). Hyperparameter optimization for large language model instruction-tuning. *arXiv preprint arXiv:2312.00949*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, B., Shin, R., Liu, X., Polozov, O., and Richardson, M. (2019). Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

Yao, Q., Wang, M., Chen, Y., Dai, W., Li, Y.-F., Tu, W.-W., Yang, Q., and Yu, Y. (2018). Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 31.

Zöller, M.-A. and Huber, M. F. (2019). Survey on automated machine learning. *arXiv preprint arXiv:1904.12054*, 9:844.