Implementation of system code generation based on Large Language Model Fine-tune with QLoRA

1st Muhamad Raihan Syahrin Sya'bani
School of Computing
Telkom University
Bandung, Indonesia
raihansyahrin@student.telkomuniversity.ac.id

2nd Donni Richasdy
School of Computing
Telkom University
Bandung, Indonesia
donnir@telkomuniversity.ac.id

3rd Dana Sulistyo Kusumo
School of Computing
Telkom University
Bandung, Indonesia
danakusumo@telkomuniversity.ac.id

***Abstract*—This study aims to improve code generation performance by applying parameter-efficient fine-tuning using Quantized Low-Rank Adaptation (QLoRA). Currently, large language models (LLMs) in code generation continue to face deployment challenges in low-resource environments, particularly due to high computational demands. The core problem addressed in this study is the inefficiency and limited adaptability of pre-trained models in producing correct code under constrained resource conditions, which results in decreased output quality and restricts accessibility for low-resource users. While previous approaches have employed full fine-tuning on large-scale datasets to mitigate these issues—yielding improvements in generalization—they remain hindered by substantial memory usage and computational cost. This study analyzes a compact fine-tuning pipeline utilizing QLoRA, applied to the Qwen2.5-Coder-0.5B-Instruct model, to address these constraints and improve generation accuracy with minimal resource consumption. The proposed system was fine-tuned using two benchmark datasets—CodeExercise-Python-27k and Tested-22k-Python-Alpaca—and demonstrated performance improvements of up to 7.3% on HumanEval and 4.3% on HumanEval+ in pass@1 metrics, compared to the base model. These findings confirm that fine-tuning with specific datasets, with lightweight methods like QLoRA, significantly enhances the effectiveness of compact LLMs in code generation, contributing to advancements in software engineering, AI-assisted learning, and low-resource-constrained development platforms.**

Keywords—Code Generation, Large Language Models, QLoRA, Fine-Tuning, HumanEval.

I. INTRODUCTION

Every year, the integration of artificial intelligence (AI) into software development pipelines sees a big increase, according to the statistics [1], [2]. The Artificial Intelligence Index Report 2023 shows a massive surge in AI-related projects on GitHub, rising from just 1,536 in 2011 to 347,934 by 2022 [3]. This explosive growth reflects the trends of using AI-powered tools to streamline software development, reduce manual effort, and improve overall productivity. At the same time, this expansion opens up completely challenges—particularly concerning the accuracy, reliability, and transparency of AI-generated code—highlighting the need for more effective integration of AI within the software development process [4].

In recent years, advances in the field of AI technologies—particularly in the Large Language Models (LLMs)—have opened new possibilities for automated code generation [5], [6]. These developments have significantly transformed the software engineering landscape, affecting tasks such as coding, debugging, testing, and system design [7], [8]. Notable models such as CodeLlama [9], Qwen2.5-Coder [10], and CodeBERT [11] are examples of this transformation, demonstrating this advancement by converting natural language descriptions into functional source code. Although various innovations have been achieved, the rapid progress in the development of LLMs also raises fundamental challenges related to context understanding and syntactic appropriateness, which have a direct impact on the effectiveness of their use in real-world programming practices [12], [13].

This problem is critical in the domain of AI-assisted software development due to its effect on model performance, scalability, and trustworthiness, particularly as software complexity increases [1]. Ensuring that AI-generated code is not only syntactically correct but also semantically valid and maintainable becomes essential for practical adoption [14]. Therefore, the challenge lies not only in generating code but also in doing so with high accuracy, contextual relevance, and computational efficiency. These concerns highlight the urgent need for more effective fine-tuning strategies that address these shortcomings.

Although various methods have been developed, the full fine-tuning process of LLM still requires enormous resources [15], [16], making it impractical for many real-world applications, especially in models with billions of parameters. This challenge has led to the emergence of alternative approaches that are more resource-efficient while still maintaining high performance. One example is Low-Rank Adaptation (LoRA) [17] and its extension, Quantized LoRA (QLoRA) [18], which offers a more storage-efficient solution. By using the QLoRA approach, the fine-tuning process becomes more scalable and accessible, enabling a wider application of LLM in various real-world software engineering tasks.

Research in this domain has experienced rapid development, with many previous studies emphasizing the use of PEFT methods to optimize models like CodeGen [19] and CodeT5+ [20]. However, recent works demonstrate that QLoRA enables fine-tuning models of up to 65 billion parameters on a single 48GB GPU, while maintaining performance even under 16-bit precision [18]. These advancements motivate further investigation into QLoRA's applicability for smaller open-source models under code generation tasks.

This research contributes by proposing a fine-tuning approach using QLoRA that is both computationally efficient and performance-oriented for the task of code generation. The proposed method involves two main stages: first, learning rate optimization using a 50% data subset to minimize resource usage; and second, full-scale training and evaluation using metrics like CodeBLEU [21], as well as Pass@k accuracy measured on HumanEval [22], and HumanEval+ [23]. This pipeline is designed to maximize output quality while minimizing hardware demands.