Quantization Impact on LLaMA 3.2 and Mistral Code Repair Accuracy in BugsInPy

Assignee Research

June 8, 2026

Abstract

This report synthesises findings from 5 peer-reviewed papers addressing the following research question: How does 4-bit quantization affect the code repair accuracy of LLaMA 3.2 versus Mistral on the BugsInPy dataset compared to FP16 baselines. 10 claims were extracted from source literature; 6 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 6.7/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Are Large Language Models Memorizing Bug Benchmarks?. Research question: How does 4-bit quantization affect the code repair accuracy of LLaMA 3.2 versus Mistral on the BugsInPy dataset compared to FP16 baselines?.

2 Methodology

Systematic literature search across multiple databases yielded 5 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 6.7/10.

3 Results

5 papers retrieved. 10 claims extracted; 6 independently verified. Quality review score: 6.7/10.

4 Limitations

This report is a machine-generated literature synthesis and does not constitute original research. Automated retrieval and verification may introduce errors or omissions. Review scores reflect automated assessment, not human peer review. Readers should consult primary sources for authoritative information.

5 Extracted Claims

Claim	Verified	Confidence
Large Language Models (LLMs) are used for code generation, bug detection, and repair.	✓	0.24
Numerous bug benchmarks containing real-world bugs from software projects have been developed to evaluate LLM performanc	✓	0.29
There is a concern in the software engineering community that bug benchmarks may not reliably reflect true LLM performan	✓	0.32
Limited research has been conducted to quantify the impact of potential data leakage in bug benchmarks.	✓	0.27
The study uses benchmark membership analysis within commonly used training datasets to identify potential leakage.	✓	0.21
The study uses negative log-likelihood analysis to identify potential leakage.	✓	0.16
The study uses n-gram accuracy analysis to identify potential leakage.	×	0.13
The model codegen-multi exhibits significant evidence of memorization in the Defects4J benchmark.	×	0.15
The model LLaMa 3.1 exhibits limited signs of data leakage.	×	0.13
LLaMa 3.1 was trained on larger datasets compared to models showing significant memorization.	×	0.12

References

- https://doi.org/10.48550/arxiv.2411.13323
- https://openalex.org/W7158423025

- https://doi.org/10.48550/arxiv.2303.17651