Codestral and Llama3 Inference Latency and Throughput on LiveCodeBench Programming Tasks

Assignee Research

June 4, 2026

Abstract

This report synthesises findings from 4 peer-reviewed papers addressing the following research question: How do inference latency and token throughput differ between Codestral and Llama3 when generating solutions for LiveCodeBench's multi-step programming problems. 8 claims were extracted from source literature; 8 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 8.5/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Performance analysis of localised large language models in resource-constrained edge for Python and Rust APIs. Research question: How do inference latency and token throughput differ between Codestral and Llama3 when generating solutions for LiveCodeBench's multi-step programming problems?.

2 Methodology

Systematic literature search across multiple databases yielded 4 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 8.5/10.

3 Results

4 papers retrieved. 8 claims extracted; 8 independently verified. Quality review score: 8.5/10.

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
Edge deployments of large language models (LLMs) often suffer from significant latency due to the overhead of high-level	✓	0.36
Four quantised LLMs – Llama 3.2:1b, Gemma 3:1b, Granite 3.1-MoE:1b, and Qwen 2.5:0.5b – were tested on a Raspberry Pi 4	✓	0.36
Each model was served via a local Ollama inference server.	✓	0.17
A fixed suite of twenty prompts – covering factual retrieval, arithmetic reasoning, translation, code synthesis, and cre	✓	0.36
Rust markedly reduces cold-start delays: mean model load times fall from 1 648.7 ms (Python) to 52.8 ms (Rust) for Llama	✓	0.43
Corresponding end-to-end latencies decrease by 1.4-2.0 s across models.	✓	0.18
In warm-start conditions, both clients deliver nearly identical decoding throughput – $\approx$2.7 tokens/s for Llama 3.2:1b, 4.	✓	0.39
Rigorous statistical testing, including paired t-tests, Mann-Whitney U tests, and bootstrap confidence intervals, confir	✓	0.34

References

- https://www.semanticscholar.org/paper/2e7364728eb82206211c43c08eb580e322bbe4f8
- https://arxiv.org/abs/2602.16760
- https://www.semanticscholar.org/paper/79baa1ab34b5e2623c2936d1bf670fd65cb45551