Scaling Accuracy-Efficiency Trade-offs in Lugha-Llama Models for Code Generation Benchmarks

Assignee Research

June 7, 2026

Abstract

This report synthesises findings from 11 peer-reviewed papers addressing the following research question: How does the accuracy-efficiency trade-off of Lugha-Llama-8B-wura-Pre-TLI scale with model size (e.g., 8B vs 34B) on HumanEval-V compared to text-only code generation benchmarks like HumanEval or MBPP. 15 claims were extracted from source literature; 2 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 4.8/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: HumanEval Pro and MBPP Pro: Evaluating Large Language Models on Self-invoking Code Generation. Research question: How does the accuracy-efficiency trade-off of Lugha-Llama-8B-wura-Pre-TLI scale with model size (e.g., 8B vs 34B) on HumanEval-V compared to text-only code generation benchmarks like HumanEval or MBPP?.

2 Methodology

Systematic literature search across multiple databases yielded 11 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 4.8/10.

3 Results

11 papers retrieved. 15 claims extracted; 2 independently verified. Quality review score: 4.8/10.

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
o1-mini achieves 96.2% pass@1 on HumanEval but only 76.2% on HumanEval Pro	✓	0.25
Instruction-tuned models are less efficient on self-invoking code generation than traditional code generation tasks	✓	0.32
HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) serve as fundamental benchmarks, focusing on Python functio	×	0.05
Several benchmarks have expanded code evaluation benchmarks to encompass multiple programming languages (Zheng et al., 2	×	0.05
Benchmarks have expanded to include complex tasks like program repair (Haque et al., 2022; Jiang et al., 2023; Muennigho	×	0.03
Benchmarks have expanded to include dynamic problem sets (Jain et al., 2024)	×	0.04
Benchmarks have expanded to include code reasoning through code summarization (Barone and Sennrich, 2017; Hasan et al.,	×	0.05
Deepseek-V2.5 is used to generate self-invoking problems, candidate solutions, and test inputs	×	0.08
Generated solutions are executed with test inputs in a controlled Python environment to obtain ground truth outputs	×	0.02
An iterative method involving Python execution check and manual review ensures all test cases pass successfully	×	0.03
Final execution results are used to construct complete test cases with assert command	×	0.03
Qwen2.5-Coder-7B-base has a pass rate of 59.6% on HumanEval Pro and 38.6% on MBPP Pro	×	0.11
Qwen2.5-Coder-7B-instruct has a pass rate of 64.9% on HumanEval Pro and 35.1% on MBPP Pro	×	0.09
DeepseekCoder-33B-base has a pass rate of 71.9% on HumanEval Pro and 38.6% on MBPP Pro	×	0.12
DeepseekCoder-33B-instruct has a pass rate of 80.7% on HumanEval Pro and 43.9% on MBPP Pro	×	0.09

References

- http://arxiv.org/abs/2410.12381v3
- http://arxiv.org/abs/2306.08568v2
- http://arxiv.org/abs/2412.21199v2