Sparse MoE and Dense Architectures in Code Generation: Throughput and Latency Benchmarks

Assignee Research

June 4, 2026

Abstract

This report synthesises findings from 14 peer-reviewed papers addressing the following research question: What is the difference in inference throughput and token generation latency between sparse MoE and dense architectures when evaluated on code generation tasks like HumanEval. 12 claims were extracted from source literature; 3 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: What is the difference in inference throughput and token generation latency between sparse MoE and dense architectures when evaluated on code generation tasks like HumanEval?.

2 Methodology

Systematic literature search across multiple databases yielded 14 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

14 papers retrieved. 12 claims extracted; 3 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.	✓	0.19
o1-mini achieves 76.2% pass@1 on HumanEval Pro.	✓	0.21
Instruction-tuned models are less efficient on self-invoking code generation than on traditional code generation tasks.	✓	0.30
HumanEval and MBPP serve as fundamental benchmarks focusing on Python function completion tasks with test-driven evaluat	×	0.08
Deepseek-V2.5 was used to generate self-invoking problems, candidate solutions, and test inputs for the benchmark.	×	0.03
OpenCoder-8B-base achieved a score of 56.1 on the Base Problem and 10.5 on the Self-invoking Problem.	×	0.09
OpenCoder-8B-instruct achieved a score of 75.4 on the Base Problem and 22.8 on the Self-invoking Problem.	×	0.08
DeepseekCoder-6.7B-base achieved a score of 59.6 on the Base Problem and 35.1 on the Self-invoking Problem.	×	0.09
DeepseekCoder-6.7B-instruct achieved a score of 56.1 on the Base Problem and 35.1 on the Self-invoking Problem.	×	0.09
Qwen2.5Coder-7B-instruct achieved a score of 64.9 on the Base Problem and 35.1 on the Self-invoking Problem.	×	0.08
DeepseekCoder-33B-instruct achieved a score of 80.7 on the Base Problem and 43.9 on the Self-invoking Problem.	×	0.08
Yi-Coder-9B-Chat achieved a score of 66.7 on the Base Problem and 31.6 on the Self-invoking Problem.	×	0.07

References

- http://arxiv.org/abs/2411.11217v1
- http://arxiv.org/abs/2605.11733v1
- http://arxiv.org/abs/2412.21199v2