Code-Specific Data Augmentation Effects on Pass@1 in Code Generation Models

Assignee Research

June 4, 2026

Abstract

This report synthesises findings from 11 peer-reviewed papers addressing the following research question: What is the effect of code-specific data augmentation on the pass@1 scores of code generation models across diverse programming language datasets. 9 claims were extracted from source literature; 2 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 5.2/10. 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 effect of code-specific data augmentation on the pass@1 scores of code generation models across diverse programming language datasets?.

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 5.2/10.

3 Results

11 papers retrieved. 9 claims extracted; 2 independently verified. Quality review score: 5.2/10.

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.	✓	0.26
Instruction-tuned models are less efficient on self-invoking code generation than traditional code generation tasks.	✓	0.31
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	×	0.06
The benchmark construction process involves three steps: (1) Self-invoking problem Generation, (2) Solutions Generation,	×	0.11
Deepseek-V2.5 is used to generate the self-invoking problems, as well as their candidate solutions and test inputs.	×	0.07
The final execution results are used to construct complete test cases with assert command.	×	0.03
Qwen2.5-Coder-7B-base achieves 59.6% on HumanEval Pro and 38.6% on MBPP Pro.	×	0.13
DeepseekCoder-33B-instruct achieves 80.7% on HumanEval Pro and 43.9% on MBPP Pro.	×	0.12

References

- http://arxiv.org/abs/2412.21199v2
- http://arxiv.org/abs/2410.12381v3
- http://arxiv.org/abs/2303.12869v1