Task-Specific Fine-Tuning Effects on Small vs. Large Language Models in Code Generation

Assignee Research

May 30, 2026

Abstract

This report synthesises findings from 9 peer-reviewed papers addressing the following research question: What is the impact of task-specific fine-tuning on the throughput and accuracy of small language models compared to large models in code generation benchmarks such as HumanEval. Large Language Models (LLMs) have demonstrated significant capabilities in understanding and analyzing code for security vulnerabilities, such as Common Weakness Enumerations (CWEs). However, their reliance on cloud infrastructure and substantial computational requirements pose. 12 claims were extracted from source literature; 1 was independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 5.6/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Case Study: Fine-tuning Small Language Models for Accurate and Private CWE Detection in Python Code. Research question: What is the impact of task-specific fine-tuning on the throughput and accuracy of small language models compared to large models in code generation benchmarks such as HumanEval?.

2 Methodology

Systematic literature search across multiple databases yielded 9 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 5.6/10.

3 Results

9 papers retrieved. 12 claims extracted; 1 independently verified. Quality review score: 5.6/10.

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
Farasat & Posegga [15] achieved an average accuracy of 98.6%, F1 score of 94.7%, precision of 96.2%, recall of 93.3%, an	×	0.09
Bagheri et al. [31] reported an F1 score of 99% using a hybrid model combining self-attention and CNN (Conformer).	×	0.03
Singh et al. [32] achieved an accuracy of 0.66, precision of 0.65, recall of 0.66, and F1 score of 0.64 for CWE predicti	×	0.04
Dozono et al. [6] reported Python F1 scores for various LLMs: GPT-4o=0.80, GPT-4T=0.76, Gemini 1.5 Pro=0.75, CodeLlama-7	×	0.03
Steenhoek et al. [34] evaluated LLMs on C/C++ and found a low balanced accuracy of 54.5%.	×	0.04
Shestov et al. [35] achieved an F1 score of 0.86 for binary classification in Java CWE detection using WizardCoder.	×	0.06
Li et al. [36] reported a 5-6% improvement over the base model using LoRa and IA3 fine-tuning approaches for LLMs.	×	0.06
Jiang et al. [37] achieved the best F1 score of 87% using the Llama 2-7b model with LoRa fine-tuning.	×	0.05
The un-tuned codegen-mono model failed to detect a single CWE within any of the 100 code snippets presented in the basel	×	0.12
After instruction-following fine-tuning, the codegen-mono model achieved an accuracy of 99%, precision of $\approx$98.08%, recal	✓	0.22
The MITRE Top 25 Most Dangerous Software Weaknesses list [30] was selected as the target for the detection model.	×	0.05
Google's gemini-2.0-flash-thinking-exp-01-21 model was used for generating synthetic data for the dataset.	×	0.04

References

- http://arxiv.org/abs/2504.16584v1

- http://arxiv.org/abs/2603.21389v1
- http://arxiv.org/abs/2306.08568v2