Learning Rate Annealing Effects on CodeT5+ HumanEval Pass@1 Performance

Assignee Research

June 8, 2026

Abstract

This report synthesises findings from 3 peer-reviewed papers addressing the following research question: How does learning rate annealing affect the code generation performance of CodeT5+ on HumanEval, measured by pass@1 scores across different annealing schedules. 11 claims were extracted from source literature; 10 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 6.9/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Hotfixing Large Language Models for Code. Research question: How does learning rate annealing affect the code generation performance of CodeT5+ on HumanEval, measured by pass@1 scores across different annealing schedules?.

2 Methodology

Systematic literature search across multiple databases yielded 3 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 6.9/10.

3 Results

3 papers retrieved. 11 claims extracted; 10 independently verified. Quality review score: 6.9/10.

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
Large Language Models for Code (LLM4Code) have become an integral part of developers' workflows, assisting with tasks su	✓	0.34
LLM4Code models exhibit undesired behaviors after their release, like generating buggy code, due to their extensive trai	✓	0.39
The training data for LLM4Code usually comes from open-source software.	×	0.15
Developers fix the buggy code in the training data over time.	✓	0.23
Retraining models on the updated dataset usually takes much time and resources.	✓	0.24
Hotfixing LLM4Code aims to mitigate undesired behaviors effectively and efficiently with minimal negative effects.	✓	0.27
This paper focuses on hotfixing LLM4Code to make them generate less buggy code and more fixed code.	✓	0.30
Models from the popular CodeGen family frequently generate buggy code.	✓	0.28
Three learning objectives are defined in hotfixing: (1) learn the desired behaviors, (2) unlearn the undesired behaviors	✓	0.29
Four different fine-tuning techniques are evaluated for hotfixing the models.	✓	0.16
Optimizing the three learning goals together, using LoRA (low-rank adaptation), effectively influences the model's behav	✓	0.29

References

- https://doi.org/10.48550/arxiv.2408.05727

- https://doi.org/10.48550/arxiv.2411.16341
- https://doi.org/10.48550/arxiv.2407.05700