Sliding Window vs. Full Attention in Long-Sequence Code Generation Accuracy

Assignee Research

May 31, 2026

Abstract

This report synthesises findings from 4 peer-reviewed papers addressing the following research question: What is the impact of sliding window attention on code generation accuracy compared to full attention mechanisms in long-sequence programming benchmarks. GitHub Copilot, an extension for the Visual Studio Code development environment powered by the large-scale language model Codex, makes automatic program synthesis available for software developers. This model has been extensively studied in the field of deep learning, however, a. 11 claims were extracted from source literature; 0 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 3.7/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Choose Your Programming Copilot: A Comparison of the Program Synthesis Performance of GitHub Copilot and Genetic Programming. Research question: What is the impact of sliding window attention on code generation accuracy compared to full attention mechanisms in long-sequence programming benchmarks?.

2 Methodology

Systematic literature search across multiple databases yielded 4 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 3.7/10.

3 Results

4 papers retrieved. 11 claims extracted; 0 independently verified. Quality review score: 3.7/10.

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
PSB1 was published in 2015.	×	0.00
PSB2 was published in 2021.	×	0.03
The publication of PSB1 in 2015 revived and consolidated the field of general program synthesis in GP.	×	0.12
PSB2 extends the collection of common problems from PSB1.	×	0.05
Three major categories of methods emerged from the GP field: stack-based GP, grammar-guided GP, and linear GP.	×	0.01
Stack-based approaches like PushGP use stacks to differentiate between the supported data types.	×	0.03
The success rates of PushGP in program synthesis could be increased significantly with the introduction of the UMAD oper	×	0.05
Grammar-guided GP approaches can represent common programming languages like Python.	×	0.05
Hemberg et al. used not only the GP-typical input/output examples, but also the textual problem descriptions of the benc	×	0.07
Linear GP uses data registers similar to those known from common Assembler languages.	×	0.02
Lalejini and Ofria suggested tag-based memory to make the program representation more stable against changes by variatio	×	0.04

References

- http://arxiv.org/abs/2111.07875v1
- http://arxiv.org/abs/2106.06086v1
- http://arxiv.org/abs/1909.12281v1