Convolutional and Graph Neural Networks for Code Property Graph Analysis with Commit Embeddings

Assignee Research

June 1, 2026

Abstract

This report synthesises findings from 16 peer-reviewed papers addressing the following research question: What is the effect of using convolutional neural networks versus graph neural networks on the inference efficiency and detection accuracy when processing code property graphs augmented with commit. The increasing complexity of modern software systems has led to a rise in vulnerabilities that malicious actors can exploit. Traditional methods of vulnerability detection, such as static and dynamic analysis, have limitations in scalability and automation. 12 claims were extracted from source literature; 1 was independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 4.2/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Enhancing Software Vulnerability Detection Using Code Property Graphs and Convolutional Neural Networks. Research question: What is the effect of using convolutional neural networks versus graph neural networks on the inference efficiency and detection accuracy when processing code property graphs augmented with commit message embeddings?.

2 Methodology

Systematic literature search across multiple databases yielded 16 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 4.2/10.

3 Results

16 papers retrieved. 12 claims extracted; 1 independently verified. Quality review score: 4.2/10.

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
Wang et al. applied deep learning techniques to detect software defects, using Abstract Syntax Trees (ASTs) to model the	×	0.11
ASTs are useful for understanding the syntactic structure of code, but they do not capture the dynamic aspects of progra	×	0.06
The method incorporates Control Flow Graphs (CFGs) and Program Dependency Graphs (PDGs) alongside ASTs, enabling a more	×	0.11
Zhang et al. applied graph-based deep learning models for vulnerability detection in C/C++ code.	×	0.15
Their method incorporated both syntax and semantic information, but it did not fully exploit the interdependencies betwe	×	0.04
Graph Neural Networks (GNNs) have demonstrated their efficacy in a variety of domains, including software analysis.	×	0.09
GNNs are capable of processing graph-structured data, learning relationships between nodes and edges, and have been succ	×	0.06
Li et al. introduced the Gated Graph Sequence Neural Network (GGS-NN), which extended traditional GNNs to handle sequenc	×	0.05
Their method achieved promising results, identifying patterns indicative of malware.	×	0.01
Liang et al. applied ensemble models combining decision trees and CNNs for malware classification.	×	0.03
Their results showed improved detection rates compared to individual models, particularly in complex malware scenarios.	×	0.05
Yamaguchi et al. introduced Code Property Graphs (CPGs), which combine syntactic, control flow, and data flow informatio	✓	0.18

References

- http://arxiv.org/abs/2603.29216v1

- http://arxiv.org/abs/2503.18175v1
- http://arxiv.org/abs/2211.12792v2