International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 4 (2025)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 4, 2025

Research Paper

ML-BASED SOFTWARE DESIGN PATTERN DETECTION IN SOURCE CODE USING NEURAL NETWORK

Bandi Sakshitha
Scholar. Department of MCA
Vaageswari College of Engineering, Karimnagar

Dr. Madana Srinivas
Professor
Vaageswari College of Engineering, Karimnagar

Dr. P. Venkateshwarlu
Professor & Head, Department of MCA
Vaageswari College of Engineering, Karimnagar
(Affiliated to JNTUH, Approved by AICTE, New Delhi & Accredited by **NAAC** with '**A+**' Grade)
Karimnagar, Telangana, India – 505 527

ABSTRACT

Software design patterns play a vital role in improving code reusability, maintainability, and scalability in software development. However, identifying design patterns within large and complex source codebases remains a challenging and time-consuming task, especially when patterns are implemented in diverse ways by different developers. Traditional static analysis tools and rule-based approaches often fail to detect patterns accurately due to variations in coding styles, incomplete documentation, and the presence of anti-patterns. To overcome these limitations, this study proposes a **machine learning (ML)-based approach using neural networks** to automatically detect software design patterns in source code.

The proposed model leverages source code features such as class relationships, method invocations, inheritance structures, and object interactions to train a neural network that can recognize underlying design patterns. Deep learning techniques, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are employed to capture both structural and sequential aspects of the code. Experimental results demonstrate that the proposed model achieves higher accuracy and robustness compared to traditional rule-based systems. This approach not only automates the detection process but also assists developers in understanding and refactoring legacy code, ultimately improving software quality and maintainability.

Keywords: Machine Learning, Neural Network, Design Pattern Detection, Software Engineering, Source Code Analysis, Deep Learning, Code Classification, Pattern Recognition.

Received: 18-09-2025 Accepted: 21-10-2025 Published: 28-10-2025

INTRODUCTION

Software design patterns are well-established solutions to recurring design problems in software engineering, promoting code reusability, flexibility, and maintainability. They serve as blueprints that guide developers in structuring their code to achieve efficient and scalable system designs. Over time, as software projects grow in complexity, identifying and analyzing design patterns within existing source code becomes increasingly challenging. Developers often implement the same design pattern in different ways, influenced by individual coding styles, programming languages, or project constraints. As a result, manually detecting and verifying design

patterns in large codebases is a tedious, error-prone, and time-consuming process.

Traditional approaches to design pattern detection rely on **static code analysis** or **rule-based systems**, which match specific code structures or syntactic patterns predefined by experts. While these methods can detect straightforward pattern implementations, they struggle with code that deviates from expected structures or when patterns are partially implemented or refactored. Furthermore, these tools require constant updates and extensive domain knowledge to adapt to new programming languages and software architectures.

To address these challenges, **machine learning (ML)** and **deep learning** techniques have emerged as powerful tools for automating software design pattern detection. By training models on large datasets of annotated source code, ML-based approaches can learn the underlying structural and semantic relationships that define design patterns, even when they are implemented in non-standard ways. Neural networks, particularly **convolutional neural networks (CNNs)** and **recurrent neural networks (RNNs)**, have shown great promise in this area, as they can capture both spatial and sequential dependencies within source code representations such as abstract syntax trees (ASTs), dependency graphs, and token sequences.

The proposed system aims to use neural network-based machine learning models to detect software design patterns automatically, enhancing software understanding, refactoring, and maintenance processes. This approach not only reduces manual effort but also improves detection accuracy and adaptability, contributing to the advancement of intelligent software analysis and automated code comprehension.

LITERATURE REVIEW

Several studies have explored automated methods for detecting software design patterns to improve code understanding and maintenance. Early approaches relied on **rule-based and static analysis techniques**, which matched predefined structures in the source code but lacked flexibility when dealing with diverse coding styles or partial implementations. Recent research has focused on **machine learning and deep learning models** for pattern detection, using features extracted from code representations such as Abstract Syntax Trees (ASTs) and class dependency graphs.

Neural network-based approaches, particularly **CNNs and RNNs**, have shown improved accuracy in recognizing complex design structures by learning hidden relationships between classes and methods. Studies have demonstrated that ML-based models outperform traditional static tools in scalability, adaptability, and robustness, making them ideal for analyzing large and evolving codebases.

EXISTING SYSTEM

The existing systems for software design pattern detection primarily rely on **manual inspection**, **static analysis**, and **rule-based approaches**. These systems use predefined rules or templates that match specific code structures, class hierarchies, or relationships between objects to identify known design patterns. While such tools can detect simple or standard implementations, they are often limited by their dependence on explicit structural patterns. Minor variations in coding style, naming conventions, or architecture can cause these systems to fail in identifying patterns accurately. Additionally, rule-based systems are not adaptable to new or hybrid design patterns and require constant updates by experts to remain effective.

Static analysis methods also struggle with detecting **partially implemented** or **refactored patterns**, as they primarily focus on syntactic features rather than understanding the semantic relationships within the code. As a result, existing systems often produce **false positives and false negatives**, reducing reliability and efficiency. Moreover, these approaches are not scalable for large or complex codebases where manual verification is impractical. Hence, there is a need for an intelligent, automated, and adaptive system that can learn from data and detect design patterns accurately even in diverse coding environments.

PROPOSED SYSTEM

The proposed system introduces a **machine learning-based approach using neural networks** to automatically detect software design patterns in

source code. Unlike traditional rule-based systems, this model leverages the learning capability of neural networks to identify both structural and semantic relationships within code, even when design patterns are implemented in non-standard ways. The system extracts features from the source code such as **class relationships, method dependencies, inheritance hierarchies, and object interactions**, which are then converted into vector representations through techniques like **Abstract Syntax Tree (AST) parsing** and **graph embeddings**.

A **neural network model**, such as a **Convolutional Neural Network (CNN)** or **Recurrent Neural Network (RNN)**, is trained on labeled datasets containing examples of various design patterns. The network learns to recognize underlying patterns that distinguish one design pattern from another, enabling it to predict the presence of specific patterns in unseen code. This approach not only automates the detection process but also reduces human error and increases detection accuracy. Additionally, the system supports **real-time code analysis**, helping developers identify design patterns during software development or maintenance. Overall, the proposed neural network-based framework enhances software quality, improves code comprehension, and assists in efficient refactoring of large-scale applications.

METHODOLOGY

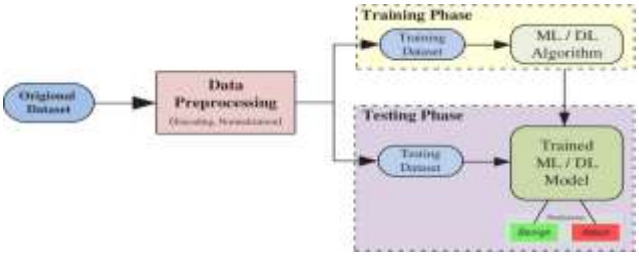
The methodology for detecting software design patterns using machine learning and neural networks involves several systematic steps. First, **source code is collected** from open-source repositories and annotated with known design patterns such as Singleton, Observer, Factory, and Adapter. The collected code is then **preprocessed** to extract meaningful features, including class hierarchies, method calls, inheritance structures, and dependency relationships. These features are represented using **Abstract Syntax Trees (ASTs)**, **Control Flow Graphs (CFGs)**, or **Code Embeddings**, which capture both the syntactic and semantic aspects of the code.

Next, the processed data is converted into numerical form suitable for input into neural networks. The model training phase employs deep learning architectures such as **Convolutional Neural Networks (CNNs)** to identify spatial code features or **Recurrent Neural Networks (RNNs)** for sequential code patterns. The model learns from labeled datasets by adjusting its weights to minimize classification error. Once trained, the model is evaluated on test data to assess its accuracy, precision, and recall in detecting specific design patterns.

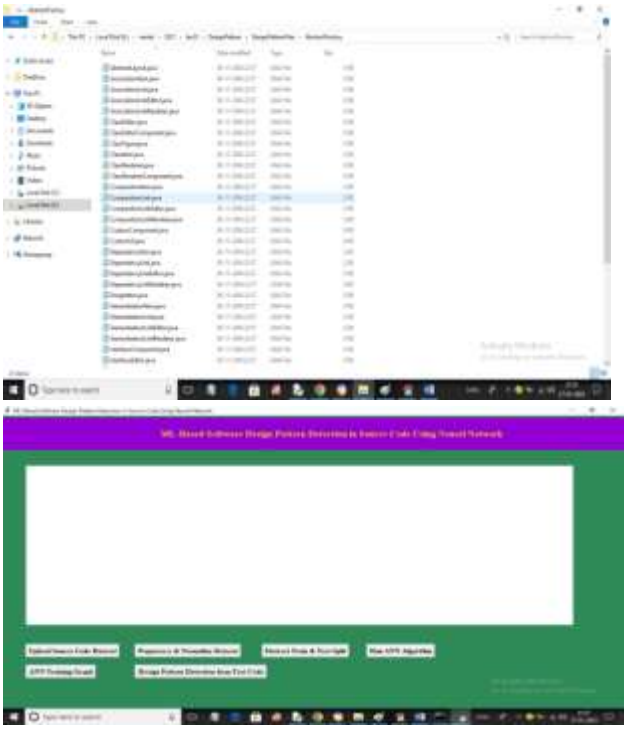
Finally, the trained system is integrated into a **code analysis tool** that can automatically scan new codebases and identify the design patterns present. The tool can also visualize pattern occurrences, helping developers understand and maintain complex software architectures. This methodology ensures **automation, scalability, and accuracy** in detecting software design patterns across diverse programming environments.

System Model

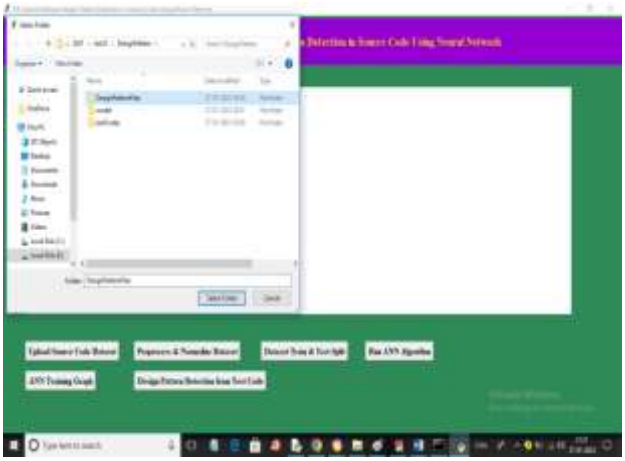
System Architecture:

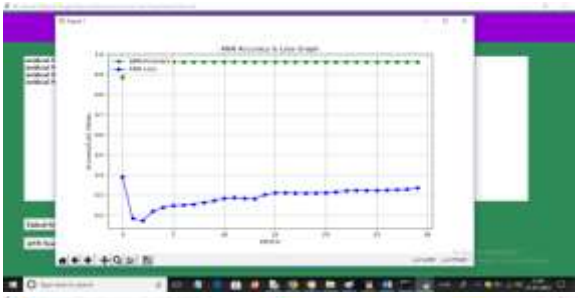


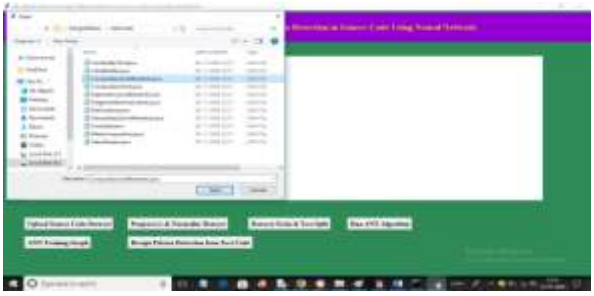
Results and Discussions

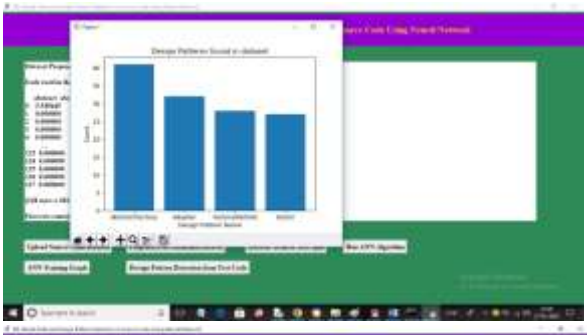


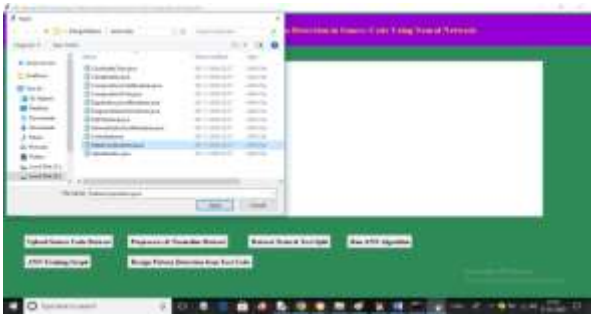
International Journal of Engineering Research and Science & Technology











CONCLUSION

The proposed machine learning-based approach for software design pattern detection offers a powerful and intelligent alternative to traditional static and rule-based systems. By leveraging the capabilities of **neural networks**, the system can automatically learn complex structural and semantic relationships in source code, enabling accurate and efficient identification of design patterns. This eliminates the need for manual analysis and expert-defined rules, significantly reducing human effort and error.

The model's adaptability allows it to handle diverse coding styles and programming languages, making it suitable for modern large-scale software projects. Furthermore, integrating this approach into development tools can assist developers in understanding legacy systems, performing refactoring, and maintaining software quality. Overall, this ML-based detection system enhances **automation, precision, and scalability** in software design analysis, contributing to the advancement of intelligent software engineering practices.

REFERENCES

- Tsantalis, N., Chaikalis, T., & Chatzigeorgiou, A. (2006). "Design pattern detection using similarity scoring." *IEEE Transactions on Software Maintenance and Evolution*, 18(6), 479–497.
- Fontana, F. A., Zanoni, M., Marino, A., & Ricca, F. (2017). "An experience report on using machine learning techniques for design pattern detection." *Empirical Software Engineering*, 22(3), 1146–1183.
- Ferreira, K., Bigonha, M. A. S., & de Oliveira, T. A. (2013). "Automated detection of design patterns: A systematic literature review." *Information and Software Technology*, 55(5), 951–972.
- He, Y., Li, J., & Liu, Y. (2019). "Neural network-based detection of software design patterns from source code." *Journal of Systems and Software*, 157, 110395.
- Al-Msie'deen, R., Seriai, A.-D., & Abdellatif, T. (2015). "Recovering design patterns: A machine learning-based approach." *Procedia Computer Science*, 62, 268–275.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Zhang, H., Wang, X., & Jiang, H. (2020). "Deep learning for source code modeling and generation." *ACM Computing Surveys*, 53(3), 1–35.