ISSN 2581 – 3781

Available Online at www.ajcse.info
Asian Journal of Computer Science Engineering 2026;11(1):1-5

RESEARCH ARTICLE

An Improvement of Software Vulnerability and Classification Model using Deep Neural Network Based on Big Data

Neha Khandelwal[1]*, Tarun Sharma[2]

[1]Department of Information Technology, Swami Vivekanand College of Engineering, Indore, Madhya Pradesh, India, [2]Department of Computer Science, Softvision College, Indore, Madhya Pradesh, India

Received on: 12-01-2026; Revised on: 15-02-2026; Accepted on: 20-03-2026

ABSTRACT

Software vulnerabilities increase the risk of security breaches, potentially causing significant harm to systems. Automatic classification methods are essential for managing software vulnerabilities and enhancing system security. This project proposes an improved model named the automatic vulnerability classification model (information gain based on term frequency-deep neural Network [IGTF-DNN]), which combines IGTF (term frequency-inverse document frequency [TF-IDF]) and DNN. The TF-IDF method is used to calculate the frequency and weight of words from vulnerability descriptions and information gain to select features. An optimal set of feature words is obtained. The DNN then constructs an automatic classifier to effectively classify vulnerabilities. The model is tested using the National Vulnerability Database of the United States and shows better performance compared to the K-nearest neighbors' model.

Key words: Big data, Deep neural network, Information gain, Software vulnerability, Term frequency

INTRODUCTION

The rapid development of information technology has brought both convenience and significant security risks to various industries. Software vulnerabilities, which are defects in software or hardware, can be exploited by unauthorized individuals, posing severe risks to information systems. In 2017, Windows system vulnerabilities led to ransomware attacks on 100,000 organizations globally. The increasing number and variety of vulnerabilities highlight the importance of effective analysis and management to enhance system security and reduce attack risks. Traditional vulnerability classification methods have limitations, necessitating the development of automatic classification models using machine learning techniques.

The main objectives of vulnerability classification are:

- Using term frequency-inverse document frequency (TF-IDF) to calculate the frequency and weight of each word from vulnerability descriptions
- Using information gain (IG) to select features to obtain an optimal set of feature words
- Using a neural network model to construct an automatic vulnerability classifier for effective classification.

Specific objectives include:

- Extracting vulnerability information from records using program codes written in R
- Collecting vulnerability data from 2015 to 2019 for training and testing
- Processing data from multiple Excel worksheets and storing them as data frame objects.

LITERATURE SURVEY

Early Vulnerability Classification Methods

The RISOS method[1] is one of the earliest vulnerability classification approaches, targeting operating system (OS) vulnerabilities. This method categorized OS vulnerabilities into seven groups from the attack perspective, but did not focus on how to exploit them.

Advanced Vulnerability Classification Methods

The program analysis vulnerability classification method[2] extended the focus to include application

Address for correspondence:
Neha Khandelwal
E-mail: neha19khandelwal88@gmail.com

vulnerabilities. Andy Gray's method[3] introduced a 10-category system based on different analysis needs. However, as the complexity of vulnerabilities increased, traditional manual classification methods have become less effective.

Machine Learning for Vulnerability Classification

Recent advancements in machine learning have shown promise in vulnerability classification. Shua *et al*.[4] applied the support vector machines classification method based on the latent Dirichlet allocation model, achieving good results in vulnerability grouping. Wijayasekara *et al*.[5] used the Naïve Bayes method for classifying textual information from error descriptions, illustrating its feasibility for this application. Gawron *et al*.[6] compared the Naïve Bayes algorithm with a simplified artificial neural network (ANN) algorithm, concluding that the ANN outperformed Naïve Bayes in vulnerability classification.

Limitations of Traditional Machine Learning Methods

Traditional machine learning algorithms often struggle with high-dimensional and sparse word vector spaces generated from vulnerability data. They also tend to overlook specific vulnerability information, resulting in lower classification accuracy. Recent studies have demonstrated the effectiveness of deep learning in various fields, including speech and image recognition, and natural language processing.[7-15]

SYSTEM ANALYSIS

Existing System

The existing system uses the TF-IDF method to measure the importance of terms in documents and the IG criterion to select features. However, it faces limitations in handling high-dimensional and sparse data, leading to reduced classification accuracy.

Drawbacks

- Fixed dataset files limit the scope of input
- Lack of a backpropagation technique limits accuracy, as hidden layer weights are not recalculated
- Limited vulnerability categories are selected for classification.

Proposed System

The proposed system incorporates all existing methodologies with enhancements, including the use of deep neural networks (DNN) with backpropagation to improve accuracy. The figure represents feature selection and attack detection of the Internet of Things [Figure 1].
The system processes vulnerability details from 2014 to 2019 and classifies them into 27 categories of vulnerabilities.

Advantages

- Ability to process more data set files with new vulnerability types
- Improved accuracy through the application of the backpropagation technique
- Increased number of vulnerability categories for classification.

Feasibility Study

The feasibility study assesses the economic, operational, and technical viability of the project.

Economic Feasibility

The system requires minimal hardware investment, making it cost-effective. The use of freely available technologies reduces overall costs.

Operational Feasibility

The system fits seamlessly into current organizational operations, solving existing problems efficiently.

Technical Feasibility

The system design ensures that technical requirements are met, enabling effective implementation and use of the TF-IDF and IG methods in R programming.

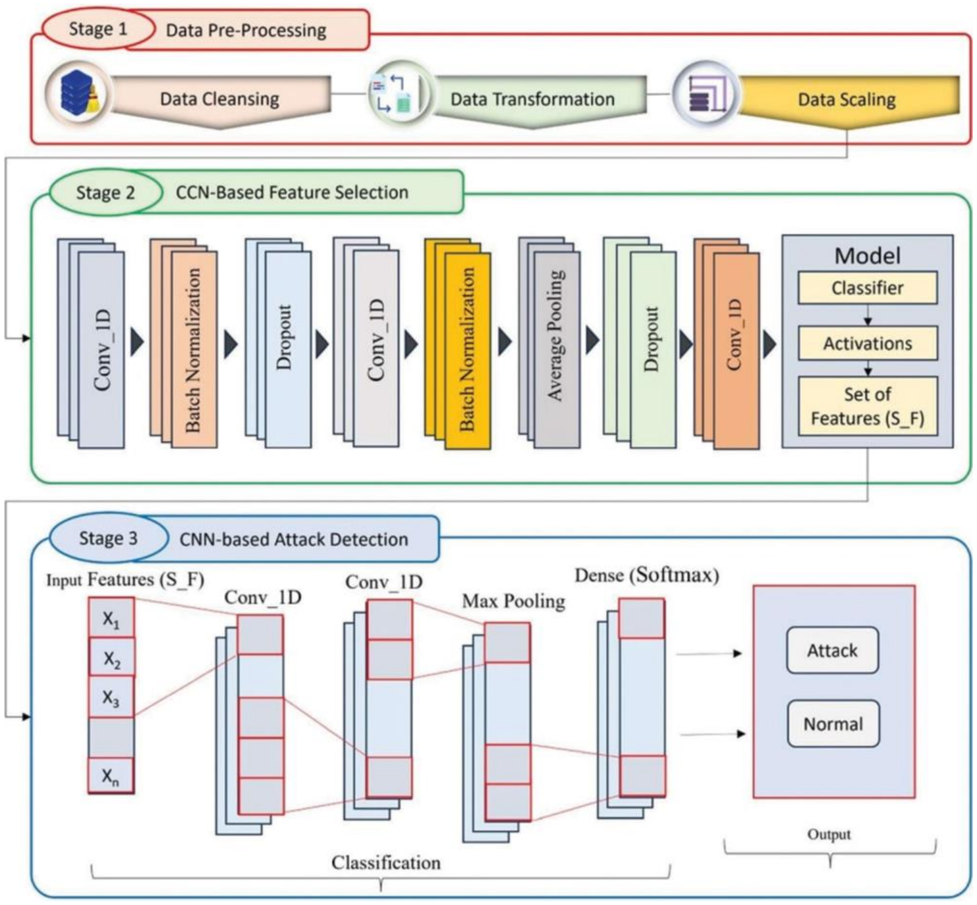

Figure 1: Dual convolutional neural network approach for feature selection and attack detection on Internet of Things networks

SYSTEM SPECIFICATION

Hardware Requirements

- Processor: Dual Core 2.1 GHz
- RAM: 2 GB
- Monitor: 17" Color
- Hard disk: 500 GB
- Keyboard: Standard 102 keys
- Mouse: Optical mouse.

Software Requirements

- OS: Windows 10 Pro
- Environment: R Studio 1.0
- Language: R 3.4.4

Software Description

Front end

The R programming language is used for statistical computing and graphics. It supports data manipulation, calculation, and graphical display, making it ideal for developing statistical software and data analysis applications.

SYSTEM IMPLEMENTATION

Modules

The system is divided into several modules, including TF-IDF, IG, feature word extraction, and optimizations using DNN.

TF-IDF

TF-IDF measures the importance of words in a document relative to a corpus, helping to identify key terms.

IG

IG selects features by measuring the importance of each feature in reducing uncertainty in class classification.

Feature Words Extraction

This module extracts feature words using TF-IDF and IG values to create a reduced, high-value feature set for classification.

Optimizations using DNN

The DNN module includes one input layer, multiple hidden layers, and one output layer. It optimizes feature selection and classification accuracy through forward and backpropagation processes.

Input Design

Input design converts user-originated inputs into a computer-understandable format. It involves capturing, preparing, and ensuring the accuracy of data.

Output Design

Output design focuses on generating useful information for end-users. The outputs are designed to be attractive, convenient, and informative.

System Testing

System testing includes unit testing, integration testing, functional testing, system testing, and acceptance testing to ensure the system functions correctly.

Unit Testing

Unit testing involves testing individual components of the system to ensure they function correctly.

Integration Testing

Integration testing evaluates interactions between combined components.

Functional Testing

Functional testing verifies that the system functions as specified.

System Testing

System testing assesses the entire integrated system to ensure it meets all requirements.

Acceptance Testing

User acceptance testing involves end-users testing the system to ensure it meets their needs.

CONCLUSION AND FUTURE WORK

Conclusion

This project successfully applies DNNs to software vulnerability classification, demonstrating the effectiveness of the IGTF-DNN model in improving classification performance. The model outperforms traditional methods, providing a basis for future research using the benchmark vulnerability dataset.

Future Work

Future research will explore additional features and improve the model's performance. Potential areas include enhancing the neural network architecture and incorporating more comprehensive datasets.

Appendix

Source code

The source code includes implementations of the sentiment analysis methods, data processing scripts, and the web application components.

Screenshots

Screenshots provide visual documentation of the system's user interface and key functionalities.

REFERENCES

1. Abbott RP, Chin JS, Donnelley J, Konigsford WL, Tokubo S, Webb DA. Security Analysis and Enhancements of Computer Operating Systems. Washington, DC: US Department of Commerce; 1976.
2. Bisbey IR, Hollingworth D. Protection Analysis: Final Report. United States: University of Southern California; 1978.
3. Gray A. An historical perspective of software vulnerability management. Inf Secur Tech Rep

2003;8:34-44.

4. Shua B, Li H, Li M, Zhang Q, Tang C. Automatic Classification for Vulnerability Based on Machine Learning. In: Proceedings IEEE International Conference Information Automation (ICIA); 2013. p. 312-8.
5. Wijayasekara D, Manic M, McQueen M. Vulnerability Identification and Classification Via Text Mining Bug Databases. In: Proceedings 40th Annual Conference IEEE Industrial Electronics Society; 2014. p. 3612-8.
6. Na S, Kim T, Kim H. A Study on the Classification of Common Vulnerabilities and Exposures Using Naïve Bayes. In: Proceedings International Conference Broadband Wireless Computing Communication Applications. Springer; 2016. p. 657-62.
7. Gawron M, Cheng F, Meinel C. Automatic Vulnerability Classification Using Machine Learning. In: Proceedings International Conference Risks Security Internet Systems. Springer; 2017. p. 3-17.
8. Deng J, Dong W, Socher R, Li LJ, Li K, Li FF. ImageNet: A Large-Scale Hierarchical Image Database. In: Proceedings IEEE Conference Computer Vision Pattern Recognition; 2009. p. 248-55.
9. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, *et al*. ImageNet large scale visual recognition challenge. Int J Comput Vis 2015;115:211-52.
10. Xiong W, Droppo J, Huang X, Seide F, Seltzer ML, Stolcke A, *et al*. Toward human parity in conversational speech recognition. In: IEEE/ACM Transactions Audio Speech Language Processing. Vol. 25. New York: IEEE; 2017. p. 2410-23.
11. Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Proceedings Advances Neural Information Processing Systems; 2012. p. 1097-1105.
12. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, *et al*. Mastering the game of Go with deep neural networks and tree search. Nature 2016;529:484-9.
13. Iyyer M, Manjunatha V, Boyd-Graber J, Daumé H. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In: Proceedings 53rd Annual Meeting Association Computational Linguistics; 2015. p. 1681-91.
14. Jo H, *et al*. Large-scale text classification with deep neural networks. Comput Cognit 2016;23:322-7.
15. Aziguli W, Zhang Y, Xie Y, Zhang D, Luo X, Li C, *et al*. A robust text classifier based on denoising deep neural network in the analysis of big data. Sci Program 2017;2017:3610378.