An *et al. J Wireless Com Network* (2023) 2023:41
https://doi.org/10.1186/s13638-023-02255-2

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

A CNN-based automatic vulnerability detection

Jung Hyun An[1], Zhan Wang[1] and Inwhee Joe[1*]

*Correspondence:
iwjoe@hanyang.ac.kr

[1] Department of Computer Engineering, Hanyang University, Seoul, Korea

Abstract

With the advent of the Internet, the activities of individuals and businesses have expanded into the online realm. As a result, vulnerabilities that result in actual breaches can lead to data loss and program failure. The number of breaches is increasing every year, as is the number of vulnerabilities. To address this problem, current research focuses on the detection of vulnerabilities using static analysis techniques. To prevent the propagation of vulnerabilities, a new paradigm is needed to quickly detect vulnerabilities, analyze them, and take actions such as blocking or removing them. Recently, artificial intelligence algorithms such as deep learning have been introduced for vulnerability detection. In this paper, we propose a vulnerability detection model, V-CNN, which aims to detect CWE/CVE (Common Weakness Enumeration/Common Vulnerabilities and Exposures) using CNN (convolutional neural network). We trained CWE for deep learning and redefined vulnerabilities based on CWE. We propose an experimental algorithm to improve vulnerability detection. The accuracy of the proposed V-CNN model is 98%, which exceeds the 95% of the random forest model. Therefore, our V-CNN has excellent correctness detection performance in the field of vulnerability detection. The V-CNN vulnerability detection algorithm can be used instead of static analysis to detect various security vulnerabilities.

Keywords: Convolutional neural networks, Vulnerabilities, Security, Deep learning, CVE (common vulnerabilities and exposures), CWE (common weakness enumeration)

1 Introduction

With the growth of the Internet, the activities of individuals and businesses have expanded into the online realm. As more individuals and businesses become active, hardware and software become more complex and diverse. Flaws or errors in this hardware and software can develop into security vulnerabilities and lead to data breaches [1]. As a result, not only do internal data breaches and system failures occur as a result of increased security vulnerabilities, but companies and individuals can also suffer significant losses.

Security breaches can occur in many ways, including hardware, operating systems and software [2]. Therefore, the main causes of security breaches are software development flaws, unauthenticated user input, software coding using insecure operating systems, ports or protocols [3], and unknown code.

In this paper, we present algorithms for detecting security vulnerabilities using CNNs (Convolutional Neural Networks), such as CVE (Common Vulnerabilities and Exposures) [4] and CWE (Common Weakness Enumeration) [5]. This is an experimental algorithm that can redefine CNN-based vulnerabilities, and CWE can be used for deep learning correctness detection. Thus, experiments can achieve high accuracy of the algorithm. Higher accuracy is achieved through experimental evaluation, such as optimizing the loss function and also adjusting the weight values. We show that the model outperforms the random forest algorithm [6]. In this paper we call it V-CNN (Vulnerable Convolutional Network). The dataset used for the experiments consists of 10,000 CVE and 5,000 CWE data. The preprocessing of the data is divided into a refinement step and an encoding step. Once the entire preprocessing is completed, a dataset is generated that can be used for the model. In the training step, the preprocessed dataset (70% for training and 30% for testing) is trained by the V-CNN model. To improve the recognition accuracy, the optimal value can be found by adjusting the weight values.

2 Related works

Breiman Leo proposed the Random Forest algorithm [6]. Random Forest is an algorithm that integrates multiple trees through the idea of integrated learning. Its basic unit is a decision tree [7], and each decision tree is a classifier, and for an input sample, N trees will have N classification results. The random forest integrates all the classification voting results and designates the category with the most votes as the final output. The random forest is designed to solve the weak generalization ability of decision trees, and the training can be highly parallelized, which has advantages for the training speed of large samples in the era of big data. Since the nodes of the decision tree can be selected randomly to divide the features, the model can still be trained efficiently when the dimensionality of the sample features is high, and the importance of each feature for the output can be derived after training. Compared to Adaboost's boosting series, RF is relatively easy to implement. However, in some noisy data sets, RF models tend to overfit. And the features that take more value divisions tend to have more influence on the decision of RF, thus affecting the effect of the fitted model.

Yoav Freund and Robert Schapire proposed Adaptive Boosting (AdaBoost) [8] in 1997. AdaBoost assigns weights to each data sample with weights following a probability distribution and initial weights following a uniform distribution, trains M models serially, determines the current model in each training round based on the error rate of the model, and updates the weights of the training samples. The weights of the final model are determined based on the error rate of the models trained in each round, and the weights of the training samples are updated, increasing the weights of misclassified samples and decreasing the weights of correctly classified samples. It is increased by the weights of samples misclassified by the previous basic classifier and decreased by the weights of correctly classified samples and used again to train the next basic classifier. At the same time, a new weak classifier is added in each round of iterations until a predefined small enough error rate or a predefined maximum number of iterations is reached before the final strong classifier is determined. In Adaboost, weak learners can be constructed using different regression classification models, which is very flexible. Compared to the Random Forest algorithm, AdaBoost fully considers the weights of each

classifier. However, it is more sensitive to anomalous samples, which may receive higher weights in the iterations and affect the prediction accuracy of the final strong learner.

In 1995, static analysis was proposed by Wichmann et al [9]. Static analysis is the analysis of a program without actually executing the program to determine whether it satisfies required properties, such as the presence of memory leaks. By analyzing lexical, syntactic, and semantic features, dataflow analysis, and model checking, static analysis can detect hidden bugs. The advantage of static analysis is that detection is fast, and analysts can use static analysis tools to quickly examine target code and perform operations in a timely manner. In practice, however, static analysis has a high failure rate. Due to the lack of an easy-to-use vulnerability detection model, static analysis tools are prone to a large number of false positives. Russell, Rebecca et al. used three open source static analysis tools (clamping, flawfinder, and cppcheck) to generate tags [10]. Each static analyzer looks for vulnerabilities from a different perspective. For example, clang mainly checks syntax and flawfinder focuses on CWE, so three static analyzers are integrated and their output is filtered to exclude those that are not vulnerable to attack in order to generate tags.

3 Proposed method

3.1 Framework of V-CNN automatic detection model

V-CNN is an automatic vulnerability detection model based on CNN. V-CNN uses one dataset (70% training, 30% testing) to learn and verify vulnerabilities. The model can be optimized by loss function, relational analysis of optimizers, weight adjustment and dataset scaling.

The framework of V-CNN is shown in Fig. 1. There are three steps in total. In the first step, the CVE/CWE source data provided by MITRE is collected and refined to verify the validity of the dataset. The source data uses the CVE/CWE officially registered with

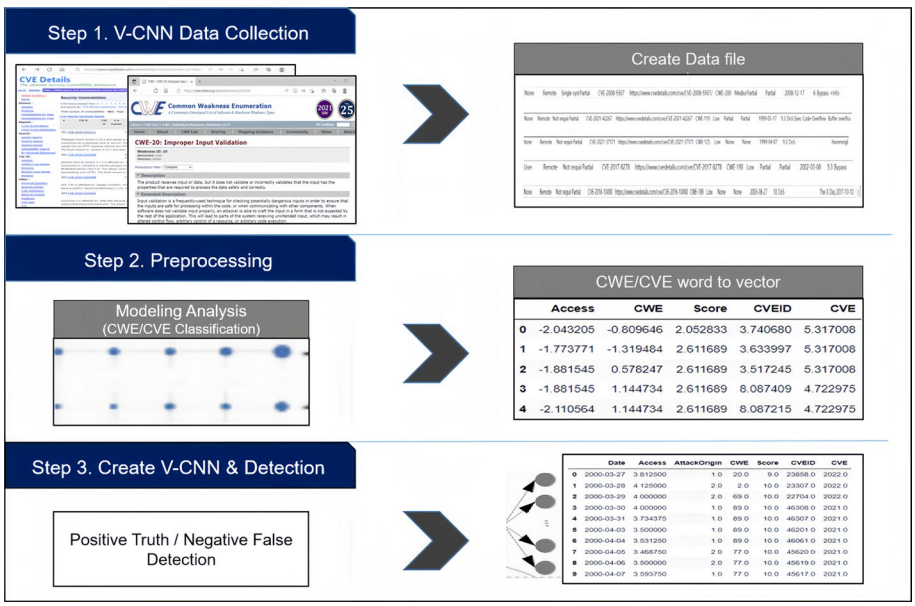


Fig. 1 Framework of the V-CNN model

MITRE, a file is created with the basic and necessary information provided, and the final dataset is created after the refinement process, such as processing and coding outliers, duplicates, and missing values. In the second step, the CVE/CWE of the modeling analysis results in text format is converted to vector format and normalized and optimized to improve the model performance. In the third step, the correct and false alarm rates of the proposed model are determined.

3.2 Definition of V-CNN-based automatic vulnerability detection method

V-CNN is a technique to learn CWE/CVE dataset by CNN algorithm and automatically detect CVE/CWE models to improve the accuracy rate.

The learning model flow of V-CNN is shown in Fig. 2.

- Preprocessing procedure: Generate the training dataset. The final dataset is created by processing the source data, handling outliers, duplicates, and missing values, and optimizing the coding.
- Learning model: The dataset is vectorized and normalized to improve the learning performance of the model. In addition, learning models are created and optimized by adjusting weights, dataset proportions, loss functions, and analyzing optimizer relationships.
- Correct detection rate and false positive rate: This is measured by automatically detecting CVE/CWE after the V-CNN model has completed learning. The performance of the algorithm is validated by comparison with the Random Forest algorithm.

3.2.1 Data collection and preprocessing

In this paper, we obtain high quality datasets by segmenting the preprocessing process. By segmenting the preprocessing process, the preprocessed dataset can be executed exactly step by step, which can improve the learning of the model.

The source dataset used in the V-CNN model uses CVEs/CWEs provided by MITRE vetting. For CVEs, items with CVSS (Common Vulnerability Scoring System) scores of 1–10 (out of 10, highest risk) are selected from the CVEs registered with MITRE from 2000 to 2021.

The process of refining the source data includes handling outliers, duplicate values, missing values, etc. Valid data were extracted for certain columns and missing data was

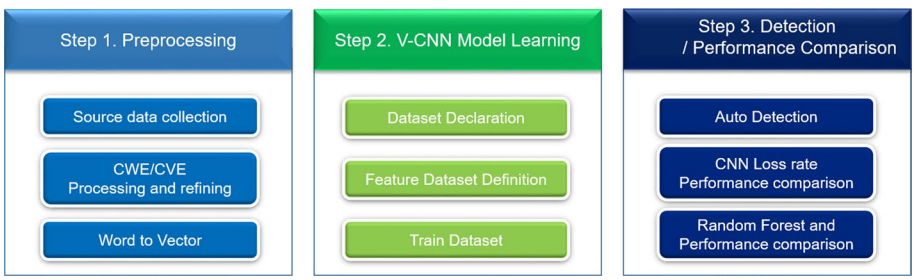


Fig. 2 Learning flow of V-CNN model

changed to the required values. In addition, a more accurate dataset is created that can be improved by eliminating unnecessary duplicate values. The resulting dataset is vectorized by two encoding steps: conversion of strings to integer data and conversion to real data actually learned.

Finally, a normalization (feature scaling) process is performed to improve the learning of the model. Figure 3 shows the steps to convert to a form that can be used for learning in the V-CNN model. This is the process of converting the source data CVE/CWE in text format into a real type of digital data. In this paper, the *Z*-score method is used for normalization.

The purpose of normalization is to reflect the same degree of scaling by changing the common scale without distorting the differences in the range of values in the dataset.

The data set for the V-CNN model is used by digitizing the source data in text form. As a result, the features of each column are in a very different range. Therefore, when performing linear regression analysis, even if the features are not significant, their inherently large values can have a large impact on the results. Therefore, the normalization process scales each data point at a different rate so that the features are of equal

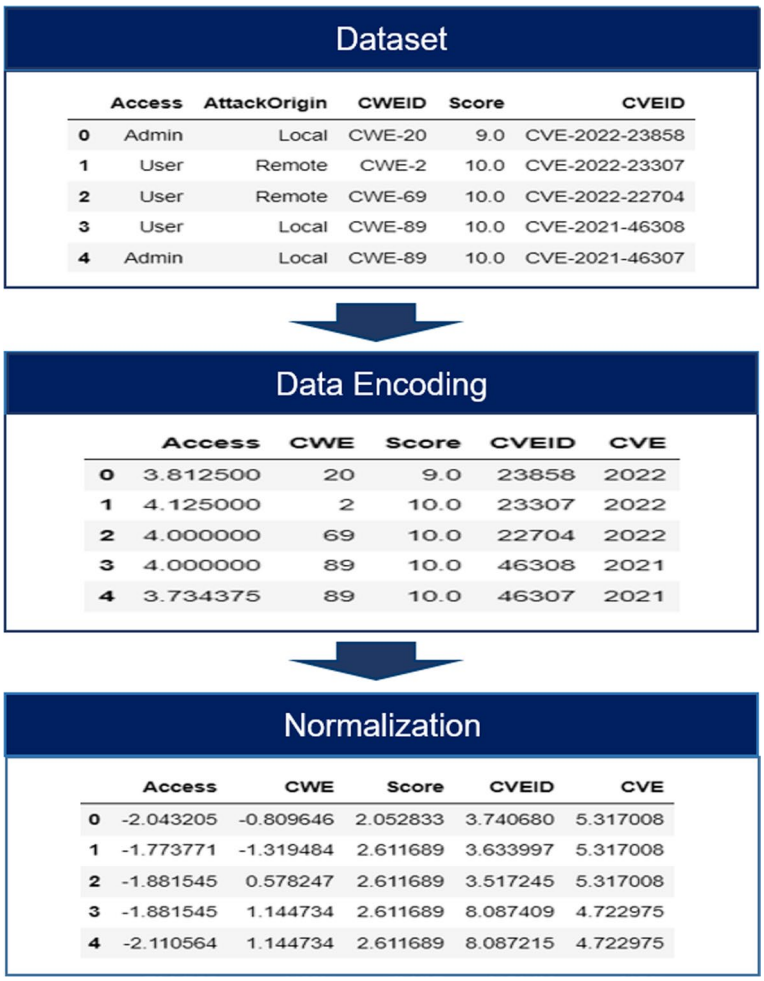


Fig. 3 Word-to-vector process

importance. Normalized data sets can improve the learning performance of V-CNN models.

There are two normalization methods: min-max scaling and Z-score normalization. Z-score normalization is a normalization strategy that can avoid the outlier problem. When Z-score normalization is performed by changing the value of X to Z-score, the features are rescaled to have the characteristics of a standard normal distribution.

3.2.2 V-CNN model learning

The V-CNN model transforms the obtained text-based dataset by refining the source data to integers and uses the final transformed dataset (by real number transformation). Data scaling of the model is performed using the Z-score method before training. Due to the nature of the dataset, the features of the real number type are in very different ranges, so insignificant features may have the greatest impact during the learning process. Therefore, the accuracy is improved by reflecting the importance of feature similarity and improving the learning performance.

The layers of V-CNN consist of a convolutional layer, a pooling layer, and a dense layer. We use 32, 64, and 128 filters of size (3, 3) in the convolutional layer of Convd 2D. Conv2D and MaxPooling2D are repeated several times to gradually abstract the high-dimensional features. Finally, the output is one-dimensional, and features are extracted from the dense layers by convolution. In this way, the features are learned based on the actual data, and softmax classifies the categories based on the labels. To prevent overfitting, we add dropout after smoothing.

The V-CNN model is designed with an active ReLU function and layer 3. In addition, the model performance is optimized by analyzing the relationship between the optimizer and the loss function. Optimizers such as Adam [11], Adagrad [12], and SGD [13] are combined with loss functions such as MSE and MAE, and optimizers such as Adam, Adagrad, and SGD are used with loss functions. In this paper, from different optimizers and loss functions, three optimizers and two loss functions are selected which are effective for the proposed model. And the selected three optimizers and two loss functions are combined as follows for experiments to find the combination that minimizes the loss rate. Combination Case of Optimizer and Loss Function:

Case (1) Adagrad + MSE
Case (2) Adagrad + MAE
Case (3) SGD + MSE
Case (4) SGD + MAE
Case (5) Adam + MSE
Case (6) Adam + MAE

The ratios of the training and test data sets are 6:4, 7:3, and 8:2, respectively, and the highest accuracy of 7:3 is chosen.

Adam, Adagrad, and SGD as optimizers are commonly used to optimize MSE, MAE loss functions. We studied the minimization loss in combination with different optimizers and loss functions and applied them to the proposed model. To improve the

Table 1 Summary of CWE category for experiments

Explanation	CWE list	Detected code number
Cross-site Scripts(XSS)	79	152
Bypass	20, 200, 254, 255, 264, 286, 287, 352	60
CFSR	352	14
Dir. Trav.	22	23
Dos, Dos Overflow, Dos Exec	16, 17, 18, 19, 20, 59, 119, 125, 189, 264, 310, 362, 399, 415, 476	387
ExecCode	17, 19, 20, 22, 77, 78, 79, 89, 94, 119, 264	101
Http R.Sql.	94, 352	3
Overflow	119, 189, 264	20
Sql	89	1

Table 2 CWE redefinition

ID	Explanation	CWE Redefinition
CWE-20	Improper Input Validation	Remote address spoofing vulnerability
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	Memory reference error in Complete Enumeration Parallel Program
CWE-125	Out-of-bounds Read	Input error
CWE-190	Integer Overflow or Wraparound	Integer overflow
CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	Local path vulnerability
CWE-399	Resource Management Errors	Resource allocation processing error
CWE-416	Use After Free	Orphan fragmentation error

efficiency of the model, we conducted experiments with a mixture of three types of optimizers and two types of loss functions.

3.3 CWE code redefinition and detection based on vulnerability

3.3.1 CWE code redefinition

The CWE categories used for classification in the V-CNN model are shown in Table 1. The classification results in the dataset are set to nine categories, as summarized in Table 1 CWE categories for experiments, and the list of CWE and the number of detected documents are summarized. We classify the main types of vulnerabilities detected by the proposed V-CNN model into nine categories. Then, the corresponding CWEs are listed according to the vulnerability types, and the number of detected codes for CWEs are summarized. For example, the CWE list corresponding to XSS is numbered 79, which means that it was found in 152 documents. 152 documents means that XSS was detected in 152 codes, corresponding to a CWE list number of 79. We classify the major types of vulnerabilities detected in the proposed V-CNN model into nine categories. Then, the corresponding CWEs are listed according to the vulnerability types and the number of detected codes for CWEs is collected.

When a CWE is registered after a MITRE review, unique codes and data structures are created, as shown in Table 2. When a newly discovered CWE was reviewed, only the new CWE was coded with reference to the data structure of the previously registered

CWE. For this purpose, the data structure format of the CWE file was redefined with reference to the actually detected source code, as shown in Table 2. As a result of the experiment, the first ten CWEs were selected and redefined in the order of the most detected CWEs.

3.3.2 Automatic detection and results

A model with high accuracy and low packet loss rate is selected by V-CNN learning to detect CVE/CWE codes. The correct and false alarm rates of CVE/CWE are measured, and accordingly, most of the CWEs are detected by the CommitCount function. We can measure both positive and negative detection. Most CWEs are detected by the V-CNN model. In addition, we can check the detection results for each software.

The preprocessing process is segmented to obtain a high-quality data set and improve the accuracy of vulnerability detection. In addition, the combination with minimum loss rate is found and applied to model optimization by weight adjustment, optimal ratio of learning dataset to test dataset, optimizer and loss function relationship analysis. Correctness is improved by the CommitCount function, which samples CWE and CVE code detection in the corresponding dataset.

V-CNN evaluates the performance of the model by measuring the correct rate and the loss rate, and the superiority of the proposed model is demonstrated by comparing the performance with that of the random forest model.

Algorithm 1 V-CNN

Input: The CWE code set reorganized by the domain name of the security domain
Output: The presence or absence of the vulnerability

Data preprocessing: handling outliers, duplicates, and missing values, and optimizing coding
Vectorization: converting strings to integer data and converting to real data for actual learning
Normalization: feature scaling - Z-score
ConV2D ← (3, 3) of 32, 64 and 128 filters
Maxpooling
ReLU + Dropout
MAE + Adam optimizer
Buffer size ← 10
malloc() ← Dynamically allocate memory to ensure there is enough space to hold the input string
while buffer overflow **do**
    strncpy() ← Copies input characters into a buffer and limits the number of copied characters to the size of the buffer minus the value of 1, so that there is always room for null terminators at the end
    Free buffer ← Manually add null to the buffer and free allocated memory to prevent memory leaks
end while
return 0

4 Results and discussion

4.1 Data source

The data source for the V-CNN model is MITRE's CVE/CWE, which has been regularly updated since 1999. Among the CVEs registered and published by MITRE from 2000 to 2021, those with CVSS scores of 1–10 (10 being the highest risk) were selected.

Table 3 shows some of the contents of the sampled dataset. The dataset is a file with a CSV extension and consists of columns representing data characteristics and actual values. Table 3 shows these columns and their corresponding values by selecting six significant columns from the entire contents of the dataset. They are CVE-ID, CVE page (CVE

Table 3 Data source for V-CNN learning

CVE-ID	CVE Page	CWE-ID	Complexity	Confidentiality	Score
CVE-2022-23307	https://www.cvedetails.com/cve/CVE-2022-23307	CWE-502	Low	None	10
CVE-2022-23227	https://www.cvedetails.com/cve/CVE-2022-23227	CWE-306	Low	None	10
CVE-2022-23221	https://www.cvedetails.com/cve/CVE-2022-23221	CWE-94	Low	None	10
CVE-2022-23178	https://www.cvedetails.com/cve/CVE-2022-23178	CWE-287	Low	Partial	10
CVE-2022-23118	https://www.cvedetails.com/cve/CVE-2022-23118	CWE-269	Low	Complete	9
CVE-2022-23009	https://www.cvedetails.com/cve/CVE-2022-23009	CWE-863	Low	Complete	9
CVE-2022-22704	https://www.cvedetails.com/cve/CVE-2022-22704	CWE-269	High	None	10

registration link site), CWE-ID, Complexity (source code complexity), Confidentiality (impact), and Score.

CVE-ID indicates a vulnerability that has been registered with MITRE. cve-2022-23307 is a 23307 vulnerability registered in 2022. cve-page is the address of the website where the vulnerability actually exists. For example, if you select the second column, CVE page (https://www.cvedetails.com/cve/CVE-2022-23307), you can view information about CVE-2022-23307. The third column, CWE-ID, is the vulnerability's registration with MITRE. Complexity is indicated by a code complexity rating of high, medium, or low. Confidentiality is the scope of the affected code. The score is a CVSS score from 1 to 10, depending on the level of risk.

4.2 V-CNN automatic detection results

To improve the accuracy of vulnerability detection, the detection rate is improved by optimizing the data pre-processing and the sampling of the CommitCount function, the model has been optimized by the variety approach has been applied to the design of layers, hyperparameter tuning, optimizers, and loss functions. The performance is evaluated by accuracy and loss rate to measure the strength of correctness detection performance. Figure 4 is a plot showing the CWE detection results sorted by their number. CWE-119, a memory-related vulnerability (improperly restricting operations within memory buffer boundaries), is the most common CWE detected by the V-CNN model. It is followed by CWE-20 and CWE-125.

The loss rate trend of V-CNN model optimization is analyzed based on the relationship between the optimizer and the loss function. Then, the combination that minimizes the loss rate is found and applied to the model. The loss function quantifies the difference between the actual value and the estimated value. And when the error is small, the value of the loss function becomes smaller. The loss function uses MAE (mean absolute error) and MSE (mean squared error), which are commonly used in regression analysis. And the optimizer supports deciding the network update method based on the loss function, so Adagrad, SGD, and Adam were selected and executed.

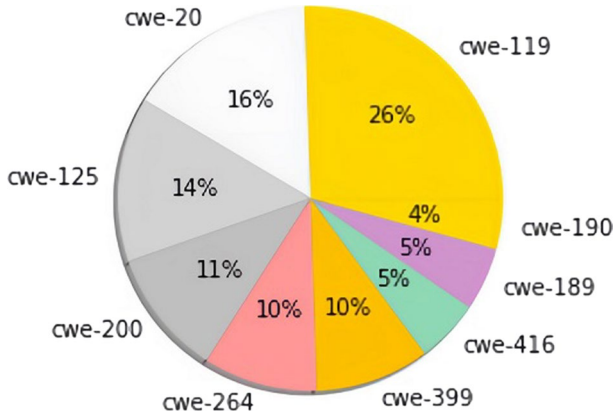
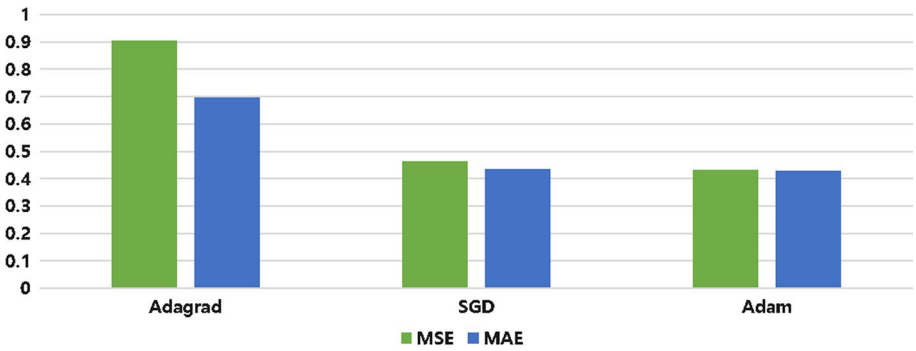


Fig. 4 Result of CWE detection



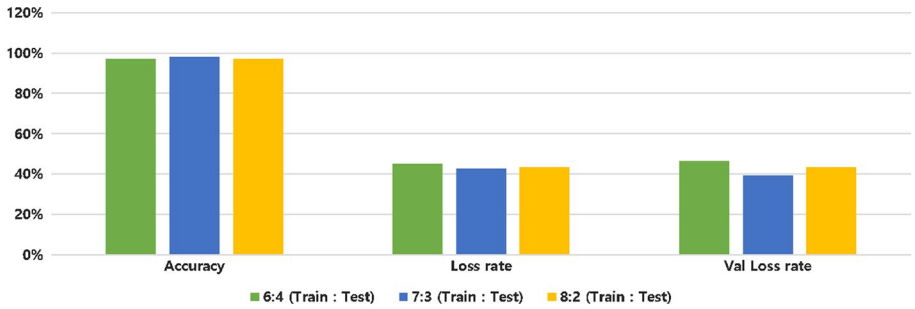
Optimizer	Adagrad	SGD	Adam
MSE	0.9060	0.4625	0.4337
MAE	0.6960	0.4364	0.4275

Fig. 5 Loss rate between optimizer and loss function according to the relationship

Figure 5 shows the relationship between the optimizer and the loss function, and the variation in the loss rate. The experimental results show that the loss rate of SGD and Adam is about half of that of Adagrad optimizer for V-CNN. There is a small difference between SGD and Adam. In MAE, the measured loss rate is slightly smaller than that in MSE. Based on the experimental results, Adam and MAE were applied to the model.

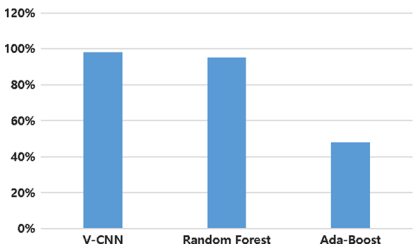
The data set is divided into training data and test data. And depending on the ratio of training and test data, significant differences in learning effects occur. According to the results of the experimental models with the ratios of training and test data of 6:4, 7:3 and 8:2, respectively, the accuracy of 7:3 in Fig. 6 is 98%, which is the highest accuracy rate.

To demonstrate the superior performance of the proposed model, experiments were performed by the random forest algorithm and compared with the performance of the proposed model. Figure 7 shows the accuracy measurements of the algorithm, the accuracy of the V-CNN model is 98%, which is better than the accuracy of the random forest model which is 95%. Therefore, V-CNN has good accuracy detection performance in the field of vulnerability detection.



	6:4 (Train : Test)	7:3 (Train : Test)	8:2 (Train : Test)
Accuracy	97%	98%	97%
Loss rate	0.4511	0.4275	0.4335
Val Loss rate	0.4656	0.3942	0.4350

Fig. 6 Performance comparison by dataset ratio



	V-CNN	Random Forest	AdaBoost
Accuracy	98%	95%	48%

Fig. 7 Performance comparison of V-CNN, random forest and AdaBoost

4.3 Performance evaluation and discussion

In the dataset presented in this paper, CVE/CWE data in text format are refined, quantified, and normalized to real number format, and the performance of the dataset is improved by model optimization. CVE/CWE are detected by V-CNN models, and the accuracy and loss rate of the results are evaluated. By comparing and analyzing the accuracy of V-CNN and random forest models, the accuracy of the proposed model is 98%. Thus, the model outperforms the random forest by 95%. The CWE code set used as the data set was reorganized by the domain name of the security domain, and the vulnerable part was detected by Vulnerabilities-CNN. In the source code, the CWE-119 (memory-related security vulnerabilities) detection classifies security vulnerabilities by domain name and proves that the predictions can be detected. Thus, its excellent performance is demonstrated by lossy optimization compared to the Harmony Model classification family of Random Forests. Thus, the part of reorganization and prediction of specific groups of domains where vulnerabilities may occur, such as CVE and CWE, is the main contribution of this paper.

Static tools are not automatically identified like artificial intelligence tools. In this paper, CNN algorithms are used instead of white-box testing techniques such as static tools, and achieve relatively high accuracy in vulnerability detection. In addition, its performance is also excellent compared to the random forest algorithm. The learning accuracy can be improved when vulnerabilities are detected by deep learning instead

of using static tools. Thus, the proposed learning model can be used instead of various static tools.

5 Conclusion

CNN as a deep learning algorithm is mainly used for image learning. However, text type codes are preprocessed by applying CNN algorithms, which are used for learning in this paper, and V-CNN models are proposed to detect vulnerabilities. Therefore, the preprocessed data of the source data can be redefined as a CWE suitable for vulnerabilities.

The accuracy of vulnerability detection is improved by optimizing the CommitCount function for data preprocessing and sampling, thus increasing the detection rate. We focus on data preprocessing and model optimization to improve the accuracy of vulnerability detection. Data selection, redundancy removal, and data size are also adjusted in preprocessing to obtain concise data.

In addition, the performance of the model is improved by analyzing the relationship between the optimizer and the loss function to minimize the loss rate and the ratio of the optimized data set. The performance of the model is evaluated using linear modeling and CNN performance measurement variables, and the accuracy of V-CNN is 98%, which proves the superiority of the proposed V-CNN model by performance comparison, while the accuracy of random forest is 95%.

The correctness detection by deep learning is more accurate than the vulnerability detection by static tools. Therefore, the learning method can be widely used as a service model for various industries and security vulnerabilities. In addition, V-CNN can be used to replace traditional static analysis tools.

Through future research, the performance of the model can be improved by increasing the correct detection rate of the proposed model. In addition, performance improvement studies can be conducted using source code, i.e., source data developed in the field for CVE/CWE detection.

Abbreviations

CVE	Common vulnerabilities and exposures
CWE	Common weakness enumeration
CNN	Convolutional neural networks
V-CNN	Vulnerable convolutional network
TMP	Temporal message propagation network
SCVDIE	Information graph and ensemble learning

Acknowledgements
The authors acknowledged the editors and reviewers for their valuable comments and suggestions.

Author contributions
All authors have contributed equally. All authors read and approved the final manuscript.

Funding
This work was supported by the Institute for Information and Communications Technology Promotion (IITP) Grant funded by the Korea government (MSIP) (Development of the technology to automate the recommendations for big data analytic models that define data characteristics and problems), under Grant 2020-0-00107.

Availability of data and materials
The data used to support the findings of this study are available from the corresponding author upon request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 8 September 2022 Accepted: 18 May 2023
Published online: 24 May 2023

References

1. M. Kalash et al., Malware classification with deep convolutional neural networks, in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (IEEE, 2018)
2. ISO-ISO/IEC 13335-1:2004-Information technology-Security techniques Management of information and communications technology security-Part 1: Concepts and models for information and communications technology security management
3. H. Jain et al., Weapon detection using artificial intelligence and deep learning for security applications, in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)* (IEEE, 2020)
4. S. Christey, R.A. Martin, Vulnerability type distributions in CVE, *Mitre report, May* (2007)
5. B. Martin et al., CWE, *SANS top* 25 (2011)
6. Leo Breiman, Random forests. Mach. Learn. **45**, 5–32 (2001)
7. Q.J. Ross, Induction of decision trees. Mach. Learn. **1**, 81–106 (1986)
8. Yoav Freund, Robert E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
9. B.A. Wichmann et al., Industrial perspective on static analysis. Softw. Eng. J. **10**(2), 69 (1995)
10. R. Russell et al., Automated vulnerability detection in source code using deep representation learning, in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (IEEE, 2018)
11. D.P. Kingma, B. Jimmy, Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. J. Duchi, H. Elad, S. Yoram, Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**(7), 2121 (2011)
13. H. Robbins, M. Sutton, A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)

Publisher's Note