FOREX Publication
Open Access | Rapid and quality publishing

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 262-268 | e-ISSN: 2347-470X

SimCoDe-NET: Similarity Detection in Binary Code using Deep Learning Network

S. Poornima[1] and R. Mahalakshmi[2]
[1]*Research Scholar, Department of Computer Science Engineering, Presidency University, Bangalore, Karnataka, India.*
[2]*Professor, Department of Computer Science Engineering, Presidency University, Bangalore, Karnataka, India.*

***Corresponding Author:** S. Poornima; Email: poornima.spa@gmail.com

ABSTRACT- Binary code similarity detection is a fundamental task in the field of computer binary security. However, code similarity is crucial today because of the prevalence of issues like plagiarism, code cloning, and recycling in software due to the ongoing increase of software scale. To resolve these issues, a novel similarity detection in binary Code using Deep learning NETwork (SimCoDe-NET) has been proposed. Initially, op-code features are extracted from the input data by using reverse engineering process and the opcode embedding is generated using N-skip gram method. The extracted features are fed into Bi-GRU neural network for classifying the similarity of the binary codes. The Bi-GRU neural network compares two data samples in feature space to identify whether they belong to similar data or non-similar data. The SimCoDe-NET framework is evaluated by using generated dataset to assess the efficiency of this method. The efficacy of the proposed SimCoDe-NET framework is assessed in terms of precision, accuracy, sensitivity, recall, similarity detection time and similarity detection rate. The accuracy of the proposed method is 99.10% which is relatively high compared to the existing method. The proposed SimCoDe-NET approach improves the accuracy by 84.9%, 88.58%, and 93.9% better than jTrans, UPPC, and HEBCS respectively.

Keywords: Binary code analysis, N-skip gram, Deep Learning, Bi-GRU Network, Internet of Things.

ARTICLE INFORMATION
Author(s): S. Poornima, R. Mahalakshmi;
Received: 31/10/23; **Accepted**: 15/01/24; **Published**: 28/03/24;
E- ISSN: 2347-470X;
Paper Id: IJEER231022;
Citation: 10.37391/IJEER.120136;
Webpage-link:
https://ijeer.forexjournal.co.in/archive/volume-12/ijeer-120136.html

Publisher's Note: FOREX Publication stays neutral with regard to jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

Similarity detection offers a wide range of applications in high-end property and cybersecurity. The number of noticeable flaws more than doubled in 2017, and open-source flaws are being discovered with increasing frequency [1]. The Internet of Things (IoT) has seen significant growth and adoption in recent years. IoT development is under pressure to get products to market quickly, yet this presents security and privacy risks [2]. An effective strategy for assuring the security of IoT devices is IoT firmware security analysis. Given the absence of source code, examining binary code has inevitably grown to be a crucial tool for firmware security analysis. These peer vulnerabilities in various firmware are rapidly found using Binary Code Similarity Discovery (BCSD) [3,4].

BCSD focuses on identifying additional peer-to-peer vulnerability functions where known vulnerability functionality exists. Malware detection [5, 6], patch analysis [7, 8], code plagiarism detection [9, 10], and patch analysis are additional security applications.

In this study, a method is offered for binary similarity detection that is both extremely accurate and faster than any existing method for binary similarity search, enabling real-world workloads to be scanned in a realistic amount of time [11,12]. A significant quantity of global information will be lost as a result of the sequential execution strategy [13]. To address these drawbacks, a novel Similarity detection in binary Code using Deep Learning Network (SimCoDe-NET) technique is proposed. The following is a summary of the research's primary contributions:

- The main goal of this research is to develop a novel Similarity detection in binary Code using Deep Learning Network (SimCoDe-NET) to improves the effectiveness of identifying the similarity of binary codes.
- Initially, a reverse engineering process extracts the features of the op-codes from the input data, and then an N-skip gram method is used to generate the opcode embedding.
- The gathered characteristics are fed into the Bi-GRU neural network, which classifies the degree of similarity between binary codes.
- The Bi-GRU neural network compares two data samples in feature space to identify whether they belong to similar data or non-similar data.

The following sections of this work are organized as follows: The literature review is covered in *section 2*, the workings of the proposed SimCoDe-NET are explained in *section 3*, the experimental findings and discussion are covered in *section 4*, and conclusions and future work are covered in *section 5*.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 262-268 | e-ISSN: 2347-470X

2. RELATED WORK

In order to enhance binary code similarity identification using the Bi-GRU neural network, researchers have recently developed a number of machine learning (ML) and deep learning (DL) techniques. We briefly go over a few strategies in this section.

In 2022 H. Wang, W. Qu, G. Katz, W. Zhu, Z. Gao, Qiu, J. Zhuge, and C. Zhang [14] suggested jTrans, a transformer-based method, is used to learn the representation of binary codes. The Binary Group dataset is used to assess the jTrans method, which performs 32.0% to 62.5% better than SOTA alternatives on this more complicated dataset in a single task. However, the performance of the jTrans framework is very low.

In 2022 W. Zhang, Z. Xu, Y. Xiao, and Y. Xue [15] suggested utilizing the capability of pseudocode for a comparison of binary codes. The UPPC framework is evaluated by using the vulnerabilities dataset and achieves an overall accuracy of 33.2% for binary code similarity detection. However, the de-compilation is only supported for a few architectures by the UPPC framework.

In 2022 S. X Jiang, X. Wang, Yu, and Y. Gong [16] suggested a cross-platform technique for binary function similarity identification using dual-layer positional encoding. Using the MISA dataset for evaluation, the DP-MIRROR technique outperformed the advanced method with an accuracy of approximately 35%. However, the accuracy of the DP-MIRROR technique is very low.

In 2023 X. Sun, Q. Wei, J. Du, and Y. Wang [17] suggested a HEBCS a very environment-friendly binary code seek method. The F1 score, accuracy, and precision of the HEBCS method's evaluation of the GNU device were each 0.943%, 0.938%, and 0.964%, respectively. However, the HEBCS method's precision is not always as accurate as anticipated. Strength and weakness of the proposed and Existing Methods shown in *table 1*.

Table 1. Strength and weakness of the proposed and Existing Methods

S.No.	Author	Proposed Method	Strength	Weakness
1.	H. Wang, W. Qu, G. Katz, W. Zhu, Z. Gao, Qiu, J. Zhuge, and C. Zhang	Transformer with jump awareness for binary code similarity (jtrans).	The jtrans approach routinely outperforms the state-of-the-art methods with considerable improvements on BCSD tasks.	The performance of the jTrans framework is very low.
2.	W. Zhang, Z. Xu, Y. Xiao, and Y. Xue	Analyze binary code similarities using pseudocode to realize its potential.	A deep pyramidal convolutional neural network (DPCNN) is used to learn the semantic properties of binary function sequences and decompiled pseudocode. Other binary similarity analysis activities, such code matching and vulnerability screening, are then performed on this network.	The de-compilation is only supported for a few architectures by the UPPC framework.
3.	S. XJiang, X. Wang, Yu, and Y. Gong	The two-layer positional encoding embedding approach (DP-MIRROR) is utilized for cross-platform binary function similarity recognition.	Using the cross-platform binary function similarity index, or DP-MIRROR, two-layer positional encoding is utilized in place of single-layer encoding to prevent OOV issues while integrating instructions.	The accuracy of the DP-MIRROR technique is very low.
4.	X. Sun, Q. Wei, J. Du, and Y. Wang	High Efficiency Binary Code Search (HEBC).	A range of characteristics, such as guiding features, syntactic features, and structural features, might take the place of the challenging semantic representation and extraction process which attains semantic extraction at the effective level.	The HEBCS method's precision is not always as accurate as anticipated.
5.	G. Liu, X. Zhou, J. Pang, F. Yue, W. Liu, and J. Wang	A GNN layer transformer model (Codeformer) is used to find similarities in binary codes.	Using a transformer, this method retrieves fundamental semantic characteristics from repeating blocks. A GNN is then used to update and aggregate the basic block features that were retrieved.	The Codeformer approach does not provide the integration of instructions as a dimension of integration.
6.	Wu, G. and Tang, H.,	Using multilayer aggregation to find vulnerabilities in binary code.	Using methods including feature concatenation, feature addition, and weighted feature combination, this study explores the effects of various features on the binary code vulnerability detection problem.	In the suggested technique the data processing phase needs to be enhanced.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 262-268 | e-ISSN: 2347-470X

7.	Proposed	Similarity detection in binary COde using DEep learning NETwork (SimCoDe-NET).	The op-code features are extracted from the input data by using reverse engineering process and the opcode embedding is generated using N-skip gram method. The Bi-GRU neural network compares two data samples in feature space to identify whether they belong to similar data or non-similar data.	To improve the process for creating a graph out of malware.

In 2023 G. Liu, X. Zhou, J. Pang, F. Yue, W. Liu, and J. Wang [18] suggested a nested transformer GNN variant for binary encoded similarity recognition. The suggested technique is evaluated by using OpenSSL, Clamav, and Curl datasets and achieves an overall accuracy of 93.38%. However, the Codeformer approach does not provide the integration of instructions as a dimension of integration.

In 2023 Wu, G. and Tang, H.,[19] suggested a version of multi-step characteristic merging forms the basis of binary coding vulnerability identification. The suggested approach achieved 87.7% F1 score for the NDSS18 dataset and 98.9% F1 score for the Juliet Test Suite dataset. However, in the suggested technique the data processing phase needs to be enhanced.

According to the literature study, many ML and DL algorithms have been applied in conjunction with the Bi-GRU neural network to find similarities in binary encoding. However, the existing system faces challenges like low accuracy rate, efficiency, and detection rate. To overcome these issues, a novel similarity detection in binary code using deep learning network (SimCoDe-NET) framework has been proposed. Ther detailed explanation of the proposed method is given in *section 3*.

3. PROPOSED SIMCODE-NET FRAMEWORK

In this section, the proposed novel similarity detection in binary COde using DEep learning NETwork (SimCoDe-NET) framework has been developed. *Figure 1* displays the actual schematic of the proposed SimCoDe-NET framework.

3.1. Opcode extraction using reverse engineering

The word embedding model is used to construct embedding instructions. By consistently upholding the linkages between words' syntactic and semantic components, it achieves this goal. *Figure 2* depicts the Opcode Extraction and vector conversion process. Word2vec and one-pass warm encoding are two integrated variants. The entry values are pre-processed and given a vector price depending on their indexing in warm encoding. Word2vec comes with two models which is the Continue Bag of Words (CBOW) version and the Skip Continue version.

3.2. Opcode embedding using n-skip gram

The N-skip gram word embedding approach is used to encode opcodes. A popular NLP version for Word2Vec is the bypass gamut version. *Figure 3* illustrates the combination of opcodes using the N-Skip-Gram process. The mathematical expression of the N-Skip-Gram model is given in *equation (1)*,

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{-t\le j\le i\ne 0}\log Pr\left(Code_{i+j}\middle|Code_i\right) \quad (1)$$

Where, $Code_1,\ldots, Code_m$ are the training texts and the volume of the training is denoted as 1 and N is the embedding vector. $Pr\left(Code_{i+j}\middle|Code_i\right)$ is the core probability. The mathematical expression of the prediction error is given in *equation (2)*,

$$Error = \frac{1}{N}\sum_{n=1}^{N}\sum_{-a\le j\le a}\log Pr\left(C_{t+j}\middle|C_t\right) \quad (2)$$

The variables N and a stand for the quantity of words and context size, and $Pr\left(C_{t+j}\middle|C_t\right)$ is the core probability of predicting the word. Using some of the design benefits of the Text-CNN approach, the model created here preserves the ordinal information of the opcode sequence. Different feature maps can be produced by combining various filters. This is how the formula is defined in *equation (3)*:

$$c_i = Relu(Conv(M, w_i) + b_i) \quad (3)$$

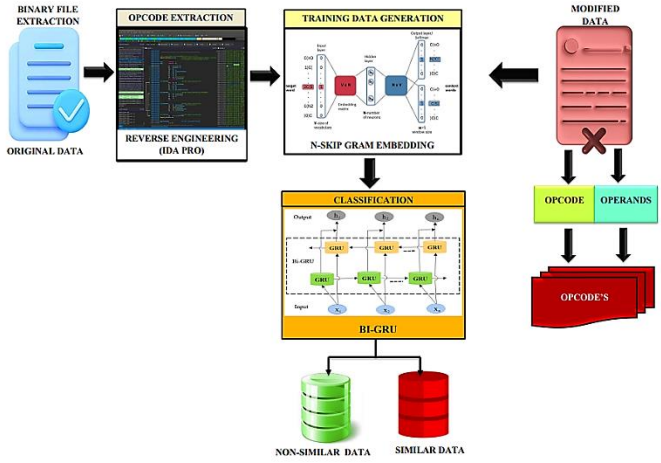


Figure 1: Overall Block Diagram of the proposed Simcode-NET Framework

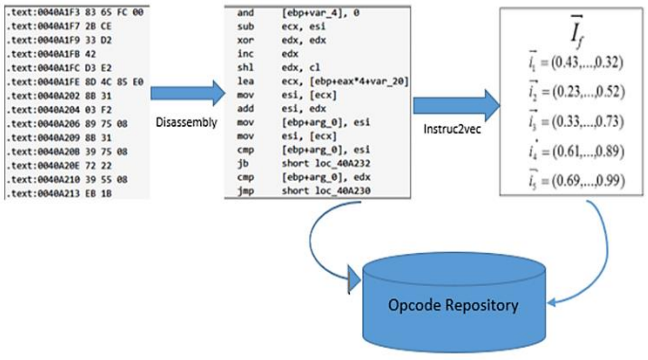


Figure 2: Opcode Extraction and vector conversion process

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 262-268 | e-ISSN: 2347-470X

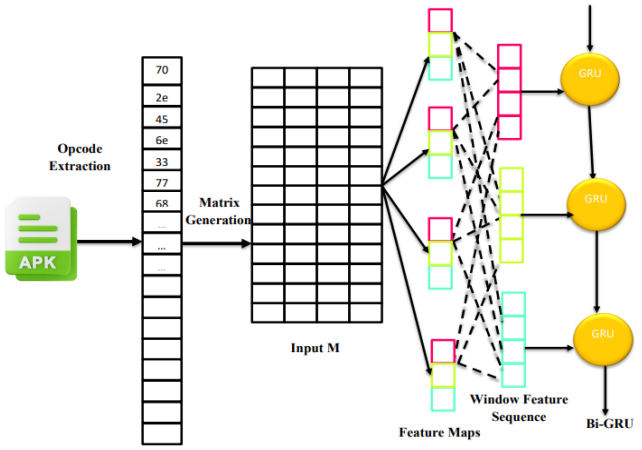


Figure 3: Opcode Embedding using N-skip gram process

The output matrix is specifically created utilizing the convolutional community layer and the opcode collection provided by using the entrance matrix. Typically, one or more significant features are chosen *via* feature mapping using the max group and k-max group after convolution.

3.3.Classification using Bi-GRU network

The deep learning model for categorizing binary code similarity via Bi-GRU is displayed in *figure 4*. The GRU is composed of three layers which is input, hidden, and output. *Figure 4* shows the Deep learning model to discriminate between similar data and non-similar data with input binary function and output prediction. The number of observed point vectors with the input word and input sequence across the temporally indexed period is given by the expression t= (1, 2, ..., n), where t is a word-length vocabulary and y_t is a binary function used by similar and non-similar data to obtain data.

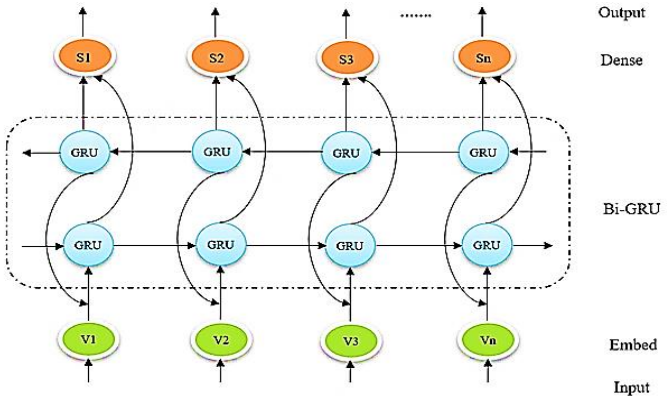


Figure 4: Binary code similarity classification using Bi-GRU

Reset output (s_t), replace gate output (u_t), and GRU output (h_t) definitions are provided. The reset and replacement gates also determine the h_t the output of the cutting-edge time by simultaneously modifying the h_{t-1} the output of the previous time and the entry of the cutting-edge time. Both the replace gate equation and the reset gate equation are contained in the *equation (4)* correctly, as well as *equation (5)*.

$$s_t = \sigma\,(W_s\,[h_{t-1}, y_t]) \quad (4)$$

$$u_t = \sigma\,(W_u[h_{t-1}, y_t]) \quad (5)$$

Where, $\sigma(x) = \frac{1}{1+e^{-y}}$ is the sigmoid function, and the weights for the reset and alternative gates, W_s and W_u, respectively, are all present. The equation contains the output equation for the calculation of *equation (6)*.

$$h_t = (1 - u_t) \times h_{t-1} + u_t \times \bar{h}_t \quad (6)$$

where $h_t$is the GRU candidate state at time t. The expression alone proves the computation of h_t *equation (7)*.

$$\bar{h}_t = \tan h\,(W_h\,[s_t \times h_{t-1}, y_t]) \quad (7)$$

The candidate state's weight is denoted as W_h. Using GRUs that can be represented as *equation (8)– (10)*, Bi-GRU is determined.

$$\vec{h}_t = GRU_f\,(y_t, \vec{h}_{t-1}) \quad (8)$$

$$\overleftarrow{h}_t = GRU_b\,(y_t, \overleftarrow{h}_{t-1}) \quad (9)$$

$$h_t = \vec{h}_t \oplus \overleftarrow{h}_t \quad (10)$$

$\vec{h}_t$ and $\overleftarrow{h}_t$ represent the state information of the forward and reverse GRUs, respectively. The GRU function is an equation, which denotes GRUf for forward and GRUb for backward which is composed of *equation (8)- (10)*. $\oplus$ which denotes concatenating the $\vec{h}_t$ and $\overleftarrow{h}_t$. As a result, Bi-GRU, which has a bidirectional GRU form, can store both similar and non-similar data records. Finally, it might classify the output as either similar data or non-similar data.

4. RESULT AND DISCUSSION

One of the six suggested Linux distributions used to produce test datasets is until-Linux, which uses Coreutils, findutils, diffutils, sg3utils, and other tools. The package supply code is obtained and each software is generated using x86-64 and ARM processors, compilers (gcc and clang), and 4 optimization levels (O0, O1, O2, and O3).

Dataset generation

The appropriate functions in various created files use the intact information available in the binary to obtain active learning versions. To produce negative examples, random vector pairs are created that result from non-equivalent processes. After making sure that no pairs exist in the match list, these pairs are labeled with the value 0 which is no match, and used as a set of negatively labelled data.

4.1.Performance analysis

This section quantifies the efficiency of binary code similarity detection using various metrics, including precision, recall, F1 score, AUC, detection time, and detection rate. Dataset Generation Properties is shown in *figure 5*.

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 262-268 | e-ISSN: 2347-470X

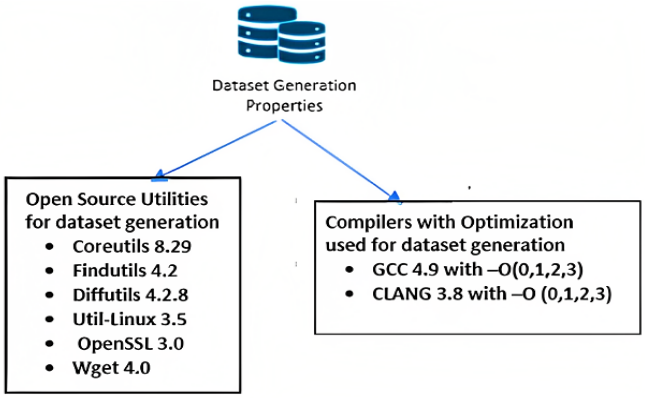
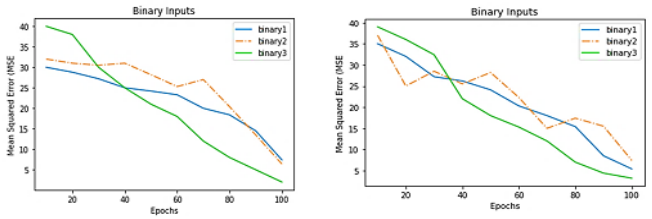


Figure 5: Dataset Generation Properties



(a) (b)

Figure 6: (a)Training and testing epochs vs (b)Average MSE among multiple Binaries

4.2. Comparative analysis

The effectiveness of the SimCoDe-NET technique was assessed using certain characteristics, including recall, F1 score, precision, and specificity. The comparison of the SimCoDe-NET methodology under consideration with existing methods such as jTrans [14], UPPC [15], and HEBC [17] is based on the dataset which is presented below. Training and testing epochs and Average MSE among multiple Binaries are shown in *figure 6.*

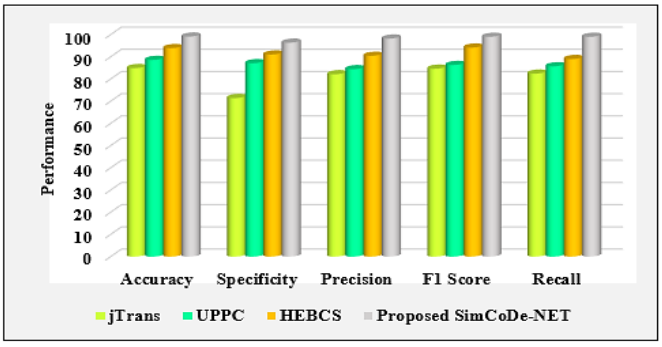


Figure 7: Performance metrics of the proposed SimCoDe-NET model

To illustrate the greater efficacy of the proposed SimCoDe-NET approach, it has been compared with other approaches. Performance is affected by TPR, sensitivity, recall, F1 score, and precision, among other factors. A graphic representation of the F1 score, specificity, sensitivity, recall, and precision are shown in *figure 7.* Performance comparison of proposed SimCoDe-NET model shown in *table 2.*

Table 2. Performance comparison of proposed SimCoDe-NET model

Method	Accuracy	Specificity	Precision	F1-Score	Recall
jTrans	84.9	71.41	82.13	84.63	82.41
UPPC	88.58	87.1	84.4	86.3	85.7
HEBCS	93.9	91	90.44	94.14	88.98
SimCoDe-NET	99.10	96.35	98.2	99	98.99

The accuracy of the proposed method is 99.10% which is relatively high compared to the existing method. The proposed SimCoDe-NET approach improves the accuracy by 84.9%, 88.58%, and 93.9% better than jTrans, UPPC, and HEBCS respectively.

4.3. Similarity Detection Accuracy

The Bi-GRU neural network uses at least 60 epochs to assess the framework's efficacy. To explain the performance, a metric called detection accuracy, which is formally defined in *equation (11)*:

$$Detection\ Accuracy = \frac{correctly\ detected\ Test\ Binaries}{total\ test\ binaries} \quad (11)$$

Figure 8 illustrates the relationship between detection accuracy and training time by using half of the binaries as instructional facts and the other half as checking information. As a result, following training, it can attain a detection accuracy of roughly 90%. It is significant to note that the detection accuracy in this instance is 85%, as opposed to the earlier detection accuracy in the dataset, which was 90% when the test data came from the same categories as binaries [20]. Successive Rate of Existing and the Bi-GRU Neural Network shown in *figure 9.*

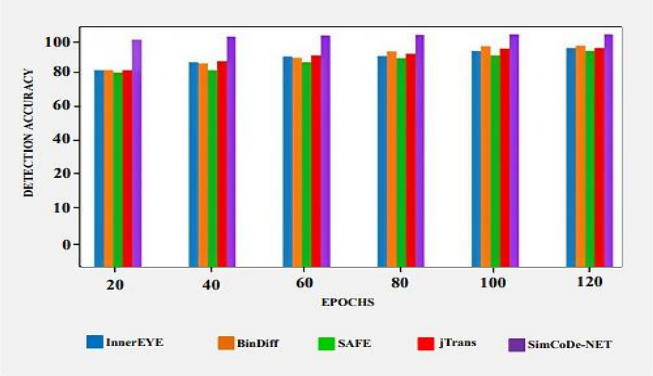


Figure 8: Validation of accuracy among proposed framework and baseline models

The purpose of this research is to develop a novel similarity detection method utilizing the DEep Learning Network (SimCoDe-NET) with the goal of increasing the efficiency of binary code similarity determination. The overall effectiveness of the suggested SimCoDe-NET approach is assessed by comparing a number of criteria with the acquired results,

FOREX Publication
Open Access | Rapid and quality publishing
International Journal of Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 262-268 | e-ISSN: 2347-470X

including recall, F1 score, accuracy, and specificity. Comparing the accuracy of the SimCoDe-NET approach to earlier techniques like jTrans [14], UPPC [15], and HEBC [17], it is very high at 99.10%. The proposed SimCoDe-NET approach improves accuracy by 84.9%, 88.58%, and 93.9% in comparison to jTrans, UPPC, and HEBCS, respectively. 90% detection accuracy for the frame and method detection accuracy were obtained by using well-known techniques as InnerEYE [23], BinDiff [24], SAFE [25], and jTrans [14]. The SimCoDe-NET is assessed. SimCoDe-NET. A number of deep learning techniques, such as RNN [21], Tree-LSTM [22], and DPCNN [14], were employed to assess the Bi-GRU method's classification performance. Particular standards were taken into account, such as recall, accuracy, precision, specificity, and F1 score. Furthermore, the same objective is also accomplished by the proposed SimCoDe-NET approach, which also determines the similarity of actual binary codes.

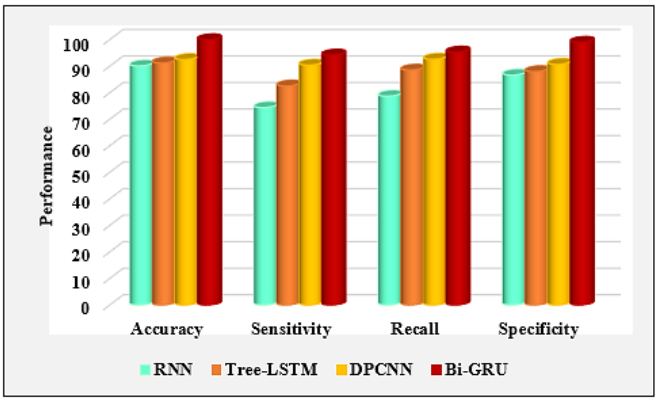


Figure 9: Successive Rate of Existing and the Bi-GRU Neural Network

4. CONCLUSIONS

In this research, a novel SIMilarity detection in binary COde using DEep learning NETwork (SimCoDe-NET) has been proposed. The Bi-GRU neural network compares two data samples in the feature space to determine whether the given data belongs to similar data or dissimilar data. With a test dataset and an unknown dataset, the model's accuracy was 0.920 and 0.704, respectively. The accuracy of the proposed method is 99.10% which is relatively high compared to the existing method. The proposed SimCoDe-NET approach improves the accuracy by 84.9%, 88.58%, and 93.9% better than jTrans, UPPC, and HEBCS respectively. As compared to current approaches, the experimental results provide promising results in terms of detection accuracy and computational efficiency. In future, we will enhance the process for creating a graph out of malware. Improve the way DLL functions are processed, learn where to look for the important functions, and stop using the default functions that the compiler produces.

5. ACKNOWLEDGMENTS

The authors would like to thank the reviewers for all of their careful, constructive and insightful comments in relation to this work.

REFERENCES

[1] Shin, E.C.R., Song, D. and Moazzezi, R. 2015. Recognizing functions in binaries with neural networks. In 24th {USENIX} Security Symposium, 611–626.

[2] Lou, A., Cheng, S., Huang J. and Jiang, F. 2019. Tfdroid: android malware detection by topics and sensitive data flows using machine learning techniques. in Proceedings of the 2019 IEEE 2nd International Conference on Information and Computer Technologies, ICICT, Hawaii, HI, USA, 30–36.

[3] Shalev, N. and Partush, N. 2018. Binary similarity detection using machine learning. In: Proceedings of the 13th workshop on programming languages and analysis for security. ACM, New York, NY, USA, 42–47.

[4] Egele, M., Woo, M., Chapman, P. and Brumley, D. 2014. Blanket execution: Dynamic similarity testing for program binaries and components. In Proceedings of the 23rd USENIX Conference on Security Symposium. Berkeley, CA, USA: USENIX Association, 303–317.

[5] Eschweiler, S., Yakdan, K. and Gerhards-Padilla, E. 2016. Discovre: Efficient crossarchitecture identification of bugs in binary code. In Proceedings of the 2016 network and distributed systems security symposium (NDSS)

[6] Wang, Y., Shen, J., Lin, J. and Lou, R. 2019. Staged method of code similarity analysis for firmware vulnerability detection, IEEE Access, 7, 14171–14185.

[7] Zhao, L., Li, D., Zheng, G. and Shi, W. 2018. Deep neural network based on android mobile malware detection system using opcode sequences. In: 2018 IEEE 18th International Conference on Communication Technology (ICCT) 1141–1147.

[8] Ananya Aswathy, Amal, T. R., Swathy, P. G., Vinod, P. and Mohammad, S. 2020. SysDroid: a dynamic ML-based android malware analyzer using system call traces. Cluster Computing 23(4), 2789–2808.

[9] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E. and Shah, R. Signature verification using a siamese time delay neural network. Advances in neural information processing systems, 737–737.

[10] Zhang, X., Sun, M., Wang, J. and Wang, J. 2018. Malware detection based on opcode sequence and resnet. in Proceedings of the International Conference on Security with Intelligent Computing and Big-Data Services, Springer, Guilin, China, 489–502.

[11] Şahın, D.Ö., Kural, O.E., Akleylek, S. and Kiliç, E. 2018. New results on permission based static analysis for android malware. In 2018 6th International Symposium on Digital Forensic and Security (ISDFS) 1–4.

[12] Yu, Z.,Cao, R.; Tang, Q., Nie, S., Huang, J. and Wu, S. 2020. Order matters: semantic-aware neural networks for binary code similarity detection. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, pp. 1145–1152.

[13] Shukla, S., Kolhe, G., PD, S.M. and Rafatirad, S. 2019. Stealthy malware detection using rnn-based automated localized feature extraction and classifier. In 2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI) (IEEE), 590–597.

[14] Wang, H., Qu, W., Katz, G., Zhu, W., Gao, Z., Qiu, Zhuge, J. and Zhang, C. 2022. jtrans: Jump-aware transformer for binary code similarity.

[15] Zhang, W., Xu, Z., Xiao, Y. and Xue, Y. 2022. Unleashing the power of pseudo-code for binary code similarity analysis. Cybersecurity, 5(1), 23.

[16] XJiang, S., Wang, X., Yu and Gong, Y. 2022. Double-Layer Positional Encoding Embedding Method for Cross-Platform Binary Function Similarity Detection. Chinese Journal of Electronics, 31(4), 604-611.

[17] Sun, X., Wei, Q., Du, J. and Wang, Y. 2023. HEBCS: A High-Efficiency Binary Code Search Method. Electronics 12(16), 3464.

[18] Liu, G., Zhou, X., Pang, J., Yue, F., Liu, W. and Wang, J. 2023. Codeformer: A GNN-Nested Transformer Model for Binary Code Similarity Detection. Electronics, 12 (7), 1722.

[19] Wu, G. and Tang, H. 2023. Binary Code Vulnerability Detection Based on Multi-level Feature Fusion. IEEE Access

[20] Guo, J., Zhao, B., Liu, H., Leng, D.; An, Y. and Shu, G. 2023. DeepDual-SD: Deep Dual Attribute-Aware Embedding for Binary Code Similarity

International Journal of
Electrical and Electronics Research (IJEER)
Research Article | Volume 12, Issue 1 | Pages 262-268 | e-ISSN: 2347-470X

Detection. International Journal of Computational Intelligence Systems 16(1), 35.

[21] Huang, C., Zhu, G., Ge, G., Li, T. and Wang, J. FastBCSD: Fast and Efficient Neural Network for Binary Code Similarity Detection.

[22] Yang, S., Dong, C., Xiao, Y., Cheng, Y., Shi, Z., Li, Z. and Sun, L.2003. Asteria-Pro: Enhancing Deep-Learning Based Binary Code Similarity Detection by Incorporating Domain Knowledge. ACM Transactions on Software Engineering and Methodology.

[23] Zuo, F., Li, X.,Young, P., Luo, L., Zeng, Q. and Zhang, Z. 2018. Neural machine translation inspired binary code similarity comparison beyond function pairs. arXiv preprint arXiv:1808.04706.

[24] Arutunian, M., Hovhannisyan, H., Vardanyan, V., Sargsyan, S., Kurmangaleev, S. and Aslanyan, H. 2021. A Method to Evaluate Binary Code Comparison Tools. In 2021 Ivannikov Memorial Workshop (IVMEM), 3-5.

[25] Massarelli, L., Di Luna, G.A., Petroni, F., Baldoni, R. and Querzoni, L. 2019. Safe: Self-attentive function embeddings for binary similarity. In Detection of Intrusions and Malware, and Vulnerability Assessment. 16th International Conference, DIMVA 2019, Gothenburg, Sweden, June 19–20, 2019 Proceedings 16, 309-329.