Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)
Vol 7, No.1, March 2024, pp. 97 – 103
p-ISSN: 2614-3372 | e-ISSN: 2614-6150

Uncovering Malware Families Using Convolutional Neural Networks (CNN)

[1*]Ruly Sumargo, [2]Handri Santoso
[1,2]Big Data and Internet of Things, Fakultas Teknologi Informasi Universitas Pradita, Tangerang
Email: [1]ruly.sumargo@student.pradita.ac.id, [2]handri.santoso@pradita.ac.id

Article Info

Article history:
Received Nov 5th, 2023
Revised Jan 30th, 2024
Accepted Feb 30th, 2024

Keyword:
Convolutional Neural Network
Cyberattack
Email
Malware
Python

ABSTRACT

Malware attacks pose significant cyber threats, with a rising number of vulnerability reports in security communities due to the continual introduction of mutations by malware programmers to evade detection. One of the most attractive targets which attacked by malware is the organization emails system. Malware's mutations within the malware family, has complicating the development of effective machine learning-based malware analysis and classification methods. To answer this challenge, this research uses an agnostic deep learning solution inspired by ImageNet's success, which efficiently classifies malware into families by analyzing visual representations of malicious software as greyscale images using a Convolutional Neural Network (CNN). The Malwizard is a flexible Python tool suitable for both organizations and end-users enabling automated and rapid malware analysis within email system. Malwizard could be use as an Outlook Email's add-in and an API service for SOAR platforms. The study evaluates this novel approach using the Microsoft Classification Challenge dataset, where image representations are encrypted to address privacy concerns. Experimental results show that the proposed approach performs comparably to the best existing model on plain text data, accomplishing the task in one-third of the time. For the encrypted dataset, adjustments to classical techniques are necessary for improved efficiency.

Corresponding Author:
Ruly Sumargo,
Department of Information Technology,
Pradita University,
Scientia Business Park, Jl. Gading Serpong Boulevard No.1, Kabupaten Tangerang, Banten 15810.
Email: ruly.sumargo@student.pradita.ac.id

DOI: http://dx.doi.org/10.24014/ijaidm.v7i1.27243

1. INTRODUCTION

Malware, commonly known as malicious software, is a harmful program that runs within a computer system without permission, with the aim of damaging or stealing personal data from the system [1]. Malware is classified into several types such as viruses, trojan horses, worms, rootkits, scareware, and spyware. Malware is not always classifiable based on its function because one malware can have multiple functions depending on its creator's intentions [2].

Over the past few years, the malware attacks have been steadily growing. In 2019, the survey related to spam emails reached an average of 14.5 billion emails per day globally. From that figure, 2.5% (equal to 365.2 million emails) were identified as email phishing [6]. According to the "Threat Landscape 2020" report from the European Network and Information Security Agency (ENISA), email and phishing messages have emerged as the leading means of malware transmission, positioning malware as the top threat among the 15 trending threats[7][8]. And in 2022, SonicWall's reported the number of malware attacks worldwide reached 5.46 billion cases. This marked a 1.52% increase compared to the previous year, which had 5.38 billion cases. If malware successfully infiltrates and performs activities that go undetected by the system, it will result in a system performance slowdown [9]. Furthermore, malware attacks can lead to various issues, including

eavesdropping, unauthorized access without the owner's knowledge or permission, manipulating bank transactions for profit, personal data theft, and financial losses. These are evidence of digital crimes committed by intruders using malware as their medium [10]. Malware is also rapidly evolving, giving rise to numerous new malware families that are challenging to detect [1]. Given those issues, malware classification is necessary to facilitate the study and differentiation of malware types [11].

Malware uses various methods to infect systems, for example infecting through email attachments, scripts from web pages, links to web files ready for installation, exploiting system bugs, USB drivers, file sharing, and even pirated applications. Malware is often camouflaged using common file types such as drivers (.drv), data (.dat), libraries (.lib), temporary files (.tmp), and others, which sometimes may not be aware by users [3].

Malware classification is one of the data mining techniques that map data into predefined groups or classes [12]. Considering that data mining will lead to a huge dataset and complex models, researchers have started to adopt machine learning classifiers to tackle the issues through logistics regression, neural networks, and decision trees [13][14]. Those methods are used to achieve a novel, swifter, and dependable approach that is not reliant on code or secure environment analysis but is capable of identifying, examining, and categorizing malware based on its content and behavior [13].

Neural networks emulate the functioning of the human brain, enabling computer algorithms to uncover patterns in a short time and address typical problems encountered in the field of machine learning. It is a good alternative method for performing malware analysis in terms of accuracy and short processing time. Convolutional Neural Networks (CNN) belong to the family of feedforward neural networks and are particularly adept at tasks like image recognition [15][16]. The primary distinguishing feature lies in the arrangement of neurons within the layers. CNNs feature neurons organized in three dimensions, encompassing width, height, and depth. These networks are characterized by three distinctive types: convolutional, pooling, and fully connected, setting them apart from traditional feedforward neural networks [17][18]. CNN methods, commonly used for image processing have exhibited strong performance in the classification process [19].

Email systems represent a vital component of any organization's infrastructure. Email server plays a pivotal role and it's under constant threat from a rising tide of security concerns like business email compromise (BEC), phishing, data leaks, and privacy breaches, among other challenges. These vulnerabilities serve as a strong motivation for organizations to fortify their email infrastructure [4].

Email servers were originally designed with security as a secondary concern. While they do employ a combination of anti-spam and antivirus tools, their primary defenses are limited to measures like grey lists, the Sender Policy Framework (SPF), and Domain Keys Identified Mail (DKIM), as well as the Domain-based Message Authentication, Reporting, and Conformance (DMARC) mechanisms. These mechanisms are effective at detecting forged sender addresses and preventing unauthorized domain use but fall short in identifying malicious content within email messages [5].

Considering the importance and critical level of email systems for an organization, utilizing the CNN algorithm in malware classification for email systems could become the best option. The CNN algorithm leverages parameter sharing, thereby reducing the overall system parameter count and lessening the computational load [16]. In addition, CNNs can automatically learn and extract relevant features from the raw binary data of malware samples.

2. RESEARCH METHODOLOGY

This research is arranged in a systematic flow, starting from identifying the problems, gathering information, deep-dive study, and development of the final solution [20][21], as shown in Figure 1.

Identifying and detecting malicious software poses a highly challenging task without a perfect solution. The landscape of malicious software has transformed into a structured marketplace with significant financial consequences, driving considerable investments in technologies and resources aimed at surpassing traditional security measures. This creates a substantial barrier for the cybersecurity community.

To investigate this matter, two pivotal datasets of malicious software were selected. The initial dataset originates from the Microsoft Malware Classification Challenge, which provides one of the most comprehensive and current publicly available collections, and the second dataset comes from the Big Data Innovators Gathering Cup (BIG 2015) [22]. Additionally, it highlights the significance of the VirusShare.com repository, widely recognized as a principal source for malware samples, granting access to security researchers, incident responders, forensic analysts, and even the curious, providing a repository of real malicious code. In order to streamline the system by swiftly conducting malware analysis, a versatile Python tool named 'Malwizard' was selected. This solution is well-suited for both corporate entities and individual end-users. The implementation of Malwizard took the form of an Outlook add-in, as well as an API service compatible with risk-based Security Orchestration, Automation, and Response (SOAR) platforms, such as “TheHive” [23].

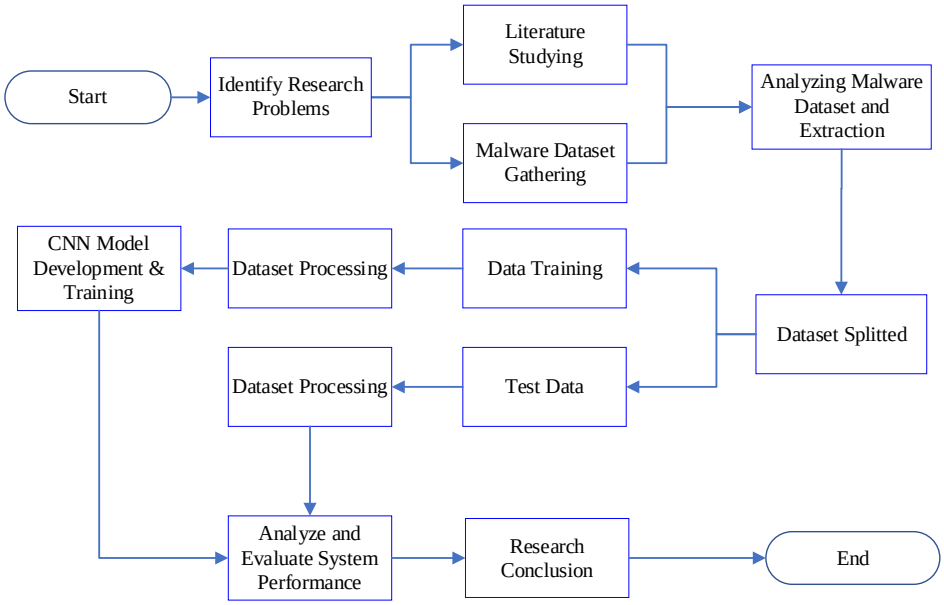


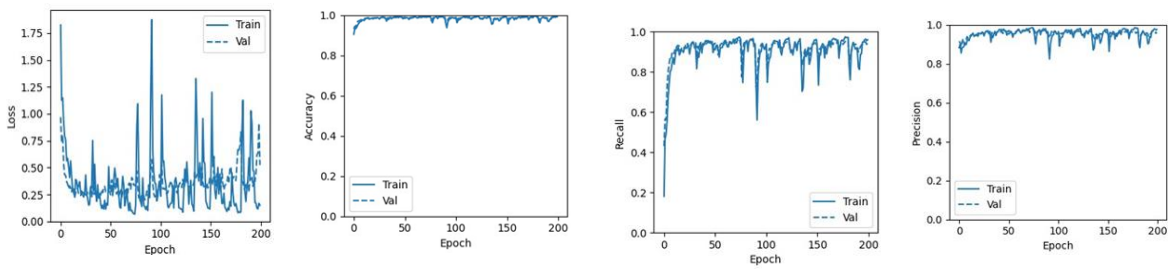
Figure 1. Research Flow Chart

To evaluate the performance of the designed system, an evaluation of the novel method for the classification of malicious software contained in email attachments was performed. This approach involves the conversion of the attachments into grayscale images and leverages CNN to acquire a hierarchical understanding of features, starting from pixel-level data and extending through the layers of the classifier. In addition, customization in this method was performed, to uphold user privacy by employing homomorphic encryption for the representation of email attachment images. This encryption technique allows for image classification without requiring the decryption of the data [24][25]. At the final stage, model evaluation was performed using the Microsoft dataset, and the result was with the most popular models such as ResNet-50, VGG16, Google's Inception, and Gilbert's model.

3. RESULT

To mitigate potential disruptions during the tests, all experiments were conducted within the same environment. A single computer using Ubuntu 20.4 LTS operating system, machine learning implementation supported by Keras and Tensorflow 2.4.2. GPU AMD RX Vega 56 (8GB VRAM) with API GPU ROCm 4.1.1. CPU AMR Ryzen 9 3900x and memory 32GB used to build the system.

Considering the highly unbalanced classes in the Microsoft dataset, where the highest number of samples has 2942 entries and the lowest number were 42, class weighting was chosen to balance the 9 classes on the Microsoft dataset. The model underwent training and validation, with a ratio of 80% portion of the data used for training and the remaining 20% for testing. Prior to testing, all images were resized to 256 × 256 pixels using area interpolation, which interpolates based on pixel relationships. The training phase encompassed 200 epochs, indicating the number of times the model was trained with the training data, using the default Keras batch size of 32.



(a) (b) (c) (d)

Figure 2. Main Metrics Model Evaluation: (a) loss (b) accuracy (c) recall (d) precision

Figure 2 shows the training and validation results. In the loss graph (a), the validation curve's loss increases as the model keeps training, while the training curve's loss tends to decrease. In the recall graph (c),

the training line stays stable after the 50th round, and the validation curve consistently hovers around a recall value of 0.9. However, it's important to note that the model seems to be overfitting during training, as seen in the erratic behavior, the accuracy, precision, and recall curves approaching a value of one. This overfitting might be due to the highly imbalanced dataset.

Table 1. Confusion Matrix for The Designed Model

Malware Family	Precision	Recall	F1-Score	Support
Kelihos_ver3	0.991843	0.996721	0.994276	610
Lollipop	0.983299	0.971134	0.977178	485
Ramnit	0.955479	0.914754	0.934673	305
Obfuscator.ACY	0.974249	0.900794	0.936082	252
Gatak	0.917874	0.989583	0.952381	192
Tracur	0.890909	0.960784	0.924528	153
Vundo	0.952941	0.941860	0.947368	86
Kelihos_ver1	0.908046	0.975309	0.940476	81
Simda	0.666667	0.888889	0.761905	9
Macro average	0.915701	0.948870	0.929874 (92.99%)	2173
Weighted average	0.963139	0.961804	0.961909 (96.19%)	2173
Accuracy			0.961804 (96.18%)	2173

Table 1 shows the designed model has achieved an F1-Score average of 96.19 percent when making predictions on the training data. While it accurately predicts most classes, it struggles with classes that have small quantity samples. Even with weight balancing, increasing the samples in these classes or using class balancing is crucial to improve overall accuracy and uniformity.

Table 2 shows the comparison of various models based on trainable parameters, weighted F1-Score, epoch average time, and overall training time. VGG16 and Gibert's models had lower accuracy due to high parameter counts leading to noise in images. ResNet-50 achieved a good F1-Score, but it required a longer training time than InceptionV3.

Table 2. Model Performance Comparison for Malware Classification Using Microsoft Dataset

Model	Parameter	Accuracy	F1-Score	EpochTime (sec/avg)	Time to Train (seconds)
ResNet-50	23 546 761	97.52%	97.52%	1350s	4509.59
VGG16	165 753 545	93.56%	93.68%	130s	434.98
InceptionV3	21 786 217	97.79%	97.80%	58s	189.33
Gibert's Model	245 386 489	81.36%	80.95%	29s	99.44
Designed Model	41 946 377	96.18%	96.19%	16s	54.39

The designed model balanced F1-Score and training time effectively. The Keras implementation of Gibert's model had an accuracy of 81.36 percent, differing from the study. K-fold tests typically show lower accuracy due to dataset splitting. This research lacks information on dataset balancing and uses macro average F1-Score instead of weighted average during the trial.

Table 3. Model Confusion Matrix for The Designed Model With Sigmoid Output

Malware Family	Precision	Recall	F1-Score	Support
Kelihos_ver3	0.996522	0.939344	0.967089	610
Lollipop	0.957606	0.791753	0.866817	485
Ramnit	0.970464	0.754098	0.848708	305
Obfuscator.ACY	0.715789	0.539683	0.615385	252
Gatak	0.509485	0.979167	0.670232	192
Tracur	0.616114	0.849673	0.714286	153
Vundo	0.951220	0.906977	0.928571	86
Kelihos_ver1	0.896552	0.962963	0.928571	81
Simda	0.380952	0.888889	0.533333	9
Macro average	0.777189	0.845838	0.785888 (78.59%)	2173
Weighted average	0.873736	0.830649	0.838520 (83.85%)	2173
Accuracy			0.830649 (83.07%)	2173

The implementation of the sigmoid activation function resulted in a weighted mean F1-Score of 83.85 percent, as shown in Table 3. This result was lower than the 96.19 percent achieved with the softmax function which is shown in Table 1. The Simda class had the lowest performance due to limited samples, while other classes achieved relatively high F1-Score values, enabling the model to distinguish between malware and non-malware samples effectively. Performance tests were conducted ten times with 20 MB files, the maximum attachment size allowed in Microsoft Outlook.

Table 4 shows baseline timings without encryption, displaying "Plain Text Time" for image resampling using the OpenCV library. The Microsoft Outlook add-in utilized OpenCV.js, which has a higher-level abstraction due to JavaScript, resulting in a processing time of 1.19 seconds. In contrast, the Python OpenCV implementation, closer to machine-level instructions, processed the task faster at 0.07 seconds.

Table 4. Plain Text Performance Comparison

	Plain Text Time (seconds)	Overall Time Request (seconds)
Microsoft Outlook Add-In	1.19	1.50
Middleware - API	0.07	1.16

Table 5 shows the performance comparison between the Javascript-based Microsoft Outlook Add-In and the Python API implementation for homomorphic encryption (HE). Image encryption is computationally intensive, with each pixel requiring encryption, leading to increased processing time. Using a 128-bit key for 256x256 images, the Javascript-based add-in takes approximately 83 seconds, while increasing the key to the recommended 2048 bits extends processing time to 336 minutes, rendering real-time processing impractical.

Table 5. Python (API) And Javascript (Microsoft Outlook Add-In) Performance Comparison

	Key Size	Encrypt Time (sec)	Overall Time Request (seconds)
Microsoft Outlook Add-In	128-bits	83.98 (≈ 1.40 min)	86.06 (≈ 1.43 min)
Middleware - API	128-bits	21.19 (≈ 0.35 min)	23.29 (≈ 0.39 min)
Microsoft Outlook Add-In	2048-bits	20 189.88 (≈ 336 min)	20 191.32 (≈ 336 min)
Middleware - API	2048-bits	5 093.69 (≈ 85 min)	5 095.80 (≈ 85 min)

From the result above, the Python API performs similarly taking 21 seconds with a 128-bit key and 85 minutes with a 2048-bit key. Despite the poor performance of strong keys, such as 2048-bit keys, they are highly recommended for security, even if they limit the solution's use in daily basis activities.

4. CONCLUSION

This research has introduced a novel approach for malware assessment, distinct from traditional signature-based and quarantine methods. It resulted in two main contributions: designing a new machine learning model and implementing it in Malwizard, a malware analysis tool for Microsoft Outlook, and API integration with security platforms. The new machine learning model involved transforming malware files into grayscale images, aiming for efficiency and improving upon existing models. Tests using the Keras framework and Scikit-learn library validated that the designed model achieved 96.18% accuracy, outperforming traditional models.

FUTURE WORK

In this section, we present some potential opportunities for advancing Malwizard development and enhancing encrypted image analysis through machine learning (ML). The first steps could involve creating new extensions utilizing the Malwizard API to expand integration with diverse security tools.

The second feasibility for the future work is an essential future focus on utilizing adepted models during analyzing encrypted images process and ensuring privacy during malware assessment. The expanding market for ML analysis demanding on a need to reduce encryption duration for files under assessment. Enhancing homomorphic encryption methods or utilizing more efficient cipher algorithms could accomplish this, maintaining sample structure akin to homomorphic encryption.

The third future work is step involves refining the image resampling process, it is necessary to find a method to downsample the malware representation images, effectively identifying and disregarding noise while retaining crucial information. This downsizing is critical for Malwizard, reducing image size for analysis, minimizing encryption time in privacy mode, and enhancing malware detection.

A final area of future work involves sourcing a dataset of legitimate files to establish a legitimate class, enabling the ML model in Malwizard to train beyond malicious files. This class facilitates the use of normalization functions, like softmax, to determine the class most similar to the sample under analysis, thereby elevating accuracy in identifying malicious files.

ACKNOWLEDGEMENTS

We sincerely wish to express our gratitude to all parties who have participated and contributed significantly to this research. Our heartfelt thanks go to all individuals and institutions who have provided invaluable assistance and exceptional support in facilitating the progress of this research. To everyone who has extended support, advice, and technical assistance throughout the research process, we extend our utmost gratitude for the meaningful contribution to the smooth completion of this study.

The Microsoft Malware Classification Challenge dataset was used to test the model but encountered challenges due to class imbalance and resource limitations. Malwizard, comprising an Outlook Add-in, API, and machine learning processing units, was developed to ease everyday malware analysis. Its potential in cyber security has been demonstrated, and its ability to adapt to various ML models has been emphasized. Malwizard's flexibility and continuity highlight its potential to contribute to malware detection and prevention, addressing ongoing challenges in machine learning-based malware analysis.

REFERENCES

[1] A. R. Yogasware, D. R. Akbi, and V. R. Nastiti, "Klasifikasi Malware Family Menggunakan Metode K-Nearest Neighbor (K-NN)," *REPOSITOR*, pp. 305–314, 2021, doi: https://doi.org/10.47065/bits.v5i1.3538.

[2] Y. D. Puji Rahayu and Nanang Trianto, "Analisis Malware Menggunakan Metode Analisis Statis dan Dinamis untuk Pembuatan IOC Berdasarkan STIX Versi 2.1," *Info Kripto*, vol. 15, no. 3, pp. 105–111, Nov. 2021, doi: 10.56706/ik.v15i3.30.

[3] F. Panjaitan, H. Yudiastuti, and M. Ulfa, "Analisis Malware dengan metode Surface dan Runtime Analysis," *J. Ilm. Matrik*, vol. 23, no. 1, pp. 1–11, Apr. 2021, doi: 10.33557/jurnalmatrik.v23i1.1148.

[4] S. Adiwal, A. Gupta, B. Rajendran, and B. S. Bindhumadhava, "A Secure Methodology for Filtering Spam & Malware in E-mail System and Secure E-mail Testbed Setup," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 10, no. 2, pp. 651–657, Apr. 2021, doi: 10.30534/ijatcse/2021/271022021.

[5] M. Office, Chrisda, and V. Ratulach, "Email authentication in EOP," 2023. https://learn.microsoft.com/en-us/microsoft-365/security/office-365-security/email-authentication-about?view=o365-worldwide (accessed Aug. 22, 2023).

[6] T. Muralidharan and N. Nissim, "Improving malicious email detection through novel designated deep-learning architectures utilizing entire email," *Neural Networks*, vol. 157, pp. 257–279, Jan. 2023, doi: 10.1016/j.neunet.2022.09.002.

[7] N. Suri, L. Barriga, M. Franco, F. Cutas, and C. Ardagna, "Work Package 4: Policy and the European Dimension Preliminary Version of Deliverable D 4.4: Cybersecurity Roadmap for Europe by CONCORDIA," 2020. [Online]. Available: https://www.concordia-h2020.eu/wp-content/uploads/2021/03/Deliverables_D4.4-M24.pdf

[8] European Union Agency for Cybersecurity (ENISA), "List of top 15 threats." https://www.enisa.europa.eu/topics/cyber-threats/threats-and-trends/etl-review-folder/etl-2020-enisas-list-of-top-15-threats (accessed Aug. 25, 2022).

[9] M. Hazri, "Analisis Malware PlasmaRAT dengan Metode Reverse Engineering," *J. Rekayasa Teknol. Inf.*, vol. 4, no. 2, p. 192, Nov. 2020, doi: 10.30872/jurti.v4i2.4131.

[10] H. Saputra, S. Basuki, and M. Faiqurahman, "Implementasi Teknik Seleksi Fitur Pada Klasifikasi Malware Android Menggunakan Support Vector Machine," *Fountain Informatics J.*, vol. 3, no. 1, p. 12, May 2018, doi: 10.21111/fij.v3i1.1875.

[11] D. Efriyani and F. Panjaitan, "Klasifikasi Malware Dengan Menggunakan Recurrent Neural Network," *J. Ilm. Matrik*, vol. 23, no. 3, pp. 310–316, 2021, doi: 10.33557/jurnalmatrik.v23i3.1592.

[12] P. B. N. Setio, D. R. S. Saputro, and Bowo Winarno, "Klasifikasi Dengan Pohon Keputusan Berbasis Algoritme C4.5," *Prism. Pros. Semin. Nas. Mat.*, vol. 3, pp. 64–71, 2020, [Online]. Available: https://journal.unnes.ac.id/sju/index.php/prisma/article/download/37650/15478/

[13] A. Djenna, A. Bouridane, S. Rubab, and I. M. Marou, "Artificial Intelligence-Based Malware Detection, Analysis, and Mitigation," *Symmetry (Basel).*, vol. 15, no. 3, p. 677, Mar. 2023, doi: 10.3390/sym15030677.

[14] J. Pavithra and S. Selvakumara Samy, "A Comparative Study on Detection of Malware and Benign on the Internet Using Machine Learning Classifiers," *Math. Probl. Eng.*, vol. 2022, pp. 1–8, Jun. 2022, doi: 10.1155/2022/4893390.

[15] F. A. Khatami, B. Irawan, and C. Setianingsih, "Analisis Sentimen Terhadap Review Aplikasi Layanan E-Commerce Menggunakan Metode Convolutional Neural Network," *e-Proceeding Eng.*, vol. 7, no. 2, pp. 4559–4566, 2020, [Online]. Available: https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/12305

[16] M. Afif, A. Fawwaz, K. N. Ramadhani, and F. Sthevanie, "Klasifikasi Ras pada Kucing menggunakan Algoritma Convolutional Neural Network(CNN)," *J. Tugas Akhir Fak. Inform.*, vol. 8, no. 1, pp. 715–730, 2020, doi: https://doi.org/10.34818/eoe.v8i1.14320.

[17] Phrabu, "Understanding of Convolutional Neural Network (CNN) — Deep Learning," 2018.

[18] P. O. A. Sunarya, R. Refianti, A. B. Mutiara, and W. Octaviani, "Comparison of accuracy between convolutional neural networks and Naïve Bayes Classifiers in sentiment analysis on Twitter," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 5, pp. 77–86, 2019, doi: 10.14569/ijacsa.2019.0100511.

[19] Yudi Widhiyasana, Transmissia Semiawan, Ilham Gibran Achmad Mudzakir, and Muhammad Randi Noor, "Penerapan Convolutional Long Short-Term Memory untuk Klasifikasi Teks Berita Bahasa Indonesia," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 10, no. 4, pp. 354–361, 2021, doi: 10.22146/jnteti.v10i4.2438.

[20] C. R. Kothari, *Research Methodology Methods & Techniques*, 2nd Editio. New Delhi: NEW AGE INTERNATIONAL PUBLISHERS, 2004.

[21] K. A. Adams and E. K. Lawrence, *Research Methods, Statistics, and Applications*, 2nd Editio. Los Angeles: SAGE, 2019.

[22] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, *Microsoft Malware Classification Challenge*.

2018. doi: https://doi.org/10.48550/arXiv.1802.10135.
[23] StrangeBee, "The Hive a 4 in 1 Security Incident Response Platform," 2019. https://thehive-project.org/ (accessed Jun. 03, 2023).
[24] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 1, pp. 342–351, 2016.
[25] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep Neural Networks over Encrypted Data," Nov. 2017, doi: https://arxiv.org/pdf/1711.05189.pdf.

BIOGRAPHY OF AUTHORS

Ruly Sumargo is a student of Big Data and Internet of Things at Pradita University, Tangerang. He is an aspiring AI enthusiast. He gained hands-on experience as an industry 4.0 expert and mastered diverse skills ranging from machine learning, data science, and industrial automation.

Handri Santoso received his bachelor's degree in Physics from the University of Indonesia in 1995 and proceeded with his professional career in the engineering field. In 2003, he continued his studies at Nagaoka University of Technology, Japan. He received his Master's degree and PhD in Information Science and Control Engineering, in 2005 and 2008 respectively. From 2008 up to now, he has been active in the education field as a researcher and lecturer, and also performs community services. Currently, he is the Dean of The Science and Technology Faculty of Pradita