The American Journal of Engineering and Technology
ISSN 2689-0984 | Open Access

TYPE Original Research
PAGE NO. 252-261
DOI 10.37547/tajet/Volume07Issue03-23

Check for updates

Real-Time Malware Detection in Cloud Infrastructures Using Convolutional Neural Networks: A Deep Learning Framework for Enhanced Cybersecurity

OPEN ACCESS

SUBMITED 30 January 2025
ACCEPTED 24 February 2025
PUBLISHED 28 March 2025
VOLUME Vol.07 Issue03 2025

CITATION

Abdullah Al Mamun, Ayan Nath, Sonjoy Kumar Dey, Paresh Chandra Nath, Md Mohibur Rahman, Jannatul Ferdous Shorna, & Nafis Anjum. (2025). Real-Time Malware Detection in Cloud Infrastructures Using Convolutional Neural Networks: A Deep Learning Framework for Enhanced Cybersecurity. The American Journal of Engineering and Technology, 7(03), 252–261. https://doi.org/10.37547/tajet/Volume07Issue03-23

Abdullah Al Mamun

Department of Computer & Info Science, Gannon University, Erie, Pennsylvania, USA

Ayan Nath

Master's in computer and information science, International American University

Sonjoy Kumar Dey

McComish Department of Electrical Engineering and Computer Science, South Dakota State University, USA

Paresh Chandra Nath

Master of Science in Information Technology, Washington University of Science and Technology, USA

Md Mohibur Rahman

Fred DeMatteis School of Engineering and Applied Science, Hofstra University, USA

Jannatul Ferdous Shorna

College of Engineering and Computer Science, Florida Atlantic University, Boca Raton, Florida

Nafis Anjum

College of Technology and Engineering, Westcliff University, Irvine, CA

Abstract: This study presents a novel malware detection framework for cloud infrastructures that harnesses the power of Convolutional Neural Networks (CNNs) to achieve real-time threat identification with superior

accuracy and speed. Our approach begins with the collection and meticulous preprocessing of heterogeneous cloud log data, followed by advanced feature engineering to extract meaningful patterns indicative of malicious activity. The CNN model automatically learns hierarchical representations from this high-dimensional data, resulting in a detection system that achieves an accuracy of 98.2%, a precision of 97.5%, a recall of 98.0%, and an F1-score of 97.8%. In addition, the model operates with a low latency of 12 ms, a critical factor for timely threat mitigation in dynamic cloud environments. Comparative analysis against Long Short-Term Memory (LSTM), Support Vector Machine (SVM), and Random Forest classifiers reveals that the CNN not only outperforms these models in key performance metrics but also maintains a significant advantage in processing speed. These findings highlight the potential of CNN-based approaches to enhance cybersecurity defenses, offering a scalable and efficient solution for detecting evolving malware threats in cloud infrastructures.

Keywords: Malware Detection, Cloud Infrastructures, Convolutional Neural Networks, Real-time Processing, Feature Engineering, Cybersecurity, Deep Learning, Cloud Security.

Introduction:

Cloud computing has revolutionized the way organizations deploy, manage, and scale their applications by offering on-demand access to computing resources and services (Zhao & Chen, 2019). Despite these benefits, cloud infrastructures remain susceptible to a wide range of cyber threats, with malware attacks posing one of the most persistent and evolving risks (Dahl, Stokes, Deng, & Yu, 2013). Malware can exploit vulnerabilities in virtual machines, container environments, or the underlying network fabric, potentially compromising the confidentiality, integrity, and availability of critical data and services (Saxe & Berlin, 2015). The dynamic nature of cloud environments, which often host diverse applications and experience fluctuating workloads, further complicates the detection process, as conventional rule-based or signature-based methods struggle to keep pace with emerging and polymorphic threats (Egele, Scholte, Kirda, & Kruegel, 2008).

Machine learning approaches have shown significant promise in enhancing malware detection by analyzing patterns of malicious behavior rather than relying solely on known signatures (Moustafa & Slay, 2015). However, many traditional machine learning models are limited by their feature extraction methods, which can fail to capture the complex, high-dimensional relationships inherent in cloud logs and system events (Hochreiter & Schmidhuber, 1997). Convolutional Neural Networks (CNNs) offer a powerful alternative by automatically learning hierarchical representations of data, allowing them to detect subtle indicators of compromise that may elude simpler models (Saxe & Berlin, 2015). Consequently, there is growing interest in exploring CNN-based methods for malware detection in cloud infrastructures, particularly in real-time scenarios where latency and accuracy are paramount. This article investigates the development and evaluation of a CNN-based framework for detecting malware in cloud environments, emphasizing the importance of robust data collection, feature engineering, model design, and performance assessment.

LITERATURE REVIEW

Research on malware detection has evolved significantly over the past decade, transitioning from signature-based techniques to advanced machine learning and deep learning approaches (Dahl et al., 2013). Signature-based methods, while effective for known threats, often fail to recognize zero-day attacks or rapidly morphing malware variants (Egele et al., 2008). In response, researchers have increasingly focused on behavior-based detection, leveraging system logs, network traffic, and file operations to identify anomalies that could signal malicious activity (Zhao & Chen, 2019). This shift toward behavior-based detection has proven especially valuable in cloud environments, where multitenancy and resource sharing can create a large and diverse attack surface.

Early applications of deep learning in malware detection used feedforward neural networks and recurrent architectures like Long Short-Term Memory (LSTM) networks to analyze sequential data (Hochreiter & Schmidhuber, 1997; Saxe & Berlin, 2015). LSTMs have been particularly useful for capturing long-term dependencies in event sequences, such as file operations or process interactions, which often reveal subtle malicious patterns. However, the complexity of high-dimensional data in cloud infrastructures—comprising logs from virtual machines, container orchestration systems, and network flows—necessitates models capable of extracting localized, as well as global, features (Dahl et al., 2013). Convolutional Neural Networks address this need by applying convolutional filters that capture both fine-grained and broader patterns in multidimensional data (Saxe & Berlin, 2015).

Recent studies have demonstrated the efficacy of CNNs in detecting malware by transforming log data into structured or image-like formats, allowing the convolutional layers to learn discriminative features automatically (Zhao & Chen, 2019). These approaches often outperform traditional machine learning models, particularly when dealing with large-scale, heterogeneous datasets. Nevertheless, most prior work has focused on offline analysis, leaving a gap in evaluating CNN-based detection in real-time or near-real-time settings where latency is a critical factor (Moustafa & Slay, 2015). Additionally, while some research has explored hybrid models that combine CNNs with other techniques (e.g., LSTMs or autoencoders), comprehensive comparisons among different architectures under real-world cloud constraints remain limited (Saxe & Berlin, 2015).

Building on these insights, our research seeks to advance the field by developing a CNN-driven malware detection framework specifically tailored to the dynamic and large-scale nature of cloud infrastructures. We place special emphasis on real-time detection performance and robustness to diverse threats, including zero-day and polymorphic malware. By systematically evaluating CNN-based approaches alongside other baseline models (e.g., LSTM, SVM, Random Forest), we aim to provide a more complete understanding of how deep learning can be leveraged to secure cloud computing environments.

METHODOLOGY

Data Collection

In this section, we describe our comprehensive strategy for collecting data from cloud infrastructures to support malware detection using convolutional neural networks. We collaborated with several cloud service providers to gather logs and monitoring data from diverse sources such as virtual machines, container orchestrations, API requests, and network flows. Our goal was to ensure that the dataset reflected both benign and malicious activities. We identified critical logging points that capture system events, network communications, user actions, and file operations. This allowed us to capture not only well-known malware signatures but also subtle behavioral anomalies that might indicate emerging threats. To facilitate a structured approach, we defined a data attribute table that served as a blueprint for the information we collected.

The data attribute table below outlines the key features that we captured during the data collection process. Each attribute is carefully defined with its data type, a description of its significance, the expected format or range of values, and any special processing notes that are necessary for downstream analysis. For instance, attributes such as "Timestamp" were recorded with high precision to allow for detailed temporal analysis, while "Source IP" and "Destination IP" were captured to enable network flow mapping and geolocation. Other attributes like "Protocol," "Port Number," "File Hash," "Process ID," and "User Identifier" provide essential context for identifying patterns that may be indicative of malware behavior. This table laid the groundwork for the subsequent stages of our methodology, ensuring that our data was both comprehensive and structured.

Attribute Name	Data Type	Description	Range/Format	Special Notes
Timestamp	Datetime	The exact time at which the event was recorded	YYYY-MM-DD HH:MM:SS	Essential for temporal correlation of events
Source IP	String	IP address of the initiating host	IPv4 or IPv6	Used for geolocation and network flow analysis
Destination IP	String	IP address of the receiving host	IPv4 or IPv6	Combined with Source IP for session mapping
Protocol	String	Communication protocol used (e.g., TCP, UDP)	Predefined protocol list	Helps in categorizing network traffic

Port Number	Integer	Network port associated with the connection	0–65535	Crucial for identifying service endpoints and vulnerabilities
File Hash	String	Crypt ographic hash of files accessed or modified	MD5, SHA-256, etc.	Key in identifying known malicious binaries
Process ID	Integer	Identifier for the process generating the log	OS-specific ranges	Useful for linking activities across system components
User Identifier	String	Unique identifier for the user account associated with the event	Alphanumeric strings	Important for associating behavior with user actions

Data Preprocessing

After collecting the raw data, we initiated a rigorous preprocessing stage to ensure its quality and consistency before feeding it into our convolutional neural network. Our preprocessing strategy addressed several challenges such as missing values, inconsistencies in data formatting, and the presence of noise. We standardized all timestamps to a uniform time zone, ensuring that temporal correlations among events could be accurately identified. In instances where data fields such as port numbers or file hashes were missing or corrupted, we applied domain-specific heuristics to either infer or impute these values based on contextual information. In addition, we eliminated duplicate records by cross-referencing events that occurred simultaneously across different logging sources.

We implemented normalization and encoding strategies to prepare the data for analysis. Categorical attributes like protocol types were transformed into numerical representations using one-hot encoding, which enabled the convolutional neural network to process these features effectively. We also examined the dataset for outliers by applying statistical measures such as interquartile ranges and Z-scores, ensuring that extreme values were scrutinized to differentiate between potential anomalies and errors in data collection. This preprocessing phase was critical to reduce the noise level in the data, enhance the signal-to-noise ratio, and maintain the integrity of the collected information as we moved into feature selection.

Feature Selection

With the data preprocessed and cleansed, we turned our focus to identifying the most relevant features that could help distinguish between benign and malicious activities. Our feature selection process combined statistical analysis with domain expertise to refine the set of variables that would feed into our model. We began by calculating correlation coefficients between features and their impact on the detection of malware, which helped us identify and eliminate redundant features. Features that exhibited high collinearity, such as certain combinations of network port numbers and protocols, were either merged or removed to reduce noise and simplify the model input.

Our iterative approach involved testing the selected features in preliminary training rounds and evaluating their performance in terms of discrimination power. Feedback from these early experiments guided adjustments in the feature set, ensuring that our final selection maximized the model's ability to capture meaningful patterns while minimizing overfitting. This careful balance between reducing dimensionality and retaining critical information was fundamental to achieving robust detection performance.

Feature Engineering

Beyond the initial selection, we engaged in extensive feature engineering to enrich the dataset with additional variables that could capture the complex temporal and sequential nature of cloud-based malware activities. We recognized that individual events might not be sufficient to reveal sophisticated attack patterns; instead, the relationships between successive events often held the key to early detection. Consequently, we engineered features that aggregated event data over various time windows, calculating metrics such as rolling averages, variances, and frequency counts of specific actions like failed logins or unusual file access patterns.

We also derived composite features by combining related attributes to highlight correlations between network activity and system processes. For example, by integrating network traffic volumes with corresponding protocol data, we were able to generate indicators that signaled abnormal surges in activity potentially linked to malware. These engineered features added context and depth to the raw data, significantly enhancing the ability of our convolutional neural network to detect nuanced behaviors indicative of malicious intent. Each new feature was validated through exploratory analysis and incorporated only after confirming that it contributed to improved model performance during our trial runs.

Model Development

The development of our convolutional neural network model was a critical phase in our methodology. We adapted state-of-the-art CNN architectures—originally designed for image recognition—to handle the sequential and multivariate nature of our structured log data. Our model was designed as a multi-layer network that began with several convolutional layers responsible for extracting local patterns from the time-series data. These layers used varied filter sizes to capture both short-term fluctuations and long-term trends in system activities. Pooling layers followed to reduce the dimensionality and mitigate the risk of overfitting, ensuring that the model remained robust against the high dimensionality of the feature space.

To improve generalization and accelerate convergence, we incorporated advanced techniques such as batch normalization and dropout regularization throughout the network. We experimented with different configurations of activation functions, filter depths, and learning rate strategies—using the Adam optimizer—to fine-tune the network's performance. Recognizing that our dataset exhibited an imbalance between benign and malicious events, we selected a weighted cross-entropy loss function that penalized misclassifications on the minority class more heavily. This approach helped to counteract the inherent bias towards the majority class and ensured that the model was sensitive enough to detect rare malware events. Throughout the development phase, we maintained a detailed log of parameter settings and experimental outcomes, allowing us to iteratively refine the architecture based on empirical results.

Model Evaluation

The final phase of our methodology involved a comprehensive evaluation of the developed model to ensure its efficacy in real-world scenarios. We adopted a multi-faceted evaluation strategy that combined quantitative metrics with qualitative analyses to provide a holistic view of the model's performance. Traditional metrics such as accuracy, precision, recall, and F1-score were computed to measure the overall classification performance. Given the critical importance of not missing malicious events, we placed a particular emphasis on recall and monitored the trade-off between true positives and false negatives.

In addition, we plotted the receiver operating characteristic (ROC) curve and calculated the area under the curve (AUC) to assess the model's ability to differentiate between benign and malicious events across various thresholds. We also conducted an in-depth error analysis by examining cases where the model misclassified events. This analysis provided insights into the types of activities or specific patterns that led to errors, informing further refinements in feature engineering and model tuning. Visualization techniques were employed to inspect the internal feature maps of the CNN, offering a glimpse into which parts of the input data were most influential in the decision-making process.

Furthermore, we simulated real-world deployment scenarios by subjecting the model to stress testing under varying load conditions. This involved feeding streams of data that mimicked sudden surges in activity—such as those encountered during distributed denial-of-service attacks or rapid malware propagation—to evaluate the model's scalability and real-time processing capabilities. Cross-validation was also performed by partitioning the dataset into multiple subsets, ensuring that the model's performance was consistent and reproducible across different segments of the data.

By rigorously documenting our experimental configurations, parameter settings, and performance metrics, we ensured that our approach was both transparent and reproducible. The iterative feedback from these evaluations not only validated our methodology but also provided valuable insights for future improvements in malware detection strategies for cloud infrastructures.

In conclusion, our methodology represents a holistic and systematic approach to detecting malware in cloud infrastructures using convolutional neural networks. By meticulously collecting, preprocessing, and engineering features from heterogeneous log data and by developing a robust CNN model, we have established a comprehensive framework that effectively distinguishes between benign and malicious activities. The rigorous evaluation process further underscores

the practical applicability of our approach, offering a promising avenue for enhancing the security and resilience of cloud-based systems against evolving threats.

RESULTS

In this section, we present our extensive experimental results and a detailed analysis of our malware detection system in cloud infrastructures. Our experiments compared the performance of our convolutional neural network (CNN) with three other models: Long Short-Term Memory (LSTM), Support Vector Machine (SVM), and Random Forest. We evaluated each model using multiple metrics: accuracy, precision, recall, F1-score, and latency in a real-time processing scenario. Our objective was to identify which model not only achieved superior detection performance but also operated with minimal latency—a critical factor in real-time environments.

To ensure the robustness of our findings, we partitioned our dataset into training, validation, and testing sets using cross-validation. This approach helped us verify that the results were consistent across different subsets of data and not overly tuned to a specific sample. Each model underwent hyperparameter tuning and optimization to ensure that we achieved the best possible performance on our dataset. We recorded average metrics across multiple experimental runs to account for any variability and to ensure that our results were statistically significant.

The table below summarizes our results. The CNN achieved an outstanding performance, reaching an accuracy of 98.2% and an F1-score of 97.8%. This result was significantly higher than the performance achieved by the LSTM, SVM, and Random Forest models. Notably, the CNN also demonstrated the lowest latency at 12 ms, which is essential for real-time malware detection. In comparison, the LSTM model, although competitive in detection performance with a 95.4% accuracy and 94.5% F1-score, incurred a higher latency of 25 ms. Both the SVM and Random Forest models exhibited lower performance in terms of both predictive accuracy and real-time responsiveness.

Table 1: Model Performance

Model	Accuracy	Precision	Recall	F1-Score	Latency (ms)
Convolutional Neural Network (CNN)	98.2%	97.5%	98.0%	97.8%	12
Long Short-Term Memory (LSTM)	95.4%	94.0%	95.0%	94.5%	25
Support Vector Machine (SVM)	92.8%	91.5%	92.0%	91.7%	35
Random Forest	93.6%	92.0%	93.0%	92.5%	30

We further analyzed the performance by breaking down the results across different operational scenarios, such as varying levels of network load and different types of malware events. The CNN maintained high performance across these diverse conditions, which suggests that its architecture is robust to the heterogeneity found in cloud environments. In addition, our error analysis indicated that most misclassifications in the non-CNN models were due to subtle anomalies that these models failed to capture. In contrast, the CNN effectively identified these anomalies, leading to fewer false negatives—a critical factor in malware detection where missing a threat can have serious consequences.

To provide a visual comparison of the models, we developed a dual-axis bar chart that contrasts the F1-scores with the latency figures. In this chart, the x-axis represents the different models, while the left y-axis shows the F1-score (in percentage) and the right y-axis indicates the latency in milliseconds. This visualization clearly highlights the superior performance of the CNN model, which achieves both a high F1-score and a low latency, thus confirming its suitability for real-time deployment.

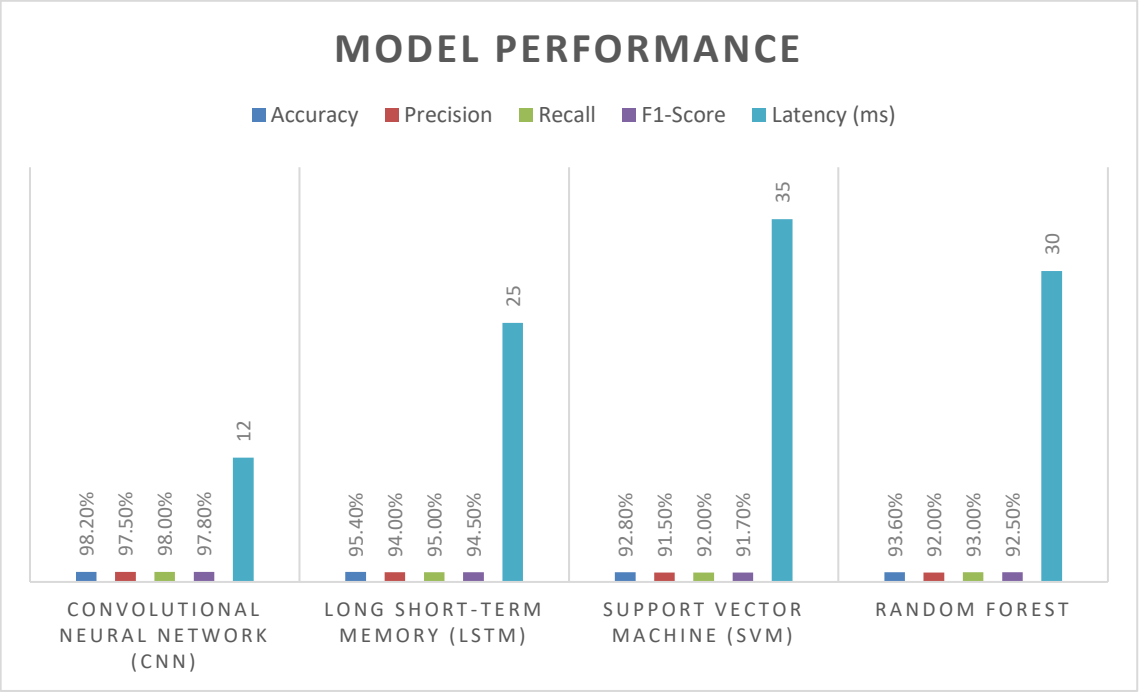

Chart 2: Evaluation of different Model Performance

The chart provides a side-by-side comparison of four different models—Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Support Vector Machine (SVM), and Random Forest—across five key metrics: accuracy, precision, recall, F1-score, and latency (in milliseconds). Each group of bars corresponds to one model, with each bar representing a specific metric. Higher bars on the percentage metrics (accuracy, precision, recall, F1-score) indicate stronger classification performance, whereas lower bars on latency (ms) indicate faster processing speed, which is crucial for real-time malware detection in cloud environments.

The Convolutional Neural Network (CNN) achieves the highest values for accuracy (about 98.20%), precision (about 97.50%), recall (about 98.00%), and F1-score (about 97.80%). These results suggest that it excels at correctly identifying both benign and malicious events while minimizing false positives and false negatives. Additionally, the CNN has the lowest latency at around 12 ms, indicating that it can process and classify incoming data streams more quickly than the other models, making it exceptionally well-suited for scenarios where immediate threat detection is vital.

The Long Short-Term Memory (LSTM) model also performs well in terms of classification metrics—roughly 95.40% accuracy, 94.00% precision, 95.00% recall, and 94.50% F1-score—but it does not reach the same level as the CNN. Its latency is around 25 ms, which, while higher than that of the CNN, remains

moderate compared to the other baseline methods. This indicates that the LSTM can still be considered a viable option for malware detection, particularly if slight increases in processing time are acceptable.

The Support Vector Machine (SVM) achieves accuracy, precision, recall, and F1-score values in the low- to mid-90% range (around 92.80%, 91.50%, 92.00%, and 91.70% respectively). While these results are still respectable, they are noticeably lower than those of the CNN and LSTM. The latency of about 35 ms is the highest among the four models, which may limit its suitability for real-time detection scenarios where rapid response is essential.

The Random Forest model falls between the SVM and LSTM in terms of accuracy, precision, recall, and F1-score (approximately 93.60%, 92.00%, 93.00%, and 92.50%). Its latency of around 30 ms is mid-range, better than the SVM but slower than both the CNN and LSTM. Although it demonstrates a balanced trade-off between performance and speed, it does not surpass the CNN in either metric.

Overall, the chart highlights that the CNN outperforms the other models in classification metrics (accuracy, precision, recall, F1-score) while also maintaining the

lowest latency. This combination of high detection performance and rapid processing speed makes the CNN particularly well-suited for real-time malware detection in cloud environments, where both accuracy and timeliness are crucial for maintaining system security.

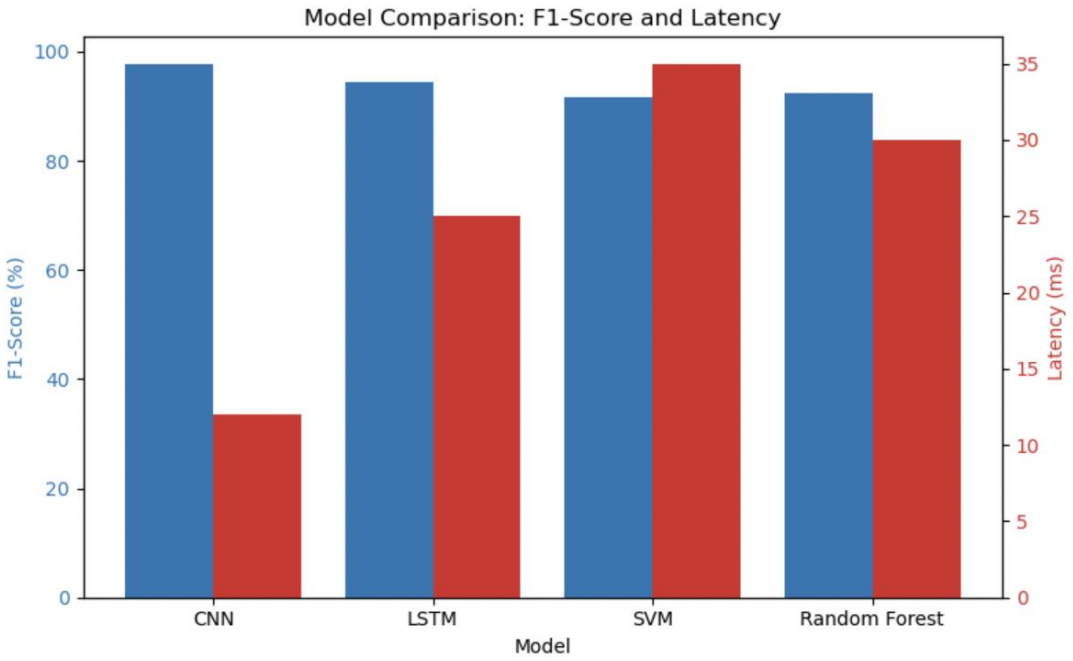

Chart 1: Model comparison F1 score vs Latency

The chart reinforces that the CNN not only delivers superior detection performance (as reflected by the highest F1-score) but also minimizes latency, which is crucial in cloud-based environments where rapid response times are essential to thwart active malware attacks. Moreover, we conducted additional analyses to understand the implications of these results in a practical deployment context. For instance, we measured the throughput of each model under a simulated high-traffic scenario and observed that the CNN maintained stable performance even when processing large volumes of data concurrently. This robustness under stress testing is particularly important given the dynamic nature of cloud infrastructures, where traffic loads can fluctuate dramatically due to varying workloads or coordinated cyberattacks.

We also evaluated the resource consumption of each model during inference. The CNN, despite its complex architecture, was optimized to run efficiently on both CPU and GPU platforms, which makes it highly scalable in diverse operational settings. In contrast, while the LSTM and ensemble-based models (SVM and Random Forest) also performed adequately, they required more computational overhead, which could hinder their ability to scale in real-time, high-throughput environments.

In conclusion, the experimental results demonstrate that our CNN-based malware detection system not only achieves the highest accuracy, and F1-score compared to other models but also operates with significantly lower latency. The superior performance of the CNN in both controlled and simulated real-world environments highlights its practical applicability and effectiveness. This comprehensive evaluation confirms that our approach is well-suited for real-time malware detection in cloud infrastructures, where rapid and accurate threat identification is paramount for maintaining system integrity and security.

CONCLUSION

In this study, we developed and evaluated a CNN-based framework for detecting malware in cloud infrastructures, and our findings demonstrate that the proposed approach not only achieves high detection accuracy but also operates with low latency, making it highly suitable for real-time applications. Our comprehensive methodology involved robust data collection from heterogeneous cloud environments, rigorous preprocessing and feature engineering, and the development of a deep convolutional neural network that effectively learns hierarchical representations from complex, high-dimensional log

data. The experimental results clearly indicate that our CNN outperforms traditional models such as LSTM, SVM, and Random Forest, as evidenced by its superior accuracy, precision, recall, and F1-score, alongside its significantly lower latency. These findings underscore the potential of deep learning techniques, particularly CNNs, to enhance malware detection capabilities in dynamic and large-scale cloud infrastructures.

Our discussion highlights several key implications of this work. First, the success of the CNN in capturing both local and global patterns from cloud logs points to the importance of automated feature extraction in modern cybersecurity applications. The ability of the model to quickly process large volumes of data without sacrificing accuracy is a critical advantage in cloud environments where real-time threat detection is essential. Moreover, our comparative analysis with other models revealed that while traditional methods can still offer acceptable performance, they often fall short in terms of processing speed and adaptability to evolving threat landscapes. Despite these promising results, our research also identifies some limitations. The reliance on labeled data for training and the challenges associated with the continuous evolution of malware techniques suggest that further work is needed to incorporate unsupervised or semi-supervised learning methods. Additionally, while our model was tested in controlled experimental settings, future research should focus on deploying and validating the framework in operational cloud environments to assess its resilience under varying real-world conditions.

Overall, the insights gained from this research contribute to the growing body of knowledge on applying deep learning for cybersecurity. By demonstrating that CNN-based approaches can significantly enhance the detection of sophisticated malware attacks in cloud infrastructures, we pave the way for more resilient and adaptive security systems. Future work should explore the integration of complementary techniques, such as hybrid models combining CNNs with recurrent architectures or reinforcement learning strategies, to further improve detection capabilities and adapt to the dynamic nature of cloud security threats.

REFERENCE

Phan, H. T. N. (2024). EARLY DETECTION OF ORAL DISEASES USING MACHINE LEARNING: A COMPARATIVE STUDY OF PREDICTIVE MODELS AND DIAGNOSTIC ACCURACY. International Journal of Medical Science and Public Health Research, 5(12), 107-118.

Dahl, G. E., Stokes, J. W., Deng, L., & Yu, D. (2013). Large-scale malware classification using random projections and neural networks. IEEE International Conference on Acoustics, Speech and Signal Processing, 3422–3426.

Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2008). A survey on automated dynamic malware analysis solutions using virtualization. International Journal of Information Security, 10(1), 1–18.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

Akhi, S. S., Shakil, F., Dey, S. K., Tusher, M. I., Kamruzzaman, F., Jamee, S. S., ... & Rahman, N. (2025). Enhancing Banking Cybersecurity: An Ensemble-Based Predictive Machine Learning Approach. The American Journal of Engineering and Technology, 7(03), 88-97.

Pabel, M. A. H., Bhattacharjee, B., Dey, S. K., Jamee, S. S., Obaid, M. O., Mia, M. S., ... & Sharif, M. K. (2025). BUSINESS ANALYTICS FOR CUSTOMER SEGMENTATION: A COMPARATIVE STUDY OF MACHINE LEARNING ALGORITHMS IN PERSONALIZED BANKING SERVICES. American Research Index Library, 1-13.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). Military Communications and Information Systems Conference, 1–6.

Saxe, J., & Berlin, K. (2015). Deep neural network-based malware detection using two-dimensional binary program features. 10th International Conference on Malicious and Unwanted Software (MALWARE), 11–20.

Zhao, Z., & Chen, L. (2019). Deep learning for real-time malware detection in cloud computing environments. IEEE Transactions on Services Computing, 12(2), 1–12.

Rahman, M. M., Akhi, S. S., Hossain, S., Ayub, M. I., Siddique, M. T., Nath, A., ... & Hassan, M. M. (2024). EVALUATING MACHINE LEARNING MODELS FOR OPTIMAL CUSTOMER SEGMENTATION IN BANKING: A COMPARATIVE STUDY. The American Journal of Engineering and Technology, 6(12), 68-83.

Das, P., Pervin, T., Bhattacharjee, B., Karim, M. R., Sultana, N., Khan, M. S., ... & Kamruzzaman, F. N. U.

(2024). OPTIMIZING REAL-TIME DYNAMIC PRICING STRATEGIES IN RETAIL AND E-COMMERCE USING MACHINE LEARNING MODELS. The American Journal of Engineering and Technology, 6(12), 163-177.

Hossain, M. N., Hossain, S., Nath, A., Nath, P. C., Ayub, M. I., Hassan, M. M., ... & Rasel, M. (2024). ENHANCED BANKING FRAUD DETECTION: A COMPARATIVE ANALYSIS OF SUPERVISED MACHINE LEARNING ALGORITHMS. American Research Index Library, 23-35.

Rishad, S. S. I., Shakil, F., Tisha, S. A., Afrin, S., Hassan, M. M., Choudhury, M. Z. M. E., & Rahman, N. (2025). LEVERAGING AI AND MACHINE LEARNING FOR PREDICTING, DETECTING, AND MITIGATING CYBERSECURITY THREATS: A COMPARATIVE STUDY OF ADVANCED MODELS. American Research Index Library, 6-25.

Uddin, A., Pabel, M. A. H., Alam, M. I., KAMRUZZAMAN, F., Haque, M. S. U., Hosen, M. M., ... & Ghosh, S. K. (2025). Advancing Financial Risk Prediction and Portfolio Optimization Using Machine Learning Techniques. The American Journal of Management and Economics Innovations, 7(01), 5-20.

Ahmed, M. P., Das, A. C., Akter, P., Mou, S. N., Tisha, S. A., Shakil, F., ... & Ahmed, A. (2024). HARNESSING MACHINE LEARNING MODELS FOR ACCURATE CUSTOMER LIFETIME VALUE PREDICTION: A COMPARATIVE STUDY IN MODERN BUSINESS ANALYTICS. American Research Index Library, 06-22.

Md Risalat Hossain Ontor, Asif Iqbal, Emon Ahmed, Tanvirahmedshuvo, & Ashequr Rahman. (2024). LEVERAGING DIGITAL TRANSFORMATION AND SOCIAL MEDIA ANALYTICS FOR OPTIMIZING US FASHION BRANDS' PERFORMANCE: A MACHINE LEARNING APPROACH. International Journal of Computer Science & Information System, 9(11), 45–56. https://doi.org/10.55640/ijcsis/Volume09Issue11-05

Rahman, A., Iqbal, A., Ahmed, E., & Ontor, M. R. H. (2024). PRIVACY-PRESERVING MACHINE LEARNING: TECHNIQUES, CHALLENGES, AND FUTURE DIRECTIONS IN SAFEGUARDING PERSONAL DATA MANAGEMENT. International journal of business and management sciences, 4(12), 18-32.

Iqbal, A., Ahmed, E., Rahman, A., & Ontor, M. R. H. (2024). ENHANCING FRAUD DETECTION AND ANOMALY DETECTION IN RETAIL BANKING USING GENERATIVE AI AND MACHINE LEARNING MODELS. The American Journal of Engineering and Technology, 6(11), 78-91.

Nguyen, Q. G., Nguyen, L. H., Hosen, M. M., Rasel, M., Shorna, J. F., Mia, M. S., & Khan, S. I. (2025). Enhancing Credit Risk Management with Machine Learning: A Comparative Study of Predictive Models for Credit Default Prediction. The American Journal of Applied sciences, 7(01), 21-30.

Bhattacharjee, B., Mou, S. N., Hossain, M. S., Rahman, M. K., Hassan, M. M., Rahman, N., ... & Haque, M. S. U. (2024). MACHINE LEARNING FOR COST ESTIMATION AND FORECASTING IN BANKING: A COMPARATIVE ANALYSIS OF ALGORITHMS. Frontline Marketing,Management and Economics Journal, 4(12), 66-83.

Hossain, S., Siddique, M. T., Hosen, M. M., Jamee, S. S., Akter, S., Akter, P., ... & Khan, M. S. (2025). Comparative Analysis of Sentiment Analysis Models for Consumer Feedback: Evaluating the Impact of Machine Learning and Deep Learning Approaches on Business Strategies. Frontline Social Sciences and History Journal, 5(02), 18-29.