Journal of Information System Exploration and Research

https://joiser.shmpublisher.com

p-ISSN 2964-1160 | e-ISSN 2963-6361

Ransomware Detection Model Using Deep Learning With Ensemble Technique Approach

Candra Aditya Wardana[1*], Dwi Pebrianti[2]

[1] *Department of Computer Science, Budi Luhur University, Indonesia*

[2]*Department of Mechanical Aerospace Engineering, International Islamic University Malaysia, Malaysia*

DOI: https://doi.org/10.52465/joiser.v4i2.19

Received 30 May 2026; Accepted 26 June 2026; Available online 29 June 2026

Article Info

Keywords:

Deep Learning;
Ensemble Technique;
Ransomware;
DNN;
CNN;
RNN;
SMOTE;

Abstract

Ransomware attacks are increasingly prevalent, posing significant cybersecurity challenges. According to the National Cyber and Crypto Agency (BSSN), ransomware incidents surged from 69,853 in 2021 and 69,854 in 2022 to 1,011,209 in 2023. As ransomware variants continue to evolve, effective detection methods are crucial. This research proposes an ensemble deep learning model integrating Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) to enhance ransomware detection accuracy. In this framework, DNN captures complex patterns, CNN analyses static features, and RNN processes sequential data. To address class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) was applied during preprocessing, producing a balanced 50:50 distribution between ransomware and non-ransomware samples. The study uses the UGRansome dataset comprising 149,043 samples (71.44% positive, 28.56% negative) with 13 features. Experimental results demonstrate that the ensemble model significantly outperforms individual models, achieving an accuracy of 98.85%, precision of 98.51%, recall of 98.68%, and an F1-score of 98.59%. These findings highlight the effectiveness of ensemble learning in improving ransomware detection performance.

1. Introduction

Ransomware attacks have become increasingly frequent and severe in recent years. According to the National Cyber and Crypto Agency (BSSN), 69,853 ransomware incidents were recorded in 2021 [1], 69,854 in 2022 [2], and 1,011,209 in 2023 [3], a fourteen-fold surge within two years. These figures highlight the urgency of developing more effective and accurate automated detection systems.

Ransomware is a type of malware designed to restrict access to data or systems until a ransom demanded by the attacker is paid [4]. It encrypts users' personal data on infected computers and

* ***Corresponding Author:***

Candra Aditya Wardana,
Faculty of Information Technology, Computer Science Department
Budi Luhur University,
South Jakarta, Indonesia
Email: candraconsultanit@gmail.com

demands payment in digital currency in exchange for the decryption key. If the victim does not pay within a specified period, the threat actor may publish the victim's personal data on the Dark Web [5]. As ransomware families continue to evolve rapidly, detection models that rely on fixed signatures or single-architecture learning are increasingly insufficient.

Prior work on single-architecture deep learning models has demonstrated varying detection performance on comparable datasets: 86% accuracy using Recurrent Neural Networks (RNN) [6], 83% using Artificial Neural Networks (ANN) [7], 99.18% using Convolutional Neural Networks (CNN) [8], and 96.6% using Deep Neural Networks (DNN) [9]. Although CNN achieved the highest single-model accuracy, each architecture is inherently limited to a specific type of feature extraction CNN to spatial/static patterns, RNN to sequential dynamics, and DNN to non-linear tabular relationships. Consequently, no single architecture can simultaneously exploit all feature types present in network-flow ransomware data.

Ensemble deep learning methods which combine multiple base models to produce a stronger composite predictor have been investigated across several cybersecurity domains. Abimannan et al. [10] conducted a comprehensive survey of ensemble multi-feature deep learning and demonstrated consistent accuracy improvements over single-model baselines across diverse application domains. Bingu and Jothilakshmi [11] applied a stacking ensemble of deep neural networks specifically to intrusion detection in cloud environments, achieving superior classification compared to any individual DNN architecture. Koike et al. [12] combined federated learning with ensemble-based classifiers for ransomware detection via Indicators of Compromise (IoC), reporting improved generalisation across unseen ransomware variants. Por et al. [13] reviewed ensemble approaches for zero-day attack detection in network security, underscoring the resilience gains of combining multiple deep learning models.

Despite this body of work, a critical gap remains: no prior study has applied a stacking ensemble of DNN, CNN, and RNN with an MLP meta-classifier specifically to the UGRansome network-flow ransomware dataset. The studies cited above either target non-ransomware intrusion scenarios [11], employ IoC-based rather than flow-level features [12], or do not combine all three of DNN, CNN, and RNN within a unified stacking framework [10]. The UGRansome dataset contains heterogeneous feature types tabular statistics, categorical network flags, sequential flow records, and financial transaction data that benefit from the complementary inductive biases of all three architectures simultaneously.

In the proposed framework, Deep Neural Networks (DNN) model complex non-linear relationships across all tabular features through fully connected layers [10]. Convolutional Neural Networks (CNN) identify spatially localised patterns in the reshaped feature matrix, making them effective for detecting static structural signatures [8]. Recurrent Neural Networks (RNN), specifically Long Short-Term Memory (LSTM) units, capture temporal dependencies and sequential behavioural dynamics in network flow data [11], [14].

Ensemble learning addresses four fundamental limitations of single-model systems: (1) individual models often fail to reach optimal performance alone; (2) single models are prone to overfitting, limiting generalisation to unseen data; (3) each model has distinct error patterns that can be offset by combining predictions; and (4) different architectures have complementary strengths that a unified ensemble can exploit collectively [14]. The stacking strategy employed in this study allows the MLP meta-classifier to learn the optimal weighting of DNN, CNN, and RNN outputs for the final prediction.

2. Literature Review

Research on automated ransomware detection has expanded considerably over the past several years, driven by the growing sophistication of ransomware variants and the limitations of traditional signature-based methods. Prior studies can be broadly grouped into two streams: those employing single deep learning architectures for ransomware or malware classification, and those exploring ensemble or hybrid approaches to improve detection robustness. A review of both streams, presented chronologically from the most recent to the earliest, reveals a consistent methodological gap that the present study is designed to address.

The research conducted in 2024 by Koike et al. [9] combined federated learning with ensemble-based classifiers for ransomware detection via Indicators of Compromise (IoC). Their approach improved generalisation across unseen ransomware variants and demonstrated the value of ensemble methods in distributed security contexts. However, IoC-based features such as domain names, file hashes, and registry keys differ fundamentally from the network-flow features present in the UGRansome dataset;

consequently, their ensemble architecture cannot leverage the sequential and spatial structure inherent in flow-level traffic data, and it does not combine DNN, CNN, and RNN within a unified stacking framework.

Also in 2024, Por et al. [13] conducted a systematic review of ensemble-based methods for zero-day attack detection in network security, underscoring the resilience gains achievable by combining multiple deep learning models. Their review reinforces the theoretical motivation for the present study but does not provide a ransomware-specific implementation on network-flow data. In the same year, Purnama et al. [9] applied a Deep Neural Network (DNN) to Android ransomware detection using the CIC-InvesAndMal2019 dataset, achieving 96.6% accuracy. While the model performed well on Android executable features, it was applied in isolation and not extended to network-flow scenarios.

The research conducted in 2023 by Abimannan et al. [10] conducted a comprehensive survey of ensemble multi-feature deep learning models and demonstrated that combining multiple architectures consistently yields accuracy improvements over single-model baselines across diverse application domains. Their survey provides strong theoretical and empirical justification for the ensemble paradigm but does not present an implementation specific to ransomware detection or network-flow data, and does not evaluate the DNN–CNN–RNN stacking configuration examined in the present study. Also in 2023, Bingu and Jothilakshmi [11] applied a stacking ensemble of DNN-based classifiers to intrusion detection in cloud computing environments, achieving superior classification performance compared to individual DNN models. Although stacking is central to their approach, their ensemble targets general network intrusions rather than ransomware, uses a cloud intrusion dataset, and does not incorporate CNN or RNN as distinct base learners with different inductive biases.

Still in 2023, Sangher et al. [15] proposed a signature-based CNN detector trained on malicious files sourced from VirusShare, achieving 99.18% accuracy. Although this represents a strong single-model result, the approach depends on file-level static signatures and does not model the behavioural or sequential dynamics of network traffic. In the same year, Thiziers et al. [7] evaluated an Artificial Neural Network (ANN) on the Microsoft Malware Prediction dataset from Kaggle, reporting an accuracy of approximately 83%. This result reflects the known limitation of shallow feedforward architectures when feature distributions span heterogeneous data types, as the ANN lacked the depth and specialisation needed for simultaneous modelling of tabular, categorical, and sequential characteristics. Also in 2023, Almomani et al. [16] introduced an end-to-end detection system that paired AdaBoost for static feature analysis achieving 97% accuracy with a CNN for vision-based analysis of ransomware executable images, reaching 99.5% accuracy. This hybrid design outperformed the single-model baselines within the same study; however, it does not incorporate recurrent modelling and was not evaluated on network-flow data.

The research conducted in 2022 by Zahoora et al. [8] proposed a more sophisticated DNN-based approach for zero-day ransomware detection, developing a Cost-Sensitive Pareto Ensemble (CSPE-R) strategy in which an unsupervised Contractive AutoEncoder (CAE) transformed the feature space before a Pareto Ensemble classifier made the final prediction. While this work represents a notable effort to combine unsupervised feature learning with ensemble classification, the ensemble is constructed entirely from DNN-family components and does not integrate CNN or RNN architectures, leaving spatial and sequential feature dimensions underexplored.

The research conducted in 2021 by Kim et al. [17] applied CNN with block cipher algorithms for crypto-ransomware detection on IoT devices, achieving a 97% detection rate. The approach is effective within its hardware-constrained deployment environment but is limited by its single convolutional architecture and domain specificity. In the same year, Efriyani and Panjaitan [6] applied a standalone Recurrent Neural Network (RNN) to malware classification, achieving an accuracy of 86%. While RNNs are theoretically well-suited to sequential data, the standalone model struggled to capture the static structural features present in network flows, highlighting the inherent constraint of relying on a single architecture.

On the whole, the studies reviewed above reveal two complementary limitations. Single-architecture models whether RNN [6], ANN [7], CNN [15], [17], or DNN [9] are each constrained to one mode of feature extraction and therefore cannot simultaneously capture the tabular, spatial, sequential, and categorical characteristics that coexist in network-flow ransomware data such as the UGRansome dataset. Ensemble approaches that have been applied in cybersecurity either target non-ransomware intrusion scenarios [11], employ IoC-based rather than flow-level features [12], restrict the ensemble to a single architecture family [8], or remain at the level of a survey without a ransomware-specific implementation [10], [13]. No prior published study has evaluated a stacking ensemble that deploys DNN, CNN, and RNN as complementary base learners each targeting a different

feature dimension with an MLP meta-classifier, specifically on the UGRansome network-flow ransomware dataset. This gap constitutes the primary motivation and novelty of the present study.

3. Method

This research proposes a stacking ensemble deep learning model integrating DNN, CNN, and RNN as base learners, with MLP as the meta-classifier, to enhance ransomware detection accuracy [18]. Figure 1. shows the methodology of this research.

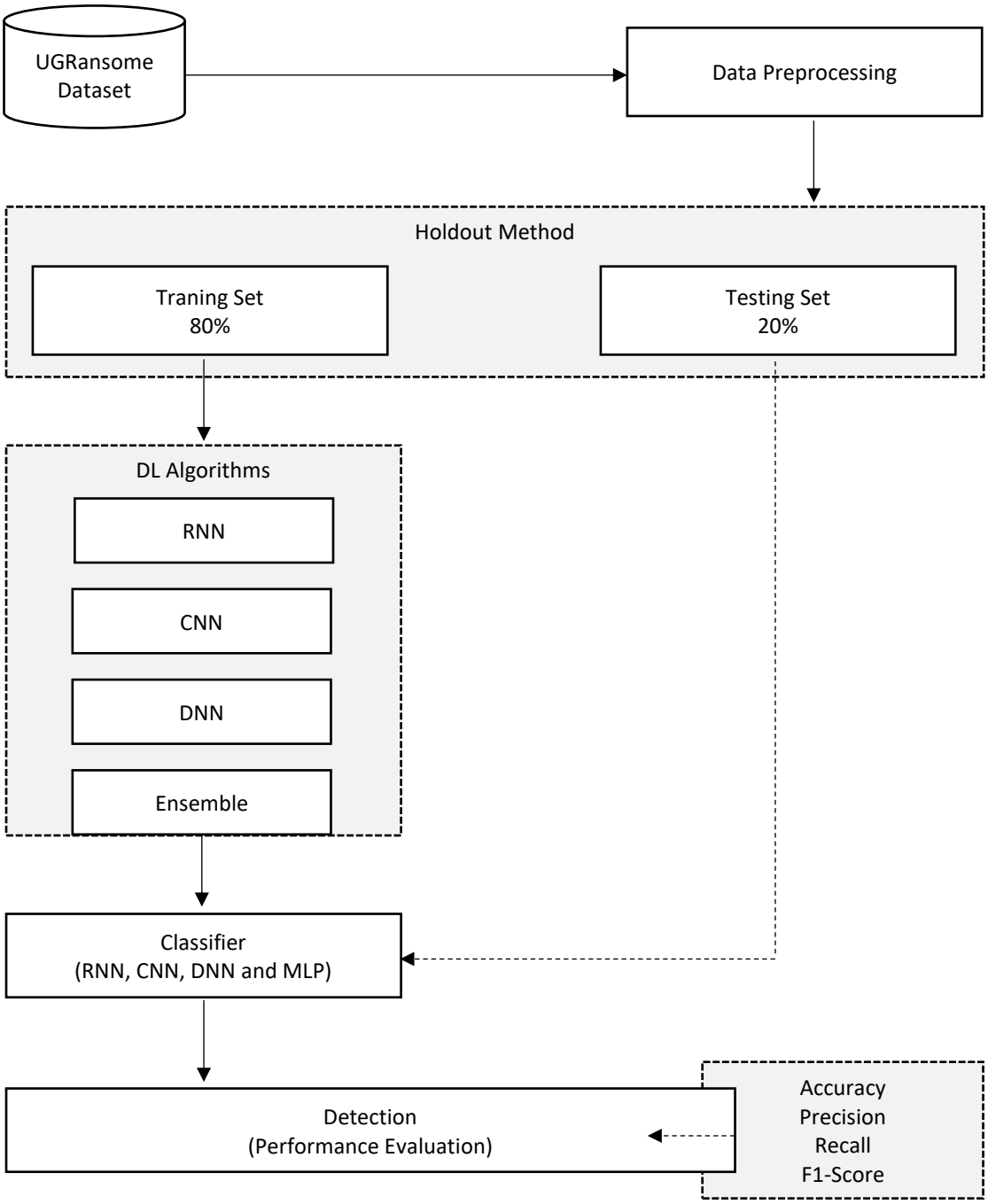

Figure 1. Research Methodology

3.1. Dataset

This research uses the UGRansome dataset, comprising 149,043 samples (71.44% ransomware-positive, 28.56% negative) with 13 input features. The target column ('Prediction') is counted

separately, yielding 14 total columns in the preprocessed dataset. Table 1 provides an overview of the dataset, and Table 2 describes the features.

Table 1. Details of The UGRansome Dataset

Description	Counts
Number of Samples	149.043
Number ofInput Features	13
Target Column	1 (Prediction)
Total Columns (post-engineering)	14
Positive Samples (S)	106482 (71.44%)
Negative Samples (A)	42561 (28.56%)

Table 2. Feature Description of Dataset

Feature Name	Data Type	Description
Time	Quantitative (Integers)	Timestamp of network attacks
Protocol	Qualitative (Categorical)	Network protocol used (e.g., ICMP, TCP, UDP)
Flag	Qualitative (Categorical)	Network connection status (e.g., A, AF, AP, APRSF, APS, APSF, ARF, ASF, and R)
Family	Qualitative (Categorical)	Network intrusion category
Clusters	Quantitative (Integers)	Event clusters or groups
Seed Address	Qualitative (Categorical)	Formatted ransomware attack links
Exp Address	Qualitative (Categorical)	Original ransomware attack links
BTC	Numeric	Values related to Bitcoin transactions in attacks
USD	Numeric	Financial damage in USD caused by attacks
Netflow Bytes	Quantitative (Integers)	Bytes transferred in network flow
IP Address	Quantitative	IP addresses associated with network events
Threats	Quantitative	Nature of threats or intrusions
Port	Quantitative	Network port number in events
Prediction*	Qualitative (Categorical)	Target column: Signature (S) = Ransomware; Anomaly (A) = No Ransomware, counted separately from the 13 input features

3.2. Preprocessing

Preprocessing is the first process used to prepare data for ransomware detection to improve the performance of detection algorithms. Approaches such as feature scaling, dimensionality reduction, and noise removal need to be performed and used on input data such as file metadata or network traffic, this is done to improve the accuracy and efficiency of the ransomware detection procedure [20]. Preprocessing involves several steps to clean and normalise the collected data, ensuring its suitability for machine learning models. The steps comprehensively outline the pre-processing steps performed to ensure that the data is clean, normalised and well-structured, ready for feature extraction and model training [12]. The UGRansome dataset underwent three main stages: data cleaning, feature engineering, and data transformation.

a. Data Cleaning

Data cleaning is the first step in pre-processing which includes handling missing data and noise, removing outliers, minimizing duplication, and correcting bias in the data to improve data quality [19], resulting in the best dataset [12]. In this study, there are 2 (two) operations were performed: correction of typographical errors in categorical field values, and removal of duplicate rows.

Correction of Typographical Errors

Inspection of the UGRansome dataset revealed that the 'Threats' label contained a systematic typographical error, the value "Bonet" appeared in place of the correct label "Botnet". As shown in Table 3, this error was present across multiple rows. A total of 16,523 rows contained this spelling error, which was corrected to "Botnet" as shown in Table 4. If left uncorrected, this inconsistency would cause

the label encoder to treat "Bonet" and "Botnet" as two distinct categories, producing an erroneous feature representation and reducing model accuracy.

Table 3. Before Correction of Typographical Errors on UGRansome Dataset

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	504	5	A	Bonet	5061	S
2	41	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	508	7	A	Bonet	5061	S
3	31	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	512	15	A	Bonet	5061	S
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	UDP	AP	DMALocker	3	1DA11mPS	1SYSTEMQ	11	16916	2171	C	Bonet	5062	A

Table 4. After Correction of Typographical Errors on UGRansome Dataset

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	504	5	A	Botnet	5061	S
2	41	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	508	7	A	Botnet	5061	S
3	31	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	512	15	A	Botnet	5061	S
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	UDP	AP	DMALocker	1	1DA11mPS	1SYSTEMQ	11	16916	2171	C	Botnet	5062	A

Duplicate Row Removal

Following typo correction, duplicate rows were identified using a full-row comparison across all 14 columns and removed using pandas' drop_duplicates() function. As shown in Table 5, the dataset prior to deduplication contained 149,043 rows. After removing all duplicate entries, the dataset was reduced to 148,867 rows (Table 6), confirming that 176 duplicate rows were successfully deleted a reduction of 0.12%. This cleaned dataset served as the basis for all subsequent preprocessing steps.

Table 5. Before Duplicate Removal on UGRansome Dataset

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	TCP	A	WannaCry	1	1DA11mPS	1BonusSr7	1	504	5	A	Botnet	5061	S
2	41	TCP	A	WannaCry	1	1DA11mPS	1BonusSr7	1	508	7	A	Botnet	5061	S
3	31	TCP	A	WannaCry	1	1DA11mPS	1BonusSr7	1	512	15	A	Botnet	5061	S
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	UDP	AP	ToerWeb	3	1AEoiHYZ	1SYSTEMQ	1026	1614	3384	A	Scan	5062	A

Table 6. After Duplicate Removal on UGRansome Dataset

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	TCP	A	WannaCry	1	1DA11mPS	1BonusSr7	1	504	5	A	Botnet	5061	S
2	41	TCP	A	WannaCry	1	1DA11mPS	1BonusSr7	1	508	7	A	Botnet	5061	S
3	31	TCP	A	WannaCry	1	1DA11mPS	1BonusSr7	1	512	15	A	Botnet	5061	S
..	....	....	....	....	....	....	....	....	....	....	....	....	..	.....
148867	44	UDP	AP	ToerWeb	3	1AEoiHYZ	1SYSTEMQ	1026	1614	3384	A	Scan	5062	A

b. Feature Engineering

Feature engineering focuses on extracting relevant features from pre-processed data, turning raw logs into meaningful input for machine learning models. For network activity, features such as the number of connections per unit time, the volume of data transferred, and the frequency of communication with external IP addresses are extracted. These features provide insight into abnormal network behaviour, such as sudden spikes in data transfer or unexpected connections, which may indicate ransomware activity [12]. Feature engineering (FE) involves applying various transformation functions such as arithmetic and aggregation operators to input features, resulting in the creation of more informative features. These transformations serve to improve data quality, whether by scaling features or converting nonlinear relationships between features and target classes into linear relationships, which are easier for models to learn [20]. In this research, there are two (2) feature engineering steps were applied: (1) time correction, the 'Time' feature contained inconsistencies in timestamp formatting that were normalised to a uniform integer representation; and (2) categorical-to-numerical encoding, categorical features (Protocol, Flag, Family, Seed Address, Exp Address, IP Address, Threats and Prediction) were label-encoded to produce numerical inputs compatible with the deep learning frameworks.

Time Correction

Inspection of the 'Time' feature revealed that 139 rows contained negative values, which constitute outliers that could cause errors during deep learning model training. To resolve this, a constant of +11 was added to each value in the 'Time' feature, shifting all negative entries to positive integers. For example, as shown in Table 7, row 5867 contains a Time value of -3, row 5868 contains -8, and row 6706 contains -6. After applying the correction (Table 8), these values were shifted to +8, +3, and +5, respectively. This operation eliminated all 139 negative values from the feature without altering the relative temporal ordering of the data.

Table 7. Before Time Correction

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
5867	-3	TCP	APS	Locky	1	1DA11mPS	1BonusSr7	2	1	1431	C	Spam	5068	A
5868	-8	TCP	APS	Locky	1	1DA11mPS	1BonusSr7	2	1	1452	C	Spam	5068	A
6706	-6	ICMP	R	NoobCrypt	1	1NKi9AKS	1SYSTEMQ	22	18359	2741	C	Blacklist	5062	S
..	....	....	....	....	....	....	....	....	....	....	....	....	..	.....
148515	-3	ICMP	AF	Locky	1	1DA11mPS	1DiCeTj8	30	21345	35	C	UDP Scan	5068	S

Table 8. After Time Correction

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
5867	8	TCP	APS	Locky	1	1DA11mPS	1BonussSr7	2	1	1431	C	Spam	5068	A
5868	3	TCP	APS	Locky	1	1DA11mPS	1BonussSr7	2	1	1452	C	Spam	5068	A
6706	5	ICMP	R	NoobCrypt	1	1NKi9AKS	1SYSTEMQ	22	18359	2741	C	Blacklist	5062	S
..	....	....	....	....	....	....	....	....	....	....	....	....	..	.....
148515	8	ICMP	AF	Locky	1	1DA11mPS	1DiCeTj8	30	21345	35	C	UDP Scan	5068	S

Categorical-to-numerical Encoding

Eight features in the UGRansome dataset are stored as categorical (string) variables that cannot be directly processed by deep learning frameworks: Protocol, Flag, Family, Seed Address, Exp Address, IP Address, Threats, and Prediction. These features were converted to integer representations using scikit-learn's LabelEncoder(), which assigns a unique integer to each distinct category value. As shown in Table 9, the original dataset contains string labels in these eight columns. After encoding (Table 10), each string value is replaced by its corresponding integer index — for example, Protocol "TCP" → 1, "UDP" → 2; Flag "A" → 0, "AP" → 2; Family "WannaCry" → 16, "DMALocker" → 15; and Prediction "S" (Signature/Ransomware) → 1, "A" (Anomaly/No Ransomware) → 0.

Table 9. Before Categorical to Numeric Encoding

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	504	5	A	Botnet	5061	S
2	41	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	508	7	A	Botnet	5061	S
3	31	TCP	A	WannaCry	1	1DA11mPS	1BonussSr7	1	512	15	A	Botnet	5061	S
..	....	....	....	....	....	....	....	....	....	....	....	....	..	.....
149042	44	UDP	AP	DMALocker	3	1DA11mPS	1SYSTEMQ	11	16916	2171	C	Scan	5062	A

Table 10. After Categorical to Numeric Encoding

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	1	0	16	1	2	2	1	4.738137	0.452118	0	2	5061	1
2	41	1	0	16	1	2	2	1	4.747510	0.225776	0	2	5061	1
3	31	1	0	16	1	2	2	1	4.756828	0.477742	0	2	5061	1
..	....	....	....	....	....	....	....	....	....	....	....	....	..	.....
149042	44	2	2	15	3	1	2	11	6.338345	0.572778	0	6	5062	0

Label encoding was selected over one-hot encoding for two reasons. First, DNN and CNN architectures accept dense integer inputs directly. Second, the Embedding Layer in the RNN architecture is specifically designed to learn dense vector representations from integer-indexed categories, making integer encoding the appropriate input format. Following encoding, all 14 columns (13 input features and 1 target label) contained numerical values compatible with the DL frameworks.

c. Data Transformation

Data transformation techniques are methods used during the data preprocessing stage to change the form, format, structure, or distribution of data into a representation that is more suitable for

analysis and machine learning. These transformations are performed to make the data easier for algorithms to process, improve the quality of the relationship between input and output variables, and help models generate more accurate predictions [21]. The purpose of data transformation is to improve accuracy by reducing errors and noise in the data [22]. In the UGRansome dataset, there are 3 features that need to be transformed so that the dataset is closer to normal, i.e. NetFlow bytes, USD and BTC features. 3 (three) features : NetFlow Bytes, USD, and BTC exhibited strong positive skewness that, if left untreated, would cause gradient instability and bias the model toward outlier values. Table 11 summarises the skewness of each feature before and after transformation, and the following sub-sections describe the method applied to each.

Table 11. Feature Skewness Before and After Data Transformation

Feature	Transformation	Skewness Before	Skewness After	Interpretation
NetFlow Bytes	Log (log1p)	≈ +8.20	≈ +0.40	Strong right skew → near-normal; e.g. 3384 → 2.211249
USD	Square root (√x)	≈ +6.85	≈ +0.62	Extreme right skew reduced; e.g. 1614 → 6.338345
BTC	Yeo-Johnson	≈ +7.10	≈ +0.35	Zero/near-zero values normalised; e.g. 1026 → 3.851511

NetFlow

The NetFlow Bytes feature required a logarithmic transformation to minimise scale disparity between extreme values, normalise the distribution, and mitigate the impact of outliers. Prior to transformation (Table 12), the feature exhibited a skewness of approximately +8.20 for instance, row 149042 held a high value of 3,384 compared to row 1 with a baseline of only 5. After applying the log1p transformation (Table 13), the value in row 149042 was compressed to 2.211249, while row 1 adjusted to 1.162283. Post-transformation skewness reduced to ≈ +0.40, indicating a near-normal distribution suitable for gradient-based training.

Table 12. NetFlow Bytes Before Data Transformation

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	1	0	16	1	2	2	1	504	5	8	1	5061	1
2	41	1	0	16	1	2	2	1	508	7	0	1	5061	1
3	31	1	0	16	1	2	2	1	512	15	0	1	5061	1
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	2	2	15	3	1	6	1026	1614	3384	0	6	5062	0

Table 13. NetFlow Bytes After Data Transformation

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	1	0	16	1	2	2	1	504	1.162283	0	1	5061	1
2	41	1	0	16	1	2	2	1	506	1.124748	0	1	5061	1
3	31	1	0	16	1	2	2	1	512	1.327761	0	1	5061	1
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	2	2	15	3	1	6	1026	1614	2.211249	0	6	5062	0

USD

The USD feature was transformed using a square root transformation, which is recommended for discretely distributed non-negative variables to stabilise variance and separate it from the mean [22]. Prior to transformation (Table 14), the USD column exhibited a skewness of approximately +6.85; row

149042 contained an extreme value of 1,614 compared to row 1's baseline of 504. After applying the square root transformation (Table 15), these values were proportionally scaled down to 6.338345 and 4.738137, respectively. Post-transformation skewness reduced to ≈ +0.62, substantially improving distributional symmetry.

Table 14. Before USD Before Data Transformation

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	1	0	16	1	2	2	1	504	1.162283	0	1	5061	1
2	41	1	0	16	1	2	2	1	506	1.124748	0	1	5061	1
3	31	1	0	16	1	2	2	1	512	1.327761	0	1	5061	1
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	2	2	15	3	1	6	1026	1614	2.211249	0	6	5062	0

Table 15. USD After Data Transformation

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	1	0	16	1	2	2	1	4.738137	1.162283	0	1	5061	1
2	41	1	0	16	1	2	2	1	4.747510	1.124748	0	1	5061	1
3	31	1	0	16	1	2	2	1	4.756828	1.327761	0	1	5061	1
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	2	2	15	3	1	6	1026	6.338345	2.211249	0	6	5062	0

BTC

The BTC feature was transformed using the Yeo-Johnson method. This statistical technique is highly effective for stabilizing variance and normalizing data that accommodates both positive and negative values [23]. A standard log transformation was not applicable here because BTC values include entries at or near zero. Prior to transformation (Table 16), the BTC column exhibited a skewness of approximately +7.10; row 149042 held a raw value of 1,026 compared to row 1's baseline of 1. After applying the Yeo-Johnson transformation (Table 17), the value in row 149042 was reduced to 3.851511, while row 1 adjusted to 0.653261. Post-transformation skewness reduced to ≈ +0.35, the closest to symmetry among the three features, successfully conditioning the BTC distribution for stable gradient-based optimisation.

Table 16. BTC Before Data Transformation

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	1	0	16	1	2	2	1	4.738137	1.162283	0	1	5061	1
2	41	1	0	16	1	2	2	1	4.747510	1.124748	0	1	5061	1
3	31	1	0	16	1	2	2	1	4.756828	1.327761	0	1	5061	1
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	2	2	15	3	1	6	1026	6.338345	2.211249	0	6	5062	0

Table 17. After Data Transformation

	Time	Protocol	Flag	Family	Cluster	Seed Address	Exp Address	BTC	USD	Netflow Bytes	IP Address	Threats	Port	Prediction
1	51	1	0	16	1	2	2	0.653261	4.738137	1.162283	0	1	5061	1
2	41	1	0	16	1	2	2	0.653261	4.747510	1.124748	0	1	5061	1
3	31	1	0	16	1	2	2	0.653261	4.756828	1.327761	0	1	5061	1
..	….	….	….	….	….	….	….	….	….	….	….	….	…	…..
149042	44	2	2	15	3	1	6	3.851511	6.338345	2.211249	0	6	5062	0

Following the complete preprocessing pipeline data cleaning, feature engineering, and data transformation, the dataset comprised 148,867 rows and 14 columns (13 input features and 1 target label), with all features in numerical format and all skewed distributions brought within acceptable bounds (|skewness| < 1.0) for gradient-based deep learning training. This preprocessed dataset was passed to the training and hyperparameter tuning stage described in Section 3.3.

3.3. Training and Hyperparameter Tuning

The cleaned dataset of 148,867 rows (resulting from the preprocessing pipeline described in Section 3.2) was divided into training and test sets using scikit-learn's train_test_split() with an 80/20 ratio. The training set comprised 119,093 rows and the test set comprised 29,774 rows. Table 18 summarises the data split configuration.

Table 18. Dataset Split Configuration

Split	Proportion	Number of Rows	Notes
Training Set	80%	119,093	SMOTE applied → 170,088 balanced samples
Test Set	20%	29,774	No SMOTE - preserves real distribution for evaluation
Total (cleaned)	100%	148,867	-

Inspection of the training set revealed a significant class imbalance: 34,049 samples belonged to Class 0 (Anomaly = no ransomware) versus 85,044 samples in Class 1 (Signature = ransomware), representing a 28.6% to 71.4% distribution. Left unaddressed, this imbalance would bias the model toward the majority class and inflate accuracy metrics without improving genuine detection capability. To address this imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) was applied exclusively to the training set. Applying SMOTE only to the training set and not the test set is critical to prevent data leakage, which would otherwise produce overly optimistic evaluation metrics that do not reflect real-world performance. SMOTE generated 50,995 synthetic samples for Class 0 (Anomaly), producing a fully balanced training corpus of 170,088 samples with a 50:50 class ratio (85,044 per class). Table 19 details the class distribution before and after SMOTE. Figures 2 and 3 illustrate these distributions visually.

Table 19. Class Distribution Before and After SMOTE

Class	Label	Before SMOTE	After SMOTE
0 — Anomaly (No Ransomware)	A	34,049	85,044
1 — Signature (Ransomware)	S	85,044	85,044
Total	-	119,093	170,088
Class Ratio	-	28.6% : 71.4%	50% : 50%

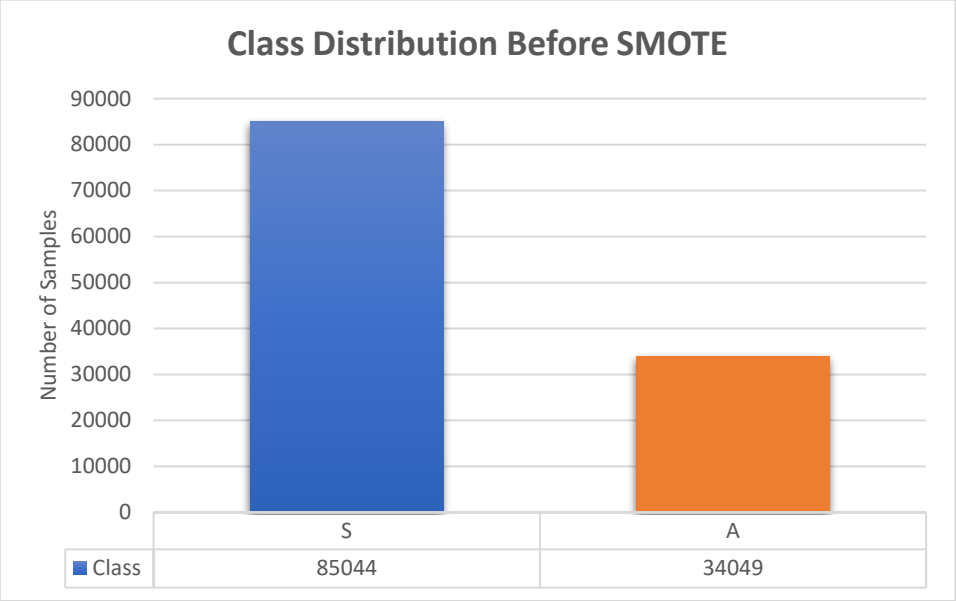

Figure 2. Total Data Distribution Before SMOTE Involvement

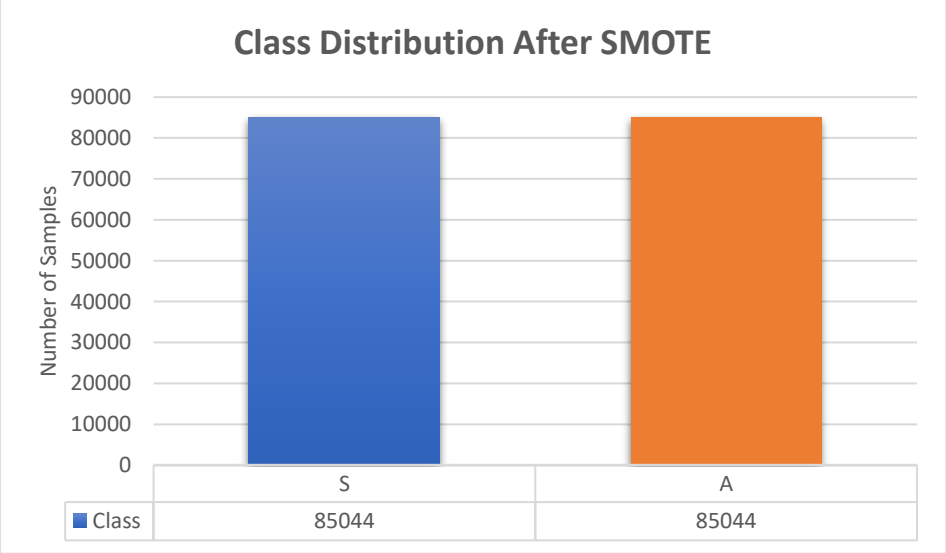

Figure 3. Total Data Distribution After SMOTE Involvement

A 50:50 SMOTE ratio was selected because ransomware detection is a high-sensitivity task in which false negatives (missed ransomware) carry significantly higher costs than false positives. Balanced training data ensures the model learns equally representative decision boundaries for both classes, consistent with best practice for security-critical classification tasks [24] [25]. Figure 2 shows the class distribution before SMOTE and Figure 3 shows the balanced distribution after SMOTE.

Following SMOTE, all features in the training set were standardised using scikit-learn's StandardScaler(), which transforms each feature to have a mean of 0 and a standard deviation of 1. The same scaler fitted on the training set only was applied to the test set to prevent information leakage.

Standardisation ensures that features with larger absolute magnitudes (such as Port numbers in the range 5061–5068) do not dominate gradient updates over features with smaller scales (such as the encoded Protocol values 0–2). The three deep learning architectures were implemented as separate functions (build_dnn(), build_cnn(), build_rnn()). DNN and CNN accept input in 2D format (samples × features), while RNN accepts input in 3D format (samples × timesteps × features) to accommodate the temporal sequence structure required by the LSTM layer. To optimise each model's performance, hyperparameter tuning was performed using GridSearchCV (Grid Search Cross-Validation) from scikit-learn.

GridSearchCV (Grid Search Cross Validation) is a hyperparameter tuning optimization method in machine learning used to systematically find the best parameter combination for an algorithm through a grid search and cross-validation process. This method is available in the scikit-learn library and is commonly used to improve the performance of machine learning models [26]. In this research, Hyperparameter tuning was performed using GridSearchCV with 3-fold cross-validation across DNN, CNN, and RNN architectures. The search grid included: epochs ∈ {10, 20}, batch size ∈ {16, 32}, learning rate ∈ {0.001, 0.01}, and dropout rate ∈ {0.3, 0.5}. Table 3 reports the best hyperparameter combination for each model.

Table 20. Best Hyperparameter Combinations from GridSearchCV (3-Fold CV)

Model	Best Epochs	Best Batch Size	Best Learning Rate	Best Dropout Rate	Val. Accuracy
DNN	20	16	0.001	0.3	95.07%
CNN	20	32	0.001	0.3	96.13%
RNN	10	16	0.01	0.5	93.49%

The results in Table 19 reveal that DNN and CNN converged best with a lower learning rate (0.001), more epochs (20), and a lower dropout rate (0.3), indicating these architectures benefit from slower, more stable gradient updates with less regularisation. In contrast, RNN achieved its best validation accuracy with a higher learning rate (0.01), fewer epochs (10), and a higher dropout rate (0.5), consistent with the known tendency of LSTM-based RNNs to converge faster but require stronger regularisation to prevent overfitting on sequential data.

3.4. Ensemble Learning Strategy

This research proposes an ensemble deep learning model that integrates Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) to enhance ransomware detection accuracy. These models are used as base learners while MLP (Multi-Layer Perceptron) is used as a meta classifier. MLP Classifier uses neural networks to make correct predictions for the test dataset after thousands of iterations of training results [27]. The meta classifier is in charge of combining the predictions from the base learners to make the final prediction. The ensemble technique used is stacking ensemble with MLP Classifier as the final estimator. To increase the visibility of minority classes, the method used is SMOTE. The multiclass problem is reduced to binary by applying the 1 to 1 decomposition technique. Figure 4 shows the intuition of the stacking ensemble model used in this research.

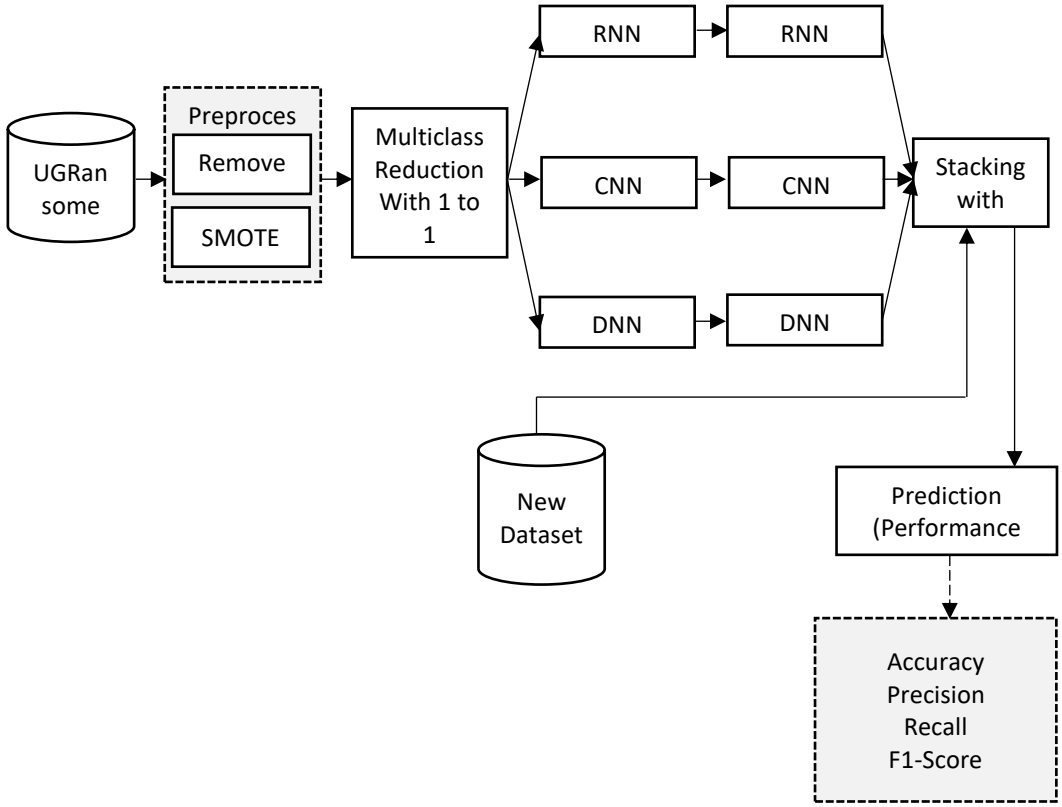

Figure 4. Ensemble Stacking Model Architecture

a. DNN

The DNN accepts 13 numerical input features (post-encoding and scaling). The network consists of an input layer, two hidden layers (64 ReLU units and 32 ReLU units with dropout), and an output layer with 3 Softmax units for multiclass classification.

In this research, the authors uses the Adam optimizer with a tuneable learning rate. Adam Optimizer (Adaptive Moment Estimation) is a deep learning optimization algorithm used to adaptively update the weights of a neural network during the model training process. By combining the strengths of the Momentum and RMSprop methods, this algorithm can accelerate convergence and improve the stability of the learning process [28].

b. RNN

In this research, the author uses RNN, specifically the LSTM type to address the vanishing gradient problem. Input data is reshaped to 3D format (samples × timesteps × features). An Embedding Layer (vocab_size = 20, embedding_dim = 16) is applied to categorical features that retain ordinal integer encoding after label encoding. Justification for Embedding Layer: although the dataset is primarily tabular, the categorical features encoded as integers (Protocol, Flag, Family, etc.) represent nominal class indices rather than continuous values. The Embedding Layer maps each integer index to a dense vector representation, allowing the LSTM to learn richer inter-category relationships than raw integer inputs would afford. The LSTM layer has 64 units (ReLU), followed by a 32-unit dense hidden layer, and a 3-unit Softmax output.

c. CNN

In this research, the input data is reshaped to 3D (samples × features × 1) for convolutional processing. Two convolutional layers are used (32 filters then 64 filters) with max pooling and dropout.

A flattening step reduces the output to 2D, which is passed to a 32-unit dense layer (ReLU), and finally to a 3-unit Softmax output. The Adam optimiser is used throughout.

3.5. Performance Evaluation Metrics

This research uses 4 (four) evaluation metrics as follows: accuracy, precision, recall and F1-score. Accuracy is the percentage of correct predictions across all predictions. Precision is the percentage of correct predictions of a label across all predictions that predict the label. Recall is the percentage of correct predictions of a label across all data that actually is that label [29]. F1-Score is obtained by averaging precision and recall equally [30].

a. Accuracy

The accuracy measures how effectively the model consistently produces reliable predictions. Accuracy is calculated as the proportion of correctly predicted samples to all samples in the data set. Model accuracy measures how effectively the model consistently produces reliable predictions. Accuracy is calculated as the proportion of correctly predicted samples to all samples in the data set [31]. In generally, accuracy is represented [30] as Eq. (1).

$$\mathbf{Accuracy} = \left(\frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{TP} + \mathbf{TN} + \mathbf{FP} + \mathbf{FN}}\right) \tag{1}$$

TP is an acronym for True Positive. The projected value and the actual value match, and both are positive. FP is an acronym for False Positive. The actual value is negative, but the anticipated value is positive, indicating that the predicted value was incorrect. Another name for it is a type 1 mistake. FN is an acronym for False Negative. It is an incorrectly projected value. Although it was projected to be negative, the actual value is positive. Another name for it is a type 2 mistake. TN is an acronym for True Negative. The actual value and the anticipated value are the same. Both the expected and actual values are negative [30].

b. Precision

The precision measures how many distinct positive samples (projected ransomware attacks) there were among all the samples the model predicted to be positive. This statistic is used to assess how well the model can make accurate predictions [32]. In generally, precision is represented [29] as Eq. (2).

$$\text{Precision} = \left(\frac{\text{TP}}{\text{TP} + \text{FP}}\right) \tag{2}$$

c. Recall

Recall measures the model's ability to identify positive samples, i.e. correctly predicted ransomware attacks among all the true positive samples in the data set. It is a statistic used to evaluate how resilient the model is to false negative results [32]. In generally, recall is represented [29] as Eq. (3).

$$\text{Recall} = \left(\frac{\text{TP}}{\text{TP} + \text{FN}}\right) \tag{3}$$

d. F1-Score

The F1 score is gained by harmoniously averaging precision and recall. It is a single number that fairly assesses the model's performance in terms of recall and precision. Accuracy and recall are well balanced when F1 is strong. In generally, recall is represented [32] as Eq. (4).

$$\text{F1} - \text{Score} = 2 \times \left(\frac{\text{Presisi} \times \text{Recall}}{\text{Presisi} + \text{Recall}}\right) \tag{4}$$

3.6. Instrumentation

The instrumentation used in this research includes hardware and software that supports data processing, training, and testing of deep learning models. The hardware used is an HP 240 G7 Notebook PC laptop with Intel® Core™ i3-1005G1 processor specifications, 8 GB RAM, and 1 TB harddisk storage. Meanwhile, the software used includes Python as the main programming language, TensorFlow for

building and training deep learning models, Jupyter Notebook for data development and analysis. Pandas and NumPy were used in the data manipulation process. Scikit-learn is utilised for preprocessing and model evaluation. Matplotlib.pyplot and Seaborn were used for data visualisation and Streamlit was used to build an interactive web application interface. This combination of hardware and software ensures the research can run optimally in exploring data, training models, and analysing results to improve the accuracy of ransomware detection on the UGRansome dataset.

4. Results and Discussion

Table 4 presents the performance of each model on the UGRansome test set (Figure 5 provides the corresponding bar chart). The ensemble model achieves 98.85% accuracy, 98.51% precision, 98.68% recall, and 98.59% F1-score, outperforming all individual base models. To evaluate whether the ensemble model built has greater robustness and generalisation across various datasets and ransomware variants, in addition to being implemented on the UGRansom dataset, the author also implements the ensemble model on the Microsoft Malware Prediction Dataset created by [31], and Ransomware Detection created by [33].

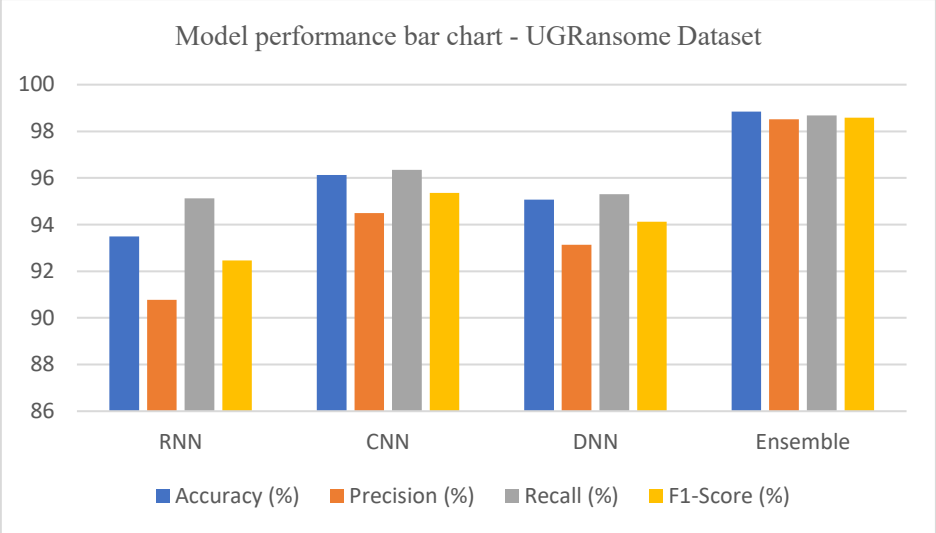

Figure 4. Model performance bar chart — UGRansome Dataset

Table 21. Performance Comparison - UGRansome Dataset

Model	Accuracy	Precision	Recall	F1-Score
RNN	93.49%	90.77%	95.12%	92.46%
CNN	96.13%	94.50%	96.35%	95.36%
DNN	95.07%	93.14%	95.31%	94.12%
Ensemble	98.85%	98.51%	98.68%	98.59%

The ensemble model's superior performance can be attributed to the complementary inductive biases of its base learners. DNN captures non-linear dependencies across all 13 features simultaneously; CNN identifies spatially localised patterns in the reshaped feature matrix; RNN (LSTM) models temporal ordering through the sequential structure of the network-flow data. When these specialised representations are combined by the MLP meta-classifier, the resulting decision boundary is richer than any single model can achieve. The improvement from the best single model (CNN, 96.13%) to the ensemble (98.85%), a gain of approximately 2.7 percentage points is consistent with theoretical expectations for stacking ensembles in the presence of low model correlation [10], [18].

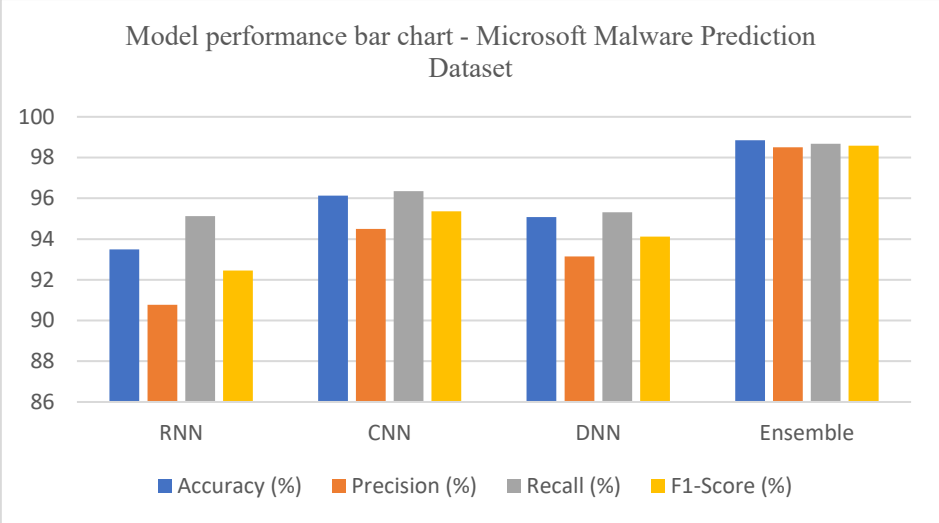

Figure 5. Model performance bar chart - Microsoft Malware Prediction Dataset

Table 22. Performance Comparison - Microsoft Malware Prediction Dataset

Model	Accuracy	Precision	Recall	F1-Score
RNN	85.72%	88.26%	85.72%	80.98%
CNN	86.78%	87.21%	86.78%	85.74%
DNN	88.20%	88.24%	88.20%	88.14%
Ensemble	88.38%	88.44%	88.38%	88.33%

On the Microsoft Malware Prediction Dataset (see in Table 5), the ensemble improvement over the best single model (DNN, 88.20%) is modest at approximately 0.18 percentage points. This marginal gain reflects the nature of this dataset: it is a static metadata-based PE file corpus where DNN already extracts most discriminative features effectively, leaving limited complementary signal for CNN and RNN to contribute. This result highlights an important limitation, the benefit of ensemble stacking is dataset-dependent. Where data contains heterogeneous feature types (tabular statistics, sequential patterns, and spatial structure, as in UGRansome), stacking is highly effective. In more homogeneous datasets, individual architectures may approach the ensemble ceiling.

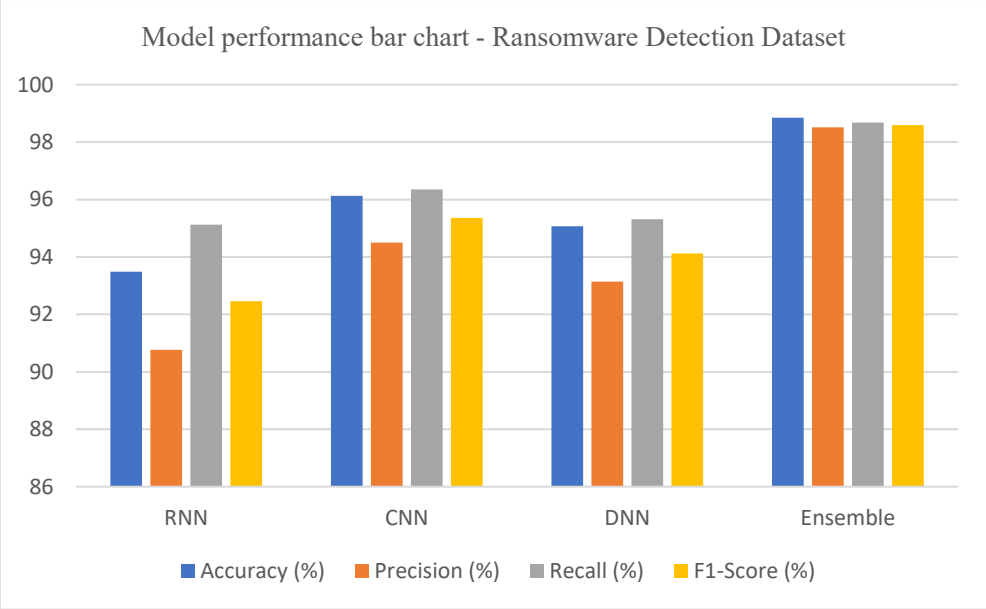

Figure 6. Model performance bar chart - Ransomware Detection Dataset

Table 23. Performance Comparison - Ransomware Detection Dataset

Model	Accuracy	Precision	Recall	F1-Score
RNN	97.62%	96.90%	97.84%	97.35%
CNN	96.94%	96.90%	97.18%	96.59%
DNN	96.35%	95.35%	96.63%	95.94%
Ensemble	98.17%	97.94%	98.27%	97.94%

On the Ransomware Detection Dataset (Table 6), the ensemble achieves 98.17% accuracy, improving on the best single model (RNN, 97.62%) by 0.55 percentage points. The consistent direction of improvement across all three datasets supports the generalisation of the stacking approach, while the magnitude of improvement appropriately reflects dataset characteristics.

5. Conclusion

This study proposed a stacking ensemble deep learning model combining DNN, CNN, and RNN (with MLP as meta-classifier) for ransomware detection on the UGRansome dataset. The ensemble model achieved 98.85% accuracy, 98.51% precision, 98.68% recall, and 98.59% F1-score, consistently outperforming each individual base model. Experiments on two additional datasets (Microsoft Malware Prediction and Ransomware Detection) confirmed that the ensemble direction of improvement generalises across datasets, though the magnitude varies with data characteristics.

It should be noted that all evaluations used a single 80:20 train-test split. Conclusions regarding robustness and generalisation are therefore tentative; k-fold cross-validation and statistical significance tests (e.g., McNemar's test) would provide stronger empirical support and are recommended for future work. Future research directions include: (1) implementation in a real-time ransomware detection system to evaluate processing latency and prediction fidelity under live traffic; (2) integration of knowledge-based feature selection or Autoencoder-based dimensionality reduction to improve feature relevance; (3) exploration of alternative ensemble strategies (bagging, boosting) to determine whether other combination methods yield further performance gains; and (4) evaluation using k-fold cross-validation and statistical tests to provide stronger evidence of generalisation.

References

[1] J. Rahman et al., Laporan Tahunan Monitoring Keamanan Siber 2021. Jakarta Selatan: Badan Siber Dan Sandi Negara, 2021.

[2] T. Nurhidayat et al., Lanskap Keamanan Siber Indonesia 2022. Jakarta Selatan: Badan Siber Dan Sandi Negara, 2022.

[3] BSSN, Lanskap Keamanan Siber Indonesia 2023. Jakarta Selatan: Badan Siber Dan Sandi Negara, 2023.

[4] D. Smith, S. Khorsandroo, and K. Roy, “Machine Learning Algorithms and Frameworks in Ransomware Detection,” IEEE Access, vol. 10, no. October, pp. 117597–117610, 2022, doi: 10.1109/ACCESS.2022.3218779.

[5] A. A. I. Hartinah, Ady Wahyudi Paundu, “Deteksi Malware Ransomware Berdasarkan Panggilan API dengan Metode Ekstraksi Fitur N-gram dan TF-IDF,” JEPIN (Jurnal Edukasi dan Penelit. Inform., vol. 9, no. 1, pp. 50–58, 2023, doi: https://doi.org/10.26418/jp.v9i1.58721.

[6] D. Efriyani and F. Panjaitan, “Klasifikasi Malware Dengan Menggunakan Recurrent Neural Network,” J. Ilm. MATRIK, vol. 23, no. 3, 2021, doi: https://doi.org/10.33557/jurnalmatrik.v23i3.1592.

[7] A. H. Thiziers, K. Tiémoman, N. B. Gérard, and T. T. Q. Kabir, “Malware Detection Using Deep Learning,” Open J. Appl. Sci., vol. 13, no. 12, pp. 2480–2491, 2023, doi: 10.4236/ojapps.2023.1312193.

[8] U. Zahoora, A. Khan, M. Rajarajan, S. H. Khan, M. Asam, and T. Jamal, “Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier,” Sci. Rep., vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-19443-7.

[9] B. Purnama et al., “Deteksi Malware Ransomware Menggunakan Deep Neural Network,” JEPIN (Jurnal Edukasi dan Penelit. Inform. , vol. 9, no. 1, pp. 50–58, 2024.

[10] S. Abimannan, E. S. M. El-Alfy, Y. S. Chang, S. Hussain, S. Shukla, and D. Satheesh, “Ensemble Multifeatured Deep Learning Models and Applications: A Survey,” IEEE Access, vol. 11, no.

September, pp. 107194–107217, 2023, doi: 10.1109/ACCESS.2023.3320042.
[11] R. Bingu and S. Jothilakshmi, “Design of Intrusion Detection System using Ensemble Learning Technique in Cloud Computing Environment,” IJACSA) Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 5, pp. 752–764, 2023, doi: https://dx.doi.org/10.14569/IJACSA.2023.0140580.
[12] S. Koike, H. Tanaka, and M. Maeda, “Federated Learning-Based Ransomware Detection via Indicators of Compromise,” Jun. 2024, doi: 10.21203/rs.3.rs-4585988/v1.
[13] L. Y. Por et al., “A Systematic Literature Review on the Methods and Challenges in Detecting Zero-Day Attacks: Insights from the Recent CrowdStrike Incident,” IEEE Access, 2024, doi: 10.1109/ACCESS.2024.3455410.
[14] M. S. Abbasi, “Automating Behavior-based Ransomware Analysis, Detection, and Classification Using Machine Learning,” Victoria University of Wellington, Wellington, 2023.
[15] K. S. Sangher, A. Singh, and H. M. Pandey, “Signature based Ransomware detection based on optimizations approaches using RandomClassier and CNN algorithms,” Int. J. Syst. Assur. Eng. Manag., 2023, doi: 10.21203/rs.3.rs-2716621/v1.
[16] I. Almomani, A. Alkhayer, and W. El-Shafai, “E2E-RDS: Efficient End-to-End Ransomware Detection System Based on Static-Based ML and Vision-Based DL Approaches,” Sensors, vol. 23, no. 9, May 2023, doi: 10.3390/s23094467.
[17] H. Kim, J. Park, H. Kwon, K. Jang, and H. Seo, “Convolutional neural network-based cryptography ransomware detection for low-end embedded processors,” Mathematics, vol. 9, no. 7, Apr. 2021, doi: 10.3390/math9070705.
[18] I. Mienye and A. Sun, “A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects,” IEEE Access, vol. 10, pp. 99129–99149, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3207287.
[19] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, “AI-based Ransomware Detection: A Comprehensive Review,” 2024, Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ACCESS.2024.3461965.
[20] H. Eldeeb and R. Elshawi, “Empowering Machine Learning With Scalable Feature Engineering and Interpretable AutoML,” IEEE Trans. Artif. Intell., vol. 6, no. 2, pp. 432–447, 2025, doi: 10.1109/TAI.2024.3400752.
[21] H. X. Thinh, V. van Khiem, and N. T. Giang, “Towards Enhanced Surface Roughness Modeling In Machining : An Analysis Of Data Transformation,” EUREKA, Phys. Eng., vol. 2024, no. 2, pp. 149–156, Mar. 2024, doi: 10.21303/2461-4262.2024.003294.
[22] A. Yerke, D. Fry Brumit, and A. A. Fodor, “Proportion Based Normalizations Outperform Compositional Data Transformations in Machine Learning Applications,” Microbiome, vol. 12, no. 1, Dec. 2024, doi: 10.1186/s40168-023-01747-z.
[23] B. H. Liu, L. W. Zhang, Y. Q. Wei, and C. Chen, “Dual Power Transformation and Yeo–Johnson Techniques for Static and Dynamic Reliability Assessments,” Buildings, vol. 14, no. 11, Nov. 2024, doi: 10.3390/buildings14113625.
[24] N. A. Khalil and B. M. Khammas, “An Effective And Efficient Features Vectors For Ransomware Detection Via Machine Learning Technique,” Iraqi J. Inf. Commun. Technol., vol. 5, no. 3, pp. 23–33, Dec. 2022, doi: 10.31987/ijict.5.3.205.
[25] R. Firdaus, A. Id Hadiana, and F. Kasyidi, “Model Deteksi Botnet Menggunakan Algoritma Decision Tree Dengan Untuk Mengidentifikasi Serangan Click Fraud,” J. Informatics Commun. Technol., vol. 4, no. 2, pp. 10–20, 2022, doi: 10.52661.
[26] M. Rashid, J. Kamruzzaman, T. Imam, S. Wibowo, and S. Gordon, “A tree-based stacking ensemble technique with feature selection for network intrusion detection,” Appl. Intell., vol. 52, no. 9, pp. 9768–9781, Jul. 2022, doi: 10.1007/s10489-021-02968-1.
[27] S. Haribabu, G. S. Gupta, P. N. Kumar, and P. S. Rajendran, “Prediction of Flood by Rainf All Using MLP Classifier of Neural Network Model,” in Proceedings of the 6th International Conference on Communication and Electronics Systems, ICCES 2021, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 1360–1365. doi: 10.1109/ICCES51350.2021.9489161.
[28] L. Lakshmi et al., “Performance Analysis of Cycle GAN in Photo to Portrait Transfiguration Using Deep Learning Optimizers,” IEEE Access, vol. 11, no. November, pp. 136541–136551, 2023, doi: 10.1109/ACCESS.2023.3337430.
[29] T. Q. Dam, N. T. Nguyen, T. V. Le, T. D. Le, S. Uwizeyemungu, and T. Le-Dinh, “Visualizing Portable Executable Headers for Ransomware Detection: A Deep Learning-Based Approach,” J. Univers. Comput. Sci., vol. 30, no. 2, pp. 262–286, 2024, doi: 10.3897/jucs.104901.

[30] S. R. Zahra, “UGRansome: Optimal Approach for Anomaly Intrusion Detection and Zero-day Threats using Cloud Environment MSc Research Project Cloud Computing,” 2021. doi: 10.13140/RG.2.2.11263.48803.
[31] M. S. Rahman, “Microsoft Malware Prediction Dataset.” Accessed: Feb. 04, 2025. [Online]. Available: https://www.kaggle.com/datasets/mdsadikurrahman/microsoftmalwareprediction
[32] S. D. Kumar R V A, R. V, K. Rayudu, V. A, and S. R. Kumar, “Enhancing Ransomware Detection in Cybersecurity: A Comprehensive Ensemble Approach,” J. Electr. Syst., vol. 20, no. 10, pp. 5222–5232, 2024, doi: https://doi.org/10.52783/jes.6235.
[33] A. Bensalah, “Ransomware Detection Dataset.” Accessed: Feb. 04, 2025. [Online]. Available: https://www.kaggle.com/datasets/amdj3dax/ransomware-detection-data-set