

Article

A Wide and Weighted Deep Ensemble Model for Behavioral Drifting Ransomware Attacks

Umara Urooj ¹, Bander Ali Saleh Al-rimy ^{2,*} , Mazen Gazzan ³, Anazida Zainal ¹ , Eslam Amer ²,
Mohammed Almutairi ⁴ , Stavros Shiaeles ²  and Frederick Sheldon ^{4,*} 

¹ Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia; uroojumara@graduate.utm.my (U.U.); anazida@utm.my (A.Z.)

² PAIDS Research Center, School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, UK; eslam.amer@port.ac.uk (E.A.); stavros.shiaeles@port.ac.uk (S.S.)

³ College of Computer Science and Information Systems, Najran University, Najran 66462, Saudi Arabia; mzgazzan@nu.edu.sa

⁴ Department of Computer Science, University of Idaho, Moscow, ID 83844, USA; almu9701@vandals.uidaho.edu

* Correspondence: bander.al-rimy@port.ac.uk (B.A.S.A.-r.); sheldon@uidaho.edu (F.S.)

Abstract: Ransomware is a type of malware that leverages encryption to execute its attacks. Its continuous evolution underscores its dynamic and ever-changing nature. The evolving variants use varying timelines to launch attacks and associate them with varying attack patterns. Detecting early evolving variants also leads to incomplete attack patterns. To develop an early detection model for behavioral drifting ransomware attacks, a detection model should be able to detect evolving ransomware variants. To consider the behavioral drifting problem of ransomware attacks, a model should be able to generalize the behavior of significant features comprehensively. Existing solutions were developed by using either a whole attack pattern or a fraction of an attack pattern. Likewise, they were also designed using historical data, which can make these solutions outdated or suffer from low accuracy for behavioral drift ransomware attacks. The detection models created using a fraction of the pre-encryption data also can not generalize the attack behavior of evolving ransomware variants. There is a need to develop an early detection model that can detect evolving ransomware variants with varying pre-encryption phases. The proposed model can detect the evolving ransomware variants by comprehensively generalizing significant attack patterns.

Keywords: ransomware; early detection; malware analysis; deep learning

MSC: 68P25



Academic Editors: Antanas Cenys and Cheng-Chi Lee

Received: 13 February 2025

Revised: 17 March 2025

Accepted: 19 March 2025

Published: 22 March 2025

Citation: Urooj, U.; Al-rimy, B.A.S.; Gazzan, M.; Zainal, A.; Amer, E.; Almutairi, M.; Shiaeles, S.; Sheldon, F. A Wide and Weighted Deep Ensemble Model for Behavioral Drifting Ransomware Attacks. *Mathematics* **2025**, *13*, 1037. <https://doi.org/10.3390/math13071037>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Ransomware is a type of malicious software that locks or encrypts a victim's files, making them partially or completely inaccessible until a ransom is paid [1–3]. Therefore, this often leads to severe financial and operational consequences for individuals and organizations, so it needs to be detected earlier [4]. Ransomware detection is critical for identifying and mitigating the threat posed by these attacks. With the increasing sophistication of ransomware, effective detection mechanisms are essential to prevent data loss, financial damage, and reputational harm. Moreover, these attacks have long-lasting and irreversible effects due to the use of encryption mechanisms [1,2]. Ransomware attacks require early detection, as they encrypt data at a very fast speed [2,3,5]. These attacks use system encryption utilities to encrypt user data. Therefore, the benign-like behavior of

these attacks makes them hard to detect [2,6]. Additionally, the demand for ransoms in the form of untraceable money has boosted the spread of this attack [7]. The risks of data loss, unauthorized sharing, service unavailability, application crashes, and damage to reputation persist even after ransom payment [1,2].

The growth of ransomware attacks highlights the rapid evolution of these threats [4,7]. This attack is evolving due to the introduction of new variants that have been developed or may develop in the future [8]. The availability of easy-to-use kits means that in-depth technical knowledge is not required, thereby facilitating the generation of these attacks. Additionally, various techniques such as variable renaming, junk code insertion, and code obfuscation are used to generate different mutants of ransomware that make detection and mitigation more challenging [9]. Code obfuscation performs evasion of detection via ransomware defense mechanisms. This technique includes various strategies like metamorphism, polymorphism, and packing [4,10,11]. According to its research, Webroot concluded that 94% of malicious executables are polymorphic [12]. Moreover, obfuscation techniques using polymorphic and metamorphic processes play a role in the generation of ransomware variants, which has resulted in the behavioral drift of attacks. Behavioral drift in ransomware attacks refers to the evolving nature of ransomware behavior over time, which poses challenges for detection systems. Attackers employ polymorphic and metamorphic techniques to avoid signature-based detection [13]. Moreover, Chen said that the changing behavior of ransomware variants due to behavioral drifting is associated with a change in attack distribution, i.e., features [7,14–16].

The development of ransomware variants is influenced by changes in the distribution of features, which in turn affect the significance of those features [17]. For example, a ransomware attack at time t_1 may exhibit an attack pattern (features), p_1 , where its relevance is higher compared to the same pattern presented by a different ransomware variant during a later attack at time t_2 using attack pattern (features) p_2 . This suggests that the importance of features varies over time across different ransomware variants. The presence of attack patterns and their key features is associated with a specific timeframe, (t) , for each ransomware variant.

Ransomware is developing at great speed; hence, a model should be well trained so that it can detect evolving variants. The identification of optimal behavioral attributes is needed and will lead to high classification accuracy [18]. To predict future ransomware, a model should be trained on the significant features of existing and new features of developing variants.

Ransomware attack detection becomes more challenging when dealing with the behavioral drifting of evolving variants. Therefore, early detection is important, as once the files are encrypted, they cannot be decrypted without a decryption key [2,3], with few exceptions. Some existing solutions have been proposed to detect ransomware attacks, but only limited studies have addressed the problem of early detection and ransomware behavioral drifting. However, these studies are limited in scope, as they do not consider the non-stationary nature of evolving ransomware attacks while ensuring early detection simultaneously. These solutions are also lacking because they were developed using the assumption of a fixed pre-encryption timeline [2,4,17,19].

A behavioral solution is required to deal with more advanced variants of ransomware attacks that display behavioral drifting [18]. To develop an early detection model for behavioral drifting ransomware attacks, a detection model should be able to detect evolving ransomware variants. To consider the behavioral drifting problem of ransomware attacks, a model should comprehensively generalize the behavior of significant features [20,21]. The existing solutions were developed by using either the whole attack pattern or a fraction of an attack pattern. Likewise, they were also developed using historical data, making

these solutions outdated or suffering from low accuracy for behavioral drift ransomware attacks [22,23].

The detection models developed using a fraction of pre-encryption data also could not generalize the attack behavior of evolving ransomware variants [5,22]. However, the existing solutions for ransomware detection are primarily based on historical data and a limited set of attack patterns, which significantly constrain their effectiveness. These approaches struggle to address the challenges posed by ransomware attacks that evolve over time, particularly those exhibiting behavioral drift. As a result, such solutions are often insufficient for the early detection of evolving ransomware variants. Since these models are designed based on the attack patterns and features prevalent within a specific timeframe, they fail to detect newly developed variants effectively. Consequently, there is a pressing need for the development of adaptive models capable of identifying behavioral changes in ransomware features that offer high detection accuracy and minimize false alarms, even as ransomware evolves.

1.1. Contribution

The contributions of this work are as follows:

1. The development of a ransomware detection model, a wide and weighted deep ensemble model, is proposed to address the challenge of behavioral drift in evolving ransomware variants;
2. We evaluated the efficacy of implementing multiple deep ensembles in the proposed model;
3. A comprehensive comparison was conducted between the proposed wide and weighted deep ensemble model and existing solutions to evaluate its relative effectiveness;
4. Experiments were conducted to check the twofold implementation of the proposed work.

1.2. Organization

The paper is organized into seven subsections. The available literature related to the proposed work is summarized in Section 2. An overview of the proposed work is presented in Section 3, describing the methodology adopted to carry out the experimental work. Section 4 consists of the experimental results obtained after the implementation of the proposed model. The results are elaborated upon and discussed in Section 5. This paper ends by presenting a conclusion at the end in Section 6.

2. Related Work

The proposed work uses different concepts, ensembles, and calculations. The detection studies are broadly categorized into single algorithm-based and ensemble-based models to perform ransomware detection [24]. This section summarizes related studies. Additionally, these studies highlight other studies that utilized relevant concepts.

A framework to identify malware applications targeting Android users was proposed in [25]. This framework utilized Android intents and permissions to detect potential threats and malicious behaviors. By performing statistical analysis on the correlation between specific permissions and associated intents, the framework leveraged a classifier to identify potential malware. In a more recent study, Ref. [26] focused on detecting ransomware attacks by extracting unique features through statistical analysis. Using machine learning techniques, the authors employed Support Vector Machines (SVMs) and logistic regression (LR) models, which were implemented on a CPU. Additionally, a neural network model was introduced to provide protection on the GPU. This GPU-based implementation was specifically designed to enhance power efficiency, reducing computational costs while maintaining detection accuracy.

The authors of [27] introduced DeepERPred, a model designed to address ransomware attacks through a predictive approach. The model employs a combination of Convolutional Neural Network (CNN) and LSTM networks to classify features identified in the dataset. This classification process is particularly effective in handling temporally changed DNA, utilizing specific sequences to detect data affected by ransomware infections. Similarly, Ref. [28] proposed a hybrid model for detecting ransomware attacks on Android devices. The model integrates and analyzes various data sources, including call logs, CPU usage, memory consumption, text data, and app permissions. The analysis was conducted using several machine learning algorithms such as AdaBoost, LR, Random Forest, SVM, and JRip, enhancing the accuracy and robustness of the detection process.

Additionally, Ref. [29] proposed a method to categorize different types of ransomware attacks using behavioral analysis. To achieve this goal, the study employed correlation values and the information gain method for feature selection. Additionally, various formulas were defined to automatically determine threshold values for the respective datasets. In a related effort, a technique called BigRC-EML was proposed to mitigate ransomware attacks targeting Windows-based applications. BigRC-EML utilizes both dynamic and static features to classify and detect different types of ransomware attacks. Feature selection in this approach was performed using Principal Component Analysis (PCA) [30]. Moreover, Ref. [31] proposed a model known as RANSOMNET+ to detect ransomware attacks on cloud-encrypted data. The model used a hybrid approach and CNN to identify local patterns and hierarchical features.

A model named Wide and Deep model of Multi-source information-Aware recommender system (WDMMA) was proposed in [32] that contained different types of information like context, a user-item interaction matrix, and characteristics. The model mainly handled the interaction and communication between the user and the relevant item. The model was implemented on two different types of datasets, resulting in WDMMA being helpful to achieve more optimized performance. A study was proposed to predict click-through rates for online advertising [33]. The study proposed a unique method to train the additional features effectively by using DNN-based models. It also helped to avoid delayed convolution for better performance. The model was extensively tested on Wiebo and public datasets. A model named FGCNN was proposed in [34] to increase the efficiency of predicting click-through rates. That model contains two main components: one as a deep classifier and the other as feature generation. MLP and CNN were used in feature generation to balance each other and identify additional and important features. The deep classifier predicted CTR based on the results of the generated feature.

The work in [35] proposed a framework to search relevant data effectively for Android using machine learning. The framework trained a linear model with feature transformation and feed-forward neural networks jointly for generic recommendations using additional inputs. Moreover, it was evaluated by using a wide range of customized apps available on the Google Play store. In [36], a study was conducted to check the impacts of an optimizer known as Adam on wide and deep neural networks (WDNNs). The dataset was taken and fed into the neural network, and the generated results were used as a benchmark. The same dataset was also fed into the wide and deep neural networks with and without the Adam optimizer to compare results. The results indicated that the Adam optimizer significantly improved the performance of the WDNN. Therefore, the WDNN model was utilized to develop a monitoring system for estimating PM2.5 concentrations. In this context, memorization techniques were applied to the features extracted from photographs, and generalization was applied to the ground truth data representing PM2.5 concentration levels [37]. This approach is conceptually similar to addressing behavior

drift in ransomware, even though the application domains differ considerably. A summary of the studies reviewed is provided in Table 1.

Table 1. Summary of related studies.

Sr#	Study	Generalization	Features Used	Behavioral Drifting
1	[25]	No	Yes	No
2	[26]	No	Yes	No
3	[27]	No	Yes	No
4	[28]	No	Yes	No
5	[29]	Yes	Yes	No
6	[30]	No	Yes	No
7	[31]	No	Yes	No
8	[35]	Yes	Yes	No
9	[33]	No	Yes	No
10	[32]	Yes	Yes	No
11	[34]	Yes	Yes	No
12	[36]	Yes	Yes	No
13	[37]	Yes	Yes	No

2.1. Existing Behavioral Drift-Related Solutions

The recent literature on malware indicates a focus on behavioral drift, with an emphasis on adaptive detection methodologies to counteract evolving threats. Concept drift refers to the phenomenon where the statistical properties of malware change over time, making previously effective detection models less reliable. As malware authors continuously innovate to evade detection, understanding and managing these drifts becomes critical for cybersecurity [38,39]. The authors of [38] discussed the concept of drift in malware classifiers, specifically examining how the distribution of malware families influences drift behavior. Their work indicates that recognizing and analyzing these drifts can enhance classifier performance in real-world contexts, which is essential for developing robust security measures. This aligns with the findings of Ref. [39], wherein the degradation of performance in machine learning-based malware detectors over time was investigated, emphasizing the need for frameworks capable of assessing and adjusting to these drifts dynamically [39,40].

Hybrid models of analysis are gaining traction as useful strategies for addressing behavioral drift. The study conducted in [41] proposed a model for integrating logistic regression and recurrent neural networks for detecting malware API calls, which emphasizes how combining different methodologies can improve detection accuracy amid varied malware behaviors. In the realm of dynamic analysis, traditional detection methods face challenges due to evasive malware techniques, which often involve dynamic binary instrumentation (DBI). The authors of [42] illustrate how DBI permits deep access to the behavior of malware, allowing for the extraction of precise and authentic behaviors from complex malware that employs evasion techniques. This methodological advancement combines both dynamic and static analysis, thus enhancing detection capabilities and making it more difficult for malware to conceal itself effectively.

Moreover, the application of reinforcement learning for malware behavior detection has shown promise, as discussed in [43]. Their work highlights an intelligent decision-making framework tailor-made for identifying malware behaviors within network security contexts, effectively adapting to dynamic environments through learned experiences and feedback. The adaptability offered by reinforcement learning models demonstrates significant potential for addressing concept drift in active malware regulation contexts. Further, the literature indicates a growing reliance on advanced machine learning techniques, including graph neural networks for behavior classification, as explored in [44]. This novel

approach emphasizes graph-based learning for recognizing malware patterns, thus enhancing detection methods against sophisticated malware that can exhibit drift over time.

The aforementioned works have largely concentrated on how the statistical distribution of malware families changes over time but often underemphasize the evolving role of specific features—especially when early-stage or partial data about new variants is limited. By focusing primarily on high-level distribution shifts, these approaches may overlook how smaller subsets of features become increasingly (or decreasingly) relevant as attacks mutate. In practice, this omission can leave detection models vulnerable to emergent ransomware variants, the behavior of which is not fully captured by historical data. Properly accounting for changing feature significance, therefore, becomes critical in accurately modeling malware drift and ensuring robust detection performance in real-world settings, where data about the newest attacks is often sparse.

2.2. Limitations of the Existing Ransomware Detection Models

The existing models did not consider the behavioral drifting of ransomware attacks. These solutions were developed by using historical data, i.e., the features of already discovered ransomware. However, the historical features either consist of insufficient attack patterns or contain data for the whole encryption process. Most of these studies were presented by using full attack patterns that did not provide early detection. Sophisticated ransomware using crypto-APIs can remain undetected for the models trained on historical patterns. Some of the solutions were developed by using an anomaly approach. However, anomaly-based detection solutions generated high false alarms, as these models were based on normal profiles. A small deviation from the set profile resulted in a false alarm. On the other hand, the other solutions used a random selection of features instead of considering all data. By using this approach, these studies did not represent each family available in the dataset. Furthermore, the available work did not use synthetic and significant features for model training to detect ransomware attacks. Therefore, these solutions did not consider the feature significance associated with the behavioral drifting of ransomware attacks. In light of the above-mentioned limitations, WWDEM is proposed to detect evolving ransomware variants displaying behavioral drifting.

3. Methodology

Developing variants associated with different timelines should be considered by utilizing the existing and evolving variants' features. The utilization of existing information and its association with developing variants could help to detect ransomware attacks. To address the problem of ransomware behavioral drifting, training on historical and developing features should be utilized. Additionally, to understand the suspicious behavior of evolving ransomware variants, all the features need to be considered at once. To address the problem of behavioral drift, a model should consider all the significant features of timeframes t_1, t_2, t_3 , up to t_n . For this reason, the left-out features of phase 1 can not be discarded; rather, they should be used comprehensively to address behavioral drifting.

3.1. Feature Selection and Data Processing

This study utilized Mutual Information Feature Selection (MIFS) to select a set of relevant and non-redundant features [45]. These most informative features help to add reliance to the results and, therefore, can be used as they are by using the memorization concept. However, to cope with behavioral concept drift, a significant portion of the remaining features might also be important. The intuition is that data limitation during the early phase of the attack makes it difficult to estimate feature significance accurately. Some of the features that were considered insignificant could be important if more attack data

are captured. Therefore, these left-out features were also used by ensembles of deep neural networks to generalize unseen feature combinations. The generalization was implemented by assigning optimized weights to each ensemble. The use of optimized weights for each ensemble will improve relevance among features. Two types of training were used here: ensemble training and joint training.

The behavior of the samples was observed in a controlled environment to gather ransomware data. Dynamic analysis is more suitable for studying polymorphic and metamorphic ransomware [46]. The dataset employed in this study was generated within the experimental environment described in [22]. Dynamic analysis was conducted using a Cuckoo sandbox, which is a flexible and feature-rich tool for analyzing malicious code [47]. It offers a realistic yet isolated environment to monitor the behavior of malware [48], enabling the generation of synthetic data. Many previous studies have utilized Cuckoo to analyze crypto-ransomware behavior [49]. The Cuckoo sandbox was deployed within VMware to create a virtualized environment that accurately reflects the behavior of ransomware samples. Separate trace files were generated for ransomware and benign samples, as each was executed individually. These trace files documented all API calls made during the analysis in Cuckoo. The extracted API data were then used to construct the synthetic data for the proposed study. An overview of the experimental process, which illustrates how the dataset was generated, is shown in Figure 1.

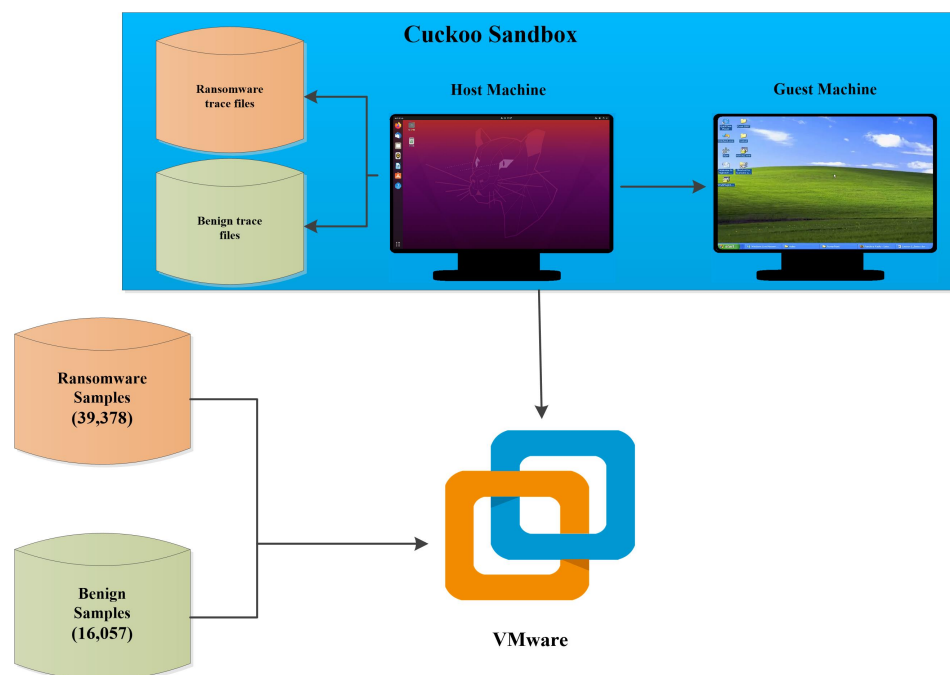


Figure 1. The experimental flow for behavioral data log generation.

3.2. Model Architecture

Joint training refers to a training technique where multiple tasks are trained simultaneously in a single model. This approach allows the model to learn to perform different tasks at the same time, which can lead to improved performance and efficiency. Joint training helps to promote generalization by extracting the shared features and relationships across the task.

In wide and weighted deep ensemble model (WWDEM), ensemble learning is used to generate diverse, more comprehensive, and reliable results. It is performed by creating a group of classifiers to address the same problem by working in a complementary manner to generate comprehensive results. Moreover, using different ensembles of different configurations and sets of numbers will emphasize the best pattern recognition from the remaining

features. In a nutshell, an ensemble model built from a combination of different ensembles that are loosely correlated will generate more stable and robust accuracy results [16,50].

In the proposed model, the deep part of the wide and deep model is modified to optimize the result of ransomware detection by using the remaining features after the implementation of the MRMR technique on the dataset of significant features. To address the problem of feature significance at time $t1$ and $t2$, we must consider all the significant features. The topmost informative features attained after the implementation of MRMR were assigned to the wide part. The redundant and left-out significant features after the implementation of MRMR were provided to the deep part. In this way, optimization was also maintained using optimization features filtered by the implementation of MRMR. The ensemble learning was implemented in the deep part, along with assigning optimized weights to each ensemble. After the integration of weight optimization and ensemble learning, the wide and weighted deep ensemble models were combined to capture the true consideration of behavioral detection (Table 2).

Table 2. Model representation and reflected behavior.

Part of Model	Representation	Reflected Behavior
Wide	f_s	Carries non-redundant and informative features.
Weight	f_d	The weighted part carries features that are used to generate hidden patterns from left-out features after applying MRMR.
Wide + Weight	$f_s + f_d$	Represents the joint training of the wide and weighted ensemble. It carries both informative features as they are and patterns from left-out features.

3.3. Joint Ensemble Learning for Behavioral Drift Detection

WWDEM, combined with joint training, effectively optimizes both the wide and deep networks during the training process. All ensembles are trained simultaneously, with the wide network employing a linear model and the deep network utilizing deep neural networks. This combination of wide and deep models offers a balance between memorization and generalization, making it well-suited for addressing the classification problem of behavioral drift. The wide network complements the deep network by capturing direct relationships, and the deep network improves generalization by learning more abstract patterns. This generalization helps identify potential combinations of past and future patterns at time $t2$, enhancing the diversity of the generated results. Additionally, the deep network focuses on less granular features; this makes it well-suited for handling left-out features in the proposed framework [35,36].

Ensemble learning is performed by building a model using a combination of models. These combinations of models work together to produce more accurate predictions. Prediction is generated by using the predictions produced by each model in the ensemble learning. Each model in the ensemble learning is implemented by using different configurations or by using different datasets. Ensemble learning is famous for its accurate, more reliable, and robust prediction. There are different ways to calculate the weights associated with each prediction. Some of the ways to produce prediction include voting, averaging, stacking, and gradient boosting. To produce the final prediction, the ensemble equation is calculated by using averaging and voting methods. For the averaging method, the decision is produced by summing up predictions produced by each model and dividing it by a number of models. Equation (1) gives an idea of the averaging method.

$$\text{Decision} = \frac{\text{Model}_1 P_r + \text{Model}_2 P_r + \text{Model}_3 P_r + \cdots + \text{Model}_n P_r}{n} \quad (1)$$

For the voting method, the decision is produced by taking the largest prediction probability of given models. Equation (2) expressed the idea.

$$\text{Decision} = \frac{\arg \max(\text{Model}_1 \text{ vote} + \text{Model}_2 \text{ vote} + \text{Model}_3 \text{ vote} + \cdots + \text{Model}_n \text{ vote})}{n} \quad (2)$$

Ensemble learning combines the decisions of individual ensembles to perform classification. Ensembling generates good results when each ensemble is loosely correlated, helping to obtain better generalization results [50,51]. Ensembling can be performed by using different approaches, including training set resampling, heterogeneous algorithms, using the same algorithms with different parameters, and using the same algorithms with different decision-combining methods [51]. In ensembling, individual models are trained independently to obtain predictions of each model. Ensemble training is performed in a disjointed manner, and predictions are calculated after ensemble training. This independent training helps to implement different configurations on different ensembles, hence generating different accuracy values. Different configurations are used to imply diversity among features. The purpose of using different configurations in different ensemble models is to generate optimized weights, thus improving the prediction accuracy of the final decision. Individual predictions from each ensemble model were calculated and employed to compute the optimized weights for each ensemble model. These optimized weights were then assigned to each ensemble of the deep model. The end results were the output of a wide and weighted deep ensemble model. The equation of ensemble learning varies depending on the specific method used to produce the final decision, and different ensembles have different equations. The proposed ensemble learning equation can be described by Equation (3) below:

$$\text{Decision} = \frac{\sum_{i=1}^n \text{Model}_i P_r \cdot w_i}{n} \quad (3)$$

Here, weights are represented using w , and each model carries its own weights. The prediction of each model is produced by using the assigned weights for each ensemble.

WWDEM takes inputs, f_s and D_b , which represent the set of selected features and significant features, respectively. Filtration is performed in lines 9 to 14 to emphasize the worth of selected features. Ensembles were defined according to the configuration set by the user. Weight calculation and optimization are performed using lines 24 to 34. Initially, the weights were calculated for each ensemble by dividing one by the number of ensembles. These initial weights provided a starting point to calculate the optimized weights. Here, upper and lower bounds were also provided as a default setting. According to the optimization algorithm settings, the optimized weights were calculated for each ensemble. Then, the optimized weight of each ensemble was assigned to the corresponding ensemble using the expression $w_n \cdot \text{model}(n)$. This weight assignment will generate weighted ensembles that will be used to form a weighted deep ensemble model. Line 35 represents the joint training of wide and weighted deep ensembles as a single model. Wide network + weightedensemble represents the proposed wide and weighted deep ensemble model. The pseudo-code of the wide and weighted deep ensemble model is presented below (Algorithm 1).

The proposed WWDEM model aims to improve the generalization of patterns to obtain hidden patterns and address the behavioral drifting problem. Figure 2 draws the general architecture of the proposed model. It is implemented using a wide and deep model, which is particularly effective due to its integration of memorization and generalization. The wide network captures memorization by learning direct feature interactions, while the deep network enables generalization by discovering complex patterns and hi-

erarchies within the data. The wide network captures established interactions among highly informative features selected through MRMR, providing strong memorization. In parallel, the deep ensemble part captures subtler patterns from the remaining features, promoting generalization and adaptability to behavioral drift. By leveraging multiple deep models, the ensemble framework learns diverse and complementary patterns, significantly reducing false positives and increasing detection reliability. This combination ensures that the model remains robust against evolving ransomware behaviors, consistently delivering improved detection accuracy and precision, even for ransomware variants not encountered during training.

Algorithm 1 Wide and Weighted Deep Ensemble Model

```

1: Input: Significant features  $D_b$ , Selected features  $F_s$ 
2: Output: Wide and weighted deep ensemble model
3:  $D_b \leftarrow$  Set of significant features
4:  $F_s \leftarrow$  Set of selected features
5:  $F_d \leftarrow$  Set of features for deep model training;  $F_d \leftarrow 0$ 
6:  $W \leftarrow$  Wide network
7:  $w_n \leftarrow$  Optimized weights
8:  $models \leftarrow []$ 
9:  $weightedensemble \leftarrow []$ 
10: Define  $W = F_s$ 
11: for all  $f \in D_b$  do
12:   if  $f = F_s$  then
13:     Discard  $f$ 
14:   else
15:      $F_d.append(f)$ 
16:   end if
17: end for
18: for  $n = 1$  to number of models do
19:   Define model ( $n$ ) using  $F_d$ 
20:   Compile model ( $n$ )
21:   Fit model ( $n$ )
22:   Evaluate model ( $n$ )
23:   Get predictions from model ( $n$ )
24:   Append model ( $n$ ) to  $models$ 
25:    $n \leftarrow n + 1$ 
26: end for
27:  $weights =$ 
28: Set upper bound = 1 and lower bound = 0
29: for  $n = 1$  to number of models do
30:   Generate mutant vector for model ( $n$ )
31:   Combine mutant vectors with model ( $n$ )
32:   Evaluate model
33:    $w_n \leftarrow$  weight selection for model ( $n$ )
34:    $weightedensemble(n) \leftarrow w_n \cdot model(n)$ 
35:   Append  $weightedensemble(n)$  to  $weightedensemble$ 
36:    $n \leftarrow n + 1$ 
37: end for
38: Wide network +  $weightedensemble$ 
39: Train [Wide network +  $weightedensemble(n)$ ]
40: Evaluate [Wide network +  $weightedensemble(n)$ ]
41: Return [Wide network +  $weightedensemble(n)$ ]

```

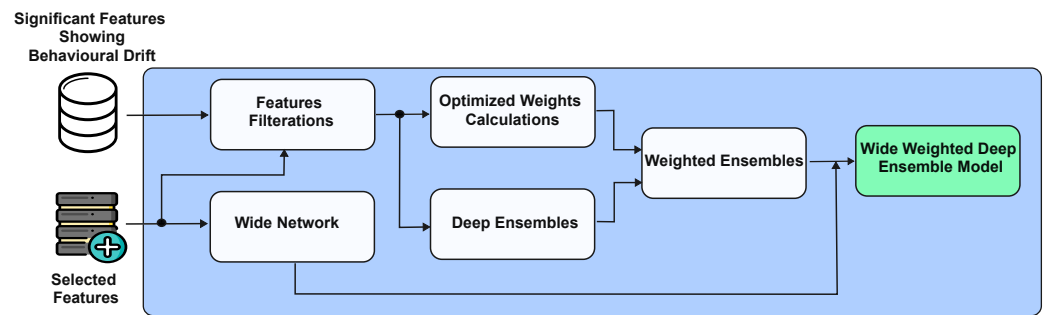


Figure 2. The architecture of wide and weighted deep ensemble model.

The proposed work is implemented using a wide and deep model, which is particularly effective due to its integration of memorization and generalization. The wide network captures memorization by learning direct feature interactions, while the deep network enables generalization by discovering complex patterns and hierarchies within the data. Ensembling is performed on the deep network by building an ensemble of different configurations. Moreover, a training and testing split is used to divide the dataset into train and test datasets in the ratio of 80:20. This division is performed to test the efficacy of the model on the dataset that is not seen by the model [49]. A hold-out validation technique is performed to obtain the data for model validation. The training dataset was further split up into training and validation datasets in a ratio of 80:20. To measure the potency of the proposed work, an evaluation was performed using evaluation metrics, including accuracy, precision, recall, F1-score, false positive rate, and detection rate. Different numbers of ensembles were used in the deep network. The same number of ensemble models were used to calculate the optimized weights for the relevant ensemble. Optimized weights, along with ensemble learning integrated into the deep part, were used to implement the weighted deep network of the proposed wide and weighted deep ensemble model. Experiments were conducted in a twofold manner. In one of the experimental setups, varying numbers of features were used, i.e., 10, 20, 30, 40, and 50, and in another setup, varying numbers of ensembles were used, including 3, 4, 5, 6, and 7 ensembles, represented as C3, C4, C5, C6, and C7 respectively. Each ensemble generated diverse results due to the implementation of different model structures of varying configurations. Each ensemble was trained in a disjointed manner, hence producing different predictions and accuracy values.

4. Results

This section explains the adopted methodology to implement the proposed model. It also briefly describes the details of the dataset used to train the wide and deep parts of the proposed model. Moreover, the experimental considerations, methods, and tools are presented afterward.

4.1. Dataset

This work utilized two different datasets to train the proposed model (WWDEM). The behavioral dataset prepared in our previous work [52] consists of pre-encryption data and synthetic data that mimic pre-encryption data. The synthetic data represented potential patterns and had a similar distribution as pre-encryption data to overcome the problem of data limitation during the pre-encryption phase. The behavioral dataset consists of significant features and is split up into two sets. One dataset contains 50 non-redundant informative features obtained by applying the TU-MRMR feature selection technique on the behavioral dataset. The second dataset contains redundant significant features by omitting informative features from the behavioral dataset.

In our dataset, many of the extracted features, which are derived from observed APIs and system calls, are inherently interpretable because they map directly to system-level operations (e.g., file creation, registry modification, and cryptographic library use). Intuitively, an API that handles encryption keys has clear relevance to ransomware behavior, offering a semantic clue as to how the malware is operating. For instance, calls like `CryptAcquireContext`, `CryptGenKey`, and `CryptEncrypt` offer insights into how encryption keys are generated and used, which is an essential part of ransomware functionality. Similarly, `CreateFileW` and `WriteFile` help indicate file manipulation behaviors, which are relevant for ransomware attempting to encrypt or overwrite user files. Registry modifications can also be detected through `RegCreateKeyEx` or `RegSetValueEx` calls. While some higher-level or aggregated features may appear more abstract, most of the core features we rely on carry domain-specific semantic meanings that cybersecurity analysts can interpret to understand the underlying behavior of each ransomware sample.

In this study, we selected ransomware families that utilize encryption methods commonly found in real-world ransomware campaigns—primarily variants of symmetric (e.g., AES) and asymmetric (e.g., RSA) ciphers. In practice, the most prevalent approach involves using a strong symmetric cipher (e.g., AES-256) for the actual file encryption, combined with an asymmetric algorithm for key exchange [2]. Our dataset reflects this industry reality by incorporating samples that rely on such hybrid mechanisms to lock user data. For example, we included well-known ransomware families like ‘WannaCry’ (which uses RSA and AES), ‘Locky’ (typically RSA-2048 plus AES), and several others that frequently appear in threat intelligence reports. This ensures that the encryption methods present in our experimental setup align with those most often encountered in current ransomware attacks, thereby providing a realistic basis for evaluating detection performance.

4.2. Experimental Environment

In this work, a ransomware detection model is introduced that detects evolving ransomware variants displaying behavioral drift. A wide and deep model is used to implement the proposed work. A wide network is considered to deal with the features obtained after the implementation of TU-MRMR. Additionally, deep networks were trained by using a behavioral dataset containing significant features. The deep network deals with features obtained by omitting selected features of TU-MRMR, i.e., left-out features. Weight optimization and ensemble learning were applied to the deep network by using the remaining features to address the behavioral drifting more precisely. This model incorporated the behavioral drifting concept by utilizing each feature to finalize the decision. Consideration of each feature is important, as it may contain information that could be useful in the future. Different configurations in different ensemble models are used to generate optimized weights, which helps to improve the prediction accuracy of the final decision. The informative features used here carried more weight as they are more relevant, and the non-redundant features, therefore, were utilized as they were. The remaining features were optimized so that none of the features were ignored, thus safeguarding confidence in the representation of behavioral drifting. Optimizing the weights of each ensemble helps to boost the performance of the resultant ensemble. Two different experimental setups were used to check the efficacy of the proposed model. In one setup, a different number of ensembles were used in the deep part of the proposed model. In another setup, the performance of the proposed work was compared against considered state-of-the-art works [22,49]. The related studies were considered to validate the obtained results.

4.3. Processing Tools

The experiments were conducted using Python 3.7 and various libraries for advanced data processing and mathematical computations: Pandas for data manipulation, NumPy for efficient numerical operations, Scikit-learn for machine learning algorithms, Skfeatures for feature selection, and SciPy for advanced scientific computations. To build and train the proposed model, the Keras and TensorFlow frameworks were utilized, with TensorFlow serving as the primary deep learning framework due to its flexibility and scalability. The proposed model was implemented using the TensorFlow framework, with heavy computational tasks accelerated by GPUs.

4.4. Evaluation Metrics

We evaluated the model's performance using precision, recall, F1-score, accuracy, and false-positive rate (FPR).

Precision represents reliability, i.e., correct predictions made by a model. It represents the relevance of predictions a model is supposed to make. Therefore, high precision means good model performance. Precision is calculated according to Equation (4).

$$\text{Precision} = \frac{\text{TruePositives(TP)}}{\text{TP} + \text{FalsePositives(FP)}} \quad (4)$$

The recall metrics are specific, i.e., they represent the sensitivity of the model by measuring its quantity. A model presenting high detection results will have a high recall value. Recall is calculated by using Equation (5).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FalseNegatives(FN)}} \quad (5)$$

The F1-score is calculated by using the two evaluation metrics, including precision and recall. The F1-score incorporates model relevance and sensitivity. It is calculated according to Equation (6).

$$\text{F1-Score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (6)$$

The correctly identified samples are represented using accuracy metrics. It is described using the ratio of correct predictions along with total predictions made. Moreover, it is described according to Equation (7).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (7)$$

The false positive rate (FPR), described by Equation (8), measures the proportion of negative instances incorrectly classified as positive by a model. It evaluates the rate of false alarms in a system.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (8)$$

Another important evaluation metric is the detection rate, as shown in Equation (9), which highlights the significance of the proposed work. The DR is calculated by dividing the number of detected ransomware samples by the total number of both ransomware and non-ransomware samples.

$$\text{DR} = \frac{\text{TP}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (9)$$

4.5. Experimental Results

The performance of the proposed WWDEM model, evaluated using varying numbers of ensembles, highlights its effectiveness in detecting evolving ransomware variants. To thoroughly assess its capabilities, a set of evaluation metrics, including precision, recall, F-score, accuracy, and FPR, was applied. WWDEM achieved the highest accuracy using a model consisting of seven ensembles trained on 50 features, whereas the lowest accuracy was observed in a model using three ensembles trained on 10 features. Similarly, the model demonstrated strong performance in terms of precision, achieving the highest precision across different ensemble and feature configurations. The variations in performance with different ensemble configurations are further discussed in Section 5. The lowest precision was observed in the model with three ensembles trained on 10 features. The details of the results obtained by the proposed model are as follows.

The results in Table 3 present the performance of the proposed WWDEM when using three ensembles (C3) and varying numbers of features (10, 20, 30, 40, and 50). The metrics include precision, recall, F1-score, accuracy, and false positive rate (FPR). Notably, the highest accuracy reaches 0.960 for the model trained on 50 features, accompanied by high precision (0.923) and recall (0.999). Conversely, the lowest reported metrics appear when only 10 features are used, with precision dipping to 0.845. Nevertheless, even the lower-range results maintain reasonable performance levels, underscoring the model's robustness across different feature subsets. These results illustrate the significance of incorporating both critical and less impactful features under an ensemble approach, especially in cases where ransomware exhibits varying behavioral traits. The gradual improvement in accuracy, precision, and recall as more features are included indicates that expanding feature coverage helps the model capture more nuanced patterns of malicious behavior. Furthermore, the relatively low FPR values suggest that the joint memorization and generalization framework effectively discriminates benign from malicious instances, a key advantage of blending linear and deep ensemble components to address the evolving signatures of ransomware.

Table 3. Evaluation metrics for the performance of the proposed WWDEM against three ensembles.

Features	Precision	Recall	F-Score	Accuracy	FPR
10	0.845	1.000	0.916	0.912	0.171
20	1.000	0.856	0.923	0.931	0.000
30	0.884	1.000	0.938	0.937	0.123
40	0.899	0.999	0.946	0.945	0.105
50	0.923	0.999	0.960	0.960	0.077

Table 4 shows the model's performance using four ensembles (C4) across the same incremental sets of features. As before, 50 features yield a high accuracy of 0.960, and 10 features result in an accuracy of 0.914. Precision peaks at a perfect 1.000 for 10 features, although the corresponding recall is lower (0.822), balancing the F1-score around 0.903. Similar patterns emerge for the other feature increments, generally showing improved recall and accuracy as the feature set grows. These results suggest that while certain minimal subsets of features can yield strong precision, they may not thoroughly capture the breadth of ransomware behaviors, hence the lower recall in some cases. In contrast, the expanded feature sets (particularly 40 and 50 features) help the model more comprehensively detect diverse ransomware activities, thus achieving a more favorable balance across all metrics. The fact that the FPR is maintained close to zero underscores the method's consistency in correctly identifying legitimate processes.

Table 4. Evaluation metrics for the performance of the proposed WWDEM against four ensembles.

Features	Precision	Recall	F-Score	Accuracy	FPR
10	1.000	0.822	0.903	0.914	0.000
20	0.878	1.000	0.935	0.933	0.130
30	0.894	0.999	0.944	0.943	0.110
40	0.999	0.894	0.943	0.948	0.001
50	0.999	0.917	0.956	0.960	0.001

Table 5 summarizes the performance of WWDEM with five ensembles for different feature sizes (10 to 50). Accuracy steadily improved from 0.915 with 10 features to 0.960 with 50 features. Similarly, precision increased significantly from 0.850 to 0.924, indicating improved accuracy in identifying ransomware samples correctly. Recall remained consistently high, ranging between 0.886 and 1.000 across different feature sets. The F1-score, combining precision and recall, also showed a clear upward trend, improving from 0.919 to 0.960. Meanwhile, the false positive rate (FPR) significantly dropped from 0.164 with 10 features to as low as 0.001 with 30 features before stabilizing around 0.076 with 50 features.

Table 5. Evaluation metrics for the performance of the proposed WWDEM against five ensembles.

Features	Precision	Recall	F-Score	Accuracy	FPR
10	0.850	1.000	0.919	0.915	0.164
20	0.879	1.000	0.936	0.934	0.128
30	0.999	0.886	0.939	0.945	0.001
40	0.905	0.999	0.950	0.949	0.097
50	0.924	0.999	0.960	0.960	0.076

The results in Table 5 demonstrate that increasing feature size generally enhanced the performance of WWDEM. A larger set of selected features improved the model's precision, indicating fewer false alerts and better reliability in distinguishing ransomware attacks from normal activities. Notably, the high recall values indicate consistent capability to detect actual ransomware instances, with only slight fluctuations. The optimal balance between low FPR and high accuracy occurs around 30–40 features, suggesting this range provides the best compromise between precision and recall. Overall, WWDEM effectively handles behavioral drift, maintaining robust and accurate detection performance as the feature set increases.

In Table 6, WWDEM leverages six ensembles (C6), again reporting metrics for 10, 20, 30, 40, and 50 features. Notably, the model trained on 10 features displays perfect precision (1.000) and a recall of 0.824, culminating in an accuracy of 0.915. As the feature set expands to 50, precision remains near-perfect at 0.997, and recall increases to 0.932, leading to an accuracy of 0.966. F1-score also climbed accordingly, indicating a sound balance between catching ransomware and avoiding false alarms. Such a pattern demonstrates the consistent enhancement of model effectiveness through a more diverse feature base. The increasing recall rates indicate that the model becomes more adept at capturing subtle ransomware behaviors, while precision remains high even when a substantial number of features are involved. Throughout all feature scenarios, the FPR values remain low, confirming that the system does not sacrifice specificity to achieve improved recall.

Table 6. Evaluation metrics for the performance of the proposed WWDEM against six ensembles.

Features	Precision	Recall	F-Score	Accuracy	FPR
10	1.000	0.824	0.904	0.915	0.000
20	1.000	0.863	0.927	0.934	0.000
30	0.898	1.000	0.946	0.945	0.105
40	0.909	1.000	0.952	0.952	0.093
50	0.997	0.932	0.963	0.966	0.003

Table 7 presents metrics under the configuration with seven ensembles (C7). Once again, distinct feature settings are provided, with 50 features achieving the highest accuracy at 0.971, a precision of 0.948, and a recall of 0.994. At the lower end, with only 10 features, the model still attains an accuracy of 0.918 and a perfect precision of 1.000, although recall is comparatively lower. Across all entries, the FPR remains below 0.120, illustrating the method's ability to avoid frequent false positives. These results highlight that increasing the number of ensembles can bolster detection rates. When the model is configured with C7, it better integrates knowledge from diverse subsets of features, pushing overall accuracy and recall to higher levels than the smaller ensemble configurations. Consequently, as with previous tables, the combination of more comprehensive feature sets and additional ensembles further refines the detection of evolving ransomware variants.

Table 7. Evaluation metrics for the performance of the proposed WWDEM against seven ensembles.

Features	Precision	Recall	F-Score	Accuracy	FPR
10	1.000	0.829	0.907	0.918	0.000
20	0.888	1.000	0.941	0.939	0.117
30	0.999	0.888	0.940	0.945	0.001
40	0.912	1.000	0.953	0.953	0.090
50	0.948	0.994	0.970	0.971	0.051

Table 8 presents the detection rate (DR) for WWDEM across all ensemble configurations (C3, C4, C5, C6, and C7) and varying feature counts (10, 20, 30, 40, and 50). The DR consistently increases as both the ensemble size and the feature subset increase. The lowest detection rate of 0.912 is observed when the model is configured with C3 on a 10-feature set, while the highest detection rate of 0.971 is obtained with C7 on 50 features. This progression reaffirms the collective findings that larger ensemble configurations and more extensive feature sets enhance the detection of sophisticated ransomware behaviors. The table clearly shows how each incremental addition of features or ensembles helps the model recognize a wider array of malicious traits, culminating in near-optimal coverage of known and emerging variants. The fact that all detection rates exceed 0.900 in each tested scenario further underscores the overall consistency of WWDEM.

Table 8. Wide and weighted deep ensemble model DR against different numbers of features and ensembles.

Features	C3	C4	C5	C6	C7
10	0.912	0.914	0.915	0.915	0.918
20	0.931	0.933	0.934	0.934	0.939
30	0.937	0.943	0.945	0.945	0.945
40	0.945	0.948	0.949	0.952	0.953
50	0.960	0.960	0.960	0.966	0.971

Table 9 compares WWDEM when using different ensemble sizes (C3 through C7) with two state-of-the-art models: the Enhanced Anomaly Behavioral Detection Model and the Hybrid Distinct Ensemble Model. WWDEM outperforms both baselines across all listed metrics: precision, recall, F1-score, accuracy, false positive rate (FPR), and detection rate (DR). Notably, when WWDEM uses C7 and 50 features, the detection rate peaks at 0.971, surpassing the 0.508 and 0.525 from the two baseline approaches by a significant margin. These results confirm the robustness and general superiority of the proposed method. While the baseline models show moderate performance, particularly in handling complex or variant-intensive attacks, WWDEM consistently reports better precision and recall. Additionally, the FPR is drastically lower in the proposed model (0.051 at best), implying fewer misclassifications of benign processes as malicious. This combination of high effectiveness and low false alarms is a hallmark of a reliable ransomware detection system.

Table 9. Comparison of wide and weighted deep ensemble model with Enhanced Anomaly Behavioral Detection Model and the Hybrid Distinct Ensemble Model.

Metric	Wide & Weighted Deep Ensemble Model					Enhanced Anomaly Behavioral Detection Model	Hybrid Distinct Ensemble Model
	C3	C4	C5	C6	C7		
Precision	0.923	0.999	0.924	0.997	0.948	0.819	0.810
Recall	0.999	0.917	0.999	0.932	0.994	0.647	0.775
F1-score	0.960	0.956	0.960	0.963	0.970	0.708	0.788
Accuracy	0.960	0.960	0.960	0.966	0.971	0.763	0.787
FPR	0.077	0.001	0.076	0.003	0.051	0.120	0.195
DR	0.960	0.960	0.960	0.966	0.971	0.508	0.525

4.6. Comparison with Related Solutions

Figure 3 compares precision scores across different ensemble configurations for the proposed WWDEM model and the two baseline models. The bars clearly demonstrate that WWDEM achieves consistently higher precision, with configurations involving larger ensembles (C6 and C7) outmatching the smaller ones (C3). In all setups, it stands notably above Enhanced Anomaly Behavioral Detection Model and Hybrid Distinct Ensemble Model. This superior precision is crucial from a practical standpoint because it indicates that WWDEM seldom flags benign processes as malicious. The high precision metrics reflect the model's ability to learn highly discriminative features, thus minimizing the risk of interrupting legitimate user activities or business processes due to false positives.

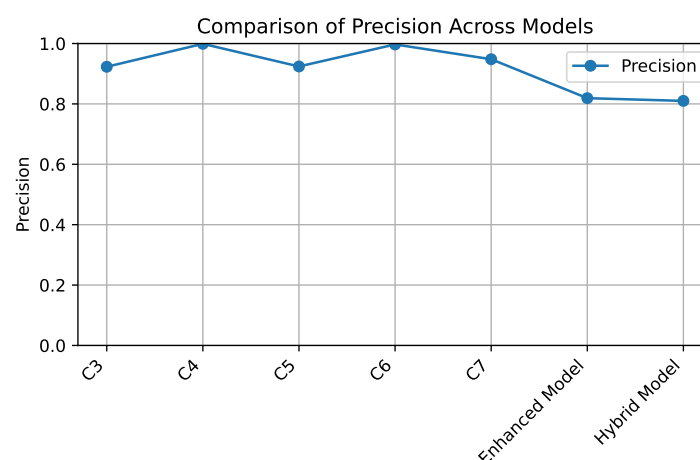


Figure 3. Precision evaluation of the wide and weighted deep ensemble model across varying numbers of features and ensembles.

The improvement in precision in Figure 3 stems from the weighted deep ensemble component of WWDEM, which methodically assigns importance to key features while filtering out less indicative ones. By ‘ensembling’ multiple deep networks, each with a unique view of the input space, the model elevates indicators that reliably distinguish malicious from normal behavior. In comparison, older or simpler anomaly-based systems may define benign baselines in a static manner, leaving them susceptible to misclassifying normal variations as threats. Consequently, the approach showcased here suggests a more sophisticated path forward, leveraging wide-and-deep synergy to enhance indicator-specific weighting and, thus, produce fewer errant alerts—a pressing concern in real-world security operations.

Figure 4 focuses on recall, highlighting how effectively each model detects ransomware among all malicious instances. WWDEM, especially when using higher ensemble numbers, consistently achieves stronger recall than the two baseline methods. This is most evident for the C7 configuration, which approaches or surpasses the 0.90–0.95 range, indicating the model’s proficiency at catching the majority of ransomware attacks in the dataset. Such a high recall is pivotal in cybersecurity, where failing to detect a ransomware threat can lead to severe operational and financial damage. Even a small gap in detection capability can be exploited, making the margin of improvement here particularly valuable in practical scenarios.

Analytically, the recall strength of WWDEM is tied to the ensemble-based generalization strategy. As ransomware evolves, certain features become temporarily dominant, and others recede in importance. The weighted deep networks within the model can dynamically re-assign emphasis to these shifts. In contrast, classical detection solutions, which rely heavily on a narrower set of historically prominent features, may miss signals tied to nascent ransomware variants. Furthermore, this outcome also mirrors the advantage of the wide part that integrates well-established indicators—sustaining recall for known threats—alongside deep ensembles that discover unfamiliar or obscure characteristics. Therefore, Figure 5 substantiates how the comprehensive approach helps the proposed model achieve near-complete coverage of malicious activities.

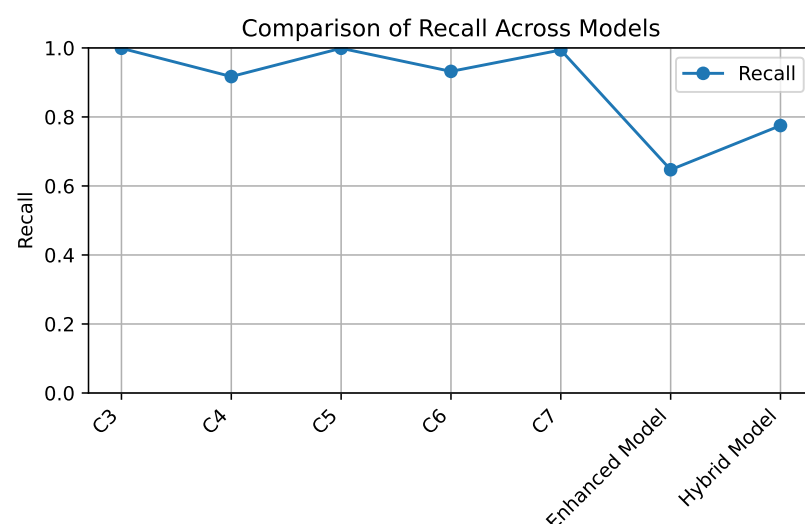


Figure 4. Recall evaluation of the wide and weighted deep ensemble model across varying numbers of features and ensembles.

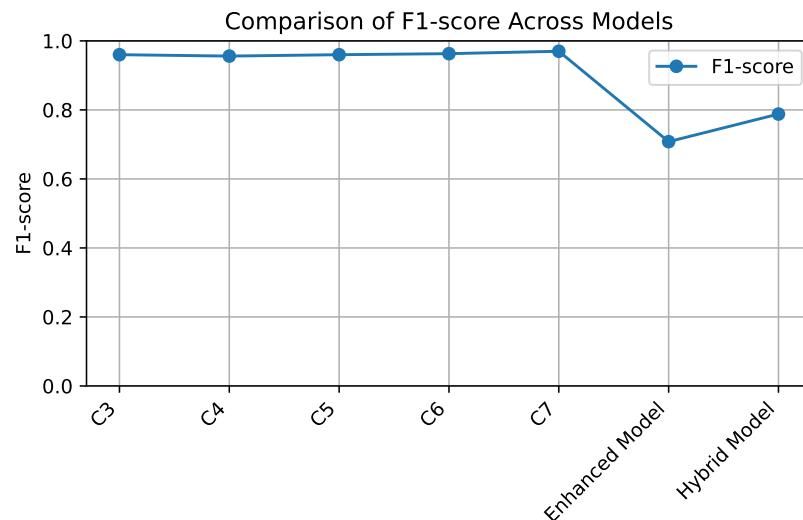


Figure 5. F1-score evaluation of the wide and weighted deep ensemble model across varying numbers of features and ensembles.

In Figure 5, the F1-scores are compared for WWDEM under various ensemble configurations and for the baseline models. The F1-score, combining both precision and recall, reveals the overall detection effectiveness of each approach. The results confirm that the proposed solution leads in F1-score, with the best-performing configuration nearing perfect equilibrium between avoiding false positives (precision) and capturing actual threats (recall). The margin by which WWDEM outperforms the existing methods is consistent across different ensemble sizes. Even the smaller ensembles (C3) show respectable F1-scores, signifying a robust foundational approach, but the metric improves further as the number of ensembles increases. This demonstrates that added ensemble diversity refines the ability to detect various strains of ransomware.

The high F1-scores illustrated in Figure 5 reinforce the fact that balancing memorization of key features and generalization for unforeseen attacks is pivotal in maintaining an all-around strong detection performance. This synergy spares the system from heavily skewing towards either precision or recall, a common pitfall in simpler models, where emphasizing one metric often compromises the other. Such balance is especially significant in organizational security environments, where both missed detections and numerous false alarms carry high stakes. By consistently excelling in F1-score, WWDEM exhibits a practical readiness to handle real-world ransomware threats. It also provides empirical evidence that addressing the shortcomings of existing methods—such as narrow feature reliance and static detection timelines—yields more balanced detection outcomes.

Figure 6 displays the accuracy rates of all tested models under various ensemble configurations, with WWDEM notably achieving the highest scores. In particular, configurations with a larger number of ensembles (C6 and C7) approach accuracy levels above 0.95. In contrast, Enhanced Anomaly Behavioral Detection Model and Hybrid Distinct Ensemble Model exhibit comparatively modest accuracies. Accuracy is a straightforward yet essential metric, depicting how many predictions out of all attempts are correct. The success of WWDEM here signifies that it can effectively distinguish benign from malicious behavior in most scenarios, validating the model's capacity for correct overall classification across a broad ransomware sample set.

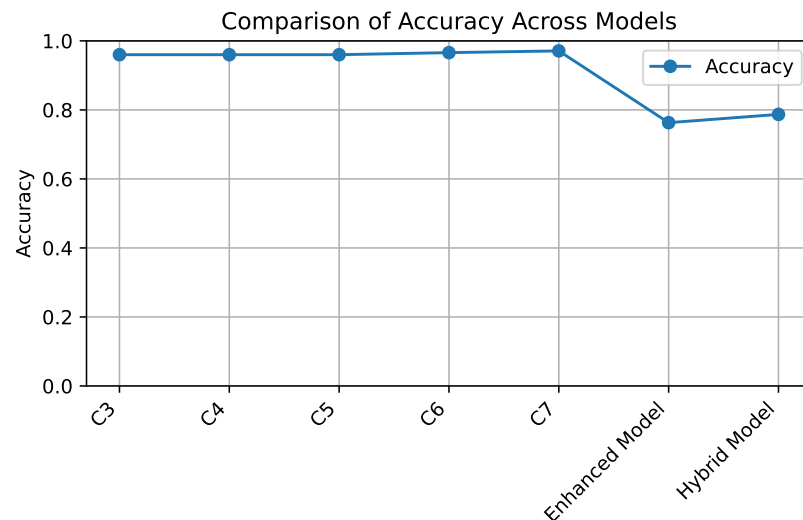


Figure 6. Accuracy evaluation of the wide and weighted deep ensemble model across varying numbers of features and ensembles.

The improved accuracy seen in Figure 6 reveals the ability of WWDEM to unify diverse feature sets and classification strategies. In cases where older models might rely on rigid, phase-specific features or purely anomaly-based thresholds, the proposed approach dynamically weighs multiple deep networks, preventing misclassifications that arise from short-lived or misleading feature signals. The consistency of high accuracy across ensembles also suggests resilience against adversarial behaviors that attempt to mimic benign processes. By relying on a wide breadth of features and the synergy of memorization-generalization, the model remains less prone to being deceived. Consequently, this adaptability represents a direct solution to the limitations of static, single-phased detection frameworks that struggle with rapidly shifting ransomware code patterns.

Figure 7 shows the comparison of the false positive rates (FPRs) for WWDEM at different ensemble sizes against the baseline methods. The graph shows that WWDEM systematically achieves much lower FPRs, especially when operating at higher ensemble counts (C6 or C7). Both baseline solutions produce substantially higher FPRs, indicating that they are more likely to label benign processes as malicious. Keeping the FPR low is critical for maintaining normal system operations. Security teams often rely on the FPR to gauge how frequently the system triggers unnecessary alerts. Excessive false positives can lead to “alert fatigue”, in which genuine threats might eventually be overlooked.

The reduced FPR in Figure 7 underscores the model’s refined approach of assigning precise weights to each ensemble’s judgment. By reconciling multiple classifiers, WWDEM offsets the tendency of any single, possibly overfitted ensemble to mistakenly flag benign behaviors. This method diverges from classical anomaly detection, which often casts a broader net at the cost of more frequent false alarms. This attribute also has direct implications for real-time ransomware defense, where false positives could disrupt legitimate application operations or tarnish the model’s credibility among users. Hence, the proposed ensemble design stands out as not only effective but also judicious in preserving the stability of everyday system usage—a key benefit over older detection systems.

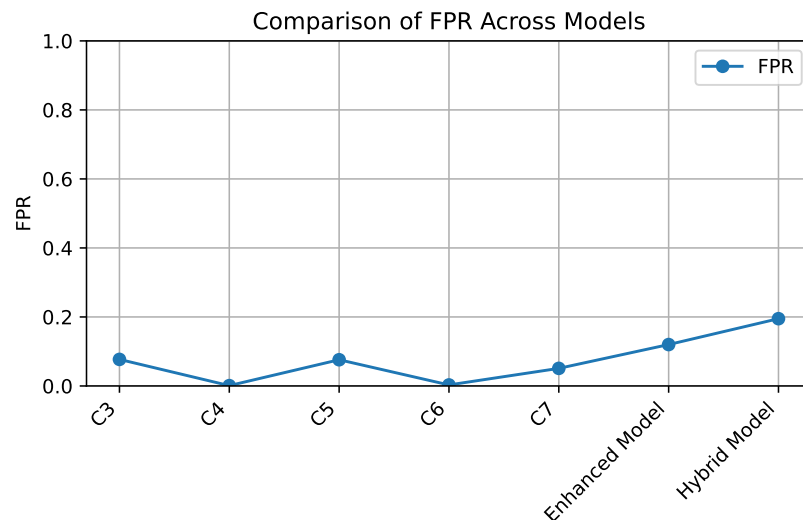


Figure 7. FPR evaluation of the wide and weighted deep ensemble model across varying numbers of features and ensembles.

Figure 8 shows the detection rate (DR) for WWDEM when using various ensemble sizes and for the comparative models. The proposed model consistently outperforms the others, achieving detection rates close to or exceeding 0.90 in all ensemble configurations. As expected, the highest DR occurs with configurations that utilize both a larger number of ensembles and a more inclusive feature set, reaching peaks above 0.95 and reinforcing the findings from the tabular data. The DR metric highlights the proportion of the total samples—both benign and malicious—that are correctly labeled as ransomware. In scenarios where there are advanced, persistent threats, timely and accurate detection is essential, and a high DR ensures that new or variant-heavy families of ransomware are not missed.

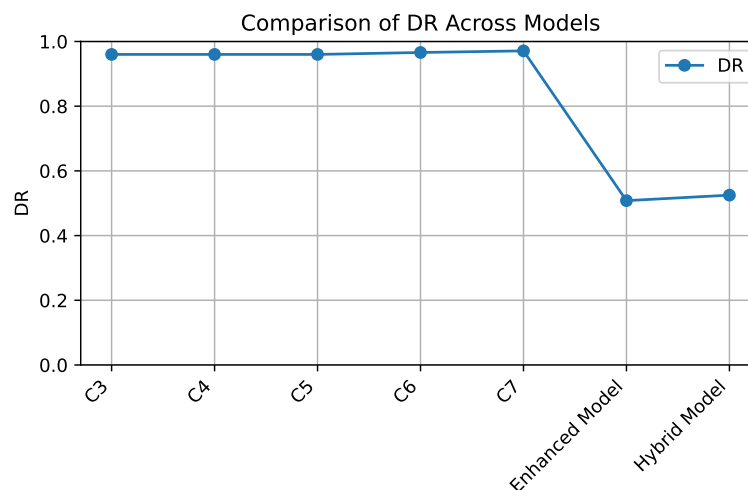


Figure 8. DR evaluation of the wide and weighted deep ensemble model across varying numbers of features and ensembles.

The higher DR displayed in Figure 8 is invaluable. It indicates that despite the constantly shifting landscape of ransomware behaviors, WWDEM is highly adept at catching even unconventional or newly emerging attack vectors. This success is attributable to the model's fundamental design principle: leveraging memorized cues in the wide part and exploring unexplored feature combinations through multiple deep ensembles. Furthermore, the comparative advantage over existing detection techniques suggests that

WWDEM effectively addresses historical challenges, such as reliance on static signatures or single-phase data. By uniting memorization, generalization, and weighted ensembling, WWDEM furnishes an advanced, future-ready tool for cybersecurity, mitigating both the risk of missing novel attacks and the disruptions from misclassifications.

Figure 9 presents a side-by-side comparison of the models' average performance metrics, including precision, recall, F1-score, accuracy, FPR, and DR. The performance curves of WWDEM remain consistently higher than the baselines across the majority of metrics, illustrating its dominant presence in detection accuracy and recall. Notably, the proposed model's FPR bars remain on the lower end, signifying a strong ability to minimize false alarms while still capturing malicious activities. The bar chart underscores that, regardless of the metric used for comparison, WWDEM generally retains its lead over the competing models. The visual representation also clarifies how each ensemble configuration (C3 through C7) contributes to incremental gains. This holistic snapshot demonstrates the efficiency of combining varied feature subsets with multiple ensemble learners, ultimately leading to heightened reliability.

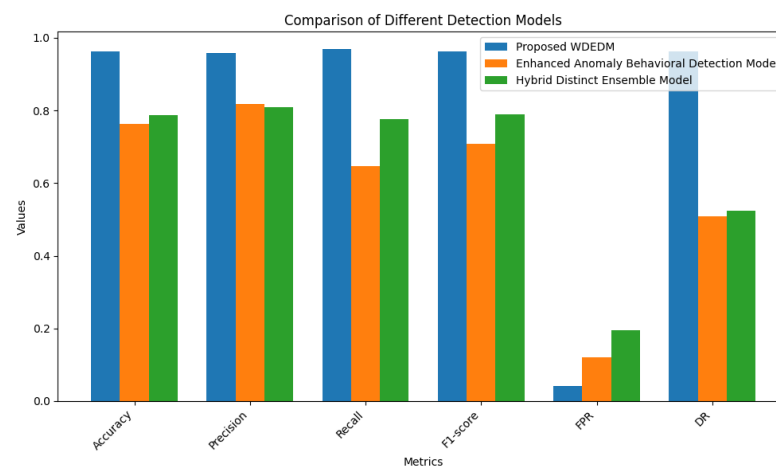


Figure 9. Performance comparison between the proposed and related models (averaged in terms of accuracy, precision, recall, f-score, false positive rate, and detection rate).

From a broader perspective, the aggregated performance in Figure 9 highlights how WWDEM systematically addresses limitations identified in older techniques. Rather than relying on a static cluster of historically relevant features, the model capitalizes on a twofold pipeline—memorizing established malicious signatures in the wide network while deploying ensemble-based deep learners to generalize potential new attack vectors. In practical terms, this combination aligns with the industry's growing recognition that adaptive, multi-module architectures can offer stronger resilience against ransomware rapid evolution. The results, thus, confirm that the proposed approach effectively closes the following gap in the literature: while many systems excel at either memorization or generalization, few have successfully integrated both in a single pipeline to tackle concept drift and the varied nature of ransomware threats.

To validate the effectiveness of the proposed wide and weighted deep ensemble model (WWDEM), a comparative analysis against state-of-the-art approaches was conducted. Table 10 summarizes the comparative performance of WWDEM against baseline models using three key parameters: accuracy, F1-score, and false positive rate (FPR). This comparison shows that the proposed WWDEM outperformed the baseline models across all three evaluation metrics, demonstrating its effectiveness in detecting evolving ransomware behaviors while minimizing false alarms. Specifically, the results show that WWDEM achieved notably higher accuracy (0.937), recall (0.971), and precision (0.937), reflecting its balanced

performance in correctly identifying ransomware instances while limiting false positives. WWDEM also achieved a lower false positive rate (FPR) of 0.095 compared to 0.120 and 0.195 from Enhanced Anomaly Behavioral Detection Model and Hybrid Distinct Ensemble Model, respectively. This indicates WWDEM is both more accurate and reliable, effectively detecting ransomware while minimizing false alerts. Overall, these averages confirm that WWDEM consistently outperforms baseline methods across all measured parameters.

Table 10. Comparative performance of WWDEM (average) and baseline models.

Model	Accuracy	Precision	Recall	F1-Score	FPR	DR
WWDEM (Proposed)	0.937	0.910	0.971	0.937	0.095	0.937
Enhanced Anomaly Behavioral Detection	0.763	0.819	0.647	0.708	0.120	0.508
Hybrid Distinct Ensemble Model	0.787	0.810	0.775	0.788	0.195	0.525

5. Analysis and Discussion

This section presents a detailed discussion of the results obtained in the prior sections. The analysis focuses on evaluating the improvements demonstrated by the proposed model while considering the various rationales for performance degradation observed in the comparison models. Existing studies have primarily focused on static timeline concepts, which assume that behavioral patterns over time remain constant. These studies did not consider the concept of behavioral drift associated with evolving ransomware variants over different timeframes. By relying on historical data within static timeframes, these models are limited in their ability to detect emerging or evolving threats, particularly those caused by ransomware variants that adapt over time. Furthermore, many anomaly detection studies have attempted to address behavioral drift, but they remain limited in their detection precision and accuracy because they were developed only by considering “normal” profiles. The benign crypto-ransomware attacks have shown, to some extent, the potential to bypass anomaly-based solutions.

Thus, anomaly-based solutions may struggle to detect new variants of ransomware consistently. Another limitation arises from studies that use pre-encryption data. These models are restricted by the limited availability of data, which can lead to false alarms, as both benign programs and ransomware often utilize the same cryptographic APIs. Moreover, these studies did not consider the evolving nature of ransomware variants, which further limits their applicability and effectiveness. Comparison studies also reported lower results due to the reasons mentioned above. These studies were developed without considering attack progression in different timeframes, which is crucial for detecting evolving ransomware variants. On the other hand, the proposed model addresses the concept of behavioral drift by incorporating informative and redundant features to uncover hidden patterns that could be developed. Effectively addressing behavioral drift is critical for designing a strong model capable of detecting evolving ransomware variants. If behavioral drift is not properly considered, the model may misclassify future ransomware variants as benign programs that could lead to a failure in detection.

The results of the proposed model, which are presented in Figures 3–8, demonstrate its superior performance across several metrics like precision, recall, F1-score, accuracy, FPR, and DR, respectively. The accuracy of the wide and weighted deep model in Figure 6 is higher than that of the comparison studies, highlighting the reliability of the proposed work. The model’s ability to correctly identify ransomware is reflected in its high precision in Figure 3 and recall in Figure 4, which together support the model’s effectiveness. In Figure 5, the F1-score further emphasizes the model’s relevance and sensitivity to the problem, with the highest value indicating that the proposed model performs an optimal balance between precision and recall. Figures 7 and 8 show the model’s performance in

minimizing FPR and maximizing DR, which are crucial for ensuring the quality and accuracy of the detection system. Table 10 summarizes the performance across all evaluation metrics. These summaries demonstrate the effectiveness of the proposed model in detecting ransomware variants exhibiting behavioral drift. By considering all relevant features, the model successfully addresses the behavioral drift concept and improves detection performance. The proposed work utilized the potential patterns of left-out features representing different timeframes, enhancing the model's ability to predict future behavior.

The reliability and confidence in detecting unseen patterns improve as the number of classifiers increases. Moreover, the inclusion of more informative features significantly enhances model relevance. The use of the most informative features has a good impact on the evaluation metrics for the proposed work. An increase in the number of informative features also displayed a significant effect on the generation of better results. The improved results against comparison studies are due to the availability of enough patterns to emphasize the certainty of detection as the characterization of ransomware behavior is mostly captured.

In contrast, studies based only on pre-encryption data underperform due to the limited volume of available data that is insufficient to develop a detection profile. The slight degradation in the results could be due to the availability of redundant features in the left-out features or to the challenge of estimating features at time t_1 , and they may better represent ransomware behavior at time t_2 . However, despite these limitations, the proposed work still outperforms comparison studies due to the consideration of memorization and generalization concepts.

Our study quantifies behavioral drift by employing a joint learning framework wherein both the 'wide' model (memorizing historically important features) and the 'deep' ensemble (generalizing emerging patterns) are trained concurrently. Within this setup, each base classifier in the deep ensemble is dynamically weighted based on its performance over newly observed data segments. As a result, when the model begins to see deviations from previously learned patterns, the weights of classifiers poorly suited to the new behavior decrease, signaling the onset of drift. Conversely, classifiers adept at recognizing newly relevant features gain higher weights, thus capturing the behavioral shift more accurately. This dynamic weighting not only serves as an internal quantification of drift but also enhances detection performance by ensuring that the most effective classifiers for recent ransomware variations drive the final decision.

Overall, the proposed model demonstrates significant improvements over existing studies. These good results suggest its implication for detecting evolving ransomware variants. The good results of the proposed model are due to the use of memorization and generalization concepts. These concepts make the model well-suited for ransomware detection displaying behavioral drifting. Additionally, this work gains worth due to the use of all the significant features left out during the second phase. The ability to consider time windows and evolving ransomware behavior further strengthens the model's performance, ensuring it remains effective over time.

6. Conclusions

This study proposes a wide and weighted deep ensemble model for early ransomware detection. The model is divided into two main components, each addressing a key concept: one focuses on memorization, and the other enhances generalization. To further improve the model, ensembling was applied to the deep part, and various ensemble configurations were created to introduce diversity and strengthen performance. Each ensemble was evaluated using standard metrics to assess its effectiveness. The results of these evaluations were compared against considered state-of-the-art studies. The comparison results showed

that the proposed work has better performance results compared to the existing work. Notably, the proposed model excelled in detecting evolving ransomware variants, showing its strengths and adaptability to new threats. This work provides a significant step forward in ransomware detection, particularly in addressing the challenges posed by behavioral drift and the evolving nature of ransomware.

Author Contributions: Conceptualization, U.U.; Methodology, U.U. and B.A.S.A.-r.; Software, U.U., B.A.S.A.-r., M.G. and M.A.; Validation, U.U., B.A.S.A.-r., A.Z., S.S. and F.S.; Formal analysis, U.U.; Investigation, U.U. and E.A.; Resources, B.A.S.A.-r., M.G. and A.Z.; Data curation, B.A.S.A.-r. and E.A.; Writing—original draft, U.U.; Writing—review & editing, B.A.S.A.-r., A.Z., M.A., S.S. and F.S.; Supervision, B.A.S.A.-r. and A.Z.; Project administration, B.A.S.A.-r., M.G. and F.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alissa, A.K.; Elkamchouchi, H.D.; Tarmissi, K.; Yafoz, A.; Alsini, R.; Alghushairy, O.; Mohamed, A.; Al Duhayyim, M. Dwarf mongoose optimization with machine-learning-driven ransomware detection in internet of things environment. *Appl. Sci.* **2022**, *12*, 9513. [\[CrossRef\]](#)
2. Al-Rimy, B.A.S.; Maarof, M.A.; Shaid, S.Z.M. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Comput. Secur.* **2018**, *74*, 144–166.
3. De Gaspari, F.; Hitaj, D.; Pagnotta, G.; De Carli, L.; Mancini, L.V. Evading behavioral classifiers: A comprehensive analysis on evading ransomware detection techniques. *Neural Comput. Appl.* **2022**, *34*, 12077–12096.
4. Cen, M.; Jiang, F.; Qin, X.; Jiang, Q.; Doss, R. Ransomware early detection: A survey. *Comput. Netw.* **2024**, *239*, 110138.
5. Kok, S.; Abdullah, A.; Jhanjhi, N. Early detection of crypto-ransomware using pre-encryption detection algorithm. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 1984–1999.
6. Hull, G.; John, H.; Arief, B. Ransomware deployment methods and analysis: Views from a predictive model and human responses. *Crime Sci.* **2019**, *8*, 1–22.
7. Chen, Q.; Islam, S.R.; Haswell, H.; Bridges, R.A. Automated ransomware behavior analysis: Pattern extraction and early detection. In Proceedings of the Science of Cyber Security: Second International Conference, SciSec 2019, Nanjing, China, 9–11 August 2019; pp. 199–214.
8. Cen, M.; Deng, X.; Jiang, F.; Doss, R. Zero-Ran Sniff: A zero-day ransomware early detection method based on zero-shot learning. *Comput. Secur.* **2024**, *142*, 103849.
9. Micro, T. *A Deep Dive into the Evolution of Ransomware Part 1*; TREND: Irving, TX, USA, 2023.
10. Singh, N.; Tripathy, S. Unveiling the veiled: An early stage detection of fileless malware. *Comput. Secur.* **2025**, *150*, 104231. [\[CrossRef\]](#)
11. Chauhan, R.; Heydari, S.S. Polymorphic Adversarial DDoS attack on IDS using GAN. In Proceedings of the 2020 International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 20–22 October 2020; pp. 1–6.
12. Ransomware, M. Behavioural Analysis of Recent Ransoms and Prediction of Future Attacks by Polymorphic. In *Computational Intelligence: Theories, Applications and Future Directions-Volume II: ICCI-2017*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 799, p. 65.
13. Daku, H.; Zavarisky, P.; Malik, Y. Behavioral-based classification and identification of ransomware variants using machine learning. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 1560–1564.
14. Chen, Y.; Ding, Z.; Wagner, D. Continuous Learning for Android Malware Detection. *arXiv* **2023**, arXiv:2302.04332.
15. García, D.E.; DeCastro-García, N.; Castañeda, A.L.M. An effectiveness analysis of transfer learning for the concept drift problem in malware detection. *Expert Syst. Appl.* **2023**, *212*, 118724.
16. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [\[CrossRef\]](#)
17. Al-Rimy, B.A.S.; Maarof, M.A.; Alazab, M.; Alsolami, F.; Shaid, S.Z.M.; Ghaleb, F.A.; Al-Hadhrami, T.; Ali, A.M. A pseudo feedback-based annotated TF-IDF technique for dynamic crypto-ransomware pre-encryption boundary delineation and features extraction. *IEEE Access* **2020**, *8*, 140586–140598. [\[CrossRef\]](#)

18. Fernando, D.W.; Komninos, N.; Chen, T. A Study on the Evolution of Ransomware Detection Using Machine Learning and Deep Learning Techniques. *IoT* **2020**, *1*, 551–604. [\[CrossRef\]](#)
19. Gazzan, M.; Sheldon, F.T. Opportunities for early detection and prediction of ransomware attacks against industrial control systems. *Future Internet* **2023**, *15*, 144. [\[CrossRef\]](#)
20. Jiang, Y.; Natekar, P.; Sharma, M.; Aithal, S.K.; Kashyap, D.; Subramanyam, N.; Lassance, C.; Roy, D.M.; Dziugaite, G.K.; Gunasekar, S. Methods and analysis of the first competition in predicting generalization of deep learning. *Proc. Mach. Learn. Res.* **2021**, *133*, 170–190.
21. Mungoli, N. Adaptive Feature Fusion: Enhancing Generalization in Deep Learning Models. *arXiv* **2023**, arXiv:2304.03290.
22. Al-rimy, B.A.S.; Maarof, M.A.; Shaid, S.Z.M. Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Gener. Comput. Syst.* **2019**, *101*, 476–491. [\[CrossRef\]](#)
23. Sharmeen, S.; Ahmed, Y.A.; Huda, S.; Koçer, B.; Hassan, M.M. Avoiding future digital extortion through robust protection against ransomware threats using deep learning based adaptive approaches. *IEEE Access* **2020**, *8*, 24522–24534. [\[CrossRef\]](#)
24. Kritika, E. A comprehensive literature review on ransomware detection using deep learning. *Cyber Secur. Appl.* **2025**, *3*, 100078. [\[CrossRef\]](#)
25. Idrees, F.; Rajarajan, M.; Conti, M.; Chen, T.M.; Rahulamathavan, Y. PIndroid: A novel Android malware detection system using ensemble learning methods. *Comput. Secur.* **2017**, *68*, 36–46. [\[CrossRef\]](#)
26. Sharma, S.; Challa, R.K.; Kumar, R. An ensemble-based supervised machine learning framework for android ransomware detection. *Int. Arab J. Inf. Technol.* **2021**, *18*, 422–429. [\[CrossRef\]](#)
27. Saminathan, Y.; Lourdasamy, R. Deep Ensemble Classifier for Ransomware Identification Using Digitalized DNA Genotyping System. *Int. J. Intell. Eng. Syst.* **2022**, *15*, 503–510.
28. Ahmed, U.; Lin, J.C.W.; Srivastava, G. Mitigating adversarial evasion attacks of ransomware using ensemble learning. *Comput. Electr. Eng.* **2022**, *100*, 107903.
29. Tasnim, N.; Shahriar, K.T.; Alqahtani, H.; Sarker, I.H. Ransomware family classification with ensemble model based on behavior analysis. In *Proceedings of the Machine Intelligence and Data Science Applications: Proceedings of MIDAS 2021*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 609–619.
30. Aurangzeb, S.; Anwar, H.; Naeem, M.A.; Aleem, M. BigRC-EML: Big-data based ransomware classification using ensemble machine learning. *Clust. Comput.* **2022**, *25*, 3405–3422. [\[CrossRef\]](#)
31. Singh, A.; Mushtaq, Z.; Abosaq, H.A.; Mursal, S.N.F.; Irfan, M.; Nowakowski, G. Enhancing ransomware attack detection using transfer learning and deep learning ensemble models on cloud-encrypted data. *Electronics* **2023**, *12*, 3899. [\[CrossRef\]](#)
32. Yuan, W.; Wang, H.; Hu, B.; Wang, L.; Wang, Q. Wide and deep model of multi-source information-aware recommender system. *IEEE Access* **2018**, *6*, 49385–49398. [\[CrossRef\]](#)
33. Niu, C.; Zhong, G.; Liu, Y.; Zhang, Y.; Sun, Y.; He, A.; Chen, Z. Structured Semantic Model supported Deep Neural Network for Click-Through Rate Prediction. *arXiv* **2018**, arXiv:1812.01353.
34. Liu, B.; Tang, R.; Chen, Y.; Yu, J.; Guo, H.; Zhang, Y. Feature generation by convolutional neural network for click-through rate prediction. In *Proceedings of the 2019 World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019*; pp. 1119–1129.
35. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M. Wide & deep learning for recommender systems. In *Proceedings of the 2016 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016*; pp. 7–10.
36. Jais, I.K.M.; Ismail, A.R.; Nisa, S.Q. Adam optimization algorithm for wide and deep neural network. *Knowl. Eng. Data Sci.* **2019**, *2*, 41–46. [\[CrossRef\]](#)
37. Gu, K.; Liu, H.; Xia, Z.; Qiao, J.; Lin, W.; Thalmann, D. PM2.5 monitoring: Use information abundance measurement and wide and deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4278–4290. [\[CrossRef\]](#)
38. Chow, T.; Kan, Z.; Linhardt, L.; Cavallaro, L.; Arp, D.; Pierazzi, F. Drift forensics of malware classifiers. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, Singapore, 18–22 September 2023*; pp. 197–207.
39. Manikandaraja, A.; Aaby, P.; Pitropakis, N. Rapidrift: Elementary Techniques to Improve Machine Learning-Based Malware Detection. *Computers* **2023**, *12*, 195. [\[CrossRef\]](#)
40. Ali, E.; Batool, N.; Rizwan, M.; Sarwar, S. Assessing Concept Drift in Malware: A Comprehensive Review and Analysis. In *Proceedings of the 2024 21st International Bhurban Conference on Applied Sciences and Technology (IBCAST), Murree, Pakistan, 20–23 August 2024*; pp. 564–569.
41. Almaleh, A.; Almushabb, R.; Ogran, R. Malware API calls detection using hybrid logistic regression and RNN model. *Appl. Sci.* **2023**, *13*, 5439. [\[CrossRef\]](#)
42. Gaber, M.; Ahmed, M.; Janicke, H. Defeating evasive malware with peekaboo: Extracting authentic malware behavior with dynamic binary instrumentation. *Preprint* **2024**. [\[CrossRef\]](#)

43. Cui, J.; Leng, B.; Wang, X.; Wang, F.; Yang, J. Malware behavior detection method based on reinforcement learning. In Proceedings of the International Conference on Computer Application and Information Security (ICCAIS 2022), Wuhan, China, 23–24 December 2022; Volume 12609, pp. 380–389.
44. Feng, P.; Yang, L.; Lu, D.; Xi, N.; Ma, J. BejaGNN: Behavior-based Java malware detection via graph neural network. *J. Supercomput.* **2023**, *79*, 15390–15414.
45. Al-Rimy, B.A.S.; Maarof, M.A.; Alazab, M.; Shaid, S.Z.M.; Ghaleb, F.A.; Almalawi, A.; Ali, A.M.; Al-Hadhrami, T. Redundancy coefficient gradual up-weighting-based mutual information feature selection technique for crypto-ransomware early detection. *Future Gener. Comput. Syst.* **2021**, *115*, 641–658.
46. Choudhary, S.; Vidyarthi, M.D. A simple method for detection of metamorphic malware using dynamic analysis and text mining. *Procedia Comput. Sci.* **2015**, *54*, 265–270. [[CrossRef](#)]
47. Alshamrani, A.F.; Alliheedi, M.A. Enhancing Malware Detection by Integrating Machine Learning with Cuckoo Sandbox. *arXiv* **2023**, arXiv:2311.04372.
48. Guarnieri, C. Installation. 2022. Available online: <https://cuckoo.readthedocs.io/en/latest/installation/> (accessed on 13 February 2025).
49. Ahmed, Y.A.; Koçer, B.; Huda, S.; Al-rimy, B.A.S.; Hassan, M.M. A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *J. Netw. Comput. Appl.* **2020**, *167*, 102753.
50. Gupta, D.; Rani, R. Improving malware detection using big data and ensemble learning. *Comput. Electr. Eng.* **2020**, *86*, 106729.
51. Kuncheva, L.I. *Combining Pattern Classifiers: Methods and Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
52. Urooj, U.; Al-Rimy, B.A.S.; Binti Zainal, A.; Saeed, F.; Abdelmaboud, A.; Nagmeldin, W. Addressing Behavioral Drift in Ransomware Early Detection Through Weighted Generative Adversarial Networks. *IEEE Access* **2023**, *12*, 3910–3925.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.