scientific reports

Check for updates

OPEN

Transfer learning with XAI for robust malware and IoT network security

Ahmad Almadhor[1], Shtwai Alsubai[2], Natalia Kryvinska[3]✉, Abdullah Al Hejaili[4], Belgacem Bouallegue[5], Mohamed Ayari[6] & Sidra Abbas[7]✉

Malware that exploits user privacy has increased in recent decades, and this trend has been linked to shifting international regulations, the expansion of Internet services, and the growth of electronic commerce. Furthermore, it is very challenging to detect privacy malware that uses obfuscation as an evasion tactic due to its behaviour, resilience, and adaptability during runtime. Forensic techniques, such as memory dumping analysis, must be used to enable a system to identify and classify patterns and behaviours that facilitate its eventual identification. This research developed a deep learning model for malware classification on an obfuscated malware dataset, called the MalwareMemoryDump dataset. It implemented transfer learning (TL) to adapt the trained model to NF-TON-IoT and UNSW-NB15, improving intrusion detection in IoT and network traffic. We conducted extensive experiments showing improved accuracy and efficiency in cross-domain detection scenarios. Further, we demonstrate that transfer learning minimises training time and computational requirements compared to training separate models from scratch. Additionally, it offers XAI-based explainability to enhance model transparency and interoperability. We demonstrated the effectiveness of the proposed model in handling diverse heterogeneous cybersecurity threats across memory-based malware analysis, IoT security, and traditional network intrusion detection. The effectiveness of the proposed methodology is evaluated using several key metrics to demonstrate its advantages over conventional methods. Experimental findings show that the proposed framework attains 99.9% accuracy on the MalwareMemoryDump dataset, 96% on the NF-Ton-IoT dataset and UNSW-NB15 datasets. Because of its innovative methodology and ability to generalise datasets, the model is a highly effective approach that outperforms many of the most recent malware detection and other security techniques.

Keywords Memory dump analysis, Transfer learning, Intrusion detection system, Deep neural networks, Shapley additive explanations, Malware attacks

In the digital world, individuals have an unalienable right to privacy, which must be protected by laws and technological advancements[1]. These include regulations for secure processing, archiving, and retention of sensitive data, such as the General Data Protection Regulation (GDPR) in Europe and the Health Insurance Portability and Accountability Act (HIPAA) in the United States[2]. However, these legal actions have been significantly diminished by specialised spyware that targets victims' privacy, either directly or through agents who have the authority to manage their private data[3]. According to[4], the three primary malware kinds with various families that currently threaten user privacy are Trojan horses, ransomware, and spyware. Spyware is a type of malware that poses the greatest harm to user privacy, as it eavesdrops and snoops to gather information[5]. The ability of ransomware to create command and control connections to an attacker's computer system, allowing it to take data before encrypting it, is one of its distinctive features[6]. Trojan horse malware can then produce backdoors, which enable it to deceive the user by appearing to be a genuine application and spy and steal data[7].

[1]Department of Computer Engineering and Networks, College of Computer and Information Sciences, Jouf University, Sakaka 72388, Saudi Arabia. [2]College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, AlKharj 16273, Saudi Arabia. [3]Department of Information Management and Business Systems, Comenius University Bratislava, Odbojárov 10, 82005 Bratislava 25, Slovakia. [4]Computer Science Department, Faculty of Computers and Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia. [5]Department of Computer Engineering, College of Computer Science, King Khalid University, Abha 61421, Saudi Arabia. [6]Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Arar, Saudi Arabia. [7]Department of Computer Engineering, COMSATS University Islamabad, Sahiwal Campus, Sahiwal 57000, Pakistan. ✉email: natalia.kryvinska@uniba.sk; sidraabbas@ieee.org

Cyberattacks using these three forms of privacy viruses may have evolved in response to the expansion of online services and electronic commerce[8]. Despite differences in their architecture and methods, the majority of modern families of specialized spyware, ransomware, and Trojan horses have a single feature in common: once a legitimate operating system process launches them, security controls are unlikely to catch them until the malware has completed, either completely or partially, its target[9]. This is primarily due to their avoidance strategy of obfuscation. There is currently no uniform and clear method to identify and comprehend the tendencies and behaviours of obfuscated privacy malware at runtime, and the majority of privacy malware detection strategies focus on detecting non-obfuscated privacy malware[4,10].

As shown in Fig. 1[1], when obfuscated privacy malware violates a system, memory dumping and analysis can identify particular patterns and behaviours that the system activities encounter in order to create a classifier for obfuscated privacy malware. Several features can be used to differentiate and even classify the various kinds of obfuscated privacy malware by family. These variables include the variety of sockets designed for communication to remote places, the number of mutexes and semaphores utilized, and the number of handlers the operating system opened in response to a procedure request. Lashkari et al.[11], and Mu et al.[12] contributions provide an extensive overview of this process. However, based on the patterns and behaviours of the many families and classes of obfuscated privacy malware, conventional programming techniques are unable to determine a generalisation. As a result, machine learning (ML) techniques, especially DNNs, are widely used to analyze camouflaged privacy spyware. Consequently, this study suggests a TL method for categorising obfuscated privacy malware that can attain comparable metrics by applying advanced strategies suggested by prior studies.

Research motivation

Malicious software that utilises obfuscation as a primary method of evading security measures at runtime includes programs that conceal their features, capabilities, and activities. Detecting obfuscated malware is extremely difficult since it is immune to signature-based methods employed by security controls such as antivirus applications, IDS, and intrusion protection system engines[13]. Lower detection metrics are obtained against this resistance as opposed to malware that is not obfuscated. Furthermore, it becomes more challenging to identify

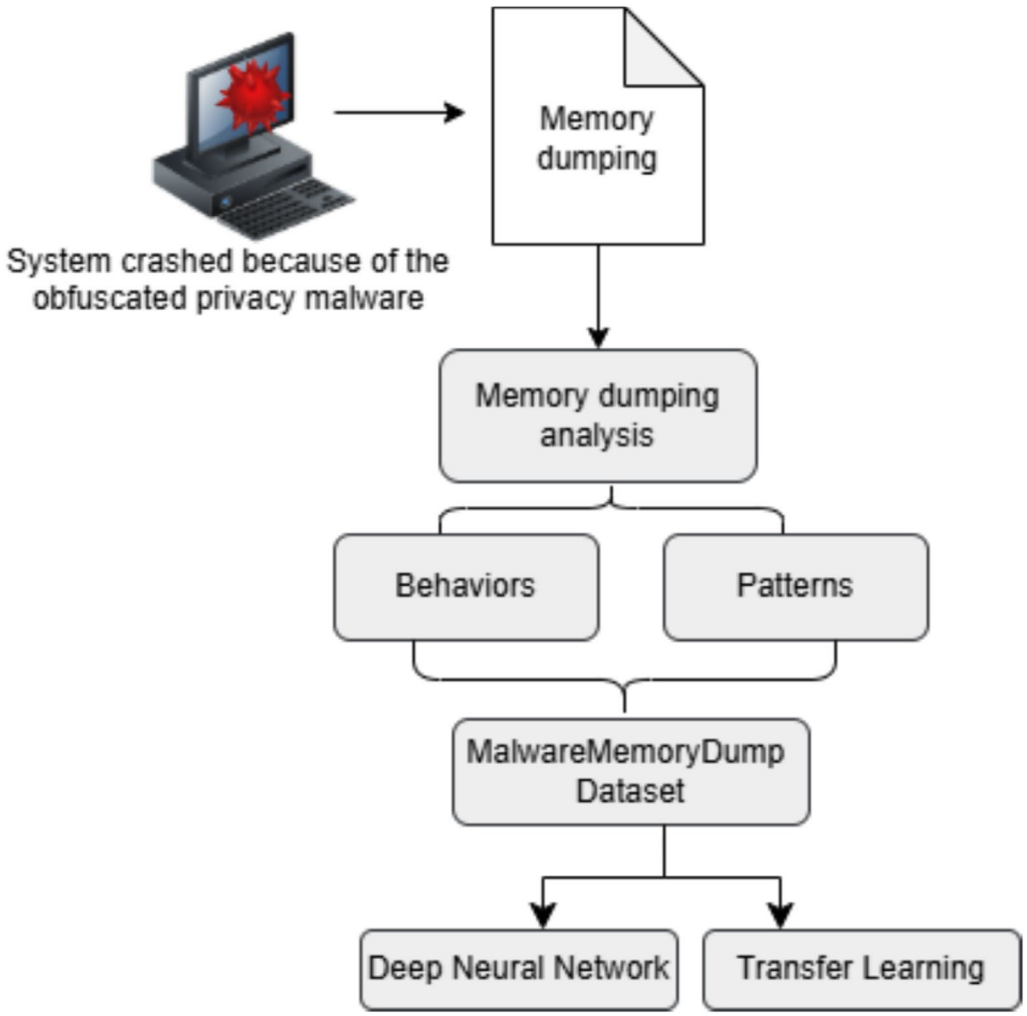


Fig. 1. Memory dumping analysis.

patterns and behaviours for recognition due to the hardiness and polymorphism of this kind of malware, which allows it to change and show itself in many ways after being activated[1]. Furthermore, multiclass classification with obfuscated malware is particularly challenging, as it makes it difficult to create characteristics that allow for a distinct division of malware into multiple families and categories[14]. Understanding the family and category of privacy malware is necessary to implement efficient security controls and countermeasures and to comprehend the nature of an attack[15].

Although obfuscation is a technique that many malware types can use to evade detection, the majority of specialised privacy malware makes extensive use of it, which has a greater impact on users and organisations than malware designed to target other security principles, such as the availability or integrity of the information[16]. This is due to its unpredictable patterns and actions, which are intentionally designed to facilitate theft, snooping, and the extraction of private information, ultimately leading to data leaks[9]. Several methods have been proposed for gathering data that is necessary for the subsequent identification and categorisation of obfuscated privacy malware. Domain Generation Algorithms (DGA) analysis[17], command and control sequence pattern detection[18], and DNS pattern analysis for malware hosting[19] are some examples. However, memory dumping analysis is the most intriguing[9] since it enables the analysis of post-mortem scenarios, providing information about the key features and actions of the virus after it has completed its full attack process[4].

The objective of this research endeavour is to create a sophisticated malware detection system with enhanced accuracy and interpretability by utilising transfer learning and SHAP. To guarantee data integrity, the method begins by preparing malware memory dump datasets. In order to optimise the parameters and produce SHAP values for explainability, a DNN is first trained on MalwareMemoryDump data. While including SHAP-based feature extraction, the pre-trained model is improved using new datasets (NF-TON-IoT and UNSW-NB15). A fully connected layer and softmax activation enhances the classification process. For obfuscated malware families, this two-step approach increases detection accuracy while maintaining transparency. By evaluating its effectiveness against existing methods, this work contributes to the development of a scalable and interpretable malware detection system.

Research contribution

This research enables more accurate and efficient detection of obfuscated malware attacks. The following is a list of the main scientific discoveries and contributions.

- *Obfuscated Malware Classification and Transfer Learning for IDS* – Developed a deep learning model for malware classification and implemented Transfer Learning (TL) to adapt the trained model to NF-TON-IoT and UNSW-NB15, improving intrusion detection in IoT and network traffic.
- *Reduced Computational Cost*– Demonstrated that transfer learning minimizes training time and computational requirements compared to training separate models from scratch.
- *Explainability and Transparency*- Provide XAI-based explainability on improving model transparency and interoperability.
- *Generalisation Across Cybersecurity Domains* – Demonstrated the effectiveness of the proposed model in handling diverse heterogeneous cybersecurity threats across memory-based malware analysis, IoT security, and traditional network intrusion detection.
- *Cross-Dataset Threat Detection* – Conducted extensive experiments showing improved accuracy and efficiency in cross-domain detection scenarios.

Organization

The notions used throughout the paper are shown in Table 1. The following part provides further structure for the paper. Section "Related work" provides background related to obfuscated malware detection. In section "Proposed methodology", the suggested approach is explained. In section "Experimental results and analysis", the efficiency of the proposed method is assessed and compared with the baseline techniques. Section "Conclusion" concludes the entire paper and contains recommendations for future research.

Related work

The increasing complexity of obfuscated malware presents a constant challenge to cybersecurity, posing a significant obstacle to maintaining digital security. With an emphasis on memory analysis and the growing significance of machine learning (ML) approaches, this literature review examines both classic and recent studies in the field of malware detection. This study critically assesses current models, identifies their inherent limits, and assesses their success rates to investigate this emerging concern. This analysis highlights the need for contemporary detection methods that can address more elusive cyber threats while also illuminating the current state of cybersecurity defences.

Obfuscated malware background

According to the authors in[20,21], the notion of "obfuscated malware" describes a substantial category of online threats that fall outside the purview of conventional detection techniques in the constantly evolving area of cybersecurity. These extremely talented adversaries attempt to conceal their true identities and activities within the vast world of online operations by employing evasion techniques with unparalleled skill. Obfuscated malware accomplishes this using a variety of obfuscation techniques that challenge standard signature-based detection algorithms, such as code encryption and polymorphic behaviour. In the study[22], the researchers explore that memory dumps are a vital battlefield for revealing the covert operations of obfuscated malware in the field of cybersecurity research. In essence, a memory dump is an impression that shows the complex information stored in a computer's random-access memory (RAM) at a particular point in time. This transient archive becomes

Abbreviation	Description
AI	Artificial Intelligence
ANN	Artificial Neural Network
APT	Advanced Persistent Threat
NB	Naive Bayes
CNN	Convolutional Neural Network
CWT	Continuous Wavelet Transform
DDAD	Data-Driven Anomaly Detection
DDoS	Distributed Denial of Services
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Services
DT	Decision Tree
EDA	Exploratory Data Analysis
FALC	Federated Averaging Learning Classifier
GRU	Gated Recurrent Unit
GDPR	General Data Protection Regulation
HPC	Hardware Performance Counter
HADRL	Hierarchical Adversarial
HIPAA	Health Insurance Portability and Accountability Act
IoT	Internet of Things
ISD	Intrusion Detection System
KNN	K-Nearest Neighbour
LSTM	Long Short Term Memory
ML	Machine Learning
NB	Naive Bayes
NIDS	Network Intrusion Detection System
PARNet	Attention Pyramid Network
RAM	Random Access Memory
RF	Random Forest
RNN	Recurrent Neural Network
RT	Random Tree
SVM	Support Vector Machine
SHAP	Shapley Additive Explanations
SMOTE	Synthetic Minority Oversampling Technique
TL	Transfer Learning

Table 1. Abbreviations list.

an invaluable resource for malware research, providing unmatched insights into how processes and programs behave during runtime. Being aware of RAM's instability, obfuscated malware deliberately hides in memory to capitalise on the constantly evolving digital landscape. Since these hostile entities frequently masquerade as legitimate processes, it becomes challenging to distinguish between benign and malicious activity, making memory dump analysis a delicate art. Finding these evil actors and comprehending the complex transformations they carry out within the boundaries of volatile memory are both difficult tasks. By navigating the complexities of memory dumps, researchers can interpret the behavioural patterns of obfuscated malware, thereby overcoming the limitations of conventional detection techniques. The importance of memory dump analysis continues to grow as digital threats become increasingly complex, necessitating creative and flexible strategies to enhance cybersecurity defences[23].

Machine and deep learning techniques

Authors in[24] proposed a straightforward and reasonably priced method for detecting obfuscated malware through memory dump analysis and a range of machine learning techniques. This work utilises the CIC-MalMem-2022 dataset, which is designed to simulate real-world scenarios and evaluate memory-based obfuscated malware detection. They assess the potential for machine learning (ML) algorithms to detect malware hidden in memory dumps. The results show that the XGBoost classifier outperformed the others in malware detection and classification, achieving an accuracy of 0.88 on the original data. Authors in[25] present an ML-based, lightweight, obfuscated malware detection. Only five features extracted from memory dumps are used in the extreme gradient boost-based suggested method, which achieves a detection accuracy of more than 99%. Recursive feature elimination was used to choose these five features based on their relative relevance. The

study's assessment showed that the system could identify malware instances in as little as 0.413 µs. SHAP were used to describe the model. The authors in[1] provide three classifiers for obfuscated privacy malware that were trained on the CIC-MalMem-2022 dataset using logistic regression (LR). In these solutions, malicious samples are separated from benign ones using a binary classifier. Trojans, spyware, ransomware, and benign samples are further separated from obfuscated privacy malware using a multiclass classifier. A more advanced multiclass classifier is able to separate benign samples from fifteen different families of obfuscated privacy malware. The study employed a unique deep neural network (DNN) in conjunction with several traditional ML techniques to develop multiclass classifiers. According to the study's findings, DNN outperforms conventional ML techniques and yields significant statistical improvements in certain parameters.

Authors in[26] evaluated the effectiveness of machine learning techniques in detecting obfuscated malware using the CIC-MalMem-2022 dataset. Among the algorithms assessed are J-48 (C4.5), Random Tree (RT), Random Forest (RF), Naive Bayes (NB), and XGBoost. RF, J-48, and XGBoost are effective in achieving high accuracy rates across a range of classification tasks, as indicated by experimental results. Although NB performs competitively as well, it struggles with multiclass classification and unbalanced datasets. The study's findings, which achieved a 99.9% accuracy rate for binary classification, emphasise the significance of using cutting-edge ML approaches to improve obfuscated malware detection abilities and provide insightful information to researchers and cybersecurity practitioners. The author of the study[27] presents MeMalDet, an innovative memory-based malware detection technique. It utilises deep autoencoders and stacked ensemble learning. The authors propose an improved dataset with temporal features (temporal data split) to give more accurate evaluations of memory-based malware detection techniques. To avoid human feature engineering, MeMalDet utilises deep autoencoders to extract optimal features from memory dumps. Then, extremely accurate malware detection is performed using a stacked ensemble. MeMalDet can successfully detect obfuscated malware under temporal splits, as demonstrated by extensive tests on our enhanced large-scale public dataset. Modern memory analysis-based malware detection methods are greatly outperformed by the study, which achieved up to 98.82% accuracy and 98.72% F1-score in identifying previously observed advanced obfuscated malware. In research[4], the author enhances the development of VolMemLyzer, one of the most recent memory feature extractors for learning systems, by employing a stacked ensemble ML model to target hidden and obfuscated malware. This enables the development of a framework for effectively identifying malware. To assess and validate this method, a specific malware memory dataset (MalMemAnalysis2022) was developed, with a focus on closely replicating real-world obfuscated malware. The results show that using memory feature engineering, the proposed method can rapidly detect hidden and obfuscated malware with accuracy and F1-Score of 99.00% and 99.02%, respectively.

In[28], the authors proposed BotDefender, an integrated system designed to defend against botnet attacks. BotDefender prevents botnet attacks by combining a machine-learning technique with a proposed network traffic analyzer. A live botnet attack strategy is designed and developed to assess BotDefender's performance. Throughout the live test, BotDefender attains a 100% overall accuracy rate and filters out 99.8% of the botnet traffic. In[29], the authors introduced PhiUSIIL, a system for detecting phishing URLs that use incremental learning and similarity indexes. However, squatting, combo squatting, homograph, Punycode, homophone, zero-width characters, and other visual similarity-based attacks can each be successfully identified with the use of the similarity index. When using a fully incremental training technique, PhiUSIIL achieved an accuracy of 99.24%; when using a pre-training approach, it achieved an accuracy of 99.79%.

Although obfuscated malware detection has advanced significantly, several research gaps remain that limit the efficacy and generalizability of current methods. Numerous studies primarily focus on specific datasets, such as network traffic captures or memory dumps, which may not accurately reflect the obfuscation strategies employed in the real world. The suggested models' ability to adapt to evolving malware threats and changing attack tactics is challenged by this dataset's dependency. Furthermore, although deep learning and machine learning models have proven to be highly accurate in controlled environments, little is known about their resilience to adversarial attacks and evasion strategies. Practical implementation in resource-constrained contexts is difficult since several studies emphasize detection performance without providing a thorough review of computing efficiency. Furthermore, the use of existing works in cybersecurity operations, where interpretability is crucial for threat analysis and response, is limited because they often prioritise model correctness over explainability. The summary of the existing literature is provided in Table 2.

Proposed methodology

In this section, a transfer learning model for detecting obfuscated malware is described. The description begins by providing an overview of the data collection and system model preparation before moving on to the DL and TL models, which are used to classify malware attacks. Figure 2 and Algorithm 1 present an extensive malware detection methodology based on transfer learning and SHAP (SHapley Additive exPlanations) values for model interpretability. Data preprocessing, which includes handling missing values, normalization to scale features appropriately, and label encoding to convert categorical labels into numerical form, is the initial step in processing a malicious memory dump dataset. This process cleans and prepares the dataset for model training. In the first step, a Deep Neural Network (DNN) model is trained using this preprocessed data to establish an initial knowledge of malware features. The learned model weights are then preserved for transfer learning. Furthermore, SHAP values are determined to provide explainability by identifying the most significant features influencing the model's predictions, and two datasets, NF-TON-IoT and UNSW-NB15, are added for further evaluation. In the transfer learning phase, the knowledge of the pre-trained DNN model is applied to detect malware attacks more accurately. NF-TON-IoT and UNSW-NB15 datasets are used to fine-tune the model in the target domain, while the original DNN model was trained on general malware data in the source domain. The pre-trained model and preprocessed data with SHAP features constitute the source domain, while new datasets that undergo similar preprocessing and SHAP analysis are included in the target domain. By using the pre-

Refs.	Focus	Dataset	Findings & results	Limitations
20	Graph Neural Networks (GNN) for obfuscated malware detection	Custom dataset based on malware graph structures	Achieved 94.3% accuracy in detecting obfuscated malware	Requires large labeled datasets; computationally expensive
21	Impact of obfuscation on malware detection techniques	Multiple malware datasets, including public repositories	Showed significant drop in detection accuracy for traditional methods	Lack of a proposed mitigation strategy; limited real-world testing
22	Smart memory forensics for Windows malware detection	Memory dumps from Windows devices	Demonstrated 92% accuracy using memory analysis techniques	Focuses only on Windows devices; lacks comparison with other OS
23	Machine learning for obfuscated malware detection in memory dumps	Public and synthetic memory dump datasets	Improved detection rates compared to traditional heuristics	May suffer from adversarial attacks; requires frequent retraining
24	Real-world obfuscated malware detection through memory analysis	Memory snapshots of real-world malware samples	Achieved over 90% detection accuracy in various scenarios	Performance may vary with unseen malware samples; potential overfitting
25	Explainable AI for obfuscated malware detection	Lightweight memory-based dataset	XMal model achieved competitive results with lower resource consumption	Limited interpretability for complex obfuscation techniques
1	Privacy-focused malware detection via memory dumping analysis	Large-scale memory dump dataset	Effective classification with minimal false positives	High computational cost; privacy concerns with memory analysis
26	Malware detection using machine learning models	Various malware repositories	Compared multiple ML models, with deep learning achieving the highest accuracy	Feature selection requires refinement; high false positive rate
27	Deep autoencoders for malware detection using memory analysis	Temporal evaluation-based dataset	Stacked ensemble model achieved over 95% accuracy	Model performance depends on proper hyperparameter tuning
4	Memory feature engineering for obfuscated malware detection	Experimental dataset from controlled memory environments	Demonstrated effective feature engineering for malware detection	Requires extensive feature extraction; high dependency on dataset quality

Table 2. Summary of related work on obfuscated malware detection.

trained weights as an initial base, the learnt information is transferred, enabling the model to adapt and further develop its malware classification abilities. The pre-trained DNN model is then enhanced with a fully linked layer to improve classification using an entire transfer learning architecture. The model's outputs are transformed into probability distributions across the malware and benign classes using a softmax activation function. The final classifications, indicating whether a sample is malicious or benign, are generated by matching the source and target labels to their respective datasets. The model benefits from existing knowledge while adapting to new datasets through this two-stage process, which comprises initial training and transfer learning. By adding SHAP values at every stage of the procedure, model transparency is improved, and malware detection decisions become easier to comprehend.

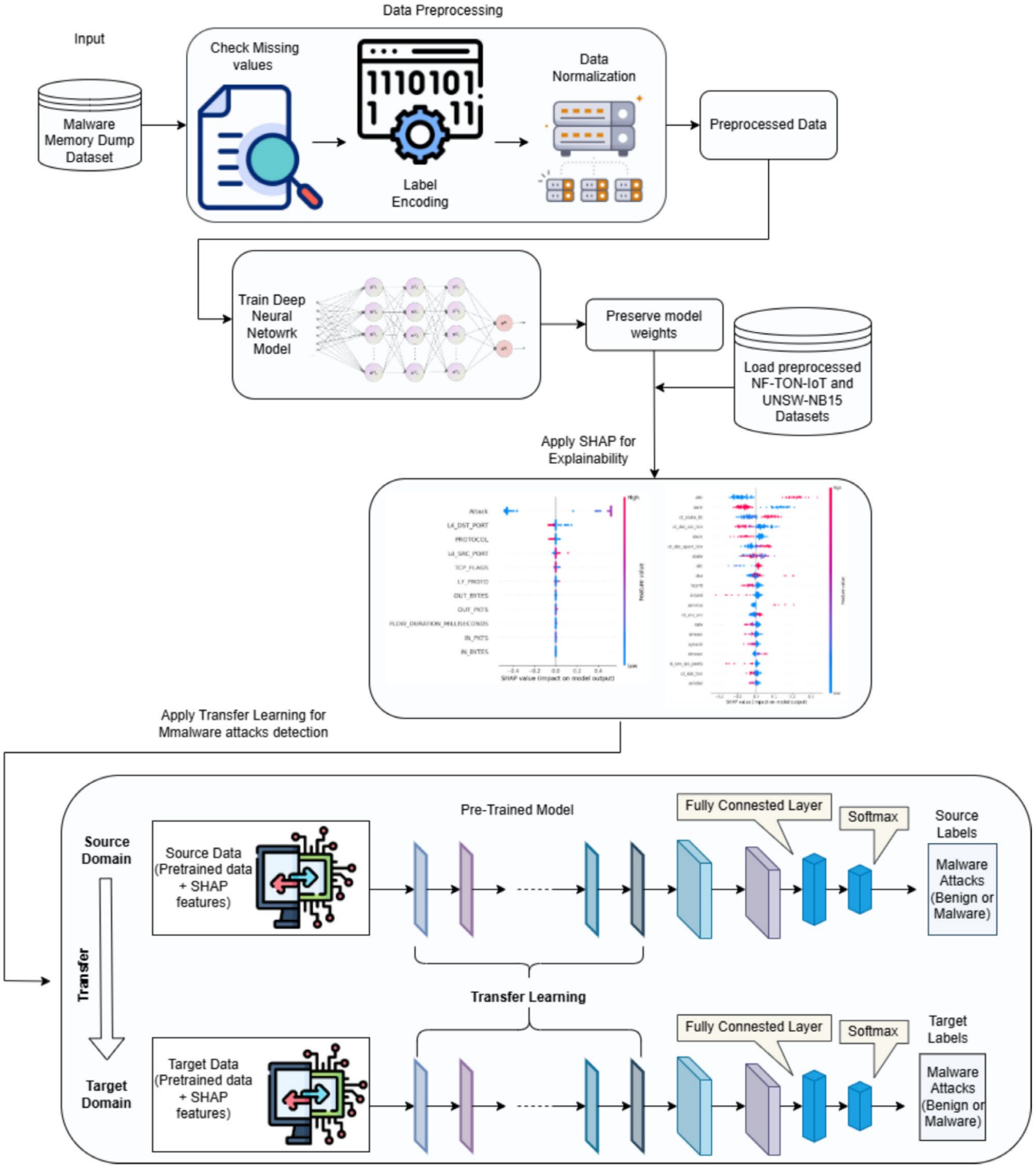


Fig. 2. Proposed architecture for obfuscated malware detection.

Input: Source Dataset D_{source}, Target Datasets D_{target} (NF-TON-IoT, UNSW-NB15)
Output: Malware classification $y \in \{\text{Benign}, \text{Malware}\}$
Preprocessing(D)
$D \leftarrow \text{clean}(D)$ *Handle missing values*
$X \leftarrow \frac{X-\mu}{\sigma}$ *Normalize features*
$y \leftarrow \text{encode}(y)$ *Encode labels*
Return X, y
Initial Training(X, y)
Build Neural Network Model:
$f_\theta = \text{Sequential}([\text{Dense}(128,\ \text{ReLU}),\ \text{Dropout}(0.3),$
$\text{Dense}(64,\ \text{ReLU}),\ \text{Dropout}(0.3),\ \text{Dense}(1,\ \text{Sigmoid})])$
Compile model using Adam optimizer and binary cross-entropy loss:
$\mathscr{L}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$
$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon}$ where:
$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta \mathscr{L}_t$
$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta \mathscr{L}_t)^2$
$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$
Initialize EarlyStopping on validation loss: stop if there is no improvement for $p = 3$ epochs
for $e = 1$ to E **do**
Train model f_θ on training set using batch size = 32
Evaluate model on validation set and compute $L_e^{(val)}$
if $L_e^{(val)}$ has not improved for p consecutive epochs **then**
Stop training and restore the best weights
end if
end for
Compute SHAP values $S = \text{SHAP}(f_\theta, X)$
Return f_{θ^*}, S
Transfer Learning$(f_{\theta^*}, D_{target})$
Preprocess D_{target}
Extract SHAP features $S_{target} = \text{SHAP}(f_{\theta^*}, X_{target})$
Fine-tune model:
$\theta' = \arg\min_\theta \frac{1}{M}\sum_{j=1}^{M}\mathscr{L}(f_\theta(X_{target,j}), y_{target,j})$
Add fully connected layer $h_\phi(z)$ where $z = f_{\theta'}(X)$
Apply softmax:
$p(y|X) = \frac{e^{h_\phi(z)}}{\sum_k e^{h_\phi(z_k)}}$
Return Adapted Model $f_{\theta',\phi}$
Malware Classification$(f_{\theta',\phi}, X_{new})$
Predict Malware:
$\hat{y} = \arg\max_y p(y|X_{new})$
Return Label $\in$ {Benign, Malware}

Algorithm 1. Malware Detection using Transfer Learning

Data description

This study uses three datasets: MalwareMemoryDump, NF-TON-IoT and UNSW-NB15 datasets for obfuscated malware detection.

MalwareMemoryDump dataset

Obfuscated malware is a type of malicious software that deliberately conceals itself to avoid detection and expulsion; it can be identified from the memory dump of the infected device. The obfuscated malware dataset was specifically designed to evaluate the efficacy of methods for detecting obfuscated malware using memory analysis, and it includes common malware categories such as Trojan Horses, ransomware, and spyware to resemble real-world scenarios. Testing the effectiveness of obfuscated malware detection systems can be performed with a well-balanced sample set. To ensure accuracy and prevent the memory dumps from revealing any indications

Malware type	Infectious file	Instances
Trojan Horse	Zeus	195
	Emotet	196
	Refroso	200
	Scar	200
	Reconyc	157
Spyware	180Solutions	200
	Coolwebsearch	200
	Gator	200
	Transponder	241
	TIBS	141
Ransomware	Conti	200
	MAZE	195
	Pysa	171
	Ako	200
	Shade	220

Table 3. Distribution of infection types and their instances.

Attribute	Details
Dataset type	NetFlow-based IoT Security Dataset
Total data flows	1,379,274
Benign flows	270,279 (19.6%)
Attack flows	1,108,995 (80.4%)
Attack categories	DoS, DDoS, MITM, Injection, Password Attacks, Ransomware, Scanning, XSS
IoT devices	Various IoT devices and protocols
Application areas	Intrusion Detection, Malware Analysis, Threat Intelligence
Data format	NetFlow records (network flow summary)

Table 4. NF-ToN-IoT dataset overview.

of the dumping process, the dataset operates in debug mode during the memory dump procedure. With the balanced dataset, 50% of memory dumps are malicious, and 50% are benign. Table 3 describes the malware families. The set contains 58,596 records, comprising 29,298 malicious and 29,298 benign records.

NF-TON-IoT dataset

An adaptation of the ToN-IoT dataset based on NetFlow, the NF-ToN-IoT dataset, is intended especially for assessing cybersecurity applications in Internet of Things (IoT) network environments [https://research.unsw.edu.au/projects/toniot-datasets]. It records network traffic as NetFlow records, which provide a simplified but useful representation of network flows, as opposed to full packet captures. Its structure renders it suitable for cybersecurity research, particularly in the fields of intrusion detection, malware analysis, and threat intelligence. The dataset encompasses a wide range of Iot devices and protocols, providing a realistic simulation of Iot network traffic, as shown in Table 4. One of the main characteristics of the NF-ToN-IoT dataset is its thorough attack classification, which covers threats such as DoS, DDoS, Man-in-the-Middle (MITM), injection attacks, password attacks, ransomware, scanning, and cross-site scripting (XSS). With a total of 1,379,274 data flows, the dataset contains 270,279 benign flows (19.6%) and 1,108,995 attack flows (80.4%), making it sufficiently large for training and assessing DL models.

UNSW-NB15 Dataset

The well-known UNSW-NB15 benchmark dataset for NIDS was developed by the University of New South Wales (UNSW) Cyber Security Lab [https://research.unsw.edu.au/projects/unsw-nb15-dataset]. Numerous modern attack methods, including worms, shellcodes, reconnaissance, backdoors, fuzzers, exploits, denial of service (DoS), and generic attacks, are included in the dataset in Table 5. The dataset is highly relevant for research and practical applications, as these attack types are typical of real-world cybersecurity threats. Both hostile and benign traffic were recorded by UNSW-NB15, which was developed in a practical but controlled network environment. The dataset is available in two formats: CSV, which provides a structured and preprocessed version suitable for machine learning applications, and PCAP (Packet Capture), which preserves detailed network traffic data for in-depth analysis. To facilitate supervised learning techniques, each network flow is labelled to indicate whether it constitutes an attack or typical traffic. The dataset contains approximately 2,576,118 records, comprising both normal and malicious traffic.

Attribute	Details
Dataset name	UNSW-NB15
Developed by	University of New South Wales (UNSW) Cyber Security Lab
Total records	2,576,118
Traffic type	Normal and Malicious
Attack categories	Worms, Shellcode, Reconnaissance, Backdoors, Fuzzers, Exploits, DoS, Generic
Environment	Realistic but Controlled Network
Data formats	PCAP (Raw Traffic), CSV (Preprocessed Features)
Labelling	Each network flow is labelled as an attack or normal traffic

Table 5. UNSW-NB15 dataset overview.

Data preprocessing

Data preprocessing is a crucial step in processing raw data for ML and DL models. Normalisation, standardisation, and categorical variable encoding are methods for processing data. Robust model training can be achieved by handling inconsistent data with resampling techniques or class weight adjustments. Data cleaning involves removing duplicates, correcting outliers, and imputation, which is the process of filling in missing values. The data used in this study were standardised using the min-max scaling technique. In this study, We used the *isnull*().*sum*() function, which counts the number of missing values per feature, to first determine whether null values were present in each column in order to evaluate the dataset's completeness. After that, we only showed the columns with null values, which made it easy to recognise particular elements that needed improvement. Furthermore, to provide a general understanding of the data quality, we computed and displayed the total number of null values throughout the entire dataset.

Data Scaling: The suggested method begins with data normalisation. This data-scaling process ensures that the weighted total stays within the initial work's bounds. Slow convergence and inadequate network training might result from unnormalized input. On the other hand, adding more data makes the merging process simpler and makes the data dimensionless. The following is the definition of the min-max scaling strategy (Eq. 1), which scales the data from 0 to 1.

$$D_{\text{scaled}} = \frac{D - D_{\min}}{D_{\max} - D_{\min}} \tag{1}$$

The initial value from the database is *D*, the highest value is D_{max}, the minimum value used in the scaling method is D_{min}, and the scaled data is D_{scaled}.

Shapley additive explanations

Shapley Additive Explanations (SHAP), a prevalent interpretability technique in ML, describe the way each feature affects a model's predictions. The cornerstone of cooperative game theory is the Shapley values, which divide each feature's contribution to the final prediction equally. SHAP calculates the average marginal contribution of a feature by taking into account all potential feature combinations. The SHAP value of a feature x_i can be determined mathematically in the manner described below in Eq. 2:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|F|!}{|S|!(|F| - |S| - 1)!} [f(S \cup \{i\}) - f(S)] \tag{2}$$

The model's output when only the features in *S* are considered is denoted by *f*(*S*), while *F* represents the complete feature set and *S* represents a subset of features. SHAP values provide a suitable statistic of feature significance and explainability by calculating each feature's contribution to the model's output. This is particularly useful in cybersecurity applications such as malware detection, where understanding the reasons behind a model's classification of a sample as malicious can increase confidence and facilitate threat analysis. For this reason, this study uses the SHAP for malware detection. Figure 3 represents the top-selected features based on SHAP values. A feature does not affect the prediction for that particular instance if its SHAP value is 0. A single instance from the dataset is represented by each dot on the plot; the colour of the dots indicates the feature value for that instance, which is blue for low values and red for high values. For the NF-TON-IoT dataset, "Attack," the most significant feature influencing the model's output, is critical in identifying the nature or existence of attacks and has a strong positive impact when its value is high (Fig. 3a). As the $'L4\_DST\_PORT$ feature shows, larger Layer 4 destination port values typically have a positive effect, whereas smaller values frequently cause a negative effect. This implies that particular network behaviours are closely associated with particular destination ports. $PROTOCOL$,$'L4\_SRC\_PORT$,$'TCP\_FLAGS$',$'L7\_PROTO$, and a number of flow-related metrics, including $'OUT\_BYTES$, $'OUT\_PKTS$, $'FLOW\_DURATION\_MILLISECONDS$, $'IN\_PKTS$, and $'IN\_BYTES$, are additional significant features. These features, which specify the number of bytes and packets transmitted and received, as well as the length of the network flow, also affect the model's predictions to differing degrees. However, for the UNSW-NB15 dataset in Fig. 3b , the top features influencing the model's output are different from those in NF-TON-IoT. $'ct\_state\_ttl$ (Count of connections with the same state and destination address), $swin$ (Source window size), and $dttl$ (Destination Time-To-Live) are some

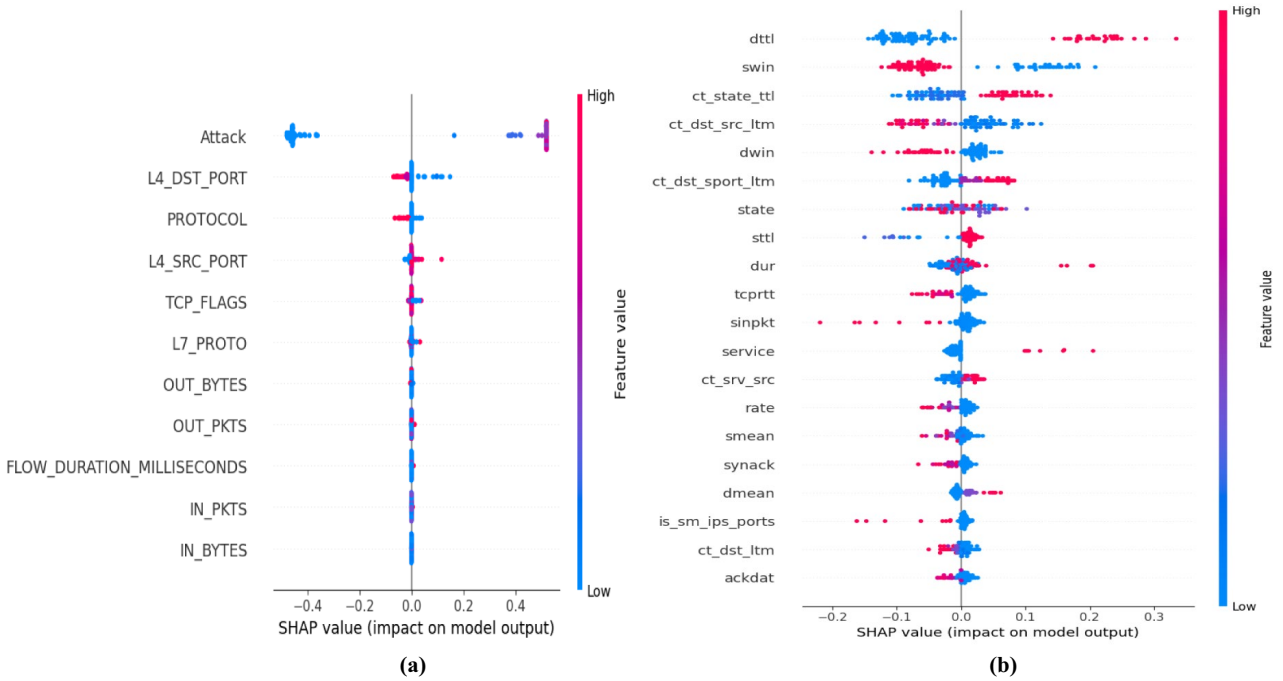


Fig. 3. Feature selection based on shapley additive explanations. (**a**) Top feature selected for NF-TON-IoT dataset. (**b**) Top feature selected for UNSW-NB15 dataset

of the noteworthy features. Additional important features include $'ct\_dst\_src\_ltm$ (count of connections with the same source and destination address), $state$ (connection state), $tcprtt$ (TCP round-trip time), $sintpkt$ (standard deviation of inter-packet arrival time), and $service$ (network service on the destination port).

Deep neural network

Deep neural networks (DNNs) are artificial neural networks (ANNs) with numerous hidden layers between the input and output layers. Due to its ability to identify complex patterns in data, it is widely utilised in various cybersecurity applications, including virus detection. The following elements comprise a standard DNN: The feature vectors obtained from malware memory dumps are transferred to the input layer. Hidden layers contain the number of neuronal layers where feature extraction and transformation happen. Activation functions represent intricate interactions and add non-linearity. The output layer generates the final categorization outcome, such as malware vs benign. A deep neural network can be represented mathematically in Eqs. 3 and 4:

$$Z^{(l)} = W^{(l)} A^{(l-1)} + b^{(l)} \tag{3}$$

$$A^{(l)} = f(Z^{(l)}) \tag{4}$$

A set of weights and biases is applied to the input from the previous layer by each layer, with the layer number denoted by the index l. At layer l, the weighted sum of inputs is represented as $Z(l)$. This is calculated by taking the activation values $A(l-1)$ from the previous layer, the weight matrix $W(l)$, and a bias term $b(l)$. Next, $Z(l)$ is subjected to the activation function $f(\cdot)$, which adds non-linearity and allows the model to learn intricate patterns. To enable the neural network to learn complex correlations within the data, this iterative process is carried out across multiple hidden layers until the final output is produced. The architecture of the deep neural network is illustrated in Fig. 4.

Transfer learning

Transfer learning is the process of applying a model that has been trained on one task to another that is comparable but distinct. Instead of building a DNN from scratch and modifying it for a new use or domain, this method utilises pre-trained models, which are often developed using large datasets. The primary motivations for transfer learning are the utilisation of learned models or components from the source task to enhance performance on the target task, reduce the need for large datasets, and conserve computing time and resources[30]. There are several important steps in the TL process represented in Fig. 5.

Before learning features, a base model is selected, which is a pre-trained model trained on a source task using a source dataset D_s (Eq. 5).

$$\min_{\theta_s} \mathcal{L}_s \left(M_s(x_s; \theta_s), y_s \right) \tag{5}$$

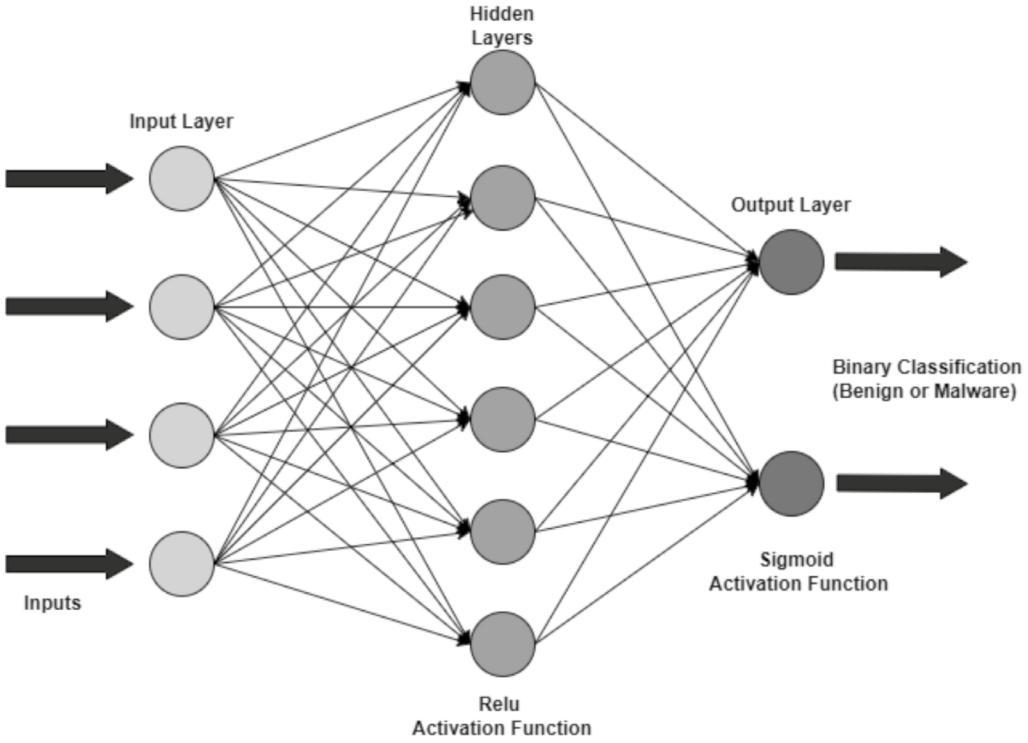


Fig. 4. Architecture of deep neural network model.

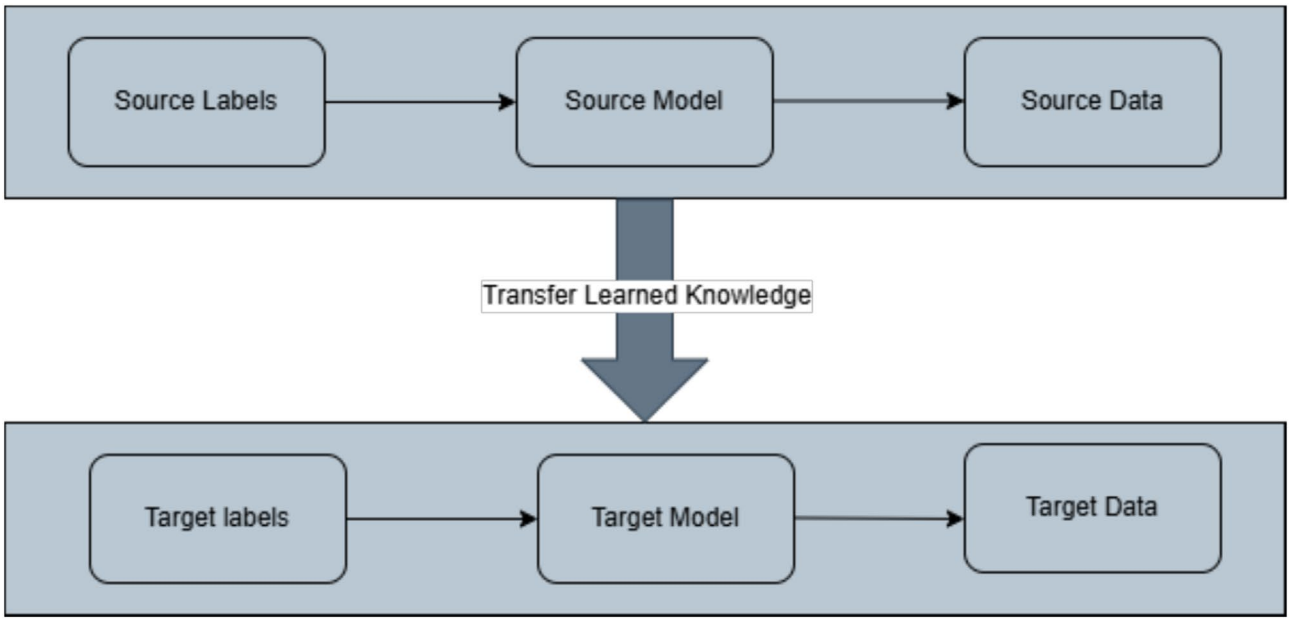


Fig. 5. Transfer learning architecture.

Layer (type)	Output shape	Param #
input_layer_1 (InputLayer)	(None, 10)	0
dense_3 (Dense)	(None, 128)	1,408
dropout_2 (Dropout)	(None, 128)	0
batch_normalization (BatchNormalization)	(None, 128)	512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 1)	65

Table 6. Model architecture summary.

Aspect	From scratch (Memory_Forensic)	Transfer learning (NF_TON_IOT / UNSW-NB15)
Hardware	Google Colab (NVIDIA Tesla T4 GPU, 16GB RAM)	Same (Colab environment reused)
Training time	∼45–60 seconds for 10 epochs	∼30–45 seconds for 20 epochs with frozen layers
Inference time	∼2.5 ms/sample (using `.predict()` on test set)	∼1.2 ms/sample (due to fewer trainable parameters)
Memory usage	∼800MB GPU memory (from TensorFlow Profiler)	∼500MB GPU memory (smaller input and frozen base)
Trainable parameters	∼1,000–2,000 (all layers trainable)	∼500–800 (most layers frozen)
Efficiency gain from TL	Not applicable (baseline)	∼35–40% reduction in memory & training time
Model size	4-layer dense NN	Reused base + minor new layers (efficient)
Conclusion	Baseline full training	Reduced cost via frozen weights, smaller input size

Table 7. Experimental setup and computational cost evaluation.

This is the process of minimizing the loss function L_s for the source model M_s given the labels y_s, model parameters θ_s, and input data x_s. The pre-trained layers F_s have frozen weights θ_s, which means they do not change (Eq. 6).

$$\frac{\partial L_t}{\partial t} = 0 \tag{6}$$

Only the new layers (target-specific layers) with weights θ_t are updated (Eq. 7).

$$\theta_t \leftarrow \theta_t - \eta \cdot \frac{\partial L_t}{\partial \theta_t} \tag{7}$$

where η is the learning rate, while θ_t is updated, the weights of some or all of the pre-trained layers are unfrozen.

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial L_t}{\partial \theta} \tag{8}$$

where θ includes both pre-trained and new parameters, the source and target tasks can be combined into the overall optimization objective for transfer learning (Eq. 8).

$$L_{\text{total}} = \lambda \cdot L_s + (1 - \lambda) \cdot L_t \tag{9}$$

In Eq. 9 L_s is the source task loss, L_t is the target task loss, and λ is the weighting factor (which regulates the balance between the source and target loss).

Transfer learning with a deep neural network (DNN) involves transferring knowledge from a source domain to a target domain using pre-trained models. On a source dataset D_s, a DNN model M_s is first trained to learn feature representations by minimizing a source loss function $L_s(\theta_s)$. The target domain then uses the lower layers of M_s, which extract generic features $f_s = g_s(x; \theta_s^{base})$. These layers are then frozen. In order to minimize the target loss $L_t(\theta\_t)$, a new model M_t is then constructed by appending trainable task-specific layers h_t to the frozen base and trained on a target dataset D_t. The summary of transfer learning architecture is illustrated in Table 6.

Experimental results and analysis

The performance of the framework is assessed using a variety of assessment criteria, experimental results are examined and reviewed, and data gathered for this study are interpreted. These factors provide crucial information about the model's performance. Table 7 lists all the tools and equipment used in the experiments, along with a detailed comparison between creating a DL model from scratch and applying TL on Iot datasets. Both approaches were used with Google Colab, the same computer environment that includes an NVIDIA

Tesla T4 GPU and 16GB of RAM, to ensure fair and unbiased comparisons. The Memory_Forensic dataset requires approximately 45–60 seconds for 10 epochs of training for the model built from scratch. Conversely, even after 20 epochs, the transfer learning method, which utilised frozen base layers, completed training in a mere 30 to 45 seconds. The smaller number of trainable parameters and the reuse of pretrained weights, which reduces backpropagation efforts, are responsible for this efficiency. The TL method was also preferred in terms of inference time per sample. Due to its lighter architecture and fewer active parameters, the TL model achieved faster inference at approximately 1.2 milliseconds per sample, whereas the scratch model required about 2.5 milliseconds per sample. Due to this, real-time anomaly detection scenarios in Iot applications are more suitable for the TL technique. Based on TensorFlow Profiler's memory measurements, the scratch model used roughly 800MB of GPU memory. On the other hand, since the input size was smaller and the majority of the model layers were frozen, the TL model required only about 500 MB. In edge computing contexts, where memory resources are frequently limited, this decrease is particularly crucial. In terms of trainable parameters, the TL model had only 500 to 800, as most of the layers were frozen, whereas the scratch model had 1,000 to 2,000. This decrease in parameter count directly impacts faster training and less memory usage. Transfer learning resulted in a significant increase in efficiency. The TL configuration resulted in a 35–40% decrease in memory use and training time, making it a suitable option for resource-conscious, lightweight deployments. In contrast, the from-scratch model was used as a baseline with no previous modifications. Lastly, in terms of model size and complexity, the TL-based architecture utilised a pretrained model as its base and added only a few new layers. In contrast, the baseline model used a conventional 4-layer dense neural network. This modular and parameter-efficient architecture further supports the usefulness and efficiency of TL in limited IoT situations.

Evaluation matrices

Accuracy is the percentage of correctly identified samples compared to the total sample size and serves as the standard for evaluating performance. The accuracy of the model in Eq. 10 reflects the confidence in its ability to produce accurate forecasts. It is crucial in assessing its predictive power and dependability despite its simplicity.

$$Acc = \frac{True_{pos} + True_{neg}}{True_{pos} + True_{neg} + False_{pos} + False_{neg}} \tag{10}$$

The precision of a model or system is the degree to which it accurately predicts the positive class. This number is correspondingly displayed in Eq. 11 to facilitate comprehension of the metric fundamental equation.

$$Pre = \frac{TrueP}{TrueP + FalseP} \tag{11}$$

Recall is a metric used to evaluate the performance of classification models, particularly when identifying positive cases is crucial. It is also known as the true positive rate or sensitivity. The capacity of a model to accurately differentiate all pertinent instances (true positives) from the actual positive cases is measured by recall. The calculation of Eq. 12 indicates the unique advantage of this diverse perspective for an estimation.

$$Re = \frac{TrueP}{TrueP + FalseN} \tag{12}$$

Since the appropriately calculated F1 score may effectively convey the essence of balanced performance, it serves as a balance between accuracy and recall. Equation 13 provides a good description of this basic estimation method despite its complexity.

$$F1 - score = 2 \times \frac{Pre + Re}{Pre + Re} \tag{13}$$

Table 8 displays the classification performance of a Deep Neural Network (DNN) model for the identification of obfuscated malware. Malicious samples are likely represented by class 1, and benign samples by class 0. The model obtained a precision of 0.99 for each class, indicating that 99% of the anticipated positive samples were accurate. The model correctly identified 99% of the real positive cases, as indicated by the recall of 0.99. A well-balanced performance is demonstrated by the F1-score, which gives a harmonic mean of precision and recall, which is likewise 0.99 for both classes. There were 5801 Class 0 samples and 5919 Class 1 samples in the dataset, indicating an essentially even distribution. The model's total accuracy is 99%, which indicates that 11720 out of

Class	Precision	Recall	F1-score	Support
0	0.99	0.99	0.99	5801
1	0.99	0.99	0.99	5919
Accuracy			0.99	11720
Macro Avg	0.99	0.99	0.99	11720
Weighted Avg	0.99	0.99	0.99	11720

Table 8. DNN model results for obfuscated malware detection.

Classes	Precision	Recall	F1-score	Support
0	0.93	0.99	0.96	192225
1	0.99	0.93	0.96	191593
Accuracy			0.96	383818
Macro Avg	0.96	0.96	0.96	383818
Weighted Avg	0.96	0.96	0.96	383818

Table 9. Transfer learning results on NF-ToN-IoT dataset.

Classes	Precision	Recall	F1-score	Support
0	1.00	0.92	0.96	9072
1	0.92	1.00	0.96	9061
Accuracy			0.96	18133
Macro Avg	0.96	0.96	0.96	18133
Weighted Avg	0.96	0.96	0.96	18133

Table 10. Transfer learning results on UNSW-NB15 dataset.

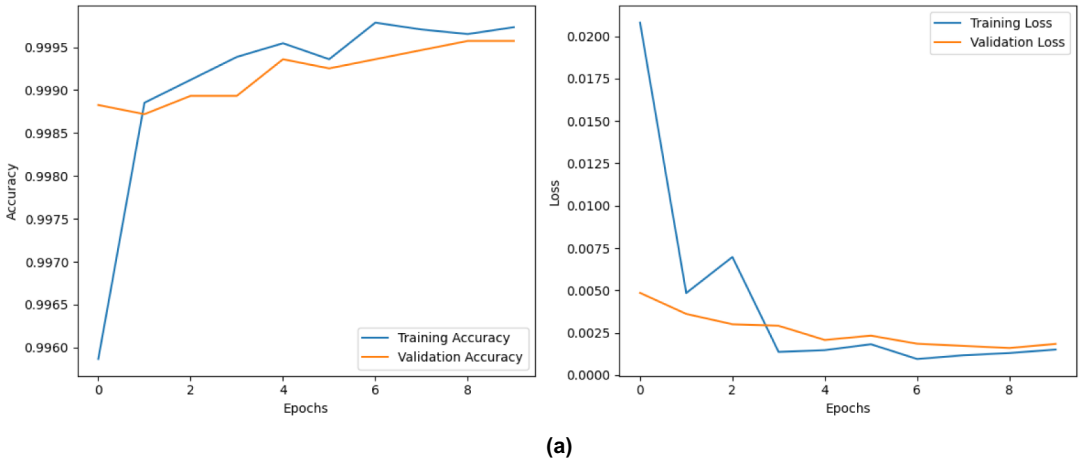
11720 cases were properly identified. Furthermore, all measures show similar performance independent of class size, with the macro average, which computes the average for each class equally, producing values of 0.99.

Table 9 demonstrates the outcomes of the transfer learning on the NF-ToN-IoT dataset for the identification of obfuscated malware. Class 0 exhibits favourable performance metrics: a recall of 0.99 implies the model correctly recognized 99% of the actual benign occurrences, while a precision of 0.93 suggests that 93% of the samples predicted as benign were, in fact, benign with an F1-score of 0.96, which maintains a balance between recall and precision. The dataset contained a total of 192,225 samples for this class. Class 1 had a precision of 0.99, indicating that almost all the malicious samples predicted were accurate. With a recall of 0.93, it was able to identify 93% of all malicious instances. It demonstrated strong performance with an F1-score of 0.96, similar to Class 0. 191,593 occurrences were identified in this class. Overall, the model performed effectively, properly classifying a substantial portion of the 383,818 samples with an accuracy of 96%.

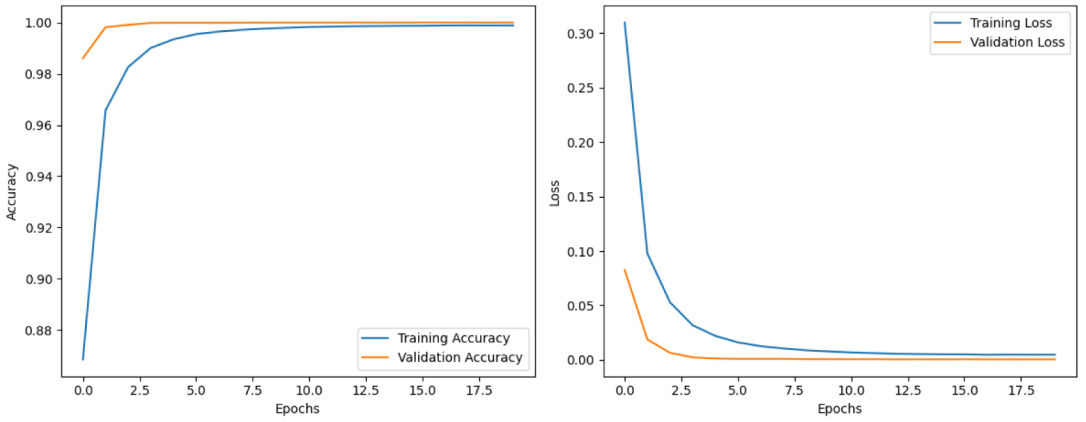
Table 10 demonstrates the results of transfer learning on the UNSW-NB15 dataset for identifying obfuscated malware. Class 0, which stands for apparently benign samples, had a precision of 1.00. The F1-score, which balances precision and recall, was 0.96, and the recall was 0.92, which indicates that 92% of real benign incidents were properly identified. There were a total of 9,072 benign samples (support). The precision for Class 1, which probably consists of dangerous or obfuscated malware samples, was 0.92, which means that 92% of the samples that were predicted to be malicious were correct. The model effectively detected every fraudulent instance, as evidenced by the recall of 1.00 and the high F1-score of 0.96. There were 9061 samples in this class. The accuracy of the overall model performance was 0.96, meaning that 96% of the 18,133 samples in total were properly identified. Precision, recall, and F1-score values of 0.96 were obtained by the macro average, which considers both classes equally regardless of size, indicating consistently satisfactory results across both classes.

Figure 6 compares the effectiveness of two models, a Transfer Learning model and a DNN model, by identifying malware that has been disguised across three distinct datasets. Figure 6a illustrates the training and validation curves of a DNN model, which is probably employed as the basis for malware detection. While the validation accuracy shows a similar pattern but remains somewhat lower, it shows reasonable generalisation with no overfitting. In contrast, the training accuracy increases significantly over the initial few epochs and maintains around 99.8%. The training and validation loss values decrease sharply in the early epochs before stabilising at very low levels, demonstrating effective learning with few errors. The DNN model's exceptional overall performance, characterised by high accuracy and minimal loss, makes it an effective choice for malware detection. The performance of the Transfer Learning model on the NF-ToN-IoT dataset, which is intended for malware analysis and most likely includes network traffic data from IoT devices, is shown in Fig. 6b . This approach utilises previously acquired data from a larger malware dataset to facilitate the identification of malware. After roughly 10 epochs, the training accuracy rises quickly to almost 100%, indicating efficient and rapid learning. Substantial generalization is demonstrated by the validation accuracy, which similarly shows a consistent rise and plateaus over 98%. The practicality of the model in identifying malware patterns is further supported by the quick decline and stabilization of both training and validation loss values at very low levels. This demonstrates Transfer Learning's tremendous efficacy on the NF-ToN-IoT dataset, where it achieved nearly flawless accuracy, making it exceedingly useful for malware detection in IoT environments.

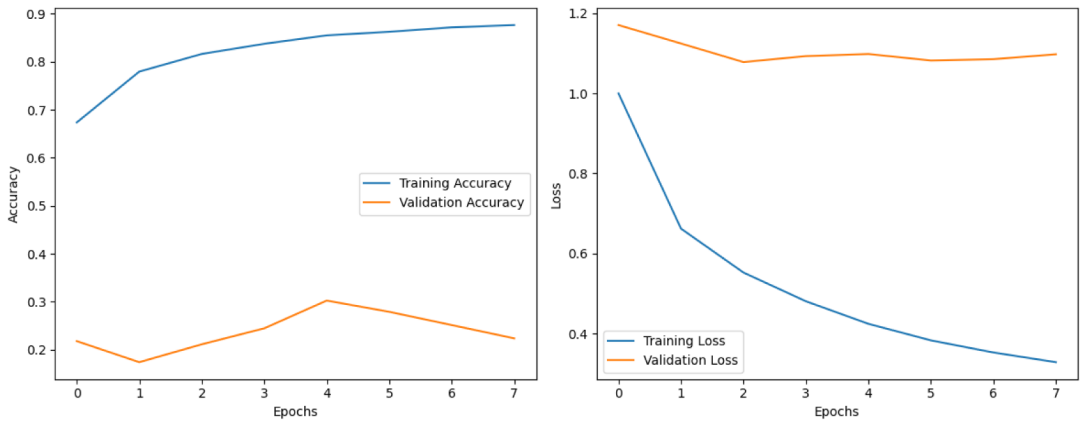
Figure 6c evaluates the effectiveness of the TL model on the UNSW-NB15 dataset, a well-known benchmark dataset that contains network traffic data with a range of attack scenarios, including malware. The training accuracy increases steadily and reaches about 88% after 7 epochs. The validation accuracy approximately follows this trend. In the initial epochs, the training and validation loss levels similarly rapidly decrease before levelling off. The loss values remain slightly higher than those of the NF-ToN-IoT dataset, though, which may indicate that the UNSW-NB15 dataset has more intricate patterns or that the pre-trained features are more difficult to transfer. The accuracy of the TL model is marginally lower than that of the NF-ToN-IoT dataset despite its



(a)



(b)



(c)

Fig. 6. Graphical visualisation of accuracy and loss curve for obfuscated malware detection. (**a**) Training and validation curve of DNN model. (**b**) Transfer learning model training and validation curve on NF-ToN-IoT dataset. (**c**) Transfer learning model training and validation curve on UNSW-NB15 dataset.

excellent performance. This suggests that obfuscated malware detection in this dataset is more difficult and might require additional fine-tuning or domain-specific feature extraction.

Table 11 provides the performance of a Fully Connected Neural Network (FCNN). Figure 7 compares two different neural network models for detecting malware that has been obscured. Figure 7a displays the training and validation curve for a FCNN. The validation accuracy has a similar pattern, levelling off at a slightly lower

Epoch	Accuracy	Loss	Val accuracy	Val loss
1	0.5840	0.7105	0.6870	0.5473
2	0.6877	0.6003	0.6849	0.5035
3	0.7147	0.5584	0.6890	0.4797
4	0.7236	0.5399	0.7362	0.4633
5	0.7357	0.5263	0.7338	0.4495
6	0.7454	0.5116	0.7186	0.4409
7	0.7486	0.5108	0.7181	0.4330
8	0.7522	0.5012	0.7180	0.4270
9	0.7534	0.4981	0.7184	0.4223
10	0.7544	0.4946	0.7184	0.4171
11	0.7567	0.4885	0.7188	0.4139
12	0.7566	0.4867	0.7192	0.4101
13	0.7566	0.4828	0.7196	0.4071
14	0.7594	0.4807	0.7198	0.4039
15	0.7623	0.4735	0.7204	0.4006
16	0.7631	0.4723	0.7211	0.3983
17	0.7605	0.4721	0.7218	0.3958
18	0.7645	0.4680	0.7221	0.3933
19	0.7644	0.4654	0.7225	0.3906
20	0.7669	0.4637	0.7234	0.3880
21	0.7625	0.4641	0.7238	0.3858
22	0.7623	0.4658	0.7241	0.3842
23	0.7667	0.4597	0.7250	0.3820
24	0.7648	0.4585	0.7265	0.3801
25	0.7664	0.4533	0.7291	0.3783
26	0.7664	0.4593	0.7320	0.3762
27	0.7656	0.4530	0.7338	0.3753
28	0.7706	0.4456	0.7428	0.3729
29	0.7695	0.4485	0.7433	0.3715
30	0.7667	0.4510	0.7432	0.3686

Table 11. Training and validation performance per epochs for fully connected neural network.

value of 0.70, while the training accuracy rises quickly in the initial few epochs before settling at 0.74. The noticeable difference between training and validation accuracy suggests some overfitting, indicating that the model performs better on the training data than on unseen data. The training loss, which decreases sharply in the initial epochs before gradually declining to less than 0.55, and the validation loss, which also decreases but plateaus at a higher value, both indicate the presence of overfitting. Overall, the fully connected neural network exhibits indications of overfitting, which may restrict its capacity for generalization, even though it attains a respectable accuracy of about 70%.

Table 12 demonstrates the performance of the hybrid CNN-DNN model. Similarly, in Fig. 7b , the training and validation curve of a CNN-DNN hybrid model is illustrated. CNNs are especially adept at extracting features from structured data, while DNNs excel at learning complex patterns. The training accuracy increases dramatically and reaches a higher plateau of about 0.78 when compared to the fully linked model. Meanwhile, the validation accuracy also increases gradually and reaches a maximum of 0.74. There seems to be less overfitting in this model, as evidenced by the reduced difference between training and validation accuracy compared to Fig. 7a . Further evidence of enhanced performance and superior generalization is provided by the training loss (blue line), which drops quickly and reaches a lower value than the fully linked model, and the validation loss (orange line), which likewise drops and plateaus at a lower level.

Findings and discussion

For detecting obfuscated malware, the proposed transfer learning-based deep neural network (DNN) model demonstrates a balance between detection performance and processing efficiency. Using feature preprocessing techniques like min-max scaling and normalisation, a multi-layered deep neural network was optimised during the initial model training on the MalwareMemoryDump dataset. The transfer learning phase utilises pre-trained model weights rather than developing a new model from scratch, which significantly reduces training time and computational overhead. Using the newly acquired feature representations from the source dataset, the pre-trained DNN is refined in this phase using the NF-TON-IoT and UNSW-NB15 datasets. In fine-tuning, lower-level layers are frozen to preserve overall malware features, while higher-level layers are adjusted to capture dataset-specific patterns. The model can more effectively generalize to hidden malware samples due

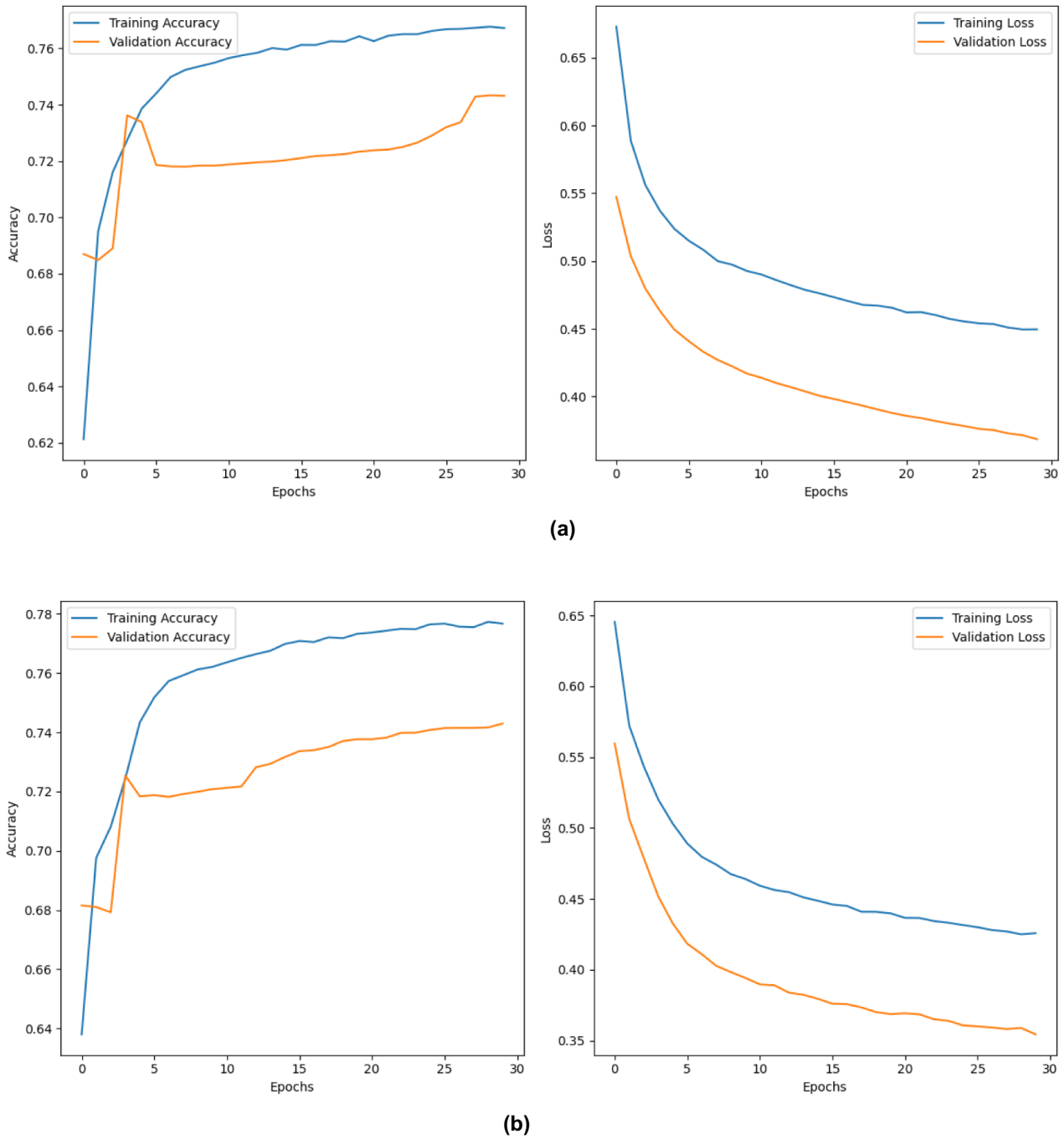


Fig. 7. Accuracy and loss curve for obfuscated malware detection. (**a**) Training and validation curve of fully connected neural net model. (**b**) Hybrid model (CNN and DNN) training and validation curve on.

to this method, which also ensures knowledge retention across datasets. The incorporation of SHAP (Shapley Additive Explanations) values, which improve interpretability by measuring feature importance in classification decisions, significantly influences the model's complexity. By iteratively assessing feature contributions to prediction probability, SHAP values are calculated, giving cybersecurity analysts insight into which features of a malware sample are the most significant for categorization. However, because many altered instances of the dataset must be created to compute individual feature attributions, this interpretability method incurs additional processing costs.

The architecture also incorporates fully connected layers, following transfer learning, which enhances feature fusion and decision-making. By producing probability distributions across binary classes, the final softmax activation function establishes whether a sample is malware or benign. Overfitting risks are mitigated by batch normalisation and dropout layers, but adjusting hyperparameters remains challenging. Model performance

Epoch	Accuracy	Loss	Val Accuracy	Val Loss
1	0.5974	0.6754	0.6815	0.5597
2	0.6914	0.5856	0.6810	0.5064
3	0.7059	0.5495	0.6792	0.4789
4	0.7203	0.5237	0.7254	0.4518
5	0.7387	0.5104	0.7184	0.4327
6	0.7471	0.4956	0.7188	0.4183
7	0.7551	0.4806	0.7182	0.4110
8	0.7596	0.4749	0.7192	0.4027
9	0.7618	0.4695	0.7199	0.3983
10	0.7625	0.4629	0.7208	0.3942
11	0.7615	0.4620	0.7212	0.3897
12	0.7653	0.4583	0.7217	0.3890
13	0.7665	0.4542	0.7282	0.3838
14	0.7657	0.4523	0.7294	0.3823
15	0.7685	0.4507	0.7317	0.3794
16	0.7723	0.4448	0.7336	0.3760
17	0.7712	0.4443	0.7340	0.3757
18	0.7719	0.4413	0.7350	0.3734
19	0.7736	0.4424	0.7370	0.3701
20	0.7742	0.4392	0.7377	0.3687
21	0.7724	0.4384	0.7377	0.3692
22	0.7736	0.4361	0.7382	0.3685
23	0.7747	0.4365	0.7398	0.3650
24	0.7735	0.4337	0.7399	0.3639
25	0.7763	0.4348	0.7408	0.3607
26	0.7762	0.4318	0.7414	0.3600
27	0.7749	0.4299	0.7415	0.3592
28	0.7737	0.4296	0.7415	0.3581
29	0.7754	0.4270	0.7416	0.3589
30	0.7776	0.4288	0.7430	0.3544

Table 12. Training and validation accuracy and loss per epoch for hybrid model.

must be balanced with the avoidance of unnecessary computational load by selecting the optimal learning rates (0.0001), dropout ratios, and layer designs. Despite the model's ability to detect obfuscated malware, problems still arise, especially when using smaller or unbalanced datasets. Biased categorisation outcomes could result from the model's inability to identify representative patterns in malware families with notably smaller sample sizes. To enhance the model's resilience to evolving obfuscation strategies, future studies should explore data augmentation methodologies, ensemble learning approaches, and adversarial training techniques.

The findings demonstrate that the proposed framework is highly applicable to real-world cybersecurity systems, particularly in security operations and forensic malware investigations. The high detection accuracy of obfuscated malware indicates that this framework may be incorporated into security solutions, including cloud-based malware detection platforms, antivirus engines, and intrusion detection systems (IDS). The interpretability of SHAP values, which provides human-readable justifications for classification decisions, enhances confidence in AI-driven security technologies. However, additional testing on dynamic malware samples and continuous model changes to accommodate evolving threats are necessary for real-world deployment. Additionally, to ensure real-time applicability in cloud and business security situations, it will be essential to optimise the computational efficiency of SHAP calculations.

Conclusion

This study employed TL and SHAP interpretability techniques to develop an effective deep learning-based system for detecting privacy-intrusive and obfuscated malware. Compared to traditional signature-based methods, which frequently struggle to detect complex or hidden malware behaviours, the proposed method demonstrated strong generalisation by initially training a DNN on the MalwareMemoryDump dataset and then fine-tuning it on two real-world cybersecurity datasets, NF-TON-Iot and UNSW-NB15. The model's capability to identify subtle malware indicators was improved by the incorporation of TL, which enabled the model to retain and adapt learnt information from memory dumps to various network traffic patterns. SHAP was also incorporated to provide insight into the model's decision-making process and identify the key components that significantly impacted categorisation outcomes. Along with making the model more transparent, this improved accountability and confidence in security systems driven by AI. The enhanced performance of the suggested

framework over current detection techniques was validated by experimental evaluation. The model's remarkable 96% accuracy on the UNSW-NB15 and on the NF-TON-IoT dataset demonstrate its resilience and versatility in a variety of attack scenarios. Overall, this study provides a scalable, interpretable, and highly accurate approach to modern malware detection, particularly when addressing obfuscated and evasive attacks.

Future studies should incorporate more real-world datasets into the proposed model to further validate its resistance to different virus types. Additionally, evaluating hybrid deep learning models may improve feature extraction and classification performance. Another option is to create adaptive models that can identify malware in real time, thereby expediting reaction times to emerging threats. Additionally, federated learning integration might enable cooperative malware detection across disparate platforms while protecting user privacy. Finally, to improve the transparency of malware categorisation, further explainability techniques, such as LIME or attention mechanisms, could be incorporated into the interpretability framework.

Data availability

All data generated or analyzed during this study are included in this published article.

Received: 14 March 2025; Accepted: 16 July 2025
Published online: 24 July 2025

References

1. Cevallos-Salas, D., Grijalva, F., Estrada-Jiménez, J., Benítez, D. & Andrade, R. Obfuscated privacy malware classifiers based on memory dumping analysis. *IEEE Access* (2024).
2. Essefi, I., Rahmouni, H. B., Solomonides, T. & Ladeb, M. F. Hipaa controlled patient information exchange and traceability in clinical processes. In *2022 IEEE 9th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, 452–460 (IEEE, 2022).
3. Jahromi, A. N., Hashemi, S., Dehghantanha, A., Parizi, R. M. & Choo, K.-K.R. An enhanced stacked lstm method with no random initialization for malware threat hunting in safety and time-critical systems. *IEEE Trans. Emerg. Top. Comput. Intell.* **4**, 630–640 (2020).
4. Carrier, T., Victor, P., Tekeoglu, A. & Lashkari, A. H. Detecting obfuscated malware using memory feature engineering. In *Icissp*, 177–188 (2022).
5. Huseynov, H., Kourai, K., Saadawi, T. & Igbe, O. Virtual machine introspection for anomaly-based keylogger detection. In *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, 1–6 (IEEE, 2020).
6. Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S. & Khayami, R. Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence. *IEEE Trans. Emerg. Top. Comput.* **8**, 341–351 (2017).
7. Shukla, S., Kolhe, G., P.D, S. M. & Rafatirad, S. Stealthy malware detection using rnn-based automated localized feature extraction and classifier. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 590–597 (IEEE, 2019).
8. Lee, Y., Woo, S., Song, Y., Lee, J. & Lee, D. H. Practical vulnerability-information-sharing architecture for automotive security-risk analysis. *IEEE Access* **8**, 120009–120018 (2020).
9. Dener, M., Ok, G. & Orman, A. Malware detection using memory analysis data in big data environment. *Appl. Sci.* **12**, 8604 (2022).
10. Chen, C.-W., Su, C.-H., Lee, K.-W. & Bair, P.-H. Malware family classification using active learning by learning. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, 590–595 (IEEE, 2020).
11. Lashkari, A. H., Li, B., Carrier, T. L. & Kaur, G. Volmemlyzer: Volatile memory analyzer for malware classification using feature engineering. In *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*, 1–8 (IEEE, 2021).
12. Mu, D. et al. Pomp++: Facilitating postmortem program diagnosis with value-set analysis. *IEEE Trans. Softw. Eng.* **47**, 1929–1942 (2019).
13. Xu, Y., Li, D., Li, Q. & Xu, S. Malware evasion attacks against IoT and other devices: An empirical study. *Tsinghua Sci. Technol.* **29**, 127–142 (2023).
14. Aurangzeb, S. & Aleem, M. Evaluation and classification of obfuscated android malware through deep learning using ensemble voting mechanism. *Sci. Rep.* **13**, 3093 (2023).
15. Shafin, S. S., Karmakar, G. & Mareels, I. Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. *Sensors* **23**, 5348 (2023).
16. Hidouri, A., Hajlaoui, N., Touati, H., Hadded, M. & Muhlethaler, P. A survey on security attacks and intrusion detection mechanisms in named data networking. *Computers* **11**, 186 (2022).
17. Yang, L., Liu, G., Dai, Y., Wang, J. & Zhai, J. Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework. *IEEE Access* **8**, 82876–82889 (2020).
18. Setiawan, H., Putro, P. A. W., Pramadi, Y. R. *et al.* Comparison of lstm architecture for malware classification. In *2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, 93–97 (IEEE, 2020).
19. Vinayakumar, R., Soman, K., Poornachandran, P., Akarsh, S. & Elhoseny, M. Improved dga domain names detection and categorization using deep learning architectures with classical machine learning algorithms. In *Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments*, 161–192 (Springer, 2019).
20. Dang, Q.-V. Detecting obfuscated malware using graph neural networks. In *International Conference on Power Engineering and Intelligent Systems (PEIS)*, 15–25 (Springer, 2023).
21. Gorment, N. Z., Selamat, A. & Krejcar, O. Obfuscated malware detection: Impacts on detection methods. In *Asian Conference on Intelligent Information and Database Systems*, 55–66 (Springer, 2023).
22. Naeem, M. R. et al. A malware detection scheme via smart memory forensics for windows devices. *Mobile Inf. Syst.* **2022**, 9156514 (2022).
23. Hossain, M. A. & Islam, M. S. Enhanced detection of obfuscated malware in memory dumps: A machine learning approach for advanced cybersecurity. *Cybersecurity* **7**, 16 (2024).
24. Hasan, S. R. & Dhakal, A. Obfuscated malware detection: Investigating real-world scenarios through memory analysis. In *2023 IEEE International Conference on Telecommunications and Photonics (ICTP)*, 01–05 (IEEE, 2023).
25. Alani, M. M., Mashatan, A. & Miri, A. Xmal: A lightweight memory-based explainable obfuscated-malware detector. *Comput. Secur.* **133**, 103409 (2023).
26. Öztürk, A. & Hızal, S. Detection and analysis of malicious software using machine learning models. *Sakarya Univ. J. Comput.Inf. Sci.* **7**, 264–276 (2024).
27. Maniriho, P., Mahmood, A. N. & Chowdhury, M. J. M. Memaldet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations. *Comput. Secur.* **142**, 103864 (2024).
28. Prasad, A. & Chandra, S. Botdefender: A collaborative defense framework against botnet attacks using network traffic analysis and machine learning. *Arab. J. Sci. Eng.* **49**, 3313–3329 (2024).

29. Prasad, A. & Chandra, S. Phiusiil: A diverse security profile empowered phishing url detection framework based on similarity index and incremental learning. *Comput. Secur.* **136**, 103545 (2024).
30. Shuai Li, A., Iyengar, A., Kundu, A. & Bertino, E. Transfer learning for security: Challenges and future directions. arXiv e-prints arXiv–2403 (2024).

Acknowledgements
The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research workthrough the project number "NBU-FFR-2025-2443-04". The authors extend their appreciation to the Deanship of Research and Graduate Studies at King Khalid University for funding this work through LargeResearch Project under grant number RGP2/473/46.

Author contributions
A.A.: Conception and design of study, Analysis and/or interpretation of data, Writing—original draft, Methodology, Acquisition of data. S.A.: Writing—original draft, Writing—review & editing, Data Analysis, Implementation, Acquisition of data. N.K.: Writing—original draft, Supervision, Writing—review & editing, Methodology. A.A.H.: Writing—original draft, Writing—review & editing, Methodology, Resources, Visualizations. B.B.: Writing—original draft, Writing—review & editing, Visualizations, Funding Acquisition. M.A.: Writing—original draft, Writing—review & editing, Methodology, Funding Acquisition, Supervision, Administration. S.A.: Writing—original draft, Acquisition of data, Conceptualization, Writing—review & editing, Methodology.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to N.K. or S.A.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.