



# OPEN A novel technique for ransomware detection using image based dynamic features and transfer learning to address dataset limitations

Jannatul Ferdous<sup>1</sup>✉, Rafiqul Islam<sup>2</sup>, Arash Mahboubi<sup>3</sup> & Md Zahidul Islam<sup>4</sup>

The increasing frequency of ransomware attacks necessitates the development of more effective detection methods. Existing image-based ransomware detection approaches have largely focused on static analysis, overlooking specialized ransomware behaviors such as encryption, privilege escalation, and system recovery disruption. Although dynamic and memory forensics-based visualization methods exist in the broader malware domain, they primarily target generic malware families and often rely on memory dumps or system snapshots without transforming behavioral features into spatially meaningful representations. Moreover, traditional machine learning methods such as Random Forest (RF), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) typically depend on manual feature engineering and large labelled datasets, limiting scalability and adaptability. To address these limitations, we propose a novel behavior-to-image ransomware detection framework that transforms dynamic behavioral features extracted from sandbox-generated JSON reports into two-dimensional (2D) grayscale and color image representations, optimized for transfer learning (TL), enabling effective classification under small-data conditions. Our approach integrates domain-specific feature filtering and impact analysis to ensure the selection of the most ransomware-relevant attributes. TL subsequently automates feature extraction and classification, eliminating the need for separate feature selection procedures and overcoming the time-consuming process of manual feature engineering. Furthermore, by leveraging prior knowledge from large-scale image datasets, TL significantly mitigates the need for extensive labelled data while maintaining high detection accuracy and strong generalization. Experimental results demonstrate that fine-tuned pretrained models, notably ResNet50, achieve up to 99.96% accuracy with a minimal loss factor of 0.0026, even with a small dataset of 500 ransomware and 500 benign samples. We further validated the model's interpretability through t-SNE visualizations and saliency maps, confirming its ability to focus on class-discriminative behavioral patterns. The low misclassification rate, along with the transparency of the model, highlights its potential for practical deployment in ransomware detection systems.

**Keywords** Ransomware, Portable executable (PE), Dynamic analysis, Convolutional neural network, Pretrained models, Transfer learning, Image classification

The exponential rise in connected devices across sectors such as IoT, industrial IoT (IIoT), healthcare, e-banking, and cyber-physical systems (CPSs) has introduced significant cyber vulnerabilities, particularly for unaware users<sup>1</sup>. Among these, ransomware stands out as one of the most persistent and damaging threats in the cybersecurity landscape, leveraging this extensive connectivity to exploit vulnerable users and organizations. The most recent report on ransomware activity indicates an alarming 18% increase in activity during the initial five months of 2024<sup>2</sup>. Additionally, the SOPHOS 2024 ransomware report shows a dramatic 500% increase in average

<sup>1</sup>School of Computing, Mathematics and Engineering, Charles Sturt University, Wagga Wagga, NSW 2650, Australia. <sup>2</sup>School of Computing, Mathematics and Engineering, Charles Sturt University, Albury, NSW 2640, Australia. <sup>3</sup>School of Computing, Mathematics and Engineering, Charles Sturt University, Port Macquarie, NSW 2444, Australia. <sup>4</sup>School of Computing, Mathematics and Engineering, Charles Sturt University, Bathurst, NSW 2795, Australia. ✉email: jferdous@csu.edu.au

ransom payments, which reached \$2.73 million in 2024—nearly \$1 million more than in the previous year<sup>3</sup>. This growing frequency of ransomware attacks and high ransom demands have become significant cybersecurity concerns, necessitating the development of effective detection methods.

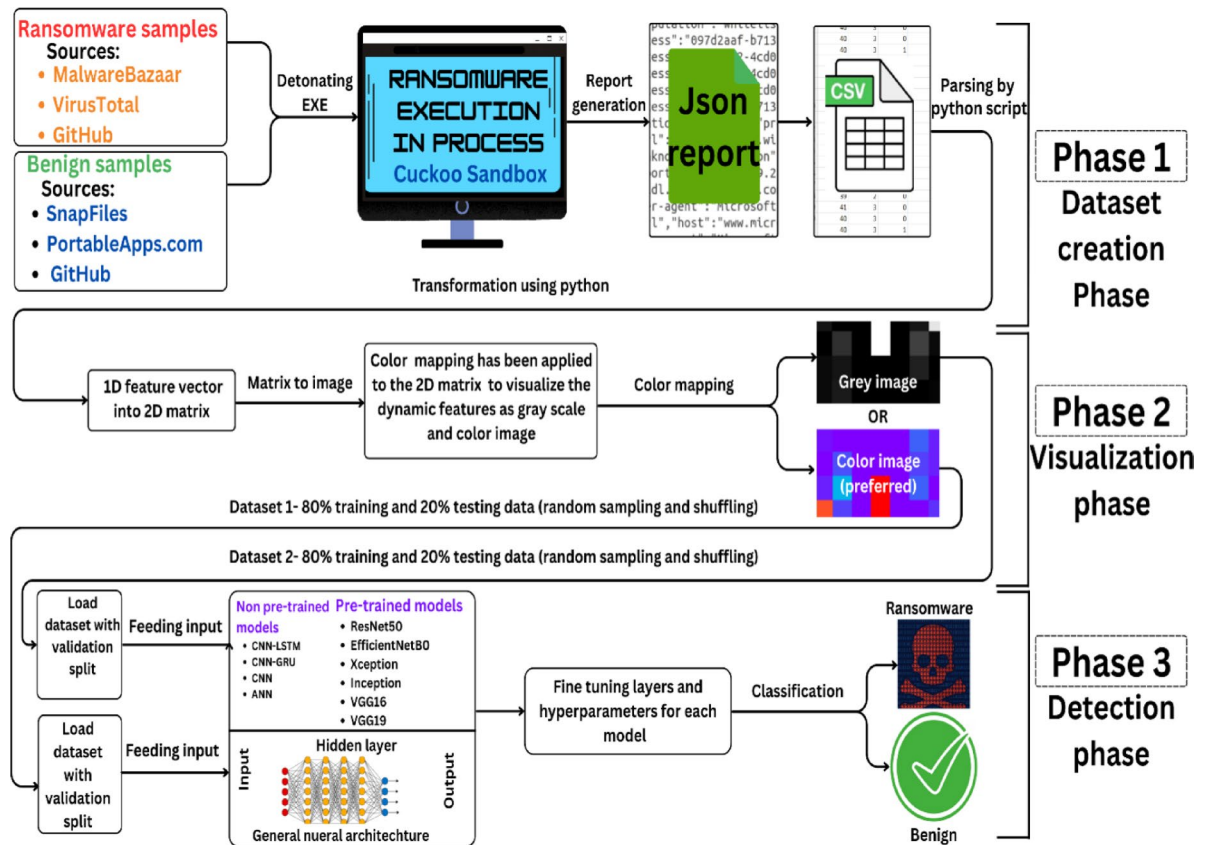
Ransomware is a type of malware that encrypts victims' files, making them inaccessible, and demands a ransom, usually cryptocurrency, for the decryption key. Early ransomware strains were relatively simple but have evolved to become more sophisticated, employing stronger encryption algorithms, targeting specific organizations, and utilizing advanced evasion techniques<sup>4</sup>. Recent high-profile ransomware attacks, such as Lockbit, Conti, Clop, and Blackbasta, have demonstrated the devastating impact of ransomware on individuals, businesses, and critical infrastructure, leading to significant financial losses, operational disruptions, and reputational damage<sup>5,6</sup>. Ransomware detection relies on two main analytical approaches, static and dynamic analysis, to identify malicious characteristics or features. Static analysis involves examining the code structure without execution, making it a faster and safer method but limited by its inability to handle obfuscated or encrypted ransomware. Identifying specific patterns or signatures in binaries is ideal for detecting known threats. In contrast, Dynamic analysis observes ransomware behavior in controlled environments, enabling detection of zero-day exploits and APTs that adapt based on the system. Although slower, Dynamic analysis provides deeper insights by capturing real-time interactions like network traffic and file manipulations, making it practical for complex ransomware<sup>7</sup>.

In the current literature, researchers have employed various machine learning (ML) and deep learning (DL) methods to identify and categorize ransomware. As traditional ML methods usually require extensive manual feature engineering, they are labor-intensive and prone to human bias. These models also lack the ability to learn abstract or spatial hierarchies from raw data, thereby limiting their adaptability to evolving ransomware behaviors<sup>8</sup>. To address this limitation, recent studies have increasingly used DL techniques for ransomware detection, yielding promising results<sup>9–11</sup>. DL automates feature extraction, thereby bypassing the requirement for manual processing. In recent studies, researchers have employed image-based approaches to improve ransomware detection by converting the complete binary executable into images via various DL techniques, including non-pretrained and pretrained models (transfer learning approaches). Several non-pretrained DL models, such as deep residual networks (DRNs), CNNs, and recurrent networks, have shown promise in image-based ransomware detection with notable approaches<sup>12</sup>, IMCFN<sup>13</sup>, CNN-LSTM<sup>14</sup>, SDIF-NN<sup>15</sup>, and malware image processing via the transfer learning (TL) approach<sup>8</sup>.

Transfer learning is a machine learning technique that involves leveraging a pretrained deep learning model on a large dataset to perform a related task on a smaller dataset. TL allows the model to apply previously learned features to new tasks. It has recently achieved significant progress in image-based malware detection and has enabled high accuracy in classifying malware images with less labelled data. This is advantageous for ransomware detection because it addresses the challenges of small training datasets by fine-tuning pretrained models for specific detection tasks. Several studies have used this technique for malware classification, including those<sup>8,15–19</sup>.

While studying existing ransomware detection research, we identified several critical challenges and research gaps. Most existing studies have focused on static analysis for image-based ransomware detection, which faces difficulties in effectively identifying advanced ransomware variants using obfuscation techniques. These sophisticated forms of ransomware necessitate the exploration of runtime behavioral features to enhance their detection capabilities<sup>20</sup>. However, visual representations of runtime behavioral features have not yet been explored in the existing image-based ransomware detection systems. Further investigations have identified two significant challenges in ransomware detection using dynamic features. First, these methods often face limitations owing to outdated datasets. Second, there is limited availability of active ransomware samples in public repositories, the rapid evolution of ransomware, and ethical and legal constraints<sup>21,22</sup>. These challenges result in substantially smaller datasets for dynamic analyses. Limited data increases the overfitting risk, reduces diverse behavior identification, and undermines reliability in detecting new threats, resulting in poor performance. Several previous studies<sup>8,23,24</sup> have utilized various ML and DL models to train CSV-formatted data obtained from JSON reports through dynamic analysis. However, the results are unsatisfactory. They obtained 95.9%, 97.28% and 96.34% accuracies. We found that this data format showed lower accuracy for small datasets.

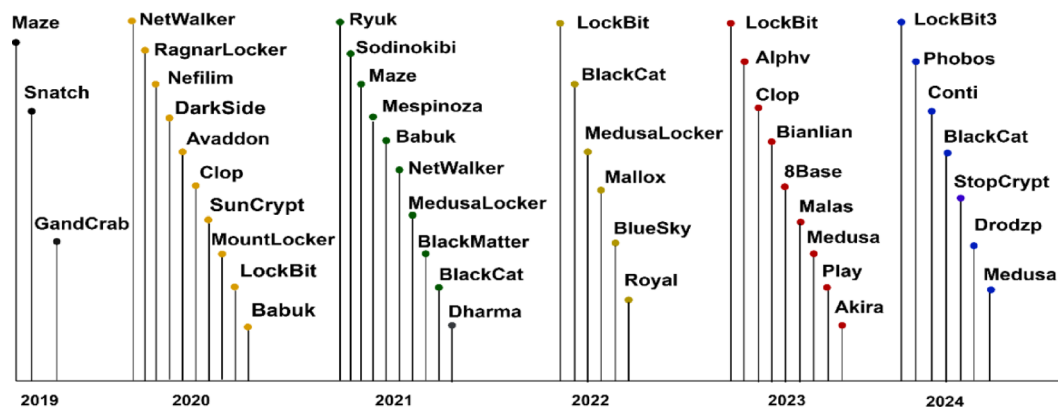
To address these limitations, this study introduces an innovative behavior-to-image ransomware detection pipeline. This pipeline transforms structured dynamic features extracted from sandbox-executed JSON reports into semantically meaningful two-dimensional images utilizing transfer learning, as depicted in Fig. 1. The process begins with dynamic analysis, which extracts the behavioral traits of ransomware in JSON format, offering rich but complex hierarchically structured insights. The JSON data were converted into structured CSV features to simplify the dataset preparation and feature engineering processes. Although this representation is suitable for numerical analysis, it inherently lacks the spatial relationships that modern deep learning models such as CNNs rely on. To address this gap, we converted CSV data into image representations by adding a new dimension for feature interpretation. For this conversion, we developed two Python scripts: one to automate feature extraction from JSON to CSV formats and the other to transform CSV data into images. This allows the use of pretrained deep learning models, which are extensively trained on large image datasets, to leverage prior knowledge and extract meaningful spatial patterns from these image-based features. To evaluate the effectiveness of the proposed approach, we employed baseline non-pretrained models, including convolutional neural networks (CNNs), artificial neural networks (ANNs), a hybrid model that combines a CNN with long short-term memory (LSTM), and another hybrid model that integrates a CNN with gated recurrent units (GRUs) to train image-based representations. These models struggled to achieve satisfactory results, indicating their limited ability to be generalized to small training datasets. This reinforces the necessity of using pretrained models via the TL approach. The key contributions of this study are as follows:



**Fig. 1.** Proposed ransomware detection workflow.

1. A Novel behavior-to-image ransomware detection pipeline: We present an innovative end-to-end framework that converts structured dynamic behavioral features derived from sandbox JSON reports into image formats. This framework is specifically tailored for TL-based classification when dealing with small datasets. Unlike prior visual methods that rely on static binary representations (e.g., raw executable bytes, PE headers), sequences of system calls, or memory dumps, our pipeline captures semantically rich runtime behaviors and enhances the ability of the model to detect polymorphic and evolving ransomware variants (e.g., Conti, Ryuk, and Lockbit) in real time.
2. Domain-guided feature engineering with automated extraction: We implement a domain-specific filtering strategy to select 100 behaviorally discriminative features, such as privilege escalation, dropped files, encryption patterns, and anti-analysis techniques. A custom Python-based extractor automates this process across large JSON datasets, thereby ensuring reproducibility and consistency. Each selected feature is numerically encoded and spatially arranged to preserve interpretability in the resulting image representation.
3. Balanced and threat-informed dataset: We curate a balanced dataset of 500 ransomware and 500 benign samples, selected from 25 active and impactful ransomware families from 2019 to 2024 (Fig. 2). Unlike many prior studies that relied on outdated or generic datasets, our dataset reflects the current threat landscapes guided by domain-specific intelligence and prevalence criteria. This addresses critical gaps in prior research, particularly the lack of behavioral diversity and the use of obsolete malware samples.
4. Empirical validation of small-data effectiveness and computational efficiency via TL: We demonstrate that fine-tuned pretrained models (especially ResNet50), combined with behavior-to-image transformation, achieve high detection accuracy even with limited datasets. Additionally, by leveraging pretrained CNN models through TL, we significantly reduced the computational burden compared with training deep learning models from scratch. This enables faster convergence, lower resource consumption, and practical applicability, even under small-data conditions.
5. Model interpretability and reliability: We enhance trust in the proposed detection pipeline by integrating t-SNE and saliency map visualizations, which provide insights into feature space separability and decision focus. These visual tools also supported our misclassification analysis, confirming that the model's rare false negatives occurred near class boundaries and could explain the model's robustness and interpretability.

Given the limitations of static and manual or rule-driven behavioral methods against modern polymorphic ransomware, this study introduces a behavior-driven, image-based detection framework. Leveraging real-time dynamic analysis, recent ransomware families, and pretrained CNNs, the approach addresses the lack of up-to-



**Fig. 2.** Timeline showing the prevalence and family wise distribution of ransomware samples collected between 2019 and 2024. The ordering reflects attack frequency and global impact based on multi-source threat intelligence reports.

date, behaviorally rich datasets, overcomes the reliance on manual feature engineering, and enables effective detection even in small-data scenarios.

The remainder of this paper is organized as follows. We begin with a review of related works, summarize the key findings, and identify the gaps that our research aims to address. This is followed by a detailed explanation of the proposed methodology, including the processes of dataset creation, image conversion, and detection strategy. We then present our experimental results, illustrate them in detailed Tables and Figures, and compare our results with those of related studies. The discussion section offers a critical analysis of the factors contributing to our model's performance and the limitations of our approach. The paper concludes by summarizing the key contributions and their implications for ransomware detection along with suggestions for future research.

## Related work

In this section, we review and compare existing research relevant to ransomware detection with a focus on methodologies involving static analysis, dynamic analysis, image-based techniques, and transfer learning. Currently, very few related studies based on TL approaches for ransomware classification exist in the literature. Hence, we describe associated studies on ransomware and more general malware-oriented papers.

Table 1 provides a comparative overview of recent ransomware and malware detection studies, summarizing their dataset sources, feature categories (e.g., static, dynamic, or visual), size of the dataset, model types, and their performance in terms of detection accuracy and key limitations. This summary highlights the lack of approaches that utilize dynamic behavior-to-image transformations, particularly for ransomware classification tasks.

Traditional approaches primarily focus on static analysis, in which features are extracted from the binary code of malicious files to detect potential threats. Manavi and Hamzeh proposed two static analysis methods for detecting ransomwares. The first approach<sup>25</sup> uses PE headers to form a graph, which is then mapped to an eigenspace and transformed into a feature vector to train a random forest (RF) classifier. This method achieved an accuracy of 93.30%. The second approach<sup>26</sup> extracted the header from executable files, built a grayscale image using a zigzag pattern, and trained a custom CNN model with an accuracy of 93.33%. The advantage of these methods is that they do not require a robust understanding of the PE header structure. One limitation is that they did not evaluate the detection of new ransomware families. The study by<sup>27</sup> focused on ransomware detection using deep reinforcement learning (DRL). This study achieved 97.9% accuracy and a 2.1% false detection rate, highlighting the importance of using advanced machine learning techniques to improve the accuracy of ransomware detection. However, the proposed method may not work for new and unknown ransomware variants, limiting static analysis adaptability to evolving threats.

By contrast, dynamic analysis monitor runtime behavior of suspected files and capture critical changes in the system, such as file operations, registry modifications, and network communications. For example<sup>8</sup>, proposed a dynamic analysis-based ransomware detection method called EldeRan, which uses regularized logistic regression (RLR). Their approach uses dynamic features like Windows API calls, registry key, file system, and directory operations. They also shared a dataset with 582 ransomware samples from 11 families. They achieved an average accuracy of 96.34%. However, this study did not evaluate the ability of the method to detect new previously unseen ransomware families. The authors in<sup>24</sup> developed a two-stage ransomware detection model using the Markov model and random forest. They first analyzed the API call patterns of ransomware and then combined them with other data such as registry keys and file extensions to enhance detection. They achieved an overall accuracy of 97.3% with 4.8% FPR. Additionally<sup>23</sup>, proposed an improved TextCNN model using API call sequences for ransomware detection, achieving an accuracy of 0.959. In<sup>28</sup>, the authors proposed Xran, an explainable deep learning-based ransomware detection system using dynamic analysis. Their method integrates features such as API call sequences, DLLs, and mutex operations to enhance the feature space.

Xran utilized a CNN architecture along with XAI techniques such as LIME and SHAP for interpretable detection. The model achieved a true positive rate (TPR) of 99.4%, outperforming state-of-the-art methods. However, this approach is computationally expensive and time consuming. The study<sup>29</sup> proposed SwiftR, a

Paper	Dataset source	Feature type (Static, dynamic or visual)	Feature name	Dataset size (R = Ransomware, M = Malware, B = Benign)	Name of the models used	Accuracy (%)	Limitation
25	VirusTotal and VirusShare	Static	PE headers	R = 6000, B = 6000	RF	93.30	Did not evaluate detection of new ransomware families.
26	VirusTotal and VirusShare	Visual static features	PE headers bytes-based images	R = 1000, B = 1000	CNN	93.30	Did not evaluate detection of new ransomware families
27	VirusShare	Static	PE headers	R = 35,367, B = 27,118	DRL	97.9	May not be effective against new and unknown ransomware variants.
8	VirusShare Software.Informer	Dynamic	API calls, file system, registry keys, directory operations etc.	R = 582, B = 942	Regularized logistic regression (RLR)	96.34	Did not assess detection of new ransomware families.
24	VirusShare, Softonic.Com	Dynamic	API calls, file system, Registry key, etc.	R = 1176, B = 1160	Markov model and random forest	97.28	May struggle to adapt with modern sophisticated ransomware.
23	Sangfor technologies	Dynamic	API call sequences	R = 1000, B = 1000	TextCNN	95.9	Convolutional layers not fully optimized and generalizability to unseen ransomware is unclear.
28	ISOT, VirusShare, Software.Informer	Dynamic	API call sequences, DLLs, and mutex operations	R = 668, B = 668	2-layered CNN	98.2	This method demands substantial computational power and is notably time-consuming.
29	vipre.com ninite.com	Static and dynamic	API Sequence, VEX sequence, Byte entropy, Behavioral analysis report	R = 10.3 K, B = 10 K	CNN, LSTM, MLP	98.75	High computational resource demand.
12	VirusShare	Visual static features	PE header bytes-based images	R = 2023, B = 2134	Xception	98.20	Relies on static features, limiting generalizability by missing dynamic ransomware behaviors.
14	VirusShare and VirusTotal	Static visual	PE files into images	M = 974,957, B = 89,004	CNN-LSTM	99.91	The study did not examine the system's scalability for larger datasets.
30	Kaggle by Microsoft 2015	Static visual	Hashes as grayscale images	M = 5000, B = 5000	CNN	98.862	Ineffective against packed or encrypted malware and need more hardware resources for faster detection.
16		Static visual	APK files images	R = 500, B = 500	ResNet50, AlexNet, InceptionV3, ResNet101, etc.	99.5	May struggle to combat with modern complex and evolving ransomware.
13	Virus-share Maling dataset	Static visual	PE files or malware binaries into images	M = 9339	VGG16, ResNet50, IMCFN	98.82	The authors overlooked the effects of image size and data imbalance on their method. Small and old dataset.
15	Maling dataset	Static visual	PE files or malware binaries into images	M = 9339	VGG16, VGG19, ResNet50, and InceptionV3, MLP	98.55	This method did not assess scalability or performance on larger datasets.
21	virussign.com	Hybrid (static and dynamic)	Visual static and hybrid (static + dynamic)	M = 512, B = 888	VGG16 network.	94.70 for hybrid and 91.41 for static image	The study did not assess the method's scalability or resource efficiency.

**Table 1.** Comparative summary of state-of-the-art ransomware and malware detection studies, detailing dataset sources, feature types (static, dynamic, or visual), model architectures used, and observed limitations in accuracy or scalability.

cross-platform ransomware fingerprinting method that uses hierarchical neural networks (HNNs) on hybrid features to achieve high accuracy.

In recent years, Image-based detection techniques have emerged as novel approaches for ransomware detection. This method converts malicious binary files into visual representations, such as grayscale or color images, and applies deep learning to classify malware from benign samples.

For example, Moreira et al.<sup>12</sup> applied static analysis to detect ransomware by converting PE header files into color images in a sequential vector pattern and classifying them via Xception convolutional neural networks without transfer learning. They obtained 98.20% accuracy and emphasized the importance of using features extracted during execution to better understand sophisticated ransomware. Bensaoud and Kalita<sup>14</sup> proposed a malware classification system using a hybrid CNN-LSTM model that transforms API calls and opcode sequences into N-gram features, achieving 99.91% accuracy on a dataset of 9,749,57 samples. While the study also explored recent deep learning architectures such as ConvNeXt-T, Swin-T, and ViT-G/14, the CNN-LSTM approach outperformed them. However, this study did not investigate the scalability of the system for larger datasets or its performance in real-world scenarios. Ni et al.<sup>30</sup> proposed a CNN model that utilized the SimHash technique, employing locally sensitive hashing to represent hashes as grayscale images, which were then fed into the CNN model for training. However, the authors noted the need to validate their approach for packed and sophisticated real-world malware, as mentioned in their future research scope. Although this method has shown promise in

leveraging visual features for detection, its reliance on static characteristics limits its ability to detect dynamic behavioral patterns, which are crucial for identifying modern ransomware variants.

The use of transfer learning and pretrained deep learning models has significantly advanced malware detection to address the limitations of traditional and image-based methods. TL leverages pretrained models on large datasets, allowing the transfer of learned knowledge to new tasks with minimal retraining. In cybersecurity, pretrained CNNs such as ResNet and AlexNet are used to detect malware by analyzing visualized malicious file features. For example, Vasan et al.<sup>31</sup> leveraged pretrained VGG16 and ResNet50 CNN models for feature extractors and combined their feature maps to achieve excellent classification accuracy using an SVM. However, the authors did not address the impact of the image size and data imbalance on their approach. Kumar and Panda<sup>15</sup> proposed SDIF-CNN, a malware detection framework that uses fine-tuned CNN models and stacked feature maps. Their method leverages fine-tuned VGG16 as a feature extractor, combined with features from VGG19, ResNet50, and InceptionV3, to create a stacked feature map. The optimized feature vector was used to train six classifiers, with MLP achieving the highest accuracy of 98.55% on the Malling dataset and 94.78% on real-world malware datasets. The framework demonstrated resilience to obfuscation techniques without relying on resource-intensive dynamic analysis. However, this study did not evaluate the scalability or performance of larger datasets. Almomani et al.<sup>16</sup> proposed E2E-RDS, a ransomware detection system that combines static ML and vision-based DL approaches. The vision-based approach converted binary executable files into 2D images and employed 19 fine-tuned and transfer-learned CNN models, with the fine-tuned ResNet50 model attaining 99.5% accuracy, surpassing alternative techniques. Additionally, Rezende et al.<sup>32</sup> proposed a malware classification method using TL with ResNet-50 on grayscale byte plot images, achieving 98.62% accuracy on 9,339 samples from 25 families. The study highlights the effectiveness of TL but lacks a discussion on handling imbalanced datasets or emerging malware variants. Huang et al.<sup>21</sup> proposed a deep-learning-based malware detection method using malware visualization techniques. Static images were generated from file features and dynamic images from behavior reports via the cuckoo sandbox and merged into hybrid images. They employed a pretrained CNN model, VGG16, to train both datasets, achieving an accuracy of 94.70% on the hybrid dataset and 91.41% on the static dataset. The results showed that the hybrid approach outperformed the static approach in terms of malware detection. However, this study did not evaluate the scalability of the method for large deployments or the resource efficiency.

## Summary

According to Table 1, while previous research has achieved promising results via advanced machine learning techniques such as deep learning and transfer learning, significant gaps remain in ransomware detection. Notably, existing studies have relied primarily on static numeric or static-visual features (e.g., PE headers or byte-based images). Dynamic analysis has been used in several studies, including EldeRan<sup>8</sup>, XRan<sup>28</sup>, and<sup>24</sup>, these approaches primarily target general malware detection and are not tailored to the behavioral aspects which is critical for addressing evolving ransomware threats. Many methods struggle with scalability because of the obfuscation techniques of small datasets, such as packing and encryption. Although some studies have explored dynamic features, they often require significant computational resources and lack real-time adaptability. Furthermore, there is a notable scarcity of research utilizing TL with pretrained models specifically for ransomware detection.

The proposed method introduces image-based dynamic features and transfer learning for ransomware detection, using our balanced dataset of the 25 most relevant ransomware families identified by cyber-threat reports. These addresses critical gaps in literature, including static feature reliance, small dataset generalization, and obfuscation resilience. Image-based dynamic features enhance resilience against obfuscation techniques like packing and encryption. Transfer learning with pretrained models ensures robust performance on small datasets. Lightweight architectures of pretrained models improve computational efficiency and scalability, increasing accuracy and generalizability beyond current methods.

## The proposed ransomware detection approach

In this section, we describe the proposed ransomware detection method. Figure 1 illustrates the high-level workflow of the behavior-to-image based ransomware detection pipeline. The workflow presents an end-to-end framework for ransomware detection that is designed in three primary phases: (1) dataset creation, (2) visualization, and (3) detection. This structured approach transforms the dynamic behavioral data of ransomware into visual formats, allowing for deep learning-based analysis via both nontrained and pretrained models. Each phase uniquely contributes to the development of a reliable and efficient detection system, which is particularly valuable for managing small and diverse datasets of active ransomware samples. The following subsections describe each phase and the workflow component.

### Phase 1: dataset creation phase

In Phase 1, the workflow encompasses the collection, detonation, and analysis of ransomware and benign samples to construct a feature-rich dataset for machine learning.

#### *Sample collection, distribution, and balancing strategy*

We created a balanced dataset by collecting 1000 binary executable samples, including 500 legitimate software applications and 500 ransomware instances, which were selected from 25 distinct ransomware families based on threat intelligence classification. All ransomware samples were collected from MalwareBazaar<sup>33</sup> and VirusShare<sup>34</sup>, whereas benign samples were obtained from SnapFiles<sup>35</sup>, PortableApps.com<sup>36</sup>, and GitHub<sup>37</sup>. This diverse collection ensured a comprehensive and representative dataset for real-world scenarios.

We applied two criteria for selecting ransomware families to ensure relevance and impact in our dataset: (1) a high prevalence of attacks and a significant impact on organizations globally, as reported by various renowned

cybersecurity sources from 2019 to 2024, and (2) the appearance of at least two independent reports from reputable cybersecurity firms.

This selection method identifies the most relevant and widespread ransomware families, ensuring that a dataset accurately represents major ransomware threats. Figure 2 presents a chronological timeline of ransomware families from 2019 to 2024 based on their prevalence and impact. The ordering is based on multi-source threat intelligence reports from 2019<sup>38</sup>, 2020<sup>39</sup>, 2021<sup>40</sup>, 2022<sup>5</sup>, 2023<sup>41</sup>, and 2024<sup>6</sup>.

This timeline highlights the evolution of ransomware, starting with prominent early families such as Maze and GandCrab, moving through increasingly sophisticated and widespread ransomware, such as NetWalker, LockBit, and BlackCat, and extending to recent threats, such as Lockbit3 and StopCrypt, in 2024. Each ransomware family was selected and positioned on the basis of its documented reach and influence within specific years, providing a clear view of the shifting ransomware landscape and progression of major threat actors over time. Focusing on prevalent ransomware families, such as LockBit, MedusaLocker, BlackCat, Phobos, and Conti, captures essential adaptive and sophisticated behaviors. This prioritization enhances the understanding of high-risk ransomware characteristics and improves the relevance of the dataset to detection models. Additionally, this study reveals trends in ransomware evolution and the techniques used by major threat actors, supporting a comprehensive analysis of current and emerging threats.

For the collection of ransomware samples, we followed the same criteria outlined by<sup>12</sup>, wherein three criteria were employed based on vendor engine detection from VirusTotal<sup>42</sup>.

- 1) If at least 45 antivirus engines on VirusTotal marked a file as malicious.
- 2) The sample was included if at least 15 antivirus engines on VirusTotal classified it specifically as ransomware.
- 3) The majority of engines or at least ten engines should identify the file as belonging to the same ransomware family.

Table 2 provides details regarding the ransomware families used in this study, including the number of samples and an overview of each.

Family name	No of samples	Overview of each family
Ryuk	15	Since 2018, Ryuk has targeted large enterprises and government institutions, often delivered through TrickBot or Emotet malware.
GandCrab	28	GandCrab, a Ransomware-as-a-Service (RaaS), spread by phishing emails, exploit kits, and compromised Remote Desktop Protocol (RDP) services.
Cerber	9	Active since 2016, Cerber has been among the first to offer RaaS and spread it by phishing and exploit kits.
Maze	8	Maze pioneered the “double extortion” tactic, encrypting data while threatening to leak it. It targeted large corporations, causing both financial and reputational harm.
Makop	12	Active since 2020, Makop has targeted small and medium businesses by exploiting RDP vulnerabilities.
RagnarLocker (Ragnarok)	23	RagnarLocker targets corporate networks and uses RDP and app vulnerabilities to spread malware and disables volume shadow copies <sup>43</sup> .
NetWalker	24	NetWalker, a C++-coded ransomware, is promoted as RaaS by the CIRCUS SPIDER group, encrypts local files network shares, and exploits logged-in user tokens for further access <sup>44</sup> .
Mespinoza (Pysa)	18	Mespinoza (Pysa) enters systems through spam or RDP credentials, using tools like Mimikatz and PowerShell Empire for credential theft and lateral movement <sup>40</sup> .
Clop	10	Over the past three years, Clop has been an active ransomware group, targeting major global organisations using advanced extortion tactics <sup>45</sup> .
REvil (Sodinokibi)	26	REvil, also known as Sodinokibi, a RaaS model, has been highly active, initiating double extortion in 2020 and exploiting a zero-day vulnerability in the Kaseya VSA server in July 2021 <sup>40</sup> .
LockBit	30	LockBit is a highly automated and self-propagating ransomware, making it unique from many other ransomware attacks.
Phobos	24	Phobos typically targets small to medium-sized businesses through insecure RDP connections
WastedLocker	25	WastedLocker is linked with the Evil Corp group and is known for highly targeted attacks on large firms.
DarkSide	24	DarkSide became notorious for attacking Colonial Pipeline, causing major fuel shortages in the U.S.
Avaddon	16	Avaddon, a RaaS operator, employed DDoS attacks for triple extortion in January 2021. It targets victims’ websites or networks to increase ransom demands <sup>40</sup> .
MountLocker	15	MountLocker is known for targeting corporate networks and exfiltrating large amounts of sensitive data.
Thanos	28	Thanos ransomware is notable for its use of the RIPlace evasion technique, which makes detection by antivirus software difficult.
Conti	18	Conti ransomware is known for its speed and efficiency, often deployed in highly coordinated attacks against large organisations.
Nefilim	11	Nefilim ransomware is like Nemty and is known for its data exfiltration before encryption, used as leverage for ransom payments.
BlackCat	30	BlackMatter is a successor to DarkSide, targeting large corporations with sophisticated attacks.
Babuk	27	Babuk ransomware targeted various sectors, including healthcare, and was one of the first groups to adopt a RaaS model for affiliate use.
BlackMatter	28	BlackMatter is a successor to DarkSide, targeting large corporations with sophisticated attacks.
MedusaLocker	28	MedusaLocker primarily targets Windows systems, using batch scripts to propagate and encrypt data.
Mallox	15	Mallox ransomware targets vulnerable servers, encrypting data and demanding ransom for decryption
BlueSky	8	BlueSky ransomware is less widespread but is known for its ability to encrypt both local and networked systems efficiently.

**Table 2.** Ransomware families used in this study offers a brief description of each family and the number of samples collected per family.

---

**Input:** JSON reports generated by Cuckoo Sandbox  
**Output:** Structured CSV file with 100 numerically encoded features (F) per sample

- 1: Initialize file list:  
files  $\leftarrow$  {1. JSON, 2. JSON, ..., n. JSON}
- 2: Initialize empty list:  
data\_list  $\leftarrow$   $\emptyset$
- 3: Define function ProcessJSON(file):
- 4: Load JSON file into dictionary D
- 5: Extract key behavioral metrics, for example:
  - Process activity (e.g., total\_processes  $\leftarrow$  D['counters']['processes']['total'])
  - File system events (e.g., created\_files, deleted\_files)
  - Network communications (e.g., dns\_request\_ip\_count, tcp\_connections)
  - Registry modifications
  - Anti-VM and anti-debugging checks
  - MITRE ATT&CK phase observations
  - ..... {F\_1, F\_2, ..., F\_100} # ( Full feature list as defined in Supplementary Table S1)
- 6: Preprocess the extracted features:
  - Boolean fields  $\rightarrow$  map to binary values (0 or 1)
  - String fields  $\rightarrow$  quantified by count or length
  - Missing fields  $\rightarrow$  set to 0
- 7: Store processed features:  
result  $\leftarrow$  {feature\_name: feature\_value for all extracted features}
- 8: return result
9. end for
- 10: for each file  $\in$  files do:  
entry  $\leftarrow$  ProcessJSON(file)  
Append entry to data\_list
- 11: Write data\_list to CSV:  
Open output.csv for writing  
Write header row from keys of first entry  
For each entry  $\in$  data\_list:  
Write feature values as row to CSV
12. end function

---

**Algorithm 1.** Extraction and structuring of dynamic behavioral features from JSON reports.

---



---

**Input:** Structured CSV file with 100 numerically encoded features per sample  
**Output:** PNG images saved in directories 'R' (ransomware) and 'G' (benign), in grayscale or color

- 1: function GenerateImagesFromCSV (file, color\_mode)
- 2: Initialize class directories:  
directories  $\leftarrow$  {'R', 'G'}  
for each directory D in directories do:  
if D does not exist, create directory D
- 3: Read CSV data:  
df  $\leftarrow$  LoadCSV(file)
- 4: Loop through Rows in DataFrame:  
for each sample i in df do:  
data  $\leftarrow$  Extract 100 feature values from row i
- 5: Preprocess Data:  
Convert all feature values to numeric format # (Features were numerically encoded during Phase 1 feature extraction and preprocessing.)
- 6: Reshape Data:  
image\_data  $\leftarrow$  Reshape data into a (10 $\times$ 10) matrix
- 7: Generate Image:  
if color\_mode == 'colorful' then  
display image\_data using 'rainbow' colormap  
else  
display image\_data using 'gray' colormap
- 8: Assign Save Directory:  
if sample label == ransomware then  
save\_directory  $\leftarrow$  'R'  
else  
save\_directory  $\leftarrow$  'G'
- 9: Save Image:  
Save image\_data as PNG file into save\_directory with filename i.png
- 10: Manage Memory:  
Close figure to release resources
- 11: end function

---

**Algorithm 2.** Generation of 2D image representations from structured CSV data.

---

#### *Execution and monitoring*

Each sample was executed within a cuckoo sandbox environment<sup>46</sup> which offers interactive real-time monitoring, a user-friendly interface, and captures detailed behavioral data in real time. These data include various system activities such as process creation, file modifications, network communications, and other actions indicative of ransomware behavior.

#### *Feature extraction and preprocessing*

Cuckoo Sandbox generated detailed behavioral reports in JSON format for each ransomware and benign sample, encompassing both static and dynamic analysis data. To enable structured downstream analysis, we implemented a domain-specific transformation pipeline that converted these hierarchical JSON files into structured data suitable for machine learning. This involved the following steps:

1. **Feature extraction:** Initially, we extracted 120 dynamic behavioral features spanning system behavior categories, including process activities (e.g., process creation and injection), network traffic (e.g., DNS and HTTP requests), registry and file system modifications, API usage, mutexes, and anti-VM techniques. We then selected the 100 most informative features based on domain-informed filtering, prioritizing features that are highly indicative of ransomware activities (e.g., file encryption, registry key modification, and back-up deletion). Features were chosen for their low sparsity, appearing consistently across diverse ransomware samples. Features like `file_created` (indicating new file creation), `regkey_written` (modifications to the Windows registry), and `dns_requests` (outbound domain name lookups) were observed across multiple ransomware families and retained for their consistent presence. Additionally, the selected features were cross-referenced with MITRE ATT&CK tactics and validated against findings from recent cybersecurity threat reports from organizations such as Sophos, CrowdStrike, and Trend Micro (2019–2024). This curated feature set ensures both technical depth and practical relevance, thereby improving the robustness and generalization of the detection model across diverse ransomware families. Preliminary experiments were conducted with 60, 80, and 120 features. However, 100 features provided an optimal trade-off between model performance and computational cost, achieving the highest accuracy and AUC without overfitting. The full list of the 100 extracted features, along with their behavioral descriptions and detection relevance, is provided in Supplementary Table S1.
2. **Preprocessing:** Following feature extraction, each ransomware and benign sample was processed through a structured pipeline such as normalization, to prepare the behavioral data for CNN-based classification. All extracted values were either originally numeric or transformed into numeric representations during preprocessing to ensure consistent formatting and compatibility with image-based modelling. The following preprocessing rules were applied:
  - I. Boolean-type features (e.g., true/false) were binary encoded:

Behavioral indicators such as privilege escalation attempt, shadow copy deletion, screen capture activity, and sandbox evasion behavior were processed as Boolean fields.

If the ransomware exhibited this behavior, the feature was encoded as 1 (true).

If a behavior was absent, the feature was encoded as 0 (false).

- II. Categorical string fields (e.g., IP origin and country names) were quantified as follows:

String length or frequency of occurrence within the dataset.

- III. List-type features (e.g., dropped files, DNS request entries, and VirusTotal positives) were represented by count values indicate the number of elements in the list.

- IV. Missing or null values were treated as follows:

0 (zero), ensuring a uniform numerical structure across all samples.

After preprocessing, each sample was represented as a row in a CSV file with 100 numerical features, with fixed column ordering to ensure uniform mapping across samples and saved in structured format. To automate feature extraction and preprocessing, we implemented a Python script 'Preprocess\_JSON\_to\_CSV', available in our public GitHub repository<sup>47</sup>. This script extracts and transforms the raw behavioral data into a tabular (CSV) format. The core logic is outlined in Algorithm 1: Extraction and Structuring of dynamic behavioral features from JSON reports.

#### **Phase 2: visualization phase (Dynamic feature vector-to-image transformation)**

This phase transforms the structured numeric features into image representations to enable CNN-based classification. Each sample was initially represented as a 1D numeric vector with 100 features. The visualization process consists of the following three key steps:

1. **2D reshaping (grid formation):**
  - To ensure compatibility with CNN architectures, 1D feature vectors were reshaped into 2D square matrices. The final model used 100 features reshaped into a 10 × 10 grid without padding, as 100 is a perfect

square. For exploratory trials involving 120 features, zero padding was applied to make it 121 and turned it into an  $11 \times 11$  matrix. These were tested with the same CNN models as the main tests but were discarded owing to poor performance and misaligned features. No resizing or interpolation was used between formats. The  $10 \times 10$  configuration was retained for its high accuracy, spatial coherence, and suitability for CNNs, effectively simulating an image in which each cell represents one feature.

## 2. Pixel mapping and color encoding:

- The reshaped matrix is normalized and converted into an image, where the numeric value of each feature is mapped to the pixel intensity.
- Two visualization modes are supported:
  - I. Grayscale: Feature values were mapped to brightness values of 0–255.
  - II. Color: A perceptual color map (e.g., “rainbow”) was applied to enhance the contrast between high and low intensity areas.

## 3. Image saving and directory naming:

- The resulting image was saved as a.PNG file in either the ‘R’ (ransomware) or ‘G’ (benign) folder, depending on its label.
- These folders are later used by TensorFlow’s image loading function, which automatically assigns class labels based on directory names.

By transforming behavioral features into visual grids, pretrained CNNs can effectively learn discriminative patterns that represent ransomware activity. This approach exploits spatial regularities in structured behavioral data to improve detection accuracy. To automate this visualization process, we developed a Python script named ‘CSV-to-image conversion,’ available in our public GitHub repository<sup>47</sup>. This script converts the structured dynamic features into 2D image representations. The core logic is outlined in Algorithm 2: Generation of 2D image representations from structured CSV data.

### Phase 3: detection phase

The detection phase is the final stage, in which the processed dataset is fed into various machine learning models to evaluate its accuracy in identifying ransomware from benign samples. We employed both pretrained and non-pretrained machine learning models to classify image-based representations of ransomware and benign samples via a small custom dataset. Owing to the challenges associated with limited data, transfer learning with pretrained models was chosen as the primary method for this task. To further validate the effectiveness of transfer learning, non-pretrained models were employed as baseline comparisons to assess how the models trained from scratch performed relative to the proposed pretrained models. This phase encompasses several critical steps: loading the input dataset, model selection, fine-tuning the hyperparameters, and classification.

#### *Loading input dataset*

In this step, the model was loaded with the preprocessed dataset and split into 80% training and 20% validation sets, comprising ransomware and benign samples depicted as grayscale or color-mapped images based on dynamic behavioral features. This study used an 80–20 training–validation split and leveraged TensorFlow image preprocessing tools for efficient data handling. A balanced 50:50 ratio of ransomware to benign samples is achieved through random sampling and shuffling. The sampling selects a balanced subset from the entire dataset. By contrast, shuffling rearranges the data points to ensure randomness before feeding them into the model. This process is essential for preventing inherent data patterns from affecting the model during training.

#### *Model selection*

The detection phase incorporates both non-pretrained and pretrained models to evaluate their respective classification efficacies on the given dataset.

### Pretrained models with transfer learning

Training deep neural networks typically requires substantial time and computational power. Consequently, the primary objective of utilizing transfer learning is to leverage the knowledge acquired from one problem domain and apply it to another, which can lead to a reduction in training time and hardware requirements<sup>48</sup>. It can be implemented by either employing pretrained models as feature extractors for target domain classification or fine-tuning models to develop new learning parameters for the target domain<sup>15</sup>. This study leveraged transfer learning by fine-tuning six pretrained CNNc1 models to achieve high ransomware detection accuracy with limited data. A brief description of these models is provided below:

**ResNet50:** ResNet stands for the residual network and is a deep CNN model. The ResNet architecture incorporates a series of residual blocks with skip connections to address the phenomenon of diminishing accuracy that occurs when the number of layers increases<sup>49</sup>. These skip connections allowed the preservation of information from earlier layers, which helped the network learn better representations of the input data. The ResNet50 CNN model includes a 50-layer residual network. This neural network architecture demonstrated exceptional performance in image classification tasks. Consequently, the ResNet50 model is frequently selected for developing transfer learning models in related applications<sup>32</sup>.

**EfficientNetB0:** EfficientNetB0 serves as the foundational model within the EfficientNet family, which employs a compound scaling method to proportionally increase the dimensions of convolutional neural networks

(CNNs) across depth, width, and resolution. This method yields efficient and effective models, achieving high accuracy with fewer parameters<sup>50</sup>.

**Xception:** Xception stands for “Extreme Inception” and is a deep convolutional neural network architecture that relies entirely on depth wise separable convolution layers, enabling easy definition and modification<sup>51</sup>.

**VGG16 and VGG19:** The VGG model architecture includes 16 (or 19) layers organized into five convolutional neural network (CNN) blocks and three fully connected layers. Each convolutional layer employed a kernel size of  $3 \times 3$ , with padding and a stride of 1. Additionally, each block of the VGG model incorporates a max pooling layer to further reduce the spatial dimensions. The max pooling layer uses a kernel size of  $2 \times 2$  with additional padding, and a stride of two is implemented to halve the dimensions compared with the preceding block of the model. The VGG model culminates in a fully connected dense layer preceding the softmax layer, which serves as the classification and prediction component<sup>52</sup>.

**InceptionV3:** InceptionV3, a 48-layer deep convolutional neural network (CNN) architecture, excels in image-processing tasks within computer vision. It reportedly outperforms other deep CNN models such as VGG-Net, GoogleNet, PreLU, and BN-Inception. The literature shows that InceptionV3 attains high performance with relatively shallow convolutional layers using symmetric and asymmetric components such as convolution, pooling, dropout, and dense layers<sup>53,54</sup>.

We selected each pretrained model based on the following key criteria: (i) architectural diversity (e.g., residual, depth-wise separable, multi-scale feature extraction capability), (ii) effectiveness on small datasets, (iii) computational feasibility in terms of training time and resource efficiency, and (iv) proven success in malware detection literature, as summarized in Table 3. This model selection ensures consistency, reproducibility, and fair performance comparisons while maintaining practical feasibility for image-based ransomware detection.

### Non-pretrained models (performance benchmark)

We employed some non-pretrained models (CNN, ANN, CNN-LSTM, and CNN-GRU) as baseline comparisons to show how the models trained from scratch performed on a small dataset compared to the pretrained models with transfer learning.

**Convolutional neural networks (CNNs):** CNNs are deep learning models for structured grid-like data like images. They use convolutional layers to learn spatial hierarchies via filters that detect edges, textures, shapes, and patterns.

CNNs excel in image classification, object detection, and visual data tasks by capturing local and global patterns with shared weights, thereby increasing computational efficiency<sup>58</sup>.

**Artificial Neural Network (ANN):** ANNs are networks of interconnected nodes inspired by biological neural networks consisting of an input layer, one or more hidden layers, and an output layer. Each “neuron” connects to the neurons in the next layer, adjusting weights to minimize the prediction error. ANNs are applied in tasks such as regression, classification, and time series prediction; however, they generally excel with simple, structured data owing to their fully connected nature and lack of spatial awareness<sup>58</sup>.

**CNN-LSTM:** The CNN-LSTM model leverages CNNs for spatial pattern processing and LSTMs for learning temporal dependencies, making it ideal for sequential and spatiotemporal data. This model is particularly effective for tasks such as video classification, action recognition, and sequence-based anomaly detection, in which both spatial and temporal patterns are crucial<sup>14</sup>.

**CNN-GRU:** This integrates a CNN with a GRU, a computationally efficient recurrent neural network that uses fewer parameters than LSTMs. The CNN layers extract spatial features, which are then processed by the GRU layers to capture sequential dependencies. This combination is ideal for tasks such as speech recognition and time series prediction, where speed and efficiency are crucial because of the large datasets or real-time demands<sup>58</sup>.

### Fine tuning to layers and hyperparameters

Fine-tuning in transfer learning modifies the pretrained layers and optimizes the hyperparameter settings to enhance the system performance. We fine-tuned each CNN-based pretrained model (discussed in the model selection section) for ransomware detection in both the color and grayscale datasets. Table 4 summarizes the key hyperparameter settings used in our experiments, including the optimizer type, learning rate, batch size,

Model name	Rationale behind the model selection	Reference
ResNet50	ResNet50's efficient feature extraction capability and relatively lighter computational requirements make it highly suitable for ransomware detection tasks. Its residual learning capabilities allow it to adapt effectively to the subtle differences between ransomware and benign samples, even when training on smaller datasets.	15,16,18,31,32,49
InceptionV3	InceptionV3 captures multiscale features through various filter sizes, enabling it to extract fine-grained feature from smaller images.	13,15–18,53
EfficientNetB0	EfficientNetB0 maintains high performance due to its architecture optimized for efficiency, making it well-suited for small datasets with fewer data points.	12,14,55
Xception	Xception uses depth wise separable convolutions to enhance learning efficiency by effectively extracting features from small images.	16,18,51,56
VGG16 and VGG19	VGG16 and VGG19, though older architectures, were included as strong baselines due to their simplicity, widespread use, and consistent performance across various small-data scenarios. Their sequential convolutional layers allow for straightforward feature extraction, and they tend to generalize well. While they may be more parameter-heavy than newer models, they offer stable performance and interpretability, which is particularly valuable in research settings where reproducibility and comparative benchmarking are important. Their inclusion helps establish a reference point for evaluating more recent architectures in our ransomware detection context.	15,16,31,52,57,15,16,18

**Table 3.** Reasons for choosing the pretrained models used in our experiment.

Hyperparameter	Value
Include top	False
Input shape	(389, 389, 3)
Weights	imagenet
Pooling	Max
Optimizer	Adam
Learning rate	0.001
Loss function	sparse_categorical_crossentropy
Epochs	25
Batch size	32
Class mode	Binary
Monitor (Early stopping)	Accuracy
Min LR (Learning rate decay)	1e-6
Patience (Early stopping)	5
Patience (Reduce LR)	3
Additional dense layer units	1024 (ResNet50, EfficientNetB0, Xception, InceptionV3)
Classifier activation	Softmax

**Table 4.** Hyperparameters for pretrained models.

---

Input: Dataset `data_path` in compressed format, target image dimensions (`img_height`, `img_width`), `batch_size`, and `epochs`

Output: Trained Model and evaluation metrics

1: function TransferLearning (`data_path`, `img_height`, `img_width`, `batch_size`, `epochs`)

2: Data Preparation

`data_dir` ← Extract and load dataset from `data_path`

Split into training and validation sets (`train_ds`, `val_ds`)

3: Model Initialization

`model` ← Load Pretrained CNN (e.g., ResNet50, EfficientNetB0)

Add custom layers: Dense layer(s) for feature learning and output layer with softmax for classification.

4: Compile Model

Choose optimizer (Adam), loss (based on label type), and accuracy metric.

5: Training with Callbacks

`history` ← Train model on `train_ds` with validation on `val_ds`

Apply early stopping and learning rate adjustment for optimal performance

6: Evaluate Model

`y_true` ← Ground truth labels from `val_ds`

`y_pred`, `y_prob` ← Predict labels and probabilities from Model

7: Compute Metrics

Calculate precision, recall, F1-score, accuracy, and AUC

8: Plot Training Curves

Visualize accuracy and loss over epochs for training sets

9: End Function

---

**Algorithm 3.** Generation of 2D image representations from structured CSV data.

loss function, and early stopping criteria, which were consistent across all models to ensure fair comparison and reproducibility.

All the models were initialized with ImageNet weights and trained on images of our dataset with a size of  $389 \times 389$  pixels. We removed (including `_top=False`) the top classification layers with 1,000 outputs initially designed for the ImageNet dataset. We replaced them with fully connected (FC) layers designed explicitly for ransomware detection, allowing the models to effectively classify ransomware and benign samples. Max pooling has been used as the pooling method to capture dominant features. We added additional dense layers to ResNet50, EfficientNetB0, Xception, and InceptionV3. The addition of dense layers to these complex feature extraction models allows the network to capture task-specific patterns by combining and refining extracted features.

In contrast, we retained the pretrained VGG16 and VGG19 models without extra dense layers to maintain computational efficiency and avoid overfitting, particularly on smaller datasets, because these architectures

are already dense and computationally intensive<sup>52</sup>. The Softmax activation function in the classifier guarantees probabilistic outputs for the binary ransomware versus the benign task. The models were optimized using the Adam optimizer, which is a first-order gradient-based algorithm with an initial learning rate of 0.001 and sparse categorical cross-entropy loss function. Sparse categorical cross-entropy was used in this study because our model is designed with two output neurons (one for “benign” and one for “ransomware”) via a Softmax activation function, effectively treating binary classification as a two-class multiclass problem. This approach aligns with the use of sparse categorical cross-entropy, as it enables integer-encoded labels (0 for “benign” and 1 for “ransomware”) without requiring one-hot encoding. This aligns with Keras guidelines<sup>59</sup> and recent research<sup>17,60,61</sup>, ensuring both efficiency and methodological validity. A batch size of 32 was used to improve computational efficiency and prevent unstable gradient updates.

To mitigate overfitting, we used early stopping and learning rate reduction. Overfitting occurs when the model memorizes training data too well and performs poorly on new data. Early stopping prevents this by monitoring model performance on validation data not used for training. It is a technique used in machine learning to stop training a model when it no longer improves its performance on validation data for a specified number of epochs, thereby preventing it from memorizing the training set. For example, if patience is set to five, the model stops training if there is no improvement in the monitored metric for five consecutive epochs. We also used callback (ReduceLROnPlateau) to decrease the learning rate by 0.2 if the validation loss did not improve over three epochs, with a minimum learning rate threshold of 1e-6.

The selected hyperparameters (e.g., learning rate = 0.001, batch size = 32) were determined based on standard practices for training deep CNNs on small datasets, as well as prior studies on ransomware and malware detection<sup>13–16</sup>. Additionally, we performed preliminary tuning experiments on the validation set to confirm that these values offer stable convergence, good generalization, and efficient training performance.

The classification performance was evaluated using metrics such as the accuracy, precision, recall, F1 score, and AUC. Algorithm 3 provides a pseudocode for training fine-tuned pretrained models, detailing the transfer learning process for CNN models in ransomware detection. The algorithm involves the following key steps:

- The image datasets were loaded and preprocessed into training and validation sets.
- Initialize a pretrained model with custom classification layers.
- The model was then combined with suitable loss functions and metrics.
- The model was trained via early stopping and learning rate scheduling.
- The performance of the model was evaluated using precision, recall, F1 score, accuracy, and AUC metrics.

This method ensures efficient training on small datasets while leveraging the robust feature extraction capabilities of pretrained CNNs.

## Experiment and results

Our investigation aimed to assess the efficiency and robustness of the transfer learning approach using visual dynamic features to defend against ransomware attacks with limited datasets. This section consists of three components: experimental setup, evaluation metrics, and experimental results and discussion.

### Experimental setup

The experimental setup of this study incorporates sophisticated tools and techniques to analyze ransomware using dynamic analysis techniques. The analysis environment used a cuckoo sandbox<sup>46</sup> to execute and monitor diverse ransomware behaviors securely. The evaluation environment leveraged Google Colab Pro<sup>62</sup> with 51 GB of RAM, offering robust cloud computing to manage resource-intensive experiments and model training. Python version 3.10.12 was used to take advantage of the latest features, with essential libraries such as TensorFlow<sup>63</sup> and Keras<sup>64</sup> to implement and train deep learning models, as well as Pandas<sup>65</sup> and NumPy<sup>66</sup> for data management and numerical computations. Preprocessing involves scripting workflows in Python and using Bash commands to analyze the input/output operations of the JSON reports and store the resulting data in a CSV file. Additionally, color-mapping techniques were employed on structured CSV data via a Python script to visually represent the behavioral patterns of ransomware and benign applications.

### Dataset splitting and evaluation strategy

The final dataset comprises 1,000 image samples derived from structured behavioral features, with an equal distribution of 500 ransomware and 500 benign samples. To ensure robust evaluation, we applied an 80/20 stratified split, resulting in 800 samples used for training and 200 for validation. The validation set was used to monitor the performance of the model during training, guide early stopping, support learning rate adjustment, and not for final testing. Although no separate test set was used owing to the small dataset size, we ensured performance generalization by-

- Stratified random sample splitting was used to preserve the class balance (50% ransomware and 50% benign) in both subsets.
- Applying early stopping and ReduceLROnPlateau callbacks based on validation metrics to avoid overfitting.
- Multiple runs were performed, and the results were compared across different models and data configurations (color vs. grayscale) for consistency.

This setup allowed us to optimize training while mitigating overfitting risks, ensuring that the results reflected the true model performance under controlled conditions.

To support the misclassification and interpretability analysis, we conducted additional evaluations of the trained model using the 200-sample validation set. These evaluations were used to generate confusion matrices and assess the generalization capabilities of the model while maintaining controlled training-validation separation.

### Evaluation metrics

To analyze and evaluate the results, we employed the precision, recall, F1 score, area under the curve (AUC), accuracy, and loss curve as performance metrics to assess the models for ransomware classification. However, accuracy was considered as the primary metric. The following equations (Eqs. (1)–(4)) describe the mathematical formulas for these metrics:

Before proceeding with the equations, we define the acronyms of the confusion matrices used in the formulas:

- TP: True positive (correctly identified ransomware samples).
- TN: true negative (correctly identified benign samples).
- FP: False positive (benign samples incorrectly labelled as ransomware).
- FN: False negative (ransomware samples incorrectly labelled benign).

**Accuracy:** Accuracy is a metric that calculates the ratio of correct predictions (including both true positives and true negatives) to the total number of observations in a dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

**Precision:** This measures the proportion of correctly predicted positive observations to the total number of predicted positive observations.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

**Recall:** This is also termed the sensitivity or true-positive rate, which measures the percentage of positive instances accurately classified and is calculated using the following formula:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

**F1 score:** F1 score combines precision and recall into a single metric, offering a balanced evaluation of false positives and false negatives.

$$F1 - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

**Area under the curve (AUC):** The AUC metric evaluates the performance of a binary classifier by plotting a receiver operating characteristic (ROC) curve that shows a trade-off between true positives and false negatives at different thresholds. High AUC values indicate effective classification with minimal error.

**Loss:** Loss measures the error between the predicted and actual labels during training, with lower values indicating better model performance.

### Results analysis

We performed a series of experiments based on the following: (1) experiments with pretrained models and (2) experiments with nontrained models. (3) Performance comparison between grayscale and color image datasets. (4) Comparison of robustness and computational efficiency (5) Model interpretability and visual analysis (6) Misclassification analysis and (7) Comparison with relevant existing studies.

#### *Experiment with pretrained models based on color and gray image datasets*

We trained six pretrained models with varying configurations, as listed in Table 4. The performance metrics (accuracy, precision, recall, F1-score, AUC, and loss) for the ransomware classification results are summarized in Tables 5 and 6 for both color and grayscale samples. The primary evaluation metric was accuracy, which was tracked across all the experiments to measure the effectiveness of each model.

ResNet50 emerged as the top-performing model, achieving 99.96% accuracy on the color dataset and 99.91% on the grayscale dataset. This 99.96% represents the highest training accuracy observed during repeated experiments (epoch 24). To further illustrate model behavior, a confusion matrix was generated from the validation set comprising 200 samples corresponding to the same training run. While training accuracy highlights the model's optimal learning performance, the confusion matrix provides practical insight into its generalization and classification behavior on held-out data used during early stopping, not final testing. The consistently high precision, recall, F1-score, and AUC confirm ResNet50's superiority.

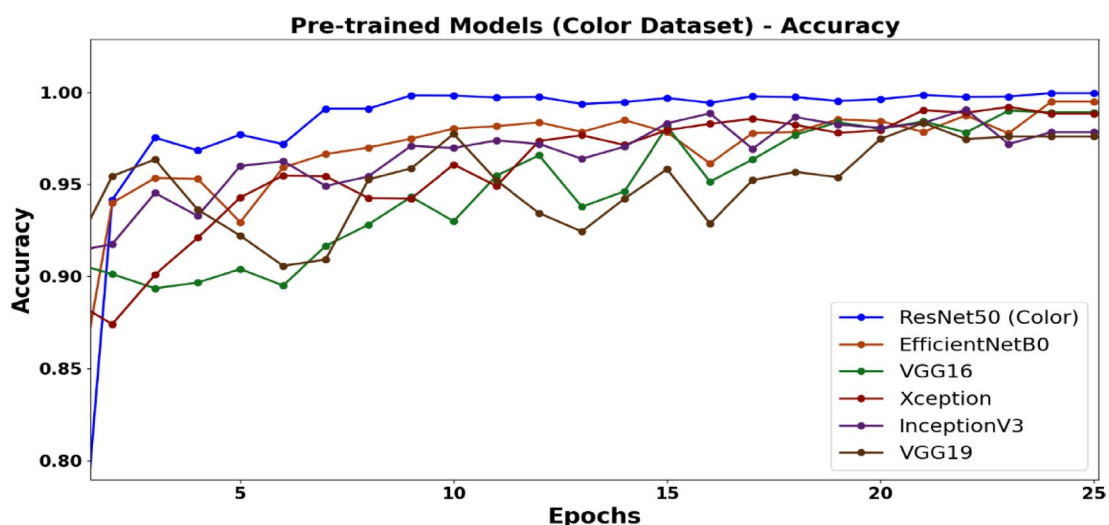
EfficientNetB0 followed closely with an accuracy of 99.91% on the color dataset and 99.70% on the grayscale dataset, thereby showing strong generalizability. Xception and InceptionV3 also demonstrate high effectiveness but with slightly lower scores, indicating their robustness in handling ransomware detection tasks. Conversely, VGG16 and VGG19 exhibited relatively low performance metrics, particularly when transitioning from the color to grayscale datasets.

Model	Accuracy	Precision	Recall	F1-score	AUC score	Loss
ResNet50	99.96	99.9591	99.96	99.959	99.8758	0.0026
EfficientNetB0	99.78	99.78	99.78	99.78	99.7725	0.0069
VGG16	98.91	98.9093	98.91	98.9096	98.8759	0.0253
Xception	98.84	98.847	98.84	98.8418	98.8349	0.0579
InceptionV3	97.84	97.84	97.84	97.84	97.9207	0.0602
VGG19	97.47	97.4701	97.47	97.4705	97.3692	0.0874

**Table 5.** Performance analysis results of the examined pretrained CNN models on the color image dataset.

Model	Accuracy	Precision	Recall	F1-score	AUC score	Loss
ResNet50	99.91	99.91	99.91	99.91	99.8492	0.0059
EfficientNetB0	99.70	99.6987	99.70	99.6993	99.61	0.0167
VGG16	98.12	98.12	98.12	98.12	98.0213	0.0471
Xception	98.04	98.034	98.04	98.0375	98.9344	0.1261
InceptionV3	97.60	97.5986	97.60	97.5992	97.6078	0.1179
VGG19	97.40	97.40	97.40	97.40	97.40	0.1194

**Table 6.** Performance analysis results of the examined pretrained CNN models on a Gray image dataset.

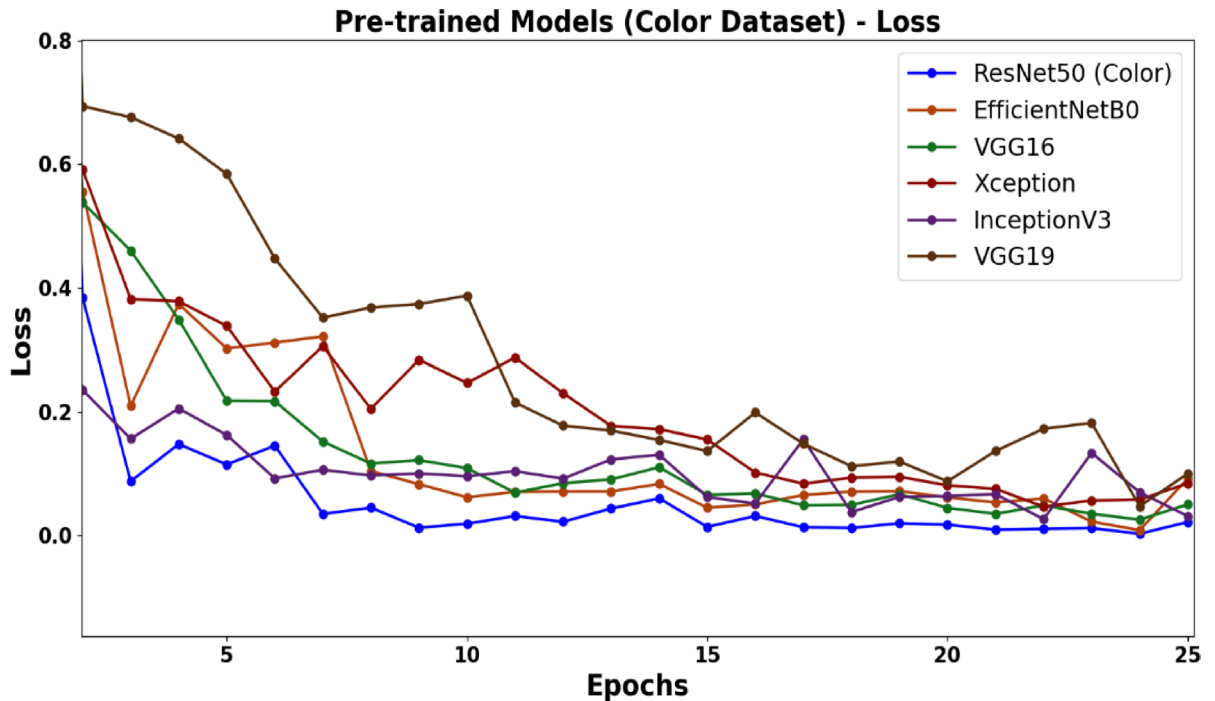


**Fig. 3.** Accuracy trends for various pretrained learning models. The ResNet50 model demonstrates superior accuracy as the number of epochs increases.

Based on the results from both datasets the color dataset consistently outperformed grayscale, with all models achieving higher accuracy on color samples. Consequently, color images were prioritized for subsequent analysis. Figures 3 and 4 further underscore ResNet50's superiority, showing its high accuracy trajectory and the lowest loss trend compared to other models. This advantage is attributed to its deeper architecture and advanced feature extraction capabilities, enabling it to capture complex behavioral patterns effectively. In contrast, the shallower VGG models exhibited limitations in handling feature-rich ransomware data.

#### Experiment with non-pretrained models based on color and gray image datasets

The performance of the non-pretrained models was evaluated on both color and gray image datasets, as shown in Tables 7 and 8. The results revealed that the integrated CNN-GRU consistently outperformed other non-pretrained models across both datasets. For color images, CNN-GRU achieved the highest accuracy of 97.49% with a corresponding loss of 0.0579. Similarly, for gray images, an accuracy of 97.07% was achieved, with a loss of 0.0680. CNN-LSTM closely followed, demonstrating a slightly lower accuracy (96.94% for color and 96.80% for gray images) and greater loss than CNN-GRU. The standalone CNN model exhibited a performance similar to that of CNN-LSTM, achieving accuracies of 96.84% for color and 96.71% for gray images but with marginally higher loss values (0.0998 and 0.1045, respectively). The ANN consistently underperformed, showing the lowest



**Fig. 4.** The loss trends of pretrained models show that reveals that the ResNet50 model consistently demonstrates the lowest loss as the number of epochs increases.

Model	Accuracy	Precision	Recall	F1 Score	AUC	Loss
CNN-GRU	97.49	97.4964	97.49	97.4919	97.4676	0.0579
CNN-LSTM	96.94	96.9464	96.94	96.9419	96.8978	0.0963
CNN	96.84	96.799	96.84	96.7838	96.8165	0.0998
ANN	90.48	90.48	90.48	90.48	92.67	0.2707

**Table 7.** Comparison of accuracy with that of the nontrained model for color images.

Model	Accuracy	Precision	Recall	F1 Score	AUC	Loss
CNN-GRU	97.07	97.0685	97.07	97.0692	97.1085	0.0680
CNN-LSTM	96.80	96.80	96.80	96.80	96.6958	0.0807
CNN	96.71	96.7107	96.71	96.7103	96.7168	0.1045
ANN	89.24	89.24	89.24	89.24	89.1613	0.2853

**Table 8.** Comparison of accuracy with that of the nontrained model for Gray images.

accuracy (90.48% for color and 89.24% for gray images) and the highest loss values (0.2707 for color and 0.2853 for gray images).

The associated graphs in Fig. 5 further clarify the analysis of the performance of these models across various image types. The graph comprises two plots that evaluate the performance of various non-pretrained models over 15 epochs in terms of accuracy and loss. ResNet50 was included as a reference pretrained model, and it displayed superior and stable performance across all epochs, quickly converging to near-perfect accuracy and minimal loss. This demonstrates the advantages of transfer learning. Among the non-pretrained models, the CNN-GRU achieved the highest accuracy and lowest final loss, highlighting its strength in capturing spatial and sequential features. These results highlight the superiority of the pretrained and hybrid models for image-based ransomware classification.

The ANN demonstrated a limited learning capacity with lower accuracy and inconsistent loss trends. ANNs lack the ability to capture spatial and temporal dependencies, which likely accounts for their poor performances. CNN-LSTM demonstrated a similar structure to CNN-GRU but with less favorable results, possibly because of the relatively heavier computational burden of LSTM for this task.

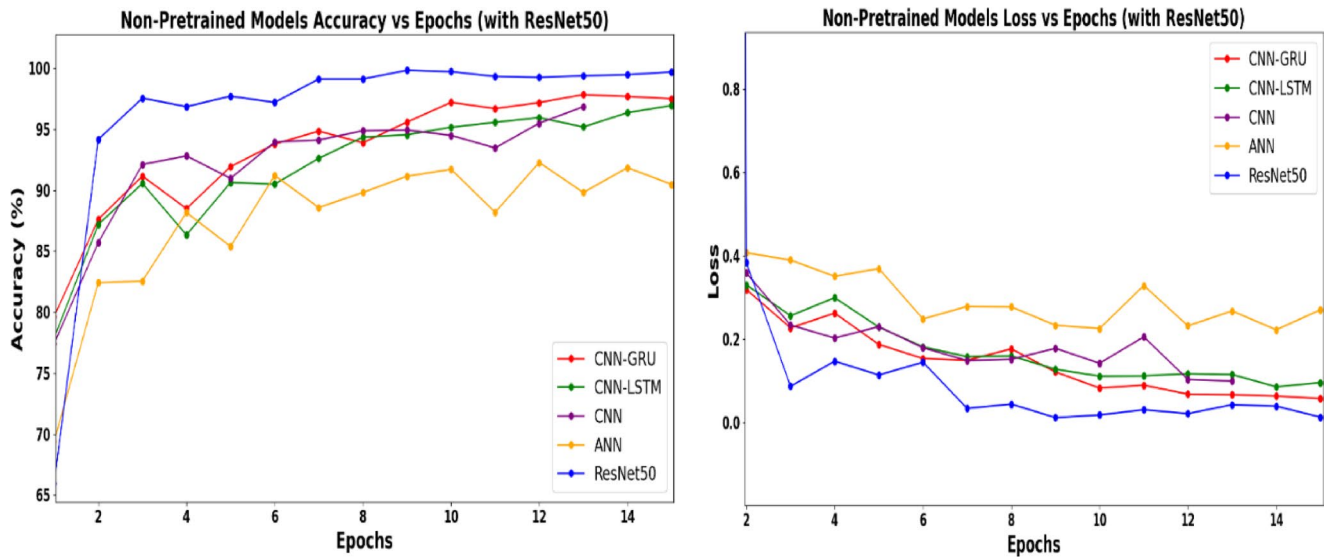


Fig. 5. Comparative performance of non-pretrained models and ResNet50 over 15 epochs.

Model	Accuracy (Color image)	Accuracy (Gray image)	Difference (%)	Loss (Color dataset)	Loss (Grayscale dataset)	Difference (%)
ResNet50	99.96	99.91	0.05	0.0026	0.0059	0.0033
EfficientNetB0	99.78	99.70	0.08	0.0069	0.0167	0.0098
Xception	98.91	98.12	0.79	0.0253	0.0471	0.0218
InceptionV3	98.84	98.04	0.80	0.0579	0.1261	0.0682
VGG16	97.84	97.60	0.24	0.0602	0.1179	0.0577
VGG19	97.47	97.40	0.07	0.0874	0.1194	0.0320
CNN-GRU	97.49	97.07	0.42	0.0579	0.0680	0.0101
CNN-LSTM	96.94	96.80	0.14	0.0963	0.0807	-0.0156
CNN	96.84	96.71	0.13	0.0998	0.1045	0.0047
ANN	90.48	89.24	1.24	0.2707	0.2853	0.0146

Table 9. Accuracy and loss or error rate comparisons between color and Gray image datasets.

*Performance comparison between color and grayscale datasets*

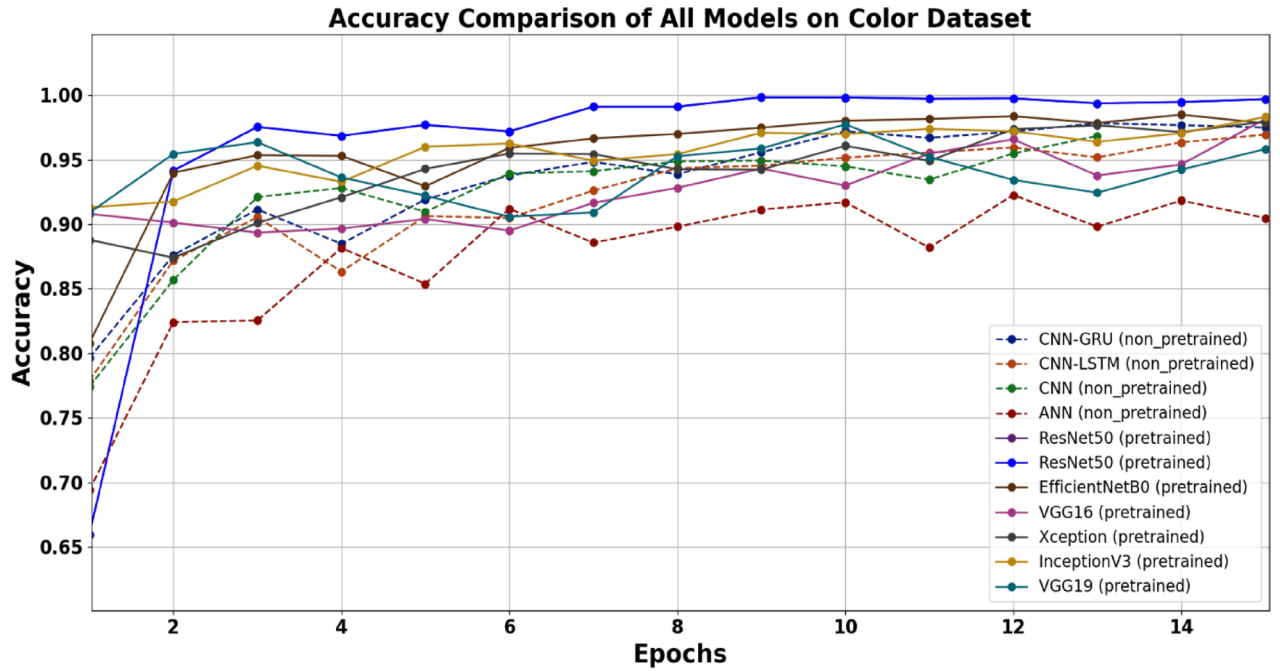
The comparative results in Table 9; Fig. 6 reveal that all the models (pretrained and nontrained) performed slightly better on the color datasets than on the grayscale datasets in terms of accuracy and loss drop.

However, the performance differences were negligible for most models.

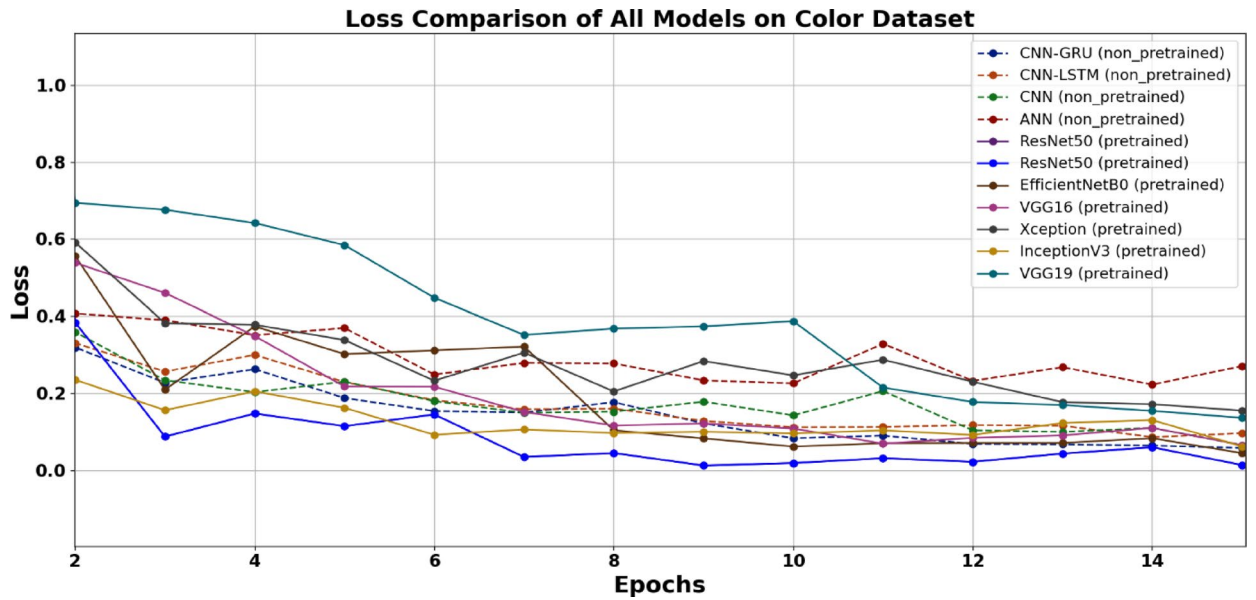
- ResNet50 achieved the highest accuracy for both datasets, with only a minor difference of 0.05%, and maintained the lowest loss gap (0.0033%), demonstrating its superior performance across both datasets and among all the models. EfficientNetB0 also demonstrated strong results, with a slight decrease in accuracy (0.12%) and a modest increase in loss (0.0114) for gray images.
- Among the non-pretrained models, the CNN-GRU outperforms the CNN-LSTM and CNN, with a relatively small accuracy gap (0.42%) and minimal loss difference (0.0101), whereas the ANN showed the largest performance decline, with a 1.24% decrease in accuracy and a 0.0146 increase in loss for gray images.
- Notably, CNN-LSTM is the only model with a marginally lower loss for gray images, indicating its unique adaptability to grayscale inputs.
- Overall, the pretrained models outperformed the nontrained models, with color datasets consistently yielding better results, as illustrated in Figs. 6 and 7.

Table 9 suggests that although color datasets provide a slight edge in performance, grayscale datasets remain highly effective for ransomware detection, making them a viable option when color data are unavailable or when computational efficiency is prioritized.

Color datasets inherently provide more information through pixel variations, which may help models distinguish between ransomware and benign behaviors. However, the negligible drop in performance across most models when grayscale datasets are used suggests that the essential features for ransomware detection can still be effectively captured with fewer color-specific details.



**Fig. 6.** Graph of the accuracy against the epoch across all models (pretrained and non-pretrained). The consistently lowest-performing model is ANN, whereas RsNet50 consistently ranks highest.



**Fig. 7.** Graph of the loss against the epoch across all models (pretrained and non-pretrained). The consistently lowest loss performing model is ResNet50, whereas ANN shows the highest loss.

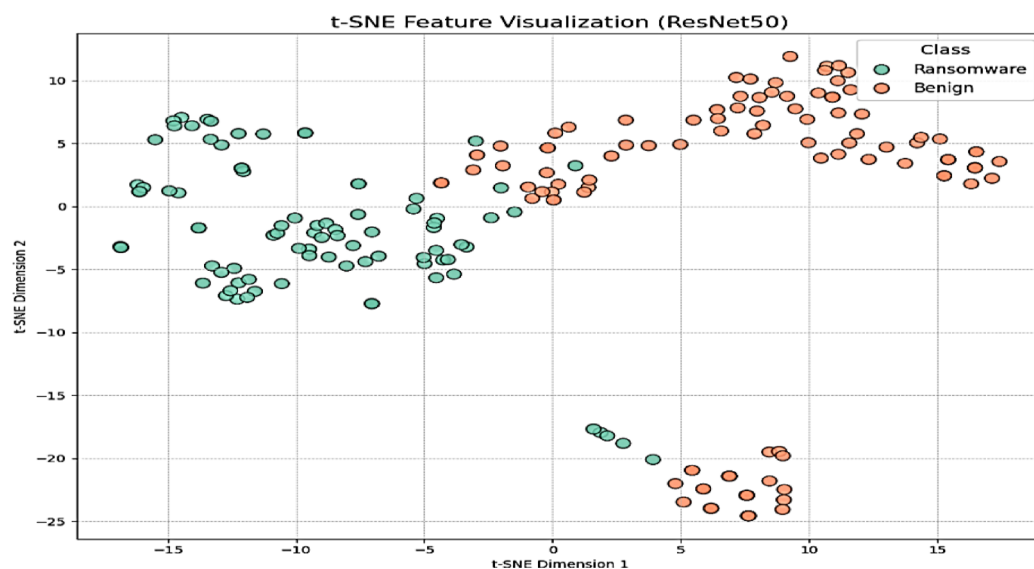
*Robustness and computational efficiency*

To assess both detection performance and computational demands, we compared six pretrained CNN models (ResNet50, EfficientNetB0, InceptionV3, Xception, VGG16, and VGG19) with and without fine-tuning. The results, summarized in Table 10, show that fine-tuning generally improved classification accuracy across most models while also reducing training time per sample.

Notably, ResNet50 achieved the highest accuracy improvement from 99.91 to 99.96%, with a reduced training time of 0.440 s/sample (down from 0.571). EfficientNetB0 also demonstrated strong performance with lower training cost. By contrast, models such as VGG16 and VGG19 required more time and showed limited gains, indicating their inefficiency for small datasets.

Model	Without fine-tuning		With fine-tuning (see Table 4)	
	Accuracy (%)	Avg. training time per sample (s)	Accuracy (%)	Avg. training time per sample (s)
ResNet50	99.91	0.571	99.96	0.440
EfficientNetB0	99.26	0.208	99.78	0.193
InceptionV3	98.77	0.380	97.84	0.098
Xception	99.01	0.549	98.90	0.262
VGG16	98.20	0.932	97.26	0.860
VGG19	97.47	1.460	98.00	1.444

**Table 10.** Training time efficiency for pretrained models.



**Fig. 8.** t-SNE feature visualization of used ransomware dataset. Orange points represent benign samples and green points represent ransomware. The two classes formed distinct clusters with limited overlap, illustrating strong feature separability and explaining the near-perfect classification accuracy of the model.

These results highlight that transfer learning with fine-tuning offers a balanced trade-off, achieving a high detection accuracy while maintaining computational efficiency. Among all the tested models, ResNet50 provided the best combination of speed and robustness for small ransomware datasets.

#### *Model interpretability and visual analysis*

To better understand the decision-making behavior of our deep learning models, we conducted a visual interpretability analysis using t-distributed stochastic neighbor embedding (t-SNE) and saliency maps. These methods help reveal how well the model separates ransomware and benign samples in the learned feature space and which regions of the input images most influence classification decisions. By employing these techniques, we aimed to provide deeper insights into the internal representations learned by pretrained CNNs and evaluate the consistency and reliability of the classification outcomes.

#### **t-SNE feature visualization**

To enhance the transparency of the model behavior, t-distributed stochastic neighbor embedding (t-SNE) was applied to the final dense layer outputs of the ResNet50 model. As shown in Fig. 8, the ransomware and benign samples are from two distinct clusters, with orange points indicating benign samples and green points representing ransomware.

Although some overlap was observed near the cluster boundary, the overall separation was clear, supporting the model's high classification accuracy of 99.96%. Rare misclassifications (one false negative) likely occurred in the overlapping region, where a ransomware sample displayed benign-like behavior.

This clustering pattern highlights the ability of the model to map behaviorally distinct classes into well-defined regions in the latent space, thereby reducing the risk of misclassification. Therefore, the rare misclassification observed is explainable and expected in boundary scenarios, further affirming the robustness and reliability of the proposed detection pipeline.

### Saliency map interpretation

To gain insight into the internal focus of the model during prediction, we applied saliency mapping using gradient-based backpropagation to the representative samples of both benign (B) and ransomware (R). This technique highlights the most influential pixel regions in the input images, allowing us to interpret which behavioral patterns the model attends to when classifying a sample as ransomware or benign.

As shown in Fig. 9, the saliency maps reveal distinct patterns of attention for the benign and ransomware inputs. For benign samples (left), the focus of the model is limited to a small, dispersed set of pixels, suggesting sparse activation. In contrast, the ransomware (right) saliency map shows intense, localized attention in specific regions, indicating that the model has learned to associate certain behavioral patterns (e.g., registry access and file encryption events) in the presence of the ransomware.

These visualizations demonstrate that the model does not respond to random noise but instead focuses on meaningful regions in the behavioral image. This further reinforces the reliability of the model and its high predictive accuracy of 99.96%.

#### Misclassification analysis

To further evaluate the classification performance of the best-performing model, ResNet50, we analyzed its misclassification behavior using a confusion matrix derived from the model weights saved at epoch 24, which corresponded to the highest training accuracy of 99.96%, and evaluated it on the 200-sample validation set (98.50%).

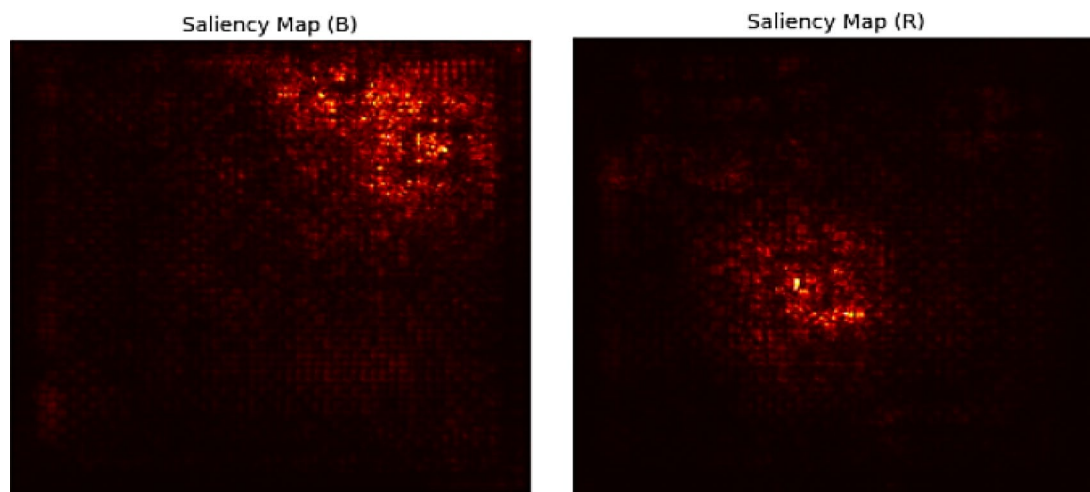
As shown in Fig. 10, the confusion matrix reveals three misclassifications: two false positives (FP) and one false negative (FN), resulting in a validation accuracy of 98.50%. This indicates that two benign samples were incorrectly predicted as ransomware and one ransomware sample was misclassified as benign. These raw counts show the performance of the model, correctly classifying 197 of the 200 validation samples. This matrix reflects the prediction performance of the model at convergence and is generated using the saved model weights after training.

Despite these minor errors, the results confirm the high classification accuracy of the model, with near-perfect identification of both benign and ransomware samples. These findings reinforce the effectiveness of the transfer-learning-based model, which maintained a consistently low misclassification rate. These rare errors are likely attributable to borderline cases either benign software exhibiting anomaly-like behavior or ransomware instances that mimic benign traits to evade detection. This interpretation is supported by the supplementary visualizations such as t-SNE and saliency map. The t-SNE plot revealed distinct clustering between benign and ransomware samples, with minimal overlap, indicating strong feature separability. Furthermore, saliency map analysis demonstrates that the model consistently focuses on behaviorally relevant regions, thereby reinforcing the reliability of its decision-making process.

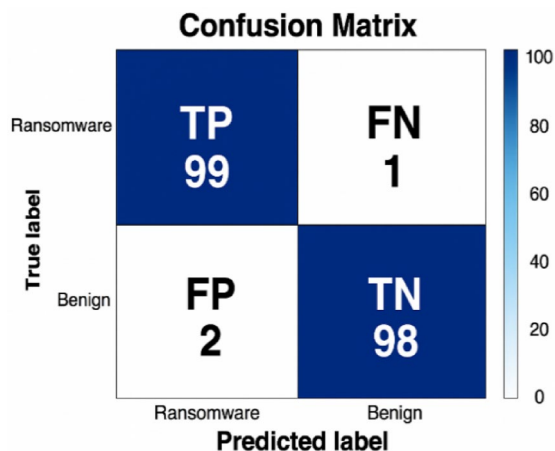
Overall, the rarity and interpretability of the misclassification patterns confirm the ResNet50-based model's robustness and generalization capability under transfer learning, even with unseen validation data.

#### Comparison of our proposed technique with the existing relevant methods

This section compares the performance of the proposed ransomware detection framework with those of several recent approaches, as summarized in Table 11. It is important to note that the studies listed in Table 11 utilized different datasets, feature engineering strategies (e.g., static vs. dynamic), and varying malware types or versions that limited the fairness of direct comparison. While our method is based on a custom dynamic behavioral dataset, most existing approaches rely on static binaries, memory dumps, or hybrid combinations. Thus, the performance results presented in Table 11 are not intended for direct one-to-one comparison but



**Fig. 9.** Saliency maps of representative benign (left) and ransomware (right) images. The highlighted regions indicate the pixels with the highest impact on the prediction of the model. The model focuses on structured behavior-encoded areas and confirms the class-specific pattern learning.



**Fig. 10.** Confusion matrix for the best performing model ResNet50, evaluated on the 200-sample validation set (98.50% accuracy).

Paper	Model	Analysis technique	Feature type	Threat type	Accuracy (%)
21	VGG416	Static	Image	Malware	91.41
21	VGG416	Hybrid	Image	Malware	94.70
12	Xception ColSeq	Static	Image	Malware	98.20
16	ResNet50	Static	Image	Malware	99.50
31	ResNet50	Static	Image	Malware	98.61
32	ResNet50	Static	Image	Malware	98.62
67	Xception CNN	Static	Image	Malware	99.20
8	Logistic Regression (RLR)	Dynamic	CSV	Ransomware	96.34
23	CNN	Dynamic	CSV	Ransomware	95.90
24	Random forest	Dynamic	CSV	Ransomware	97.28
29	CNN, LSTM, MLP	Static + Dynamic	Image + CSV	Ransomware	98.75
28	2-layered CNN	Dynamic	CSV	Ransomware	98.20
Our proposed method	ResNet50	Dynamic	Image (from CSV)	Ransomware	99.96

**Table 11.** Comparison of our method with existing recent relevant works in the literature.

rather to provide contextual benchmarking across a range of related methodologies. Against this benchmark, our proposed method achieved an accuracy of 99.96% on our balanced and threat-informed dataset, thereby demonstrating its effectiveness. To the best of our knowledge, no previous study has utilized this dynamic-to-image transformation technique for ransomware detection, making our approach a novel contribution to the literature.

Several studies have reported competitive results using static analysis. For example<sup>16,67</sup>, achieved accuracies of 99.50% and 99.20%, respectively, using image-based static features. However, static analysis methods are often more vulnerable to the obfuscation and packing techniques employed by modern ransomware. In contrast, our method incorporates dynamic behavioral data, enabling a more robust detection of evasive threats.

Further comparisons with dynamic and hybrid models using structured input formats (e.g., CSV) also support the strength of our approach. A technique proposed by<sup>8,23,24</sup>, and<sup>28</sup> achieved accuracy scores ranging from 95.90 to 98.20%. Although promising, these results remained below the performance achieved by our framework.

Overall, these findings underscore the advantage of converting structured behavioral data into images, allowing pretrained CNNs such as ResNet50 to exploit spatial patterns and significantly enhance detection performance, particularly in small dataset scenarios.

## Discussion

The proposed approach presented in this study showed exceptional performance with 99.96% accuracy, despite the use of a limited dataset. This remarkable outcome was owing to the combination of domain-guided feature engineering, behavior-to-image transformation, and transfer learning (TL) with pretrained CNN models.

Our data collection used an isolated dynamic analysis environment to capture the real-time ransomware behavior. This ensures that the dataset reflects genuine malicious activity rather than artificial samples. We conducted domain-based filtering to identify the most impactful features with highest discriminatory power

between benign and ransomware behaviors. Supplementary Table S1 in the Appendix provides concise descriptions of these features and their impacts on detection. Although our dataset was small, this domain filtering reduced the background noise, and each sample was filtered to maximize the representational value and enhance the dataset quality. This ensures that the model learns from relevant patterns for accurate classification, rather than from an extensive but noisy dataset.

We evaluated several CNN models using transfer learning, including ResNet50, InceptionV3, EfficientNetB0, Xception, and VGG variants. These models are well suited for small datasets because of their ability to generalize from prior training on large-scale datasets. By fine-tuning them on our domain-specific dataset, we achieved better classification performance, despite the limited sample size. Among these, ResNet50 consistently demonstrated superior accuracy and generalization capabilities. This makes it the optimal choice for our task of detecting ransomware.

Additional architectures such as MobileNet and DenseNet121 were also tested. MobileNet demonstrated faster training but suffered from reduced feature extraction capacity owing to its shallow depth, which limited the classification accuracy. Capturing nuanced behavioral patterns from ransomware-related image data requires richer feature hierarchies, and deeper models such as ResNet50 or InceptionV3 are more effective. Because our goal was to detect subtle distinctions between benign and malicious behaviors, we prioritized deeper models that offer more robust and reliable feature representations. Therefore, MobileNet is not included in the final model selection. We also experimented with DenseNet121, which is known for its feature reuse and compact design. Despite its theoretical advantages, DenseNet121 exhibited a significantly higher training time in our experiments compared to the models listed in Table 3. Given our focus on balancing the classification performance with computational efficiency, we excluded it from the final evaluation.

We also complemented the accuracy with other metrics, such as precision, recall, and F1 score, to ensure that our approach was evaluated holistically. The results confirmed consistent model behavior across classes with near-perfect classification metrics. A comprehensive analysis of the various figures and tables in this study clearly justifies the superiority of the proposed model in ransomware classification.

Model interpretability was assessed using visual tools such as t-SNE and saliency maps. The interpretability analysis confirms that our model not only achieves high classification accuracy (99.96%) but also provides valuable insight into its internal decision-making. The t-SNE visualization illustrates distinct clustering of ransomware and benign samples in the learned feature space, whereas saliency maps highlight class-relevant regions in the input images, confirming that the model focuses on meaningful behavioral patterns. A single false negative observed in the confusion matrix was explainable as a borderline behavioral case, further supporting the model's reliability and generalization. Although SHAP is a widely adopted tool for model explainability, it is primarily suited to structured data or natural images where pixel locations correspond to semantically meaningful features. For instance, Deep SHAP applied by<sup>68</sup> to the MNIST dataset works effectively because each pixel directly represents interpretable visual elements, such as strokes of handwritten digits. In contrast, our approach encodes abstract behavioral attributes into an image form, where pixel positions are determined by algorithmic mapping rather than a natural visual structure. Consequently, applying SHAP to our image-based data would produce attribution maps with limited human interpretability. Therefore, to support meaningful analysis, we adopted t-SNE and saliency maps, which are better suited for visualizing class separability and model attention in CNNs trained on behavior-to-image transformations.

The proposed approach also offers a significant methodological contribution. Although this study employs standard CNN-based transfer learning models, the core novelty lies in how these models are adapted within a behavior-centric ransomware detection pipeline. By transforming dynamic behavioral features extracted from sandbox logs into structured image representations, we enable these models, originally designed for natural images, to effectively learn the complex runtime patterns associated with modern ransomware. This adaptation, combined with domain-guided feature engineering, our novel behavior-to-image transformation, and visual interpretability techniques such as t-SNE and saliency maps, represents a significant advancement in applying transfer learning to small-data, behavior-aware cybersecurity tasks. This advancement is particularly important for enhancing ransomware detection in real-world settings, where limited behavioral data and rapidly evolving threat variants present ongoing challenges for conventional detection approaches.

The model also demonstrated robustness across image types, with only a minor decrease in accuracy when grayscale inputs were used. This result also suggests that the models can generalize well across both types of input data.

While the current study demonstrates several strengths, it is not without limitations. The limited size of the dataset, although balanced and meticulously curated, may not fully capture the diverse range of ransomware behaviors observed in real-world scenarios. Additionally, the dependence of the detection pipeline on a specific sandbox (e.g., Cuckoo sandbox) environment may limit its adaptability to broader deployment contexts. To enhance the resilience and practical applicability of the model, it is essential to address these limitations through ongoing dataset expansion, incorporation of evolving threats, and integration with diverse dynamic analysis platforms.

In summary, our study presents an interpretable, high-performance, and computationally efficient solution for modern ransomware detection, offering clear advantages over traditional static analysis methods.

## Conclusion and future work

This study introduced a novel approach for ransomware detection that leverages a three-phase workflow: dataset creation, visualization, and detection. By carefully curating the domain-based dataset, we ensured that only the most relevant and impactful ransomware behaviors were included even within a small dataset. This dataset was then transformed into visual representations, allowing the application of TL with pretrained deep learning models. The experimental results demonstrated remarkable accuracy, with ResNet50 achieving 99.96% accuracy

for color datasets and 99.91% accuracy for grayscale datasets, thereby demonstrating the power of this approach in leveraging small but highly relevant datasets for ransomware detection. The use of dynamic image-based features significantly enhances ransomware detection by capturing the complex behavioral patterns that static analysis methods often miss. TL further enhances the detection accuracy with small datasets by using pretrained models to reduce the training time and computational costs. In addition to accuracy, we emphasized model transparency through interpretability tools such as t-SNE and saliency maps. These visualizations confirmed that the model effectively distinguished ransomware from benign behavior by attending to meaningful regions of the input images rather than learning spurious patterns. The combined analysis of the performance and misclassification reinforces the robustness and reliability of the proposed pipeline. By capturing real-time behavior and employing sophisticated image-based classifiers, our method not only outperforms previous methods but also sets a new benchmark for scalable and accurate ransomware detection. This broadens the scope for future research in this area.

Future research should focus on addressing the limitations identified in this study. We intend to enhance this dataset by incorporating a diverse array of ransomware families and benign variants. To further improve robustness, we plan to explore the utilization of more advanced sandboxes, such as CAPE, ANYRUN, and JOE, for dynamic analysis. Additionally, although SHAP is not applicable to our current image-based workflow, future research may consider SHAP-based interpretability for structured feature branches to enhance explainability and decision accountability in conjunction with other advanced machine learning models or hybrid detection systems.

### Data availability

The code used for data preprocessing, including feature extraction, JSON-to-CSV parsing, and data visualization (CSV-to-image conversion), is available on GitHub: [Image-based Ransomware Detection with Transfer Learning] (<https://github.com/myresearchhub/Image-based-Ransomware-Detection-with-Transfer-Learning>) (<https://github.com/myresearchhub/Image-based-Ransomware-Detection-with-Transfer-Learning>). The data used in this study were obtained from various sources, with links provided in the “Phase 1: Dataset Creation Phase” section of the manuscript and the README.md file in the same repository. The dataset will be made available upon approval by the Charles Sturt University (CSU). Interested researchers may contact Jannatul Ferdous at [jferdous@csu.edu.au](mailto:jferdous@csu.edu.au) for data access.

Received: 7 March 2025; Accepted: 26 August 2025

Published online: 02 September 2025

### References

1. Yu, Z., Kaplan, Z., Yan, Q. & Zhang, N. Security and privacy in the emerging cyber-physical world: A survey. *IEEE Commun. Surv. Tutorials.* **23** (3), 1879–1919. <https://doi.org/10.1109/COMST.2021.3081450> (2021).
2. Jean, B. L. & Tour Howden's 2024 Cyber Insurance Report- Risk, Resilience and Relevance. (Accessed 05 November 2024). <https://www.howdengroupholdings.com/reports/2024-cyber-report> (2024).
3. SOPHOS. *Ransomware Payments Increase 500% In the Last Year, Finds Sophos State of Ransomware Report.* (Accessed 12 November 2024). <https://www.sophos.com/en-us/press/press-releases/2024/04/ransomware-payments-increase-500-last-year-finds-sophos-state> (2024).
4. Ferdous, J., Islam, R., Mahboubi, A. & Islam, M. Z. A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms. *IEEE Access* **11**, 121118–121141. <https://doi.org/10.1109/access.2023.3328351> (2023).
5. Sophos Maturing criminal marketplaces present new challenges to defenders, *Sophos 2023 Threat Report.* (Accessed 21 September 2024). <https://www.scribd.com/document/628559505/Sophos-Threat-Report-2023> (2023).
6. TREND Phobos Emerges as a Formidable Threat in Q1 2024, LockBit Stays in the Top Spot. (Accessed 12 October 2024). <https://www.trendmicro.com/vinfo/au/security/news/ransomware-by-the-numbers/phobos-emerges-as-a-formidable-threat-in-q1-2024-lockbit-stays-in-the-top-spot> (2024).
7. Ferdous, J., Islam, R., Mahboubi, A. & Islam, M. Z. AI-based Ransomware Detection: A Comprehensive Review. *IEEE Access* **12**, 136666–136695. <https://doi.org/10.1109/ACCESS.2024.3461965> (2024).
8. Daniele Sgandurra, E. C. L., Muñoz-González, L. & Mohsen, R. Automated dynamic analysis of ransomware: benefits, limitations and use for detection, <https://arXiv.org/abs/1609.032020>. <https://arxiv.org/pdf/1609.032020> (2016).
9. Singh, J., Sharma, K., Wazid, M. & Das, A. K. Spline interpolation-envisioned neural network-based ransomware detection scheme. *Comput. Electr. Eng.* **106**, 108601. <https://doi.org/10.1016/j.compeleceng.2023.108601> (2023).
10. Zahoor, U. et al. Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto ensemble classifier. *Sci. Rep.* **12** (1), 1–15. <https://doi.org/10.1038/s41598-022-19443-7> (2022).
11. Fernando, D. W., Komninos, N. & Chen, T. A study on the evolution of ransomware detection using machine learning and deep learning techniques. *IoT* **1** (2), 551–604. <https://doi.org/10.3390/iot1020030> (2020).
12. Moreira, C. C., Moreira, D. C. & de Sales, C. Improving ransomware detection based on portable executable header using Xception convolutional neural network. *Comput. Secur.* **130**, 103265. <https://doi.org/10.1016/j.cose.2023.103265> (2023).
13. Vasan, D. et al. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Comput. Networks.* **171**, 107138. <https://doi.org/10.1016/j.comnet.2020.107138> (2020).
14. Bensaoud, A. & Kalita, J. CNN-LSTM and transfer learning models for malware classification based on opcodes and API calls. *Knowledge-Based Syst.* **290**, 111543. <https://doi.org/10.1016/j.knsys.2024.111543> (2024).
15. Kumar, S. & Panda, K. Stacking deep image features using fine-tuned Convolution neural network models for real-world malware detection and classification. *Appl. Soft Comput.* **146**, 110676. <https://doi.org/10.1016/j.asoc.2023.110676> (2023).
16. Almomani, I., Alkhayer, A. & El-Shafai, W. E2E-RDS: efficient End-to-End ransomware detection system based on Static-Based ML and Vision-Based DL approaches. *Sensors* **23** (9), 1–27. <https://doi.org/10.3390/s23094467> (2023).
17. Ahmed, M., Afreen, N., Ahmed, M., Sameer, M., Ahamed, J. (2022). An inception approach for malware classification using machine learning and transfer learning. *Int. J. Intell. Netw.* **4** 11–18, <https://doi.org/10.1016/j.ijin.2022.11.005>
18. Chaganti, R., Ravi, V. & Pham, T. D. Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification. *J. Inf. Secur. Appl.* <https://doi.org/10.1016/j.jisa.2022.103306> (2022).
19. Escudero García, D., DeCastro-García, N. & Muñoz Castañeda, A. L. An effectiveness analysis of transfer learning for the concept drift problem in malware detection. *Expert Syst. Appl.*, **212** 118724. <https://doi.org/10.1016/j.eswa.2022.118724> (2022).

20. Aghakhani, H. et al. When malware is packin' heat; limits of machine learning classifiers based on static analysis features. *27th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)* <https://doi.org/10.14722/ndss.2020.24310> (2020).
21. Huang, X., Ma, L., Yang, W. & Zhong, Y. A method for windows malware detection based on deep learning. *J. Signal. Process. Syst.* **93**, 2–3. <https://doi.org/10.1007/s11265-020-01588-1> (2021).
22. Sharma, S., Krishna, C. R. & Kumar, R. RansomDroid: forensic analysis and detection of android ransomware using unsupervised machine learning technique. *Forensic Sci. Int. Digit. Investig.* **37**, 301168. <https://doi.org/10.1016/j.fsidi.2021.301168> (2021).
23. Qin, B., Wang, Y. & Ma, C. API Call based ransomware dynamic detection approach using TextCNN, *Proc. 2020 Int. Conf. Big Data, Artif. Intell. Internet Things Eng.* 162–166. <https://doi.org/10.1109/ICBAIE49996.2020.00041> (ICBAIE, 2020).
24. Hwang, J., Kim, J., Lee, S. & Kim, K. Two-Stage ransomware detection using dynamic analysis and machine learning techniques. *Wirel. Pers. Commun.* **112** (4), 2597–2609. <https://doi.org/10.1007/s11277-020-07166-9> (2020).
25. Manavi, F. & Hamzeh, A. A novel approach for ransomware detection based on PE header using graph embedding. *J. Comput. Virol. Hacking Tech.* **18** (4), 285–296. <https://doi.org/10.1007/s11416-021-00414-x> (2022).
26. Manavi, F. & Hamzeh, A. Ransomware detection based on PE header using convolutional neural networks\*\*. *ISecure* **14** (2), 181–192. <https://doi.org/10.22042/isecure.2021.262846.595> (2022).
27. Deng, X. Z., Cen, M. C., Jiang, M. & Lu, M. Ransomware early detection using deep reinforcement learning on portable executable header. *Cluster Comput.* **27** (2), 1867–1881. <https://doi.org/10.1007/s10586-023-04043-5> (2023).
28. Gulmez, S., Gorgulu Kakisim, A. & Sogukpinar, I. XRan: explainable deep learning-based ransomware detection using dynamic analysis. *Comput. Secur.* **139**, 103703. <https://doi.org/10.1016/j.cose.2024.103703> (2024).
29. Karbab, E. M. B., Debbabi, M. & Derhab, A. SwiftR: Cross-platform ransomware fingerprinting using hierarchical neural networks on hybrid features. *Expert Syst. Appl.* **225**, 120017. <https://doi.org/10.1016/j.eswa.2023.120017> (2023).
30. Ni, S., Qian, Q. & Zhang, R. Malware identification using visualization images and deep learning. *Comput. Secur.* **77**, 871–885. <https://doi.org/10.1016/j.cose.2018.04.005> (2018).
31. Vasan, D., Alazab, M., Wassan, S., Safaei, B. & Zheng, Q. Image-Based malware classification using ensemble of CNN architectures (IMCEC). *Comput. Secur.* **92**, 101748. <https://doi.org/10.1016/j.cose.2020.101748> (2020).
32. Rezende, E., Ruppert, G., Carvalho, T., Ramos, F. & De Geus, P. Malicious software classification using transfer learning of ResNet-50 deep neural network. *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl.* **2017** 1011–1014. <https://doi.org/10.1109/ICMLA.2017.00-19> (ICMLA, 2017).
33. Abuse.ch, MalwareBazaar database. (Accessed 02 June 2024). <https://bazaar.abuse.ch/browse/> (2024).
34. VirusShare VirusShare.com - Because Sharing is Caring. (Accessed 07 June 2024). <https://virusshare.com/> (2025).
35. SnapFiles Popular freeware categories. (Accessed 06 July 2024). <https://www.snapfiles.com/freeware/> (1997).
36. Portableapps, Portable App Directory. (Accessed 10 July 2024). <https://portableapps.com/apps> (2024).
37. Iosifache, DikeDataset. (Accessed 16 July 2024). <https://github.com/iosifache/DikeDataset/tree/main/files/benign> (2023).
38. Ang, M. J. A. Narrowed Sights, Bigger Payoffs: Ransomware in 2019. (Accessed 01 October 2024). <https://www.trendmicro.com/vinfo/au/security/news/cybercrime-and-digital-threats/narrowed-sights-bigger-payoffs-ransomware-in-2019> (TREND, 2019).
39. Networks, P. A. Unit 42 Ransomware threat report. (Accessed 10 September 2024). <https://start.paloaltonetworks.com/unit-42-ransomware-threat-report.html> (2020).
40. Zscaler, Zscaler: 2022 ThreatLabz State of Ransomware Report. (Accessed 12 October 2024). <https://www.zscaler.com/resources/industry-reports/2022-threatlabz-ransomware-report.pdf> (2022).
41. Symantec, B. The ransomware threat landscape. *Ransomware Threat Landsc.* <https://doi.org/10.2307/j.ctv1f8xc7v> (2023).
42. VirusTotal VirusTotal ransomware activity report. <https://www.virustotal.com/gui/home/upload> (Accessed 02 Jun 2024).
43. SentinelOne, "Ragnar Locker Ransomware: In-Depth Analysis, Detection, Mitigation, and Removal," 2024. <https://www.sentinelone.com/anthology/ragnar-locker/#:~:text=Summary of Ragnar Locker Ransomware&text=Ragnar Locker targets corporate networks,non-release of stolen data.> (accessed Aug. 28, 2024). [44] K. Baker, "Ransomware Examples: 16 Recent Ransomware Attacks," (2024).
44. Baker, K. Ransomware examples: 16 recent ransomware attacks. (Accessed 29 August 2024). <https://www.crowdstrike.com/cybersecurity-101/ransomware/ransomware-examples/> (2024).
45. Research, T. M. & Ransomware-spotlight-Clop (Accessed 29 August 2024). <https://www.trendmicro.com/vinfo/au/security/news/ransomware-spotlight/ransomware-spotlight-clop> (2022).
46. Cuckoo Sandbox- automated malware analysis. <https://cuckoosandbox.org/download>
47. Myresearchhub Image-based-Ransomware-Detection-with-Transfer-Learning. (Accessed 02 November 2024). <https://github.com/myresearchhub/Image-based-Ransomware-Detection-with-Transfer-Learning> (2024).
48. Algarni, M. D. et al. An Efficient Convolutional Neural Network with Transfer Learning for Malware Classification. *Wirel. Commun. Mob. Comput.* <https://doi.org/10.1155/2022/4841741> (2022).
49. He, K. *Deep Residual Learning for Image Recognition*. (2015).
50. Tan, M. & Le, Q. V. EfficientNetV2: Smaller Models and Faster Training. *Proc. Mach. Learn. Res.* **139**, 10096–10106 (2021).
51. Lo, W. W., Yang, X. & Wang, Y. An xception convolutional neural network for malware classification with transfer learning. *10th IFIP Int. Conf. New Technol. Mobil. Secur. NTMS 2019 Proc. Work.* 1–5. <https://doi.org/10.1109/NTMS.2019.8763852> (2019).
52. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *3rd Int. Conf. Learn. Represent. ICLR 2015 Conf. Track Proc.* 1–14, (2015).
53. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. Rethinking the inception architecture for computer vision. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* **2016** 2818–2826. (2016).
54. Rani, K. S. et al. Implementation of inception V3-Based CNN classification for malware detection. *2024 Int. Conf. Recent Adv. Electr. Electron. Ubiquitous Commun. Comput. Intell. RAEEUCCI 2024.* <https://doi.org/10.1109/RAEEUCCI61380.2024.10547935> (2024).
55. Tan, M. & Le, Q. V. EfficientNet: Rethinking model scaling for convolutional neural networks. *36th Int. Conf. Mach. Learn.* **2019** 10691–10700. (ICML, 2019).
56. Chollet, F. Xception: Deep learning with depthwise separable convolutions, *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognition CVPR 2017* **2017** 1800–1807. <https://doi.org/10.1109/CVPR.2017.195> (2017).
57. Kumar, S., Janet, B., DTMIC. DTMIC: Deep transfer learning for malware image classification. *J. Inf. Secur. Appl.* <https://doi.org/10.1016/j.jisa.2021.103063> (2021).
58. Manirih, P., Mahmood, A. N. & Chowdhury, M. J. M. A Survey of Recent Advances in Deep Learning Models for Detecting Malware in Desktop and Mobile Platforms. *ACM Comput. Surv.* <https://doi.org/10.1145/3638240> (2024).
59. Keras & Probabilistic losses. [https://keras.io/api/losses/probabilistic\\_losses/#sparsecategoricalcrossentropy-class](https://keras.io/api/losses/probabilistic_losses/#sparsecategoricalcrossentropy-class) (Accessed 08 February 2025).
60. Darabian, H. et al. Detecting cryptomining malware: a deep learning approach for static and dynamic analysis. *J. Grid Comput.* **18** (2), 293–303. <https://doi.org/10.1007/s10723-020-09510-6> (2020).
61. Kumar, S., Meena, S., Khosla, S. & Parihar, A. S. Autoencoder Enhanced Deep Convolutional Neural Network For Malware Classification. *Int. Conf. Intell. Technol.* <https://doi.org/10.1109/CONIT51480.2021.9498570> (2021).
62. Colab, Welcome to Colab. <https://colab.research.google.com/#scrollTo=Wf5KrEb6vrkR> (Accessed 10 August 2024).
63. TensorFlow An end-to-end platform for machine learning. <https://www.tensorflow.org/>
64. Keras About Keras 3 <https://keras.io/about/>
65. Pandas Latest version: 2.2.3. <https://pandas.pydata.org/>. (2024).

66. NumPy *The fundamental package for scientific computing with python*. <https://numpy.org/>. (2024).
67. Sharma, O., Sharma, A. & Kalia, A. Windows and IoT malware visualization and classification with deep CNN and Xception CNN using Markov images. *J. Intell. Inf. Syst.* **60** (2), 349–375. <https://doi.org/10.1007/s10844-022-00734-4> (2023).
68. Scott, M., Lundberg & Lee, S. I. A unified approach to interpreting model predictions. *31st Conf. Neural Inf. Process. Syst. (NIPS 2017)* **30** 1–10. (Long Beach, 2017).

### Author contributions

Jannatul Ferdous: Conceptualization, Methodology, Python Scripting, Investigation of experimental results, Writing – original draft, Visualization. Rafiqul Islam: Validation, Writing – review & editing, Supervision. Arash Mahboubi: Validation, Writing – review & editing, Supervision. Md Zahidul Islam: Validation, Writing – review & editing, Supervision.

### Funding

This work was supported by the Charles Sturt University Tri-Faculty Open Access Publishing Fund (2025).

### Declarations

#### Competing interests

The authors declare no competing interests.

#### Additional information

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1038/s41598-025-17647-1>.

**Correspondence** and requests for materials should be addressed to J.F.

**Reprints and permissions information** is available at [www.nature.com/reprints](http://www.nature.com/reprints).

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

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025