information

Article

Malware Classification Using Few-Shot Learning Approach

Khalid Alfarsi *, Saim Rasheed and Iftikhar Ahmad

Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; srahmed@kau.edu.sa (S.R.); iakhan@kau.edu.sa (I.A.)
* Correspondence: kayidhalfarsi@stu.kau.edu.sa

Abstract: Malware detection, targeting the microarchitecture of processors, has recently come to light as a potentially effective way to improve computer system security. Hardware Performance Counter data are used by machine learning algorithms in security mechanisms, such as hardware-based malware detection, to categorize and detect malware. It is crucial to determine whether or not a file contains malware. Many issues have been brought about by the rise in malware, and businesses are losing vital data and dealing with other issues. The second thing to keep in mind is that malware can quickly cause a lot of damage to a system by slowing it down and encrypting a large amount of data on a personal computer. This study provides extensive details on a flexible framework related to machine learning and deep learning techniques using few-shot learning. Malware detection is possible using DT, RF, LR, SVM, and FSL techniques. The logic is that these algorithms make it simple to differentiate between files that are malware-free and those that are not. This indicates that their goal is to reduce the number of false positives in the data. For this, we use two different datasets from an online platform. In this research work, we mainly focus on few-shot learning techniques by using two different datasets. The proposed model has an 97% accuracy rate, which is much greater than that of other techniques.

Keywords: few-shot learning (FSL); Prototypical; malware detection; cyber-attack; classification algorithms

Citation: Alfarsi, K.; Rasheed, S.; Ahmad, I. Malware Classification Using Few-Shot Learning Approach. *Information* **2024**, *15*, 722. https://doi.org/10.3390/info15110722

Academic Editor: Sherali Zeadally

Received: 8 September 2024
Revised: 19 October 2024
Accepted: 31 October 2024
Published: 11 November 2024

1. Introduction

In today's networked world, any system is vulnerable to cyber-attacks from malevolent actors. Attackers now have access to increasingly potent automated technology, and new threats appear almost instantaneously. It may be challenging to maintain appropriate cybersecurity as a result [1]. Malevolent software is one of the biggest problems facing the modern digital world, and sadly, things are only getting worse. Malware is software that is specifically created to harm a computer system or network to spy on people or steal money. Malware is a type of harmful programming that is intended to steal or alter essential functions, monitor user computers, encrypt or erase important data, or intercept computer systems. Malware can take many different forms, such as Trojan horses, ransomware, and computer viruses. One example that has caused the destruction of 100 thousand systems worldwide is a notorious ransomware program that caused a loss of about five billion dollars in 2017 [2]. Malware classification is a process or study to determine the origin, functionality, and potential impact of a sample of malware [3].

Malware classification is an issue that can be divided into two categories: malware classification and sample family classification. The former establishes whether or not a sample is malicious software. This study concentrates on the initial issue. In addressing the challenge of malware classification, various techniques play a pivotal role; different methods serve different purposes. Classification is the interpretation or prediction of one feature by relying on other characteristics, and it is one of the most common ways to classify large datasets [4–6].

Since thousands of new files are created on a daily basis, dangerous files offer a continual and growing threat in the digital world. Traditional antivirus solutions that are pattern-based struggle to detect these new hazardous files because they lack distinguishable patterns. Malware detection methods that work based on the concept of artificial intelligence (AI) have been looked into as a potential fix for this issue.

However, these methods often need time-consuming steps. Support Vector Machines are algorithms that detect linear separators between data points from two classes to analyze data in a multidimensional space. The SVM classification system is useful for network data analysis to support computer emergency response teams in detecting threats. Moreover, this kernel-based approach is thought to be an effective method for classifying heterogeneous online datasets for malware detection [7]. With a topological graph routing structure, neural networks are dynamic systems that process information by reacting to either a continuous or discontinuous input state. For applications like picture categorization, this deep learning technique performs better than conventional learning algorithms. To address these issues, this study explores the following research questions: How successful are classical machine learning methods for malware classification? How useful are few-shot learning approaches, such as Prototypical networks, in increasing malware detection? How accurate and efficient are these methods compared to the classical machine learning models?

Moreover, this research also aims to enhance methods for feature extraction from and classification of malware datasets, with a focus on neural network applications. The increasing quantity and variety of malware samples present analysts with an increasing challenge that requires robust automated detection and classification techniques. The current methods based on heuristics or signatures are not able to keep up, and more complex solutions are needed. The neural network discussed makes maximum use of feedforward and convolutional layers to classify harmful executables. K-nearest neighbors (K-NN), which is regarded as an administered machine learning algorithm, is used to solve regression and class problems. Predicting a data point's class or value by comparing it to its "k"-nearest neighbors in the feature space is its main goal.

This novel strategy aims to achieve a compromise between the accuracy of AI-based techniques and the speed of similarity-hash-based detection. Through the utilization of a hierarchical similarity hash and KNN classification, this study aims to offer a more resilient response to the ongoing problem of quickly detecting and classifying new harmful files in the ever-changing cybersecurity environment [8]. Effective representations for the challenge of recognition can be extracted by the deep learning recognition models. However, because they rely on a lot of labeled training data, they have poor performance in limited data situations and an inadequate generalization capacity. In this paper, few-shot learning techniques are used to train the proposed model even when the amount of training data is small. There are many classification models that behave very nicely and generate very significant and reliable results when the amount of training data is very high, but it takes time to collect data, and under various conditions, they take a long time; when it comes to training for classification, they take even longer. One of the solutions that can solve this problem is the few-shot learning technique.

The human learning process of identifying a new item or grasping a new concept after only a few occasions or one occasion is where the concept of few-shot learning originates. It is regarded as a fix for numerous kinds of deep learning issues when particular information is included in the training dataset. For learning from a few-shot instance, deep learning and machine learning modeling are therefore also being viewed with great expectation [9]. The most crucial few-shot learning strategy is meta-learning, also referred to as "learning how to learn". Its goal is to specify a workable process for upgrading learner standards. Using a meta-learning paradigm, meta-learning-based techniques train an across-task meta-learner.

It is the most natural way to learn from pertinent data and apply them to the target field when dealing with few-shot activities. The goal of data augmentation-based methods is to assist few-shot learning tasks with external data or employ generator networks to generate new instances from sparse training data. Performing data augmentation

during training that is independent of the supplementary dataset is an easy method. Generative adversarial networks are one example of how it is used (GANs). Few-shot learning technology has various applications, including computer vision, robotics, acoustic signal processing, Internet of Things analytics, and medical applications.

2. Literature Review

This section is related to the work that has been carried out so far using malware models and few-shot models for the detection of malware.

2.1. Malware and Its Types

Many terms for malware and its features have become widely used in the field of malware analysis. Although there are numerous varieties of malware, the following are the ones that are most common right now [9]:

(1) Virus: A malware that attaches itself to other software, spreading when that software is installed. It can capture keystrokes, corrupt data, and consume system resources, often infecting computers through malicious email attachments.
(2) Trojan: This malware masquerades as legitimate software, infiltrating systems via free downloads or email attachments. Once activated, it can monitor user activity, breach networks, or steal sensitive data, often indicated by unusual device behavior.
(3) Worm: A self-replicating malware that spreads from one computer to another without human intervention, typically using an Internet connection or a Local Area Network (LAN).
(4) Spyware: This software collects personal data from users and transmits it to third parties without consent. While some spyware may have legitimate purposes, malicious spyware aims to profit from stolen data, leaving users vulnerable to data breaches.
(5) Ransomware: A type of malware that locks users out of their files, demanding a ransom for access. Victims, including organizations, often pay to recover their data, with some variants also stealing data to increase the pressure to pay.

2.2. Malware Classification Using ML Methods

Initially, malware was created with straightforward objectives that made it simple to identify. Traditional malware is one type of such instance. On the other hand, newer malware might operate in kernel mode, which makes it more challenging to recognize and even harder to detect [10]. Conventional malware only has one process and does not use advanced techniques. On the other hand, new-generation malware can use several processes—either existing or newly created—at the same time and employ obfuscated techniques, increasing its capacity to hide and survive in the system [11]. Malware of the new generation is capable of more destructive operations than ever before, like persistent and targeted attacks that may involve multiple malware types.

Advances in computer and communication infrastructure have also made it harder to identify and classify malware. Malware classes have been developed as a result, allowing reverse engineers to understand the trends and classifications and deal with an infection's progression. In addition, fresh methods of analysis have been put forth to acquire a better comprehension of the actions and features of a virus. These recommendations facilitate reverse engineers comprehending a malware sample's objective and allow them to investigate it in more detail [12].

Creating classification models that tackle important malware analysis problems is the aim of the suggested approach to investigating malware activity. Its main objective is to identify whether an unknown malware instance belongs to a well-known family or is a novel strain. To this end, malware binaries are collected via spam traps and honeypots, detected by an antivirus program, and then behaviorally examined in a sandbox environment. The learning system then creates discriminative models for each malware family based on the analysis reports, producing classifiers that predict family membership. Second, the method identifies discriminative behavioral features by assigning weights to patterns

observed during training. Not only do these models have predictive power but they also draw attention to key differences between different malware families [13].

2.3. Traditional Malware Models

The classic tools for finding malware are antivirus programs and spyware detectors. On the other hand, conventional detection methods that use preset syntactic signature matching to identify malware can be readily evaded by code polymorphism and metamorphism. It is impossible for traditional signature-based antivirus programs to identify harmful executables that are polymorphic, metamorphic, or unknown. To solve this issue, a number of malware detectors have been suggested in two important analysis domains [3].

Conventional detection systems have faced significant challenges due to the rise in malware threats in recent years. Despite the ease of access to security solutions like antivirus software, firewall defense, and SSL certificate encryption, their efficacy is constrained by inherent limitations. Conventional security is defensive, offering momentary defense against recognized threats. Nevertheless, this method finds it difficult to stay up to date with the malware industry's ongoing evolution, which brings new variations and evasion strategies. Since new threats are always emerging, security systems that rely on signature databases to identify known malware need to keep up with the latest developments. They can be susceptible to signature modifications taking an excessively lengthy time.

When it comes to detecting malware or insider threats that appear to be trustworthy but are Trojan horses, the traditional approaches may not be entirely trustworthy. It is quite challenging to identify harmful code hidden in seemingly harmless systems. Certain conventional methods, including those predicated on in-depth feature analysis or heuristic procedures, may require a large computational capacity, which could jeopardize their effectiveness and real-time capabilities. Additionally, as technology develops, new attack vectors emerge, and traditional methods may not be well equipped to address threats directed at emerging technologies such as the Internet of Things (IoT), where efficient and low-cost detection processes are essential. Given these difficulties, it is becoming more and more important to employ contemporary methods and cutting-edge technologies to improve the effectiveness and versatility of malware detection systems. One must constantly conduct studies and create new tactics in order to keep ahead of the always changing world of cybersecurity threats [14].

2.4. Malware Classification Using DL Techniques

By giving minority classes more weight, deep unbalanced learning seeks to reduce the bias in model training toward majority classes. The categories for the current approaches are algorithm/model level, data level, and cost-sensitive learning level, in that order. The strategies used at the algorithm/model level concentrate on improving the classification performance by modifying the training procedures. To deal with high-dimensional data, ref. [15] created a variant long short-term memory (LSTM) model. Three loss functions were quantified and linked with a rebuilt hidden variable to enhance the anomaly detection performance. Certain sampling-based techniques, such as oversampling and undersampling, have received a lot of attention for addressing unbalanced datasets at the data level.

According to [16], who looked at several variations of under- and oversampling strategies, their impacts vary depending on the situation. However, a lot of researchers [17] have amplified particular data distributions using the generative adversarial network algorithm, which may help to lessen the noise or useless data produced by traditional data sampling techniques. To reinforce bias against infrequent but useful instances of instability, cost-sensitive learning [18] is employed. To enhance the classification performance, these techniques seek to optimize the dataset's loss function. They are unable to precisely solve the deviations in the model, though, because it is typically uncertain how much it costs to see each kind of inaccuracy.

2.5. The Few-Shot Learning Technique

Building a malware classifier that can swiftly identify an issue is crucial since malware threats are growing daily. While conventional deep learning classifiers have demonstrated efficacy in identifying complex malware based on benchmark datasets, the retention of classifiers in the event of the advent of novel malware families is a challenge. This paper's few-shot learning technique enables malware classification based on a small number of cases, negating the need to retrain the classifier for new malware families [19].

A primary concern in machine learning is the amount of data acquired. One-shot or few-shot learning is a concept for learning an object class from a small number of data points. With limited samples available, few-shot learning (FSL) aims to distinguish novel classes. During training, FSL learns a series of linked tasks from a task distribution rather than a single task. Utilizing a Memory-Augmented Neural Network in conjunction with Natural Language Processing methods, such as word2vec and n-gram, is one way to address malware classification issues utilizing this learning approach [20]. All of the malware's API calls are fed into these particular learning models after being encoded in a variety of feature spaces, which are crucial information sources for identifying the actions of malware. Previous findings showed that the results were extraordinarily accurate when compared to those of other standard approaches, leading to a new field of study in malware.

Zero-shot learning (ZSL) is a machine learning paradigm that enables a model to recognize and classify objects, tasks, or categories that it has not encountered during training. This capability is achieved by leveraging semantic information about the classes (such as attributes or textual descriptions) to generalize from known classes to unseen ones. Unlike traditional supervised learning, where models are trained on labeled data for specific classes, zero-shot learning allows models to make predictions about new classes based on their understanding of existing classes. In terms of semantic relationships, ZSL often relies on semantic embeddings (such as word vectors or attribute vectors) that capture the relationships between known and unknown classes [21].

2.6. Few-Shot Malware Models

For ransomware defense systems to create quick reaction plans, they must be able to quickly recognize and classify threats. Deep learning techniques have been shown to be applicable in several domains, but the lack of data for ransomware samples has impeded the creation of practical solutions [22]. This research advances the development of few-shot malware models by offering a novel approach that considers entropy characteristics, addresses issues with image-based features, and optimizes the model weights for improved accuracy in classifying ransomware variations. The ensuing sections examine the pertinent literature, provide details on the model, illustrate the experimental conditions and results, and conclude with recommendations for additional study.

Static analysis and dynamic analysis are the two techniques for identifying malware, as previously discussed. Malicious code needs to be run in order for a dynamic analysis detection method to function, despite its high detection rate [23]. However, this could lead to a significant scalability problem when processing the vast amount of previously unidentified binaries that show up on the desks of malware researchers. Furthermore, anti-emulation techniques prevent the extraction of crucial virus behavioral patterns. Furthermore, it is challenging to profile different malware families due to the considerable variations in their activity patterns. Some of the issues with dynamic analysis detection are resolved when malware threats are identified by static analysis before they are executed. It is possible to carry out destructive actions by looking at bytes or instruction sequences [24]. As a result, despite the obvious benefits of dynamic analysis, static code analysis is employed to ascertain malware's distinct properties.

As artificial intelligence (AI) techniques have become more and more popular, more (shallow) machine learning and deep learning approaches have been introduced. Deep learning has two main restrictions when it comes to its application. The first issue is that most of the research being undertaken today is on generating feature profiles from many

malware families and removing generic aspects from malware images. The feature profiles show the attributes of a malware sample, which can be used to assess the maliciousness of a picture sample [25]. A deep neural network cannot properly classify several variants produced by the same malware family if it uses common static data as the data input points because it cannot capture the distinctive characteristics of each malware family signature. The second issue with the current techniques is the need for huge datasets in order to identify more significant correlations between variables. With a short sample set, they are unable to identify and classify malware families [26].

Using 5-way 1-shot, 10-way 1-shot, 5-way 5-shot, and 10-way 5-shot learning, single-label and multi-label categorization are investigated. Moreover, a single-label classification task is evaluated using micro and macro F1 scores, which are supposed to quantify the average F1 scores. While the macro average provides different labels with the same weight, the micro average gives each sample the same weight regardless of the label imbalance [23].

The goal of few-shot learning (FSL) is to identify new classes when there are only a few samples available. Instead of learning a single task during training, FSL uses a task distribution to teach itself on a series of related tasks. Few-shot learning techniques that project input into the embedding space to facilitate distinguishing between similar and different samples easily are known to favor embedding-based methods. In general, task-invariant learning of the embedding function [27] allows for fast assimilation of task-specific knowledge through sample feeding without the need for retraining. Pairwise comparisons are made in the embedding space in KNN.

Many models, such as the Prototypical network proposed, which computes the mean as class prototypes and classifies according to the square Euclidean distance, extract class prototypes from a small number of samples to replace KNN with nearest neighbor classification between the queried sample and class prototypes. This helps reduce the influence of noisy samples. To decrease noise, class prototypes were obtained using a dynamic routing method and a hybrid attention-based prototype network [28] that computed the prototypes using an attention mechanism. Foregoing embedding-based work can be seen as an expansion of some prior work. Numerous studies have attempted to address few-shot learning from different angles. Table 1 shows an analysis of previous works

Table 1. Analysis of previous works.

Reference	Year	Objective	Dataset	Techniques	Issues
[1]	2022	Malware detection	Microsoft Malware Classification Challenge (2015)	RNN, CNN, DT, RF	Resolving avoidance strategies in static classifiers
[4–6]	2021	Classification techniques	Malware Detection	SVM, neural networks, Naive Bayesian classifiers, K-nearest neighbors	Predictive analysis
[9]	2020	Fast time in malware detection and classification	AI-based malware detection dataset	KNN	Dynamic analysis problems
[10]	2021	Malware classification framework based on deep learning algorithms	Malimg dataset, Microsoft Big 2015 dataset, and Malevis dataset	Deep learning	Converting from adaptable into conventional methods
[18]	2021	An SVM for malware detection	Malware detection	Support Vector Machines	Efficient malware detection in heterogeneous web datasets
[22]	2022	Few-shot learning for malware classification	Android-based malware dataset	Memory-Augmented Neural Network, Natural Language Processing	Improved accuracy with a small number of cases

Table 1. *Cont.*

Reference	Year	Objective	Dataset	Techniques	Issues
[24]	2021	Advancements in machine learning models for malware classification	Malware classification	RF, DL	Shift from traditional to adaptive approaches
[25]	2019	Efficacy of the latest FSL methods in malware detection	Classification dataset	Deep learning, data mining, big data methods	Higher accuracy rates
[26]	2019	Adaptive learning	AI-based malware detection	K-nearest neighbors, hierarchical similarity hash	Compromise between accuracy and speed
[28]	2022	Few-shot models for ransomware defense	Malware feature dataset	Deep learning techniques	Application in ransomware defense issue

3. Materials and Methods

3.1. Artificial Neural Networks (ANNs)

Artificial neural networks are a method of artificial intelligence through which computers learn to process data in a way inspired by the human brain. Artificial neural networks use nodes or neurons that are interconnected in a multi-layered structure similar to the human brain. Artificial neural networks create an adaptive system that learns from its mistakes and continually improves, thus solving complex problems [29]. The first inspiration for the idea of artificial neural networks came from the workings of the neurons in the human brain, which can be likened to biological neural networks used to process information coming into the brain. The synapses in these networks play a fundamental role in directing processing process, and this is what prompted thinking about the idea of connectivity and artificial neural networks [30]. Artificial neural networks consist of nodes, neurons, or processing units connected together to form a network of nodes created by computational programs that resemble the work of biological neurons or electronic structures. The overall behavior of the network is determined through these connections, and the mathematical model is used to process information based on the connectivity method in computing. Figure 1 shows the working principles of artificial neural networks.

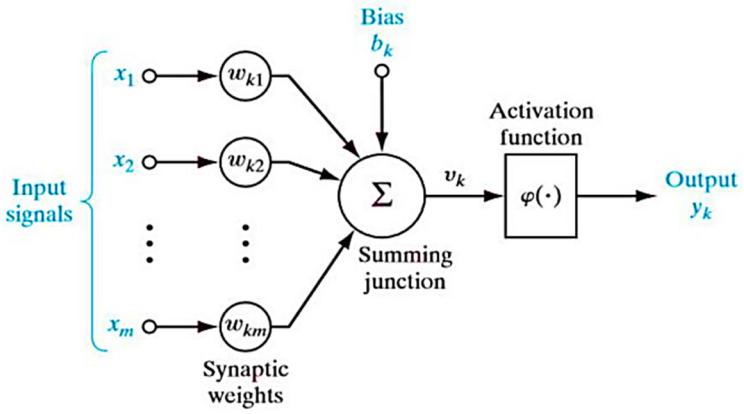

Figure 1. The working principle of artificial neural networks [31].

The artificial neural nodes are organized into parallel layers, an input layer and an output layer. These are considered the two main layers. Between these two layers, there is also a group of hidden layers, with each layer specializing in a specific type of data. All of these layers are linked to each other to form the structure of the artificial neural network [32]. Since artificial neural networks are similar to the human brain, this means that they are constantly changing and developing, meaning that artificial neural networks

are capable of learning and the way they analyze data changes, so they avoid previous errors. The artificial neural network receives a set of inputs and processes them through a set of weights associated with the input of each neuron. After that, a mathematical function called the activation function is used, which produces in its output a value representing the output of the neural circuit.

3.2. K-Nearest Neighbors (KNN)

The K-nearest neighbors (KNN) algorithm is a supervised machine learning method employed to tackle classification and regression problems. Evelyn Fix and Joseph Hodges developed this algorithm in 1951, which was subsequently expanded by Thomas Cover. The K-NN algorithm works by finding the K-nearest neighbors to a given data point based on a distance metric, such as the Euclidean distance. The class or value of the data point is then determined by the majority vote or the average of the K neighbors.

As we mentioned above, the KNN algorithm helps us identify the nearest points or groups for a query point. But to determine the closest groups or the nearest points for a query point, we need a metric. For this purpose, we use the following distance metrics:

(1) Euclidean distance: This is the Cartesian distance between two points that are in the plane/hyperplane. The Euclidean distance can also be visualized as the length of the straight line that joins the two points that are under consideration. If we have two points A ($x1$,$y1$), B ($x2$,$y2$), the Euclidean distance between these points is given by Formula (1):

$$d(A, B) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \quad (1)$$

(2) Manhattan distance: The Manhattan distance metric is generally used when we are interested in the total distance traveled by an object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions. If we have two points A (x1,y1), B (x2,y2), the Euclidean distance between these points is given by Formula (2):

$$d(A, B) = |x2 - x1| + |y2 - y1| \quad (2)$$

3.3. The Proposed Methodology

The proposed methodology is based on building a Multi-Layer Perceptron (MLP)-type artificial neural network, then applying the few-shot learning technique to training the proposed neural network to extract features, and then applying the steps of the KNN algorithm to the output of the proposed artificial neural network. The MLP acts as a feature extractor by learning complex nonlinear patterns in the data as it transforms the raw input data into a higher-level representation that captures the essential characteristics of each class. This feature extraction step is very important, especially when dealing with complex data. The output of the MLP, in the form of embeddings or feature vectors, serves as the input to the KNN algorithm. These feature vectors are more discriminative and lower-dimensional than the original input data, making KNN more effective and efficient in finding nearest neighbors. Using the features extracted by the MLP, KNN can distinguish between different classes better, even when the classes are very similar. When the MLP preprocesses the data, it improves the overall classification accuracy by providing KNN with accurate, high-quality features that represent the relationships between samples better, especially in few-shot learning scenarios, where data are sparse. Figure 2 represents the general structure of our proposed methodology.

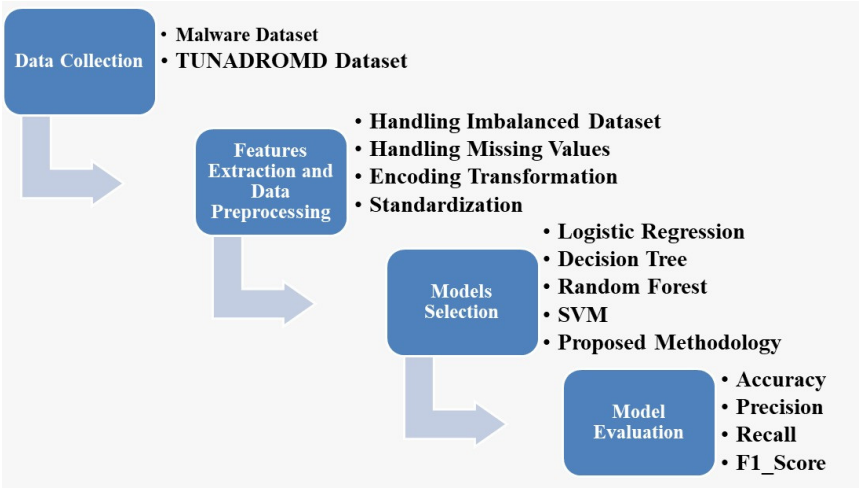

Figure 2. The general structure of our proposed methodology.

3.3.1. Data Collection

Two datasets from kaggle.com were collected. The malware dataset (https://www.kaggle.com/datasets/nsaravana/malware-detection) (accessed on 10 July 2024) has 35 characteristics and 100,000 samples. A category 0 vs. 1 is the target attribute for categorization. The second dataset TUNADROMD (https://www.kaggle.com/datasets/joebeachcapital/tunadromd-malware-detection) (accessed on 25 July 2024) has 241 characteristics and 4465 samples.

3.3.2. Data Preprocessing

Imbalanced datasets represent a major challenge in the field of artificial intelligence in general [33]. Counting imbalance occurs when a dataset contains a class that has a much larger number of samples than the other classes. This can lead to many problems, such as biased models, unreliable performance measures, and overfitting. The first dataset contains 50,000 samples each for benign and malware samples, so it is a balanced dataset. The second dataset contains 3565 samples from the malware class and 899 samples from the benign class, so this dataset needs balancing. Therefore, in this stage, the crucial step involves rectifying the imbalance present in the dataset. We used the SMOTE (Synthetic Minority Over-Sampling Technique) method, which is an oversampling method that produces synthetic samples for the minority class [34]. Unlike random oversampling, it addresses overfitting issues by concentrating on the feature space.

Due to various factors, real-world datasets frequently include missing values, typically denoted as blanks, NaNs, or other placeholders [35]. These datasets pose challenges for machine learning estimators, which expect numerical values and meaningful content in all array elements. The SimpleImputer Python class provides basic strategies for imputing missing values [36]. Missing values can be imputed with a provided constant value or using the statistics (mean, median, or most frequent) in each column in which the missing values are located. In our proposed methodology, we relied on calculating the missing values by taking the mean value of the available data.

Ensuring consistency in the numerical input data is crucial to enhancing the performance of machine learning algorithms. To achieve this uniformity, it is necessary to adjust the data to a standardized range [37]. We used StandardScaler, which is a data preprocessing technique used in machine learning and statistics. Its goal is to transform the features of a dataset to have a mean of 0 and a standard deviation of 1. This process is part of the feature scaling or normalization step, which aims to bring all the features to a similar scale. StandardScaler operates on each feature independently, ensuring that they have the same scale but preserving their relative relationships [38].

3.3.3. The Proposed Model

In this research, a model was developed to classify malware and benign files. Keras libraries in the Python programming language were relied upon to design the proposed model. The proposed ANN includes four dense layers. Figure 3 shows the steps for training the proposed model using few-shot learning. Figure 4 shows the steps for evaluating the proposed model using few-shot learning.

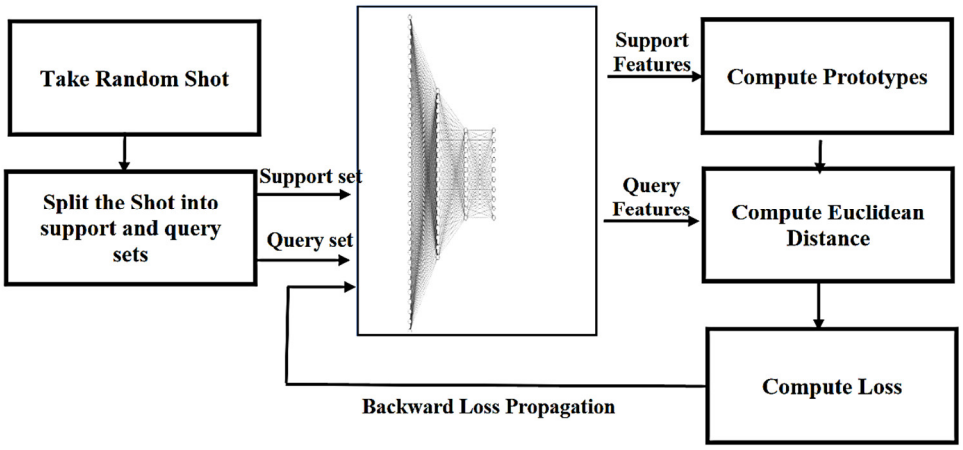

Figure 3. The steps for training the proposed model using few-shot learning.

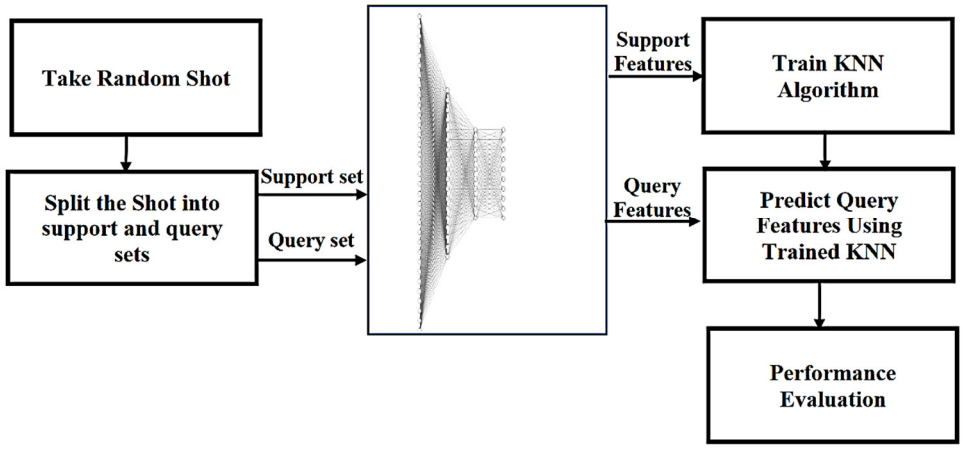

Figure 4. The steps for evaluating the proposed model using few-shot learning.

Figure 3 illustrates the steps of training the proposed model using the few-shot learning technique, where a shot is initially taken from the dataset and the size of the shot is determined by the user so that each shot contains an equal number of each category. For example, if the size of the shot is set to 5, 5 samples are taken from the dataset belonging to the malware category and 5 samples belonging to the benign category are taken. After this, the shot is randomly divided into two groups: the first group is called the support set and the second group is called the query set, where the support set is relied upon to predict the classification of the query set. Each of the two groups is entered into the artificial neural network, which works to extract the features so that the artificial neural network produces two groups: support features and query features. After that, the support features are used to calculate the prototypes, where the samples belonging to the malware category are taken from the support feature group, the arithmetic mean is calculated for each feature, and then the result is adopted as a reference for the malware category. Likewise, for the benign category, the samples belonging to the benign category are taken from the support feature group, the arithmetic mean is calculated for each feature, and then the result is adopted

as a reference for the benign category. The Euclidean distance between each sample from the query features and the reference prototype for each class is then calculated, and the sample is assigned to the class with the smallest Euclidean distance. The predicted result is then compared with the actual result in the dataset, and the loss function is calculated and back-propagated to the input of the artificial neural network to adjust the weights based on the loss. A new shot is then taken from the dataset, and the previous operations are repeated until a certain number of iterations, which are specified by the user, are completed.

Figure 4 illustrates the steps of evaluating the proposed model. First, a shot is taken from the dataset. Then, the shot is randomly divided into two sets, the support set and the query set. Each set is fed into the trained ANN model, which is established in the steps shown in Figure 3. The trained ANN model extracts features so that the ANN produces two sets, support features and query features. Then, the KNN algorithm is trained on the support feature set, and the trained KNN algorithm is used to predict the classification of the samples within the query feature set. Then, the predicted value is compared with the actual value, and the model is evaluated. Then, a new shot is taken from the dataset, and the previous processes are repeated until a certain number of iterations, which are specified by the user, are completed.

3.3.4. Performance Evaluation

Within the realm of machine learning, the confusion matrix (CM), which refers to an error matrix [39], is a table layout designed to represent the performance of a model, often one employed in supervised learning [40]. The architecture of the CM is illustrated in Figure 5.

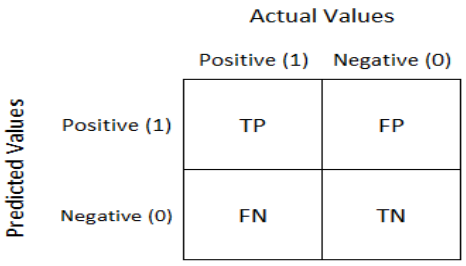

Figure 5. Confusion matrix architecture.

The following apply in the above:

- *TP* (true positive): The model classifies the sample as positive and its classification is correct.
- *TN* (true negative): The model classifies the sample as negative and its classification is correct.
- *FP* (false positive): The model classifies the sample as positive and its classification is wrong.
- *FN* (false negative): The model classifies the sample as negative and its classification is wrong.

These parameters are utilized to calculate "*Recall*", "*Precision*", and "*Accuracy*".

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

The "*Recall*" equation represents how many samples are correctly predicted of all positive samples [41].

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

The "*Precision*" equation represents how many samples are actually positive of all the samples that are predicted as positive [42].

Accuracy represents the proportion of correct predictions of the model over all the predictions [43]. Its equation is given as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

"$F1_{Score}$" is a metric that combines "precision" and "recall", providing an overall assessment of a model's performance. Its equation is gives as follows:

$$F1_{Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

4. Results and Discussion

The results of the proposed methodology were compared with decision tree (DT), random forest (RF), logistic regression (LR), and Support Vector Machines (SVMs).

4.1. Results of Dataset 1 (the Malware Dataset)

The following, Figure 6, represents the CMs for the machine learning algorithms.

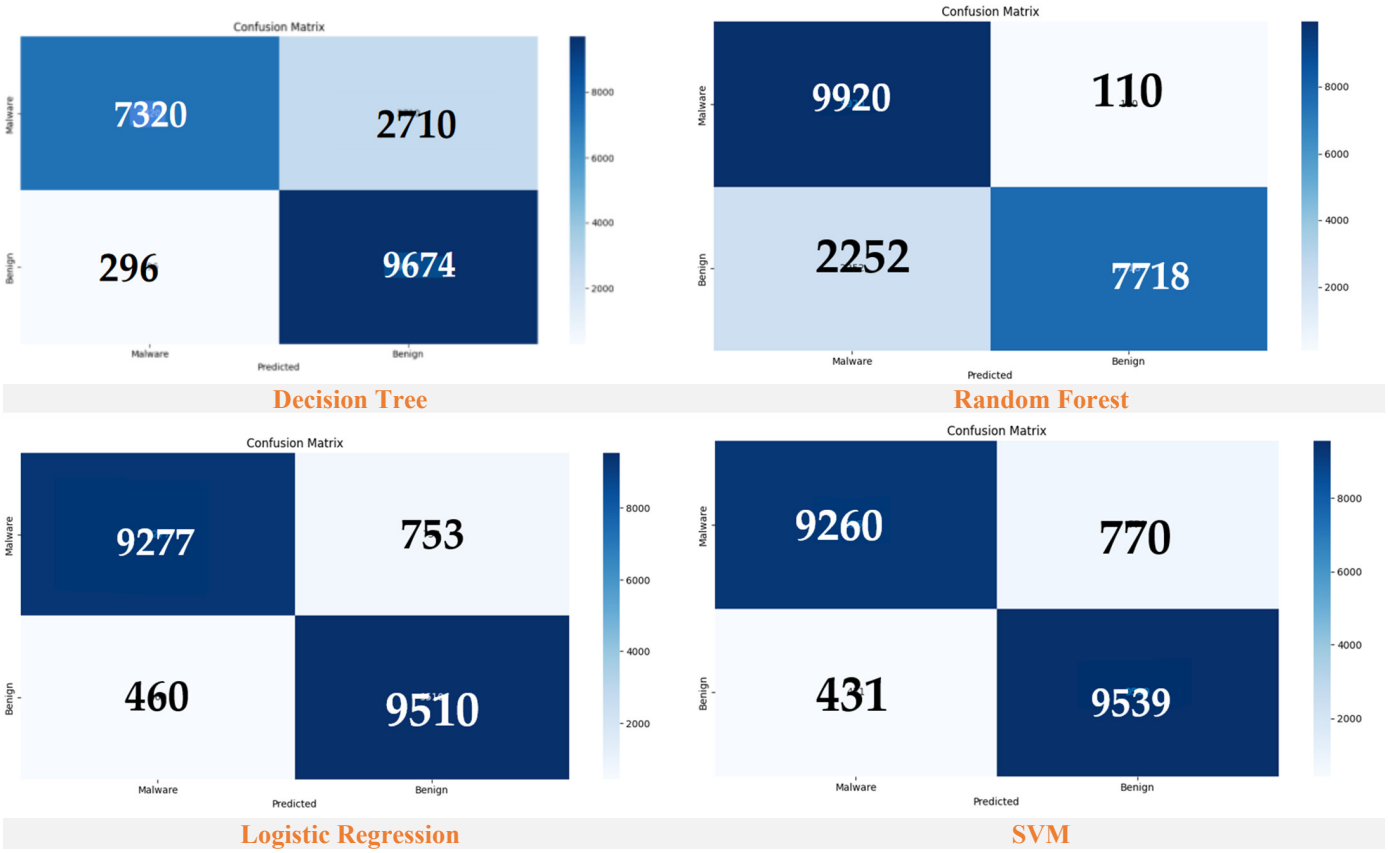

Figure 6. The CMs of the machine learning algorithms (DATASET 1).

The following, Table 2, shows the values of precision, recall, and f1_Score for each category for machine learning algorithms.

Table 2. The values of precision, recall, and f1_Score for each category for machine learning algorithms (DATASET 1).

Algorithm	Category	Precision	Recall	F1_Score
Decision Tree	Malware	0.96	0.73	0.83
	Benign	0.78	0.97	0.87
Random Forest	Malware	0.81	0.99	0.89
	Benign	0.99	0.77	0.87
Logistic Regression	Malware	0.95	0.92	0.94
	Benign	0.93	0.95	0.94
SVM	Malware	0.96	0.92	0.94
	Benign	0.93	0.96	0.94

We can see the following from Table 2:

Decision tree:

The precision value for the malware category was 0.96, which means that 96% of the files predicted as malware were actually malware and the model was very good at reducing false positives when predicting malware. We can also see that the precision value for the benign category was 0.78, which means that 78% of the files predicted as benign were actually benign. This relatively low precision indicates that there are more false positives when predicting benign cases than malware cases, which is a problem in practical applications, as a file may be predicted as benign but in fact be a malware file, which could lead to a violation of users' privacy or significant financial losses. The recall value for the malware category was 0.73, which means that the model correctly identified 73% of the actual malware files and the model missed 27% of the malware files (false negatives). The recall value for the benign class was 0.97, which means that the model correctly identified 97% of the samples that belonged to the benign class. The f1_Score value for the malware class is 0.83, which means that the decision tree algorithm achieves a good balance between precision and recall for malware but needs some improvement, especially in terms of recall. The f1_Score value for the benign class is 0.87, which means that the model performs well in recognizing benign files, with a slightly higher F1 score compared to malware.

Random forest:

The precision value for the malware class was 0.81, which means that 81% of the files predicted as malware were actually malware. This is lower than the precision of the decision tree model (0.96), indicating a higher number of false positives with the random forest algorithm. The precision value for the benign class was 0.99, which means that 99% of the files predicted as benign were actually benign, thus marking a significant improvement in precision compared to the decision tree (0.78). The recall value for the malware class was 0.99, which means that the random forest classifier correctly identified 99% of the actual malware files; thus, the random forest algorithm achieved a significant improvement over the recall of the decision tree algorithm for malware (0.73), indicating very few false negatives. The recall value for the benign class was 0.77, which means that the model correctly identified 77% of the actual benign files. This is lower than the recall of the previous model for benign cases (0.97), indicating more false negatives. The F1 score for malware is higher (0.89) compared to the F1 score for the decision tree algorithm (0.83), which gives a better balance between precision and recall. The F1 score for benign cases is the same (0.87) for both the decision tree and random forest, indicating a balanced performance.

Logistic regression:

The precision value for the malware class was 0.95, which means that 95% of the files predicted as malware were actually malware. This is an improvement over the random forest (0.81) and is close to the performance of the decision tree model (0.96). The precision

value for the benign class was 0.93, which means that 93% of the files predicted as benign were actually in the benign class. This is an improvement over the decision tree model (0.78) but slightly lower than the results of random forest (0.99). The recall value for the malware class was 0.92, which means that the logistic regression classifier correctly identified 92% of the actual malware files. This is better than the result of the decision tree model (0.73) but slightly lower than that of the random forest (0.99). The recall value for the benign class was 0.95, which means that the model correctly identified 95% of the actual malware files. This is slightly lower than the results of the decision tree model (0.97) but much better than those of the random forest (0.77). The F1 score for malware (0.94) is higher than that of both the decision tree model (0.83) and the random forest (0.89), indicating an excellent balance between precision and recall. Similarly, the F1 score for benign cases (0.94) is higher than that of both the decision tree model (0.87) and the random forest (0.87), indicating a more balanced performance than the previous classifiers.

SVM:

The precision value for the malware class was 0.96, which means that 96% of the files predicted to be malware were actually malware. This is slightly higher than that of the logistic regression model (0.95), better than that of the random forest (0.81), and similar to that of the decision tree model (0.96). The precision value for the benign class was 0.93, which means that 93% of the files predicted to be malware were actually malware. This is similar to that of the logistic regression model (0.93) and better than that of the decision tree model (0.78) but slightly lower than that of the random forest (0.99). The recall value for the malware class was 0.92, which means that the SVM correctly identified 92% of the actual malware files. This value is the same as the recall value for the logistic regression model (0.92) and better than that of the decision tree model (0.73) but slightly lower than that of the random forest (0.99). The recall value for the benign class was 0.96, which means that the SVM correctly identified 96% of the actual benign files. This value is slightly better than that of the logistic regression model (0.95), slightly lower than that of the decision tree model (0.97), and significantly better than that of the random forest (0.77). The F1 score value is 0.94 for both classes, indicating a strong balance between precision and recall, similar to the logistic regression model.

For our proposed model, the model was tested and experimented with on a set of shots, and the improvement in the performance metrics was monitored until we obtained the best performance at a value of shot = 1024. After this value, the performance remained constant without change. Therefore, a shot value = 1024 was adopted. Figure 7 shows the change in the values of the accuracy as the number of shots increased.

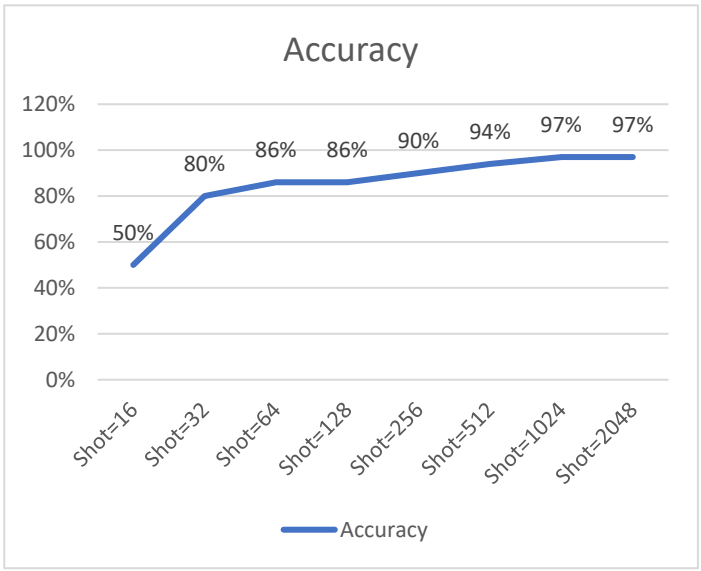

Figure 7. The change in the accuracy values as the number of shots increased (DATASET 1).

Figure 8 represents the CMs of the proposed model for each shot.

Figure 8. The CMs of the proposed model for each shot (DATASET 1).

Table 3 shows the values of precision, recall, and f1_Score for each category for the proposed model.

Table 3. The values of precision, recall, and f1_Score for each category for the proposed model (DATASET 1).

	Precision	Recall	F1_Score
Malware	0.97	0.97	0.97
Benign	0.97	0.97	0.97

The results of the proposed model indicate the excellent performance of the proposed model, as it achieved high scores across all metrics for both categories. The proposed model achieved a precision value of 0.97 for the malware category, which means that 97% of the samples predicted as malware were actually malware. This result is slightly better than those of the SVM (0.96) and logistic regression (0.95) models and significantly better than those of the random forest (0.81) and decision tree (0.96). The proposed model also achieved a precision value of 0.97 for the benign category, which means that 97% of the samples predicted as benign were actually benign. This value represents an improvement over logistic regression (0.93), SVM (0.93), and decision tree (0.78) but a slightly lower result than that of the random forest (0.99). The proposed model achieved a recall value of 0.97 for the malware class. This means that the proposed model correctly identified 97% of the actual malware files. This is a better result than that of SVM (0.92) and logistic regression (0.92), significantly better than that of decision tree (0.73), and slightly lower than random forest (0.99). The proposed model achieved a recall value of 0.97 for the benign class. This means that the model correctly identified 97% of the actual malware files. This is slightly better than the results of SVM (0.96) and logistic regression (0.95), similar to those of the decision tree (0.97), and significantly better than those of the random forest (0.77). The proposed model achieved an F1 score of 0.97 for both classes, indicating an ideal balance between precision and recall, which was better than that of all other models. Thus, it can be said that the proposed model shows an ideal performance for all performance metrics, making it a reliable choice for the task of classifying files as benign or malicious.

The results of comparing all of the previous results and calculating the accuracy values for each algorithm are shown in Figure 9.

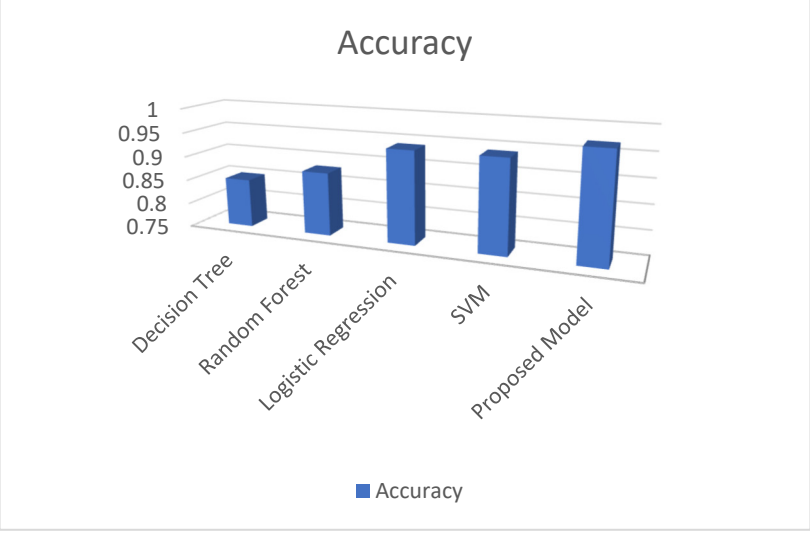

Figure 9. The accuracy values for all models (DATASET 1).

From Figure 9, we can see the following:

(1) Decision tree: An accuracy of 0.85 indicates moderate performance. Decision trees often have simpler structures and may suffer from overfitting or underfitting depending on their depth and may not capture complex patterns as effectively as more complex models, which explains their low accuracy compared to other models.
(2) Random forest: The accuracy improved compared to that of the decision tree, achieving an accuracy of 0.88, because random forests combine the results of multiple decision trees to improve the generalization ability of the model.
(3) Logistic regression and SVM: Both models achieved an accuracy of 0.94 and thus a high performance, as these models are able to handle linear discontinuities and feature engineering effectively.
(4) The proposed model: The highest accuracy of 0.97 indicates that this model not only handles basic patterns in the data but also handles more complex and subtle distinctions between classes effectively. The impressive performance of the proposed model is the result of a sophisticated hybrid approach that combines an artificial neural network (ANN) and a K-nearest neighbors (KNN) algorithm, both trained using few-shot learning techniques. By combining KNN and the ANN, the model leverages the strengths of both approaches: the ability of the ANN to learn complex high-level features and the simplicity and effectiveness of KNN in making final predictions based on proximity in the feature space.

Why does the proposed model outperform?

(1) Feature transformation by the ANN: Dense ANN layers transform raw features into a higher-dimensional space where important patterns can be more easily separated. This transformation ensures that when KNN is applied, the nearest neighbors are selected based on these well-extracted and relevant features, resulting in more accurate classifications.
(2) Augmented generalization using few-shot learning: Few-shot learning helps the ANN and KNN to generalize well from limited data, reducing the risk of overfitting and improving the robustness of the model.

4.2. Results of Dataset 2 (the TUANDROMD Dataset)

The following Figure 10 represents the CMs of the machine learning algorithms.

The following, Table 4, shows the values of precision, recall, and f1_Score for each category for the machine learning algorithms.

Table 4. The values of precision, recall, and f1_Score for each category for the machine learning algorithms (DATASET 2).

Algorithm	Category	Precision	Recall	F1_Score
Decision Tree	Malware	0.75	0.92	0.82
	Benign	0.91	0.71	0.80
Random Forest	Malware	0.90	0.95	0.92
	Benign	0.95	0.90	0.93
Logistic Regression	Malware	0.92	0.86	0.89
	Benign	0.88	0.93	0.90
SVM	Malware	0.93	0.88	0.90
	Benign	0.90	0.94	0.92

Figure 10. The CMs of the machine learning algorithms (DATASET 2).

For our proposed model, the model was tested and experimented with on a set of shots, and the improvement in the performance metrics was monitored until we obtained the best performance at a shot value = 71. After this value, the performance remained constant without change. Therefore, a shot value = 71 was adopted. The shot value = 71 was tested to verify the result we reached through the experiment on the first dataset, where the number of samples in the first dataset was 100,000 and the best accuracy was obtained at a shot value = 1024, which was approximately 1% of the total number of samples in the dataset. The number of samples in the second dataset after applying the oversampling technique was 7130 samples, so a shot value = 71 was tested, which was 1% of the total number of samples in the second dataset, and the best accuracy was achieved, which is 97%, and after that, the accuracy remained the same at a shot value = 128 and a shot value = 256. Figure 11 shows the change in the accuracy values as the number of shots increased.

The following, Table 5, shows the values of precision, recall, and f1_Score for each category for the proposed model at a shot value = 71.

Table 5. The values of precision, recall, and f1_Score for each category for the proposed model at a shot value = 71 (DATASET 2).

	Precision	Recall	F1_Score
Malware	0.97	0.97	0.97
Benign	0.97	0.97	0.97

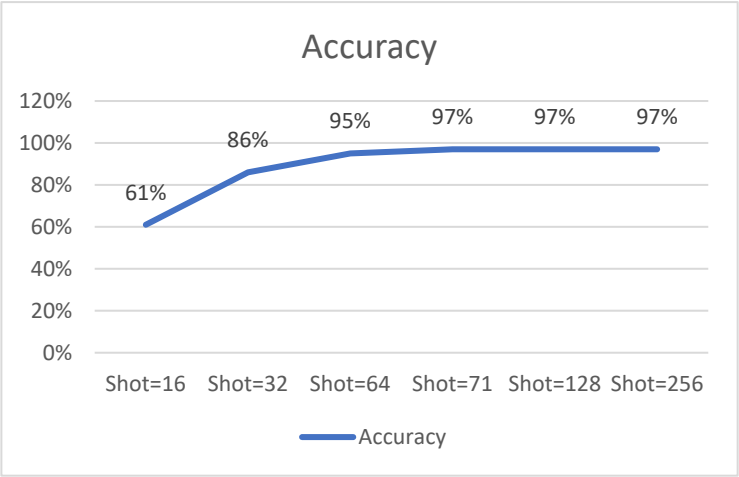

Figure 11. The change in the accuracy values as the number of shots increased (DATASET 2).

The results of the proposed model shown in Table 5 indicate an excellent and stable performance of the proposed model, as it achieved high scores in all measures for both categories and maintained the stability of its performance and results despite using a different dataset from the first dataset. The results of comparing all of the previous results and calculating the accuracy values for each algorithm are shown in Figure 12.

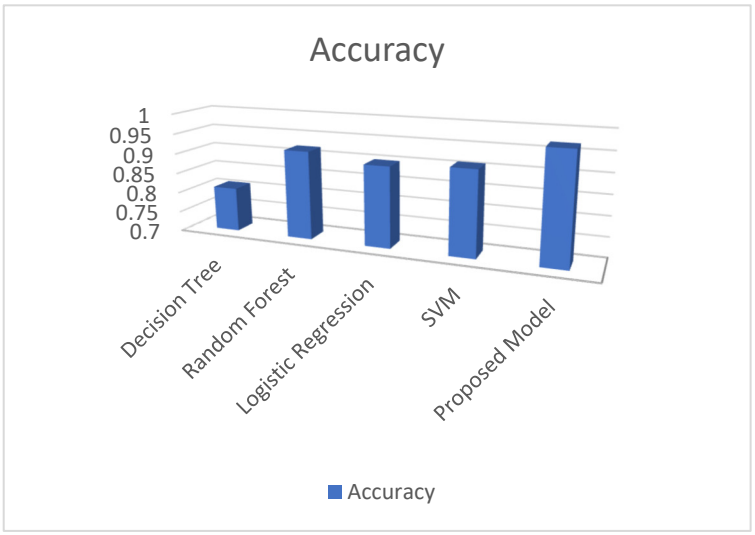

Figure 12. The accuracy values for all models on the second dataset (DATASET 2).

5. Conclusions

Few-shot learning methods, such as model networks, are particularly useful for increasing malware detection due to their ability to learn efficiently from a limited amount of labeled data. This is particularly important in the field of cybersecurity, where new and unseen malware variants emerge frequently, making it difficult to collect large, labeled datasets for training traditional machine learning models. They can also quickly adapt to new malware, thus improving their generalizability. Additionally, the simplicity and efficiency of these networks make them efficient and easy to train compared to more complex models. A comparative analysis of multiple classification models across two datasets reveals important insights into their performance and suitability for identifying malware and benign files. The proposed model, which included a hybrid artificial neural network with KNN and used few-shot learning, consistently outperformed all the other models, with the highest accuracy (0.97), balanced precision, recall, and F1 scores across both datasets. This demonstrates its robustness and ability to generalize well across diverse data complexities.

Random forest and the SVM also showed strong performance, with high accuracy (0.92 and 0.88 for random forest) and (0.91 and 0.94 for the SVM), making them reliable alternatives for scenarios that require a balance between interpretability and computational efficiency. Logistic regression, which achieved accuracy values of 0.90 and 0.94, remained a strong choice due to its efficiency and relatively high performance. However, the decision tree showed lower accuracy (0.81, 0.85) and unbalanced metrics, suggesting that it struggles with more complex datasets.

Future work should focus on improving the proposed model to enhance its performance and generalizability further, perhaps by exploring advanced neural network architectures or incorporating additional learning techniques. Continuous monitoring and periodic retraining of the deployed models will be essential for them to maintain high accuracy and adapt to new data patterns. This research is limited by the sensitivity to data quality, as few-shot learning models are very sensitive to the quality of the few training examples available. Noisy, misclassified, or unrepresentative samples can significantly impact performance, as the model has little data to cope with errors, so investigating the impact of feature engineering may lead to further improvements. Finally, evaluating the performance of the models on real-time data and in different operational environments will be crucial to ensure their practical applicability and robustness in real-world scenarios.

Author Contributions: Conceptualization, K.A., S.R. and I.A.; methodology, K.A., S.R. and I.A.; software, K.A. and S.R.; validation, K.A., S.R. and I.A.; formal analysis, K.A. and S.R.; investigation, K.A. and S.R.; resources, K.A., S.R. and I.A.; data curation, K.A., S.R. and I.A.; writing—original draft preparation, K.A; writing—review and editing, K.A.; visualization, K.A. and S.R.; supervision, S.R.; project administration, S.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data is available within the article.

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

References

1. Gopinath, M.; Sethuraman, S.C. A comprehensive survey on deep learning based malware detection techniques. *Comput. Sci. Rev.* **2023**, *47*, 100529.
2. Ryan, M. *Ransomware Revolution: The Rise of a Prodigious Cyber Threat (Vol. 85)*; Springer: Berlin/Heidelberg, Germany, 2021.
3. Wang, P.; Tang, Z.; Wang, J. A novel few-shot malware classification approach for unknown family recognition with multi-prototype modeling. *Comput. Secur.* **2021**, *106*, 102273. [CrossRef]
4. Thangaraj, M.; Sivakami, M. Text classification techniques: A literature review. *Interdiscip. J. Inf. Knowl. Manag.* **2018**, *13*, 117–135. [CrossRef] [PubMed]
5. Gorade, S.M.; Deo, A.; Purohit, P. A study of some data mining classification techniques. *Int. Res. J. Eng. Technol.* **2017**, *4*, 3112–3115.
6. Gupta, S.; Kumar, D.; Sharma, A. Data mining classification techniques applied for breast cancer diagnosis and prognosis. *Indian J. Comput. Sci. Eng. IJCSE* **2011**, *2*, 188–195.
7. Kruczkowski, M.; Szynkiewicz, E.N. Support Vector Machine for Malware Analysis and Classification. In Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, 11–14 August 2014; pp. 415–420. [CrossRef]
8. Choi, S. Combined kNN classification and hierarchical similarity hash for fast malware detection. *Appl. Sci.* **2020**, *10*, 5173. [CrossRef]
9. Aboaoja, F.A.; Zainal, A.; Ghaleb, F.A.; Al-Rimy, B.A.S.; Eisa, T.A.E.; Elnour, A.A.H. Malware detection issues, challenges, and future directions: A survey. *Appl. Sci.* **2022**, *12*, 8482. [CrossRef]
10. Aslan, O.; Samet, R. A comprehensive review on malware detection approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
11. Tang, Z.; Wang, P.; Wang, J. ConvProtoNet: Deep prototype induction towards better class representation for few-shot malware classification. *Appl. Sci.* **2020**, *10*, 2847. [CrossRef]
12. Gao, T.; Han, X.; Liu, Z.; Sun, M. Hybrid attention-based prototypical networks for noisy few-shot relation classification. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 6407–6414. [CrossRef]

13. Abusitta, A.; Li, M.Q.; Fung, B.C. Malware classification and composition analysis: A survey of recent developments. *J. Inf. Secur. Appl.* **2021**, *59*, 102828. [CrossRef]
14. Naseer, M.; Rusdi, J.F.; Shanono, N.M.; Salam, S.; Muslim, Z.B.; Abu, N.A.; Abadi, I. Malware detection: Issues and challenges. *J. Phys. Conf. Ser.* **2021**, *1807*, 012011. [CrossRef]
15. Yang, L.; Li, Y.; Wang, J.; Xiong, N.N. FSLM: An intelligent few-shot learning model based on Siamese networks for IoT technology. *IEEE Internet Things J.* **2020**, *8*, 9717–9729. [CrossRef]
16. Zhou, X.; Liang, W.; Shimizu, S.; Ma, J.; Jin, Q. Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems. *IEEE Trans. Ind. Inform.* **2021**, *17*, 5790–5798. [CrossRef]
17. Bedi, P.; Gupta, N.; Jindal, V. Siam-IDS: Handling class imbalance problem in intrusion detection systems using Siamese neural network. *Procedia Comput. Sci.* **2020**, *171*, 780–789. [CrossRef]
18. Zhou, X.; Hu, Y.; Liang, W.; Ma, J.; Jin, Q. Variational LSTM enhanced anomaly detection for industrial big data. *IEEE Trans. Ind. Inform.* **2021**, *17*, 3469–3477. [CrossRef]
19. Conti, M.; Khandhar, S.; Vinod, P. A few-shot malware classification approach for unknown family recognition using malware feature visualization. *Comput. Secur.* **2022**, *122*, 102887. [CrossRef]
20. Oprea, S.-V.; Bâra, A. Detecting Malicious Uniform Resource Locators Using an Applied Intelligence Framework. *Comput. Mater. Contin.* **2024**, *79*, 3827–3853. [CrossRef]
21. Oprea, S.-V.; Bâra, A. A Recommendation System for Prosumers Based on Large Language Models. *Sensors* **2024**, *24*, 3530. [CrossRef] [PubMed]
22. Rieck, K.; Holz, T.; Willems, C.; Düssel, P.; Laskov, P. Learning and classification of malware behavior. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Paris, France, 10–11 July 2008; Springer: Berlin/Heidelberg, Germany; pp. 108–125.
23. Zhu, J.; Jang-Jaccard, J.; Singh, A.; Welch, I.; Al-Sahaf, H.; Camtepe, S. A few-shot meta-learning based siamese neural network using entropy features for ransomware classification. *Comput. Secur.* **2022**, *117*, 102691. [CrossRef]
24. Jung, H.M.; Kim, K.-B.; Cho, H.-J. A study of Android malware detection techniques in virtual environment. *Clust. Comput.* **2016**, *19*, 2295–2304. [CrossRef]
25. Ye, H.-J.; Hu, H.; Zhan, D.-C.; Sha, F. Few-shot learning via embedding adaptation with set-to-set functions. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020.
26. Yoo, S.; Kim, S.; Kim, S.; Kang, B.B. Ai-Hydra: Advanced hybrid approach using random forest and Deep Learning for malware classification. *Inf. Sci.* **2021**, *546*, 420–435. [CrossRef]
27. Anderson, H.S.; Kharkar, A.; Filar, B.; Roth, P. Evading machine learning malware detection. *Black Hat* **2017**, *2017*, 1–6.
28. Goncalves, E.C.; Freitas, A.A.; Plastino, A. A survey of genetic algorithms for multi-label classification. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
29. Galván, E.; Mooney, P. Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Trans. Artif. Intell.* **2021**, *2*, 476–493. [CrossRef]
30. Thakur, A.; Konde, A. Fundamentals of neural networks. *Int. J. Res. Appl. Sci. Eng. Technol.* **2021**, *9*, 407–426. [CrossRef]
31. da Silveira Bohrer, J.; Grisci, B.I.; Dorn, M. Neuroevolution of neural network architectures using CoDeepNEAT and keras. *arXiv* **2020**, arXiv:2002.04634.
32. Islam, M.; Chen, G.; Jin, S. An overview of neural network. *Am. J. Neural Netw. Appl.* **2019**, *5*, 7–11. [CrossRef]
33. Fernández, A.; García, S.; Galar, M.; Prati, R.C.; Krawczyk, B.; Herrera, F. *Learning from Imbalanced Data Sets*; Springer: Cham, Switzerland, 2018; Volume 10.
34. Mohammed, A.J.; Hassan, M.M.; Kadir, D.H. Improving classification performance for a novel imbalanced medical dataset using SMOTE method. *Int. J. Adv. Trends Comput. Sci. Eng.* **2020**, *9*, 3161–3172. [CrossRef]
35. Chehal, D.; Gupta, P.; Gulati, P.; Gupta, T. Comparative Study of Missing Value Imputation Techniques on E-Commerce Product Ratings. *Informatica* **2023**, *47*, 373–382. [CrossRef]
36. Chakrabarti, S.; Biswas, N.; Karnani, K.; Padul, V.; Jones, L.D.; Kesari, S.; Ashili, S. Binned Data Provide Better Imputation of Missing Time Series Data from Wearables. *Sensors* **2023**, *23*, 1454. [CrossRef]
37. L'heureux, A.; Grolinger, K.; Elyamany, H.F.; Capretz, M.A. Machine learning with big data: Challenges and approaches. *IEEE Access* **2017**, *5*, 7776–7797. [CrossRef]
38. Misra, P.; Yadav, A.S. Impact of preprocessing methods on healthcare predictions. In Proceedings of the 2nd International Conference on Advanced Computing and Software Engineering (ICACSE), Sultanpur, India, 8–9 February 2019.
39. Haghighi, S.; Jasemi, M.; Hessabi, S.; Zolanvari, A. PyCM: Multiclass confusion matrix library in Python. *J. Open Source Softw.* **2018**, *3*, 729. [CrossRef]
40. Markoulidakis, I.; Kopsiaftis, G.; Rallis, I.; Georgoulas, I. Multi-class confusion matrix reduction method and its application on net promoter score classification problem. In Proceedings of the 14th PErvasive Technologies Related to Assistive Environments Conference, Corfu, Greece, 29 June–2 July 2021; pp. 412–419.
41. Roth, K.; Pemula, L.; Zepeda, J.; Schölkopf, B.; Brox, T.; Gehler, P. Towards total recall in industrial anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 14318–14328.

42. MacEachern, S.J.; Forkert, N.D. Machine learning for precision medicine. *Genome* **2021**, *64*, 416–425. [CrossRef]
43. Fu, G.; Sun, P.; Zhu, W.; Yang, J.; Cao, Y.; Yang, M.Y.; Cao, Y. A deep-learning-based approach for fast and robust steel surface defects classification. *Opt. Lasers Eng.* **2019**, *121*, 397–405. [CrossRef]