IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 June 2023, accepted 27 June 2023, date of publication 7 July 2023, date of current version 25 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3293117

RESEARCH ARTICLE

Detection of Data Scarce Malware Using One-Shot Learning With Relation Network

FAIZA BABAR KHAN[1], MUHAMMAD HANIF DURAD[1], ASIFULLAH KHAN[2,3], FARRUKH ASLAM KHAN[4], (Senior Member, IEEE), SAJJAD HUSSAIN CHAUHDARY[5], AND MOHAMMED ALQARNI[6]

[1]CIPMA Laboratory, DCIS, Pakistan Institute of Engineering and Applied Sciences, Islamabad 45650, Pakistan
[2]Pattern Recognition Laboratory, DCIS, PIEAS, Nilore, Islamabad 45650, Pakistan
[3]PIEAS Artificial Intelligence Center (PAIC), PIEAS, Nilore, Islamabad 45650, Pakistan
[4]Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh 11653, Saudi Arabia
[5]Department of Computer Science and Artificial Intelligence, College of Computer Science and Engineering, University of Jeddah, Jeddah 21959, Saudi Arabia
[6]Department of Software Engineering, College of Computer Science and Engineering, University of Jeddah, Jeddah 21959, Saudi Arabia

Corresponding author: Sajjad Hussain Chauhdary (shussain1@uj.edu.sa)

This work was supported by the Deputyship for Research & Innovation, Ministry of Education, Saudi Arabia, under Project MoE-IF-UJ-22-04101171-2.

ABSTRACT Malware has evolved to pose a major threat to information security. Efficient anti-malware software is essential in safeguarding confidential information from these threats. However, identifying malware continues to be a challenging task. Signature-based detection methods are quick but fail to detect unknown malware. Additionally, the traditional machine learning archetype requires a large amount of data to be effective, which hinders the ability of an anti-malware system to quickly learn about new threats with limited training samples. In a real-world setting, the majority of malware is found in the form of Portable Executable (PE) files. While there are various formats of PE files, samples of all formats such as ocx, acm, com, scr, etc., are not readily available in large numbers. Therefore, building a conventional Machine Learning (ML) model with greater generalization for data-scarce PE formats becomes a hefty task. Consequently, in such a scenario, Few-Shot learning (FSL) is helpful in detecting the presence of malware, even with a very small number of training samples. FSL techniques help to make predictions based on an insufficient number of samples. In this paper, we propose a novel architecture based on the Relation Network for FSL implementation. We propose a Discriminative Feature Embedder for feature extraction. These extracted features are passed to our proposed Relation Module (RM) for similarity measure. RM produces the relation scores that lead to improved classification. We use PE file formats, i.e., ocx, acm, com, and scr, after transforming them into images. We employ five-shot learning and then one-shot learning, which produces 94% accuracy with only one training instance. We observe that the proposed architecture outpaces the baseline method and provides enhanced accuracy by up to 94% with only one sample.

INDEX TERMS Data-scarce malware, feature embedding, meta-learning, one-shot learning, relation network.

I. INTRODUCTION

Malware is a fast-spreading epidemic in the current age of the Internet. Malware includes all programs that can cause a computer system crash, data leakage, unlawful incursion, resource damage, etc [1]. According to the latest studies, it is increasing at a distressing rate and thus can be considered lethal for the health of the confidential data. Kaspersky's detection systems detected an average of 400,000 malicious file distributions per day in the year 2022. In 2021,

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

daily 380,000 of these files were detected, signifying a 5% growth in comparison to 2021. To combat the increase in malware attacks, there is a vital need to develop an intelligent and efficient malware detection system [2], [3].

Most malware detection systems employ machine learning methods such as support vector machines, decision trees, and logistic regression. Nonetheless, these models necessitate a significant amount of data and careful management to ensure effective information representation [4]. When the available dataset is limited, these techniques become constrained [5]. Malware is grouped into different categories like PE, ZIP, XHTML, RTF, JPEG image, and Windows registry, with each category scrutinized to detect the malware and its variants. However, some groups may have an insufficient number of training examples [6], resulting in overfitting and a reduction in the models' generalization ability when dealing with rare types of malware [7], [8].

To address this problem, the idea of Few-Shot Learning (FSL) is presented. FSL is capable of conducting classification using only a few samples and generalizing rapidly to new tasks with supervised information. In few-shot learning, ''few'' generally refers to the number of labeled examples (samples) available per class or task for training the model. The exact number can vary depending on the particular experiment or task, but it typically refers to five or fewer samples per class. For our proposed approach, the term ''few'' refers to a minimum of one sample. FSL refers to the ability to generalize to new classes during training without having seen them before [9]. Since the time the idea of FSL was presented, it has emerged as an effective technique to identify unseen classes during training with limited labeled examples. One of the classic FSL scenarios is the detection of new malware where training examples with supervised information is very difficult to acquire due to smart malware developers. A minor change in malware code can change the hash of malware and for rare malware, several real records are not available at the training time. As a result, obtaining efficient learning from a restricted number of samples is critical. With FSL, the model is able to gather widespread and comprehensive information about malware through a succession of resembling tasks, and then extract task-bound information from a small number of samples to adapt to previously unseen classes during training [10]. By utilizing FSL, the challenge of limited data is overcome and the manual labeling of malware becomes less tedious. The implementation of FSL allows for the development of effective models for rare cases of malware.

Depending upon the number of instances per class in the training set, N-shot learning is called zero-shot learning (ZSL), one-shot learning (OSL), and few-shot learning. In the past, many one-shot classification algorithms have been used in different areas, i.e., learning visual models of object categories [11], semantic segmentation [12], drug discovery [13], etc. However, employing existing few-shot models in malware classification is somewhat tricky. Initially, FSL was implemented using Matching Networks, Siamese neural networks, and Prototype Networks. These techniques used embedding-based methods to modify the data to make it recognizable using some similarity measures. Relation network used the same concept with a learnable nonlinear comparison module, instead of a fixed linear comparator. But the shortcoming is that they adopted a hereditary architecture of embedding function known as Conv-4, which extracted a smaller number of features. It can lead to the misidentification of unseen malware, as important information may be lost. The vanishing gradient problem, which occurs when the gradients of the parameters during training become very small, can also slow down the learning process by making it difficult for the model to converge on a solution.

This paper presents a new architecture called ResRelNet (RRN) that addresses feature extraction and the vanishing gradient problem in malware few-shot classification. The RRN architecture improves upon the hereditary structure of convNet by incorporating residual connections, inspired by Residual Neural Networks, which allow for a smoother flow of information across multiple layers. This design pattern effectively addresses the problem of vanishing gradients in deep networks and leads to the refined extraction of effective and high-level PE features. Moreover, the proposed model is trained on a series of ''episodes,'' where each episode represents a few-shot learning task. The episodic nature of training helps the model learn to recognize common visual patterns and features that are relevant for the given tasks. The unique combination of transfer learning from ResNet, episodic training, and the focus on addressing FSL limitations distinguishes our proposed framework and establishes its novelty in the field. The results are compared with the baseline method and the experiments are performed in different phases using different graphs to highlight the factors that enhance accuracy. The contributions of this paper are:

1) The baseline method for native and malware datasets is implemented.
2) The results for five-shot and one-shot learning using the baseline method are reported.
3) The problem of detecting data-scarce malware is addressed using the proposed relation network.
4) The results of the proposed method are compared with baseline methods.
5) Future research directions in this domain are presented.

The remainder of the paper is structured as follows: Section II narrates the motivation behind this research. Section III gives the preliminary details of the techniques used in this study. Section IV covers previous studies related to FSL and relation network. Section V outlines the proposed approach. Section VI details the implementation of the proposed architecture. Section VII presents the results and analysis. Finally, the conclusion and future work are discussed in the last section.

II. MOTIVATION

The motivation for this research was the necessity to identify malware even in cases where there is only one available training data point. Various methods have been suggested in

TABLE 1. List of Acronyms.

Acronym	Definition
PE	Portable Executable
ML	Machine Learning
CNN	Convolutional Neural Network
DFE	Discriminative Feature Embedder
RM	Relation Module
FSL	Few-Shot Learning
OSL	One-Shot Learning
ZSL	Zero-Shot Learning
RRN	ResRelNet
RN	Relation Network
OCX	OLE Control EXtension
ACM	Audio Compression Manager
SCR	Screensaver File
COFF	Common Object File Format
EXE	Executable
DLL	Dynamic Link Libraries

the past to deal with the issue of inadequate training data. These techniques include ensemble learning, regularization, and data augmentation. For instance, the augmentation of a small number of samples was used in a study to forecast oil condition [14], while another study [15] employed data augmentation for small-sized datasets. However, a potential disadvantage of data augmentation is that it may introduce impractical or unrelated changes in the training data, which could harm the model's performance. Additionally, if not applied correctly, it can raise the risk of overfitting and be computationally expensive.

To address the issue of small datasets, Transfer Learning has been utilized, which refers to the process of using a pre-existing trained model as a foundation to train a new model with a smaller dataset. The pre-trained model has already acquired beneficial representations of the data, which can be adjusted on the new dataset to enhance the performance. Previous researchers, such as Brodzicki et al. [16] and Yamada et al. [17], have employed Transfer Learning to tackle this challenge. Additionally, Shen et al. [18] suggested to utilize ensemble learning and transfer learning techniques with deep convolutional neural networks in order to make an estimation of the capacity of lithium-ion batteries. However, a potential disadvantage of Transfer Learning is that it can require considerable computational resources and time to fine-tune the pre-trained model on the target dataset, particularly if the target dataset differs significantly from the pre-trained dataset. Regularization is another technique that involves adding constraints to the model during training to prevent overfitting, which is a common problem when training models on small datasets. Techniques such as L1 or L2 regularization, dropout, or early stopping can be used to improve the generalization of the model, but sometimes these techniques can be too strict and cause the model to underfit the data, resulting in poor performance. This can be problematic, especially if the dataset is small, as there may not be sufficient data to adequately capture the complexity of the underlying patterns and relationships.

While every technique has its limitations, FSL stands out as a promising approach for building machine learning models with small datasets. Compared to other techniques mentioned above, FSL offers several benefits, including its ability to efficiently train models with high accuracy, generalize well to new examples, and adapt to new tasks and domains. As a result, this study is motivated by the potential of FSL to address the challenges of building machine learning models [19] with limited data.

We started our implementation with a 5-shot approach in this study, and later extended it to include a 1-shot scenario. By conducting experiments with both 5-shot and 1-shot approaches, we were able to compare the model's performance on both tasks, which allowed us to evaluate its effectiveness and identify potential areas for improvement. Furthermore, we adopted an incremental learning approach, where we added new challenges over time. The transition from using five training samples to just one was a test of our model's performance. Our aim was to achieve high accuracy while using the minimum number of samples possible, to simulate real-world scenarios. The increase in accuracy demonstrated that our model has the potential to perform well in scenarios where there is limited training data available. By performing experiments with both 5-shot and 1-shot approaches, we were able to test how well our proposed model can generalize to new tasks with minimal training data.

III. PRELIMINARIES

The proposed approach is a novel method that presents an efficient Feature Embedder and Relation Module as the foundation of the relation network for optimal feature extraction and effective similarity measure. The formal definition of methodologies along with the structure of the PE file is presented in this section. An overview of meta-learning and relation network will be presented in the subsections.

A. PORTABLE EXECUTABLE (PE)

The Portable Executable (PE) file format is used for various types of files on Windows operating systems, such as executables (.exe), dynamic link libraries (.dll), ActiveX controls (.ocx), kernel modules (.srv), control panel applications, and screen saver file (.scr). It is built on the Common Object File Format (COFF) and contains the necessary information for the operating system to load the file into memory and run it. PE structure consists of the following parts, as shown in Figure 1:

1) DOS HEADER

The beginning of every PE file consists of a 64-byte structure known as the DOS header, which is what makes a PE file a valid MS-DOS executable.

2) DOS STUB

The DOS stub, a small MS-DOS 2.0 compatible executable, follows the DOS header and displays an error message

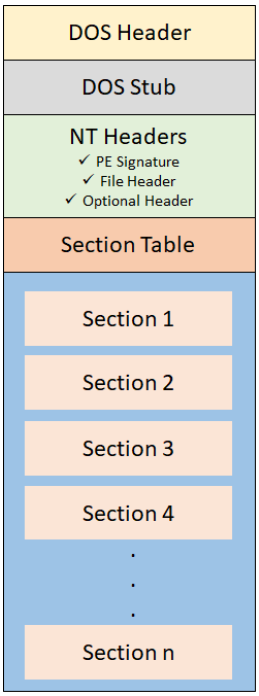

FIGURE 1. Structure of Portable Executable (PE).

indicating that the program cannot be executed in DOS mode if it is run in that environment.

3) NT HEADER

The NT Header contains the PE signature, File Header, and Optional Header.

4) SECTION TABLE

The array of Image Section Headers, also known as the section table, is located immediately after the Optional Header in a PE file. Each header in the table contains information about a specific section within the file.

5) SECTION

The sections of a file contain the actual contents, including data, resources, and code used by the program. Each section has a specific purpose.

PE file formats with a limited number of available training samples include .ocx, .acm, .com, and .scr. When the number of training instances is limited, the model may not generalize well and may become overfitted, resulting in poor performance on new data. In this research, we employed one-shot learning using these data-scarce malicious PE types.

B. META-LEARNING: LEARNING TO LEARN

Machine learning algorithms are made-up to learn from former experiences. Learning focuses to improve by getting experience at a certain task. Human learning is based on the integration of knowledge gained from different domains. Machine learning standards reflect the same process by making use of prior knowledge for new tasks, which triggers the learning process and gives a better generalization of new tasks [20]. These learning paradigms include Transfer Learning, Embedding Learning, Meta Learning, etc.

This paper presents a meta-learning approach for detecting low-data malware. Meta-learning, also known as "Learning to Learn," uses the insights gained from prior learning experiences across a variety of related tasks to improve future learning efficiency [21], [22]. This approach requires minimal data and computation compared to traditional machine learning algorithms. It adjusts various aspects of these algorithms to improve results using the knowledge acquired through multiple training sessions for various tasks. Meta-learning is particularly useful in few-shot learning scenarios, where only a limited number of examples are available for supervised learning. In these cases, conventional training methods often result in overfitting and poor generalization of test samples. Although data augmentation can help, it does not provide a guarantee of a solution. Overall, meta-learning offers unique advantages in handling few-shot learning, e.g., adapting to new tasks or domains, leveraging transfer learning, handling domain shift, enabling fast learning, and optimizing the learning process.

Meta-learning, similar to conventional supervised learning, employs different sets to assess and adjust model efficiency. These sets, referred to as meta-validation and meta-test, are organized into episodes with support and query sets. The critical distinction is that the class categories are partitioned among the meta-training, validation, and test sets to ensure that they do not intersect.

Meta-learning process can be framed in two phases, i.e., Meta-learning and Adaptation, as shown in Figure 2. Throughout the meta-learning stage, the model gradually learns a basic set of parameters that apply to all tasks; and in the adaptation phase, rapid acquisition of knowledge is performed to learn task-specific parameters [9].

During the training process, the model goes through a series of datasets, each consisting of multiple episodes. During meta-training, the model, referred to as meta-learner, begins by using the first episode's training set (also known as the support set) to create a model. This model is then used to make predictions on the test set (also known as the query set) of the same episode. The goal of meta-learning is to decrease a loss function, such as cross-entropy, based on examples in the test or query set. To achieve this, the errors are transmitted backward and utilized to modify the meta-parameters of the meta-learner, improving its performance. The model then moves on to the next episode, continuously training on the support set examples and making predictions on the query set. The process is repeated for each episode to enable frequent updates of the meta-parameters, with the ultimate goal of developing a meta-learner that possesses a robust ability to generalize. The examples in the test or query set are not encountered during the training phase, meaning that the meta-learner must learn to make predictions on unfamiliar data.

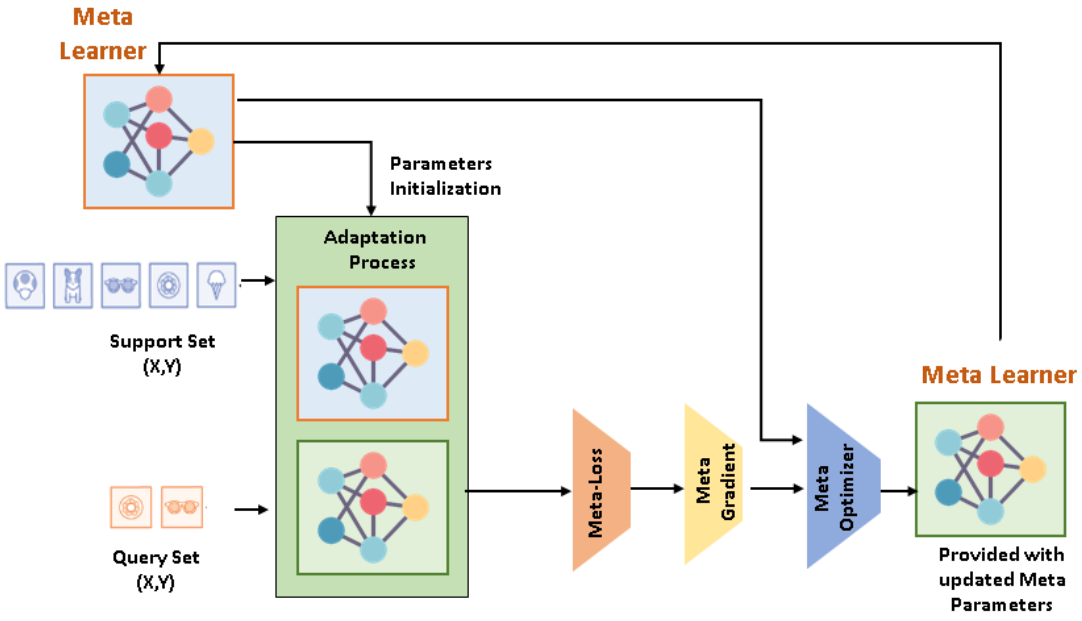

FIGURE 2. Process of Meta-Learning.

Here is a high-level overview of how the meta-learning process works:

1) First, the algorithm is given a set of tasks to learn from. Each task consists of a training set and a validation set comprising Support and Query sets.
2) The meta-learner model is initialized with some parameters, and trained on each task's training set.
3) After training on the training set, the algorithm evaluates the model on the validation set of each task and computes the loss.
4) After the computation of loss and gradients, the optimizer then updates the model's parameters using backpropagation. This update is designed to make the model generalize well to new tasks.
5) Finally, the algorithm repeats steps 2-4 for a few iterations, allowing the model to gradually adapt to the different tasks' characteristics.
6) Once the model is trained on the set of tasks, it can be used to quickly learn new tasks with only a few examples.

During the adaptation phase, the meta-learner adapts to task-specific parameters through a process known as parameter adaptation. The meta-learner learns from a set of tasks during the meta-training phase and generalizes this knowledge to adapt to new tasks during the adaptation phase. To adapt to task-specific parameters, the meta-learner utilizes the information provided by the task-specific data. It updates its internal parameters based on the observed input-output pairs of the new task. This adaptation process allows the meta-learner to fine-tune its parameters specifically for the new task at hand. The adaptation can be performed using various techniques, such as gradient-based optimization methods like stochastic gradient descent (SGD) or variants of SGD, where the meta-learner updates its parameters based on the gradients computed from the new task data. These gradients provide information about how the current parameters should be adjusted to minimize the task-specific loss or maximize the task-specific objective.

Different approaches have been proposed depending upon the adaptation phase. Meta-learning can be categorized into three main approaches:

1) Model-based
2) Optimization-based
3) Metric-based

Model-based meta-learning models depend on a strategy in which the model updates its parameters quickly with a few training steps. This update in a rapid way is attained by its inner network layout or organized by another meta-learner model. Optimization-based strategies handle the adaptation phase of the meta-learning process as an optimization problem. Metric-based approaches typically utilize non-parametric techniques for learning like k-nearest neighbors, k-means clustering, and kernel density estimation. The fundamental idea is to learn a feature space for the reconstruction of raw inputs into a depiction that allows similarity matching between the support set and the query set. Therefore, success is dependent on the selected similarity metric, e.g., cosine similarity or euclidean distance.

In metric-based learning, the forecasted likelihood, $P_\theta(y|\mathbf{x}, S)$, is determined by evaluating the weighted aggregate of the labels of a set of acknowledged labels y using a kernel function k_θ over parameters $_\theta$. The weight is determined based on the similarity between two data samples,

as determined by the kernel function (1).

$$P_\theta(y|\mathbf{x}, S) = \sum_{(\mathbf{x}_i, y_i)\in S} k_\theta(\mathbf{x}, \mathbf{x}_i) y_i \quad (1)$$

The definition of a good metric is specific to the problem at hand. It should demonstrate the connection between the inputs in the task space and aid in solving the problem.

A variety of work has been done to produce good feature vector embedding of input data and use them to design appropriate kernel functions for similarity calculations. Siamese Neural Networks are constructed by two twin networks whose outputs are conjointly trained based on a kernel function to learn the similarity between pairs of input data [23]. Matching Networks [24] perform one-shot learning using cosine similarity. Prototypical Networks employ the metric-based approach using Squared Euclidean Distance [25]. The authors in [26] introduced the concept of Relation Network (RN) which works on Siamese Network principles but with some differences. In the realm of few-shot learning, our chosen approach falls under the Metric-based category. Specifically, we employ RNs as the foundational model architecture.

Meta-learning plays a crucial role in malware detection, particularly in zero-day detection and variant detection. Zero-day detection involves generalizing from known malware patterns to identify similar malicious behaviors, enabling the identification of previously unseen malware. Similarly, meta-learning models can effectively detect and classify new variants by learning shared characteristics and features among different variants. By leveraging meta-learning in these areas, we can develop more robust and efficient malware detection systems, ultimately enhancing the security of computer systems and networks.

C. FEW-SHOT LEARNING OVERVIEW

Few-shot learning is an approach of using a few examples from each class while training a model, unlike the conventional practice of using a huge amount of data. Though this methodology is trickier as it goes against the traditional machine learning paradigm of using an extensive amount of data for training. But due to facing such limitations in real datasets, FSL can be considered a benediction for data scientists. While working on the FSL principle, for a learning task T, Dataset D consists of training and testing sets, i.e.,

$D = \{D_{train}, D_{test}\}$

$D_{train} = \{(x_i, y_i)\} \int_{i=1}^{I}$ where I is a small number

$D_{test} = \{x^{test}\}$

p(x,y) = joint probability distribution of input x and output y

h = the optimal hypothesis from x to y

The process of Few-Shot Learning involves using D_{train} to fit the model and D_{test} to test it, in order to discover h. To approximate h, the FSL model searches for the best set of parameters within a hypothesis space H, in order to parameterize the optimal h^*. The loss function (y_{pred},y_{true}) defined over the real and predicted outputs determines the performance of FSL. The definitive goal of the FSL learning algorithm is to find h^* for which the Risk R(h) is minimum. This risk cannot be computed because the distribution p(x,y) is not known to the learning algorithm. Nevertheless, we can figure out an approximation, called empirical risk, by averaging the loss function over the training set D_{train} of I samples. This risk can be affected by the tuning of any of these three elements, i.e., Data provided by D_{train}, Model to determine Hypothesis Space H, and Algorithm to search the most suitable h from H to fit into D_{train}.

In FSL, for N number of classes, we have K samples for each class of the training set. In this way, D_{train} contains N x K samples for N classes. Therefore, this learning is also called N way K shot Learning. Depending upon the number of instances per class in the training set, N-shot learning is called zero-shot learning (ZSL), one-shot learning (OSL), and few-shot learning. Zero-shot Learning is a learning method where the training instances do not appear for all the classes which lead to learning modeling without the availability of labels [27]. OSL is a technique for performing the classification using past data [28]. It aims to learn information from only a handful of labeled examples per category. The scenarios where the labeled examples are just one in number for each class, are true candidates for one-shot techniques [29]. In this paper, we perform the classification of malicious and benign PE files by learning the identification of each class from a single labeled example.

An effective approach to utilize a limited number of training examples is to simulate the few-shot learning scenario through episode-based training [30]. In each training iteration, an episode is created by randomly selecting 'C' classes from the training set, with K labeled samples arbitrarily picked from each of the 'C' classes to form the *sample set* S = (xi, yi) for i=1 to m (where m = K x C). Additionally, a portion of samples from the held-out classes among those 'C' classes are used as the *query set* Q = (xj, yj) for j=1 to n. Each training example consists of pairs of train and test data points, referred to as an episode. The purpose of dividing the sample and query sets in each episode is to mimic the support and test sets that will be encountered during the actual testing phase.

In a few-shot paradigm, meta-learning can quickly learn an initial representation of a handful of training examples, which can be served as a good starting point.

D. EPISODIC TRAINING

In this paper, we train the model in episodic manner using short episodes and one sample per class, with varying samples between episodes.

Episodic training is a sort of learning related to training the model on a succession of mini-batches where each image fits one of the sets, i.e., support or query [31], [32]. A conventional policy to train FSL approaches is to deliberate a distribution D^ over expected subsets of labels with greater alikeness to the one that came across during evaluation.

Support and query sets are formed in a way that all the classes are included along with a constant number of instances for each class. The notation used for FSL techniques is given by the following variables:

W = The total number of classes or ''ways'';
n = The number of instances per class in the support, also called ''shots'';
m = The number of examples per class in the query.

For the evaluation, the set {w, n, m} describes the problem structure. But for the training time, {w, n, m} can act as a set of hyperparameters monitoring the batch construction with utmost tuning.

E. RELATION NETWORK (RN) OVERVIEW

The relation network is capable of performing few-shot tasks by training to compare query images with a small number of labeled sample images [24], [26]. Figure 3 depicts the basic architecture of a relation network. Relation network is composed of two modules named Feature Embedding (FE) module and the Relation Module (RM). FE module is a component that transforms raw input features into a dense vector representation that captures meaningful information about the features. This vector representation is used as input to other parts of the model like RM. The process of creating feature embeddings involves mapping the input features into a high-dimensional space, where each dimension represents a feature. The goal is to create a representation that captures the relationships between features in a way that is useful for the task at hand. RM is designed to model the relationships between pairs of features in a feature map. The goal of RM is to capture the interactions and dependencies between pairs of features in the feature map, which can help the network better understand the correlation between the images. Here is a detailed explanation of how the various components of a relation network collaborate and operate together.

1) The FE module produces representations of the query and training images.
2) The RM concatenates the feature maps of the query image and support image and compares these embeddings to determine if they have the same category or not.
3) The RM generates relation scores $r_{i,j}$, ranging 0-1, for one Query image x_i and the training samples set x_j.
4) A one-shot vector is created to facilitate straightforward comparison and reporting of the outcome.

During meta-learning, the relation network learns distance metrics for carrying out comparisons of images within each episode, which simulates many few-shot tasks to mimic the test environment settings. Following episodic training, the relation network is capable of efficiently recognizing images from unseen categories by computing the relation scores between test images and the limited samples of each novel category, without requiring any additional updates to the network.

F. DEEP RESIDUAL LEARNING

Deep residual learning is a technique used to improve the accuracy of deep neural networks. It was introduced in the ResNet architecture, which stands for Residual Network. In traditional deep neural networks, each layer attempts to learn a set of features from the previous layer's output. However, as the number of layers increases, the network can become too deep and the gradients can become difficult to propagate, resulting in the vanishing gradient problem. Deep residual learning solves this problem by introducing a residual connection, which allows the network to learn the residual functions (the difference between the output and input) instead of the original functions. This means that instead of trying to learn a set of features from scratch in each layer, the network can reuse some of the features from the previous layer, reducing the chances of the gradients vanishing. By utilizing this approach, the network can attain a greater depth, which enables it to extract more intricate features, while avoiding the issue of vanishing gradients.

The formula for deep residual learning can be represented as:

$$y = F(x, W_i) + x \tag{2}$$

where
y = output of the residual block, and
x = input to the residual block.

The residual function is defined using a set of weights, and it involves adding the input x to the output of the function. This allows the network to learn the residual mapping, or the difference between the input and output, rather than the original mapping. In this way, instead of learning the identity function F(x) = x, the network can learn the residual function F(x) = x - H(x), where H(x) is a function approximated by a neural network. This allows the network to use the information from the previous layer and makes it easier for the gradients to flow through the network.

IV. RELATED WORK

Recently, there has been a significant effort in the area of FSL to tackle the issues related to insufficient and uneven datasets. One such attempt is described in [37], where a multi-tiered neural network architecture is proposed to identify images of new categories with limited examples. In [38], authors suggested a deep learning framework that incorporates streamlined end-to-end training for few-shot classification.

In this study, we introduce a network architecture for detecting malware using the relation network in a one-shot scenario. Our motivation for this work stemmed from the previous successful implementations of different approaches using the relation network. However, we noticed some gaps in those approaches and aimed to fill them by introducing new architectures in the same direction.

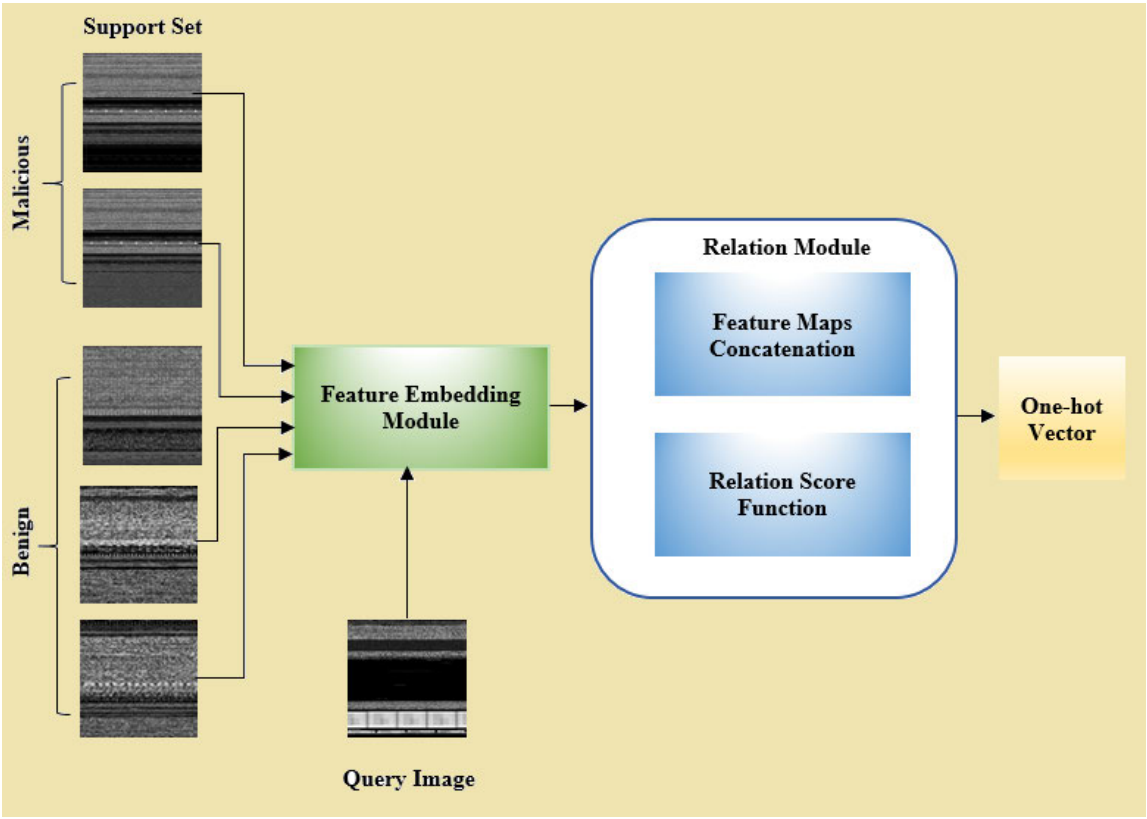

FIGURE 3. Basic Architecture of Relation Network.

TABLE 2. Comparison with Previous Work.

Features	Previous Work	This Paper
One-shot based malware detection	[11]–[13], [24], [33]	✓
Similarity metric based strategy	[9], [24]–[26], [34]–[36]	✓
Comparisons with existing approaches	[24]–[26]	✓
Malware detection with one-shot based learning technique	X	✓

A. ONE-SHOT LEARNING

One-shot learning intends to categorize the new classes unseen in the training set, provided with only one instance of each new class. In the past few years, many approaches have been adopted to tackle the problem of OSL. The authors in [11] used OSL to learn new categories with lesser training examples by developing a Bayesian learning framework founded on object categories presentation with probabilistic models. The extracted information from earlier learned categories was denoted with an appropriate prior probability density function. These models were updated with fewer training instances, which were then used for object categories detection and discrimination.

In [12], the authors used OSL for attaining semantic image segmentation. Using OSL, they generated a segmentation mask of the image for the novel class as output. They computed parameters for segmentation in a single forward pass. In [13], authors adapted OSL to predict molecule behavior in a new experimental system. The authors modified the Matching Network architecture [24] along with the residual convolutional network to learn good metrics to distinguish between support and query instances. In [24], authors proposed a novel framework for OSL, which is made up of a neural network utilizing current advances in attention and memory for rapid learning. The authors used the machine learning principle of having the same conditions for testing and training. The authors built mini-batches having only a few examples per class, switching the task from minibatch to minibatch, mimicking the test time. New benchmarks have been set for ImageNet and small-scale language modeling.

The authors in [33] combined reinforcement learning with OSL, enabling the model to agree on examples that are worth labeling for a classification task. Instead of using supervised learning, they proposed a model that learns by employing

only a minority of demanded labels and trade-off likelihood accuracy with reduced label requests through the reward function.

B. RELATION NETWORK

In our work, we modify the relation network to detect the malicious and benign nature of a PE file by transforming it into images. We focus on the sole purpose of dealing with the nature of the PE file, not with the real behavior as done in [34], where they used a similar way for unknown file image visualization to address the malware identification problem.

The authors in [26] proposed relation network, which is all over trained to identify new classes with even few examples per class. The authors proposed a novel approach for few-shot recognition called the two-branch relation network. This model incorporates an embedding module and a deep non-linear distance metric to compare query and sample items. The non-linear similarity distance metric approach learned the metric in a data-driven manner instead of manually selecting the right similarity function, e.g., Euclidean, cosine, or Mahalanobis. For effective few-shot learning, episodic training is used to tune the embedding and distance metrics. They performed their tests on Omniglot and mini-Imagenet datasets. With OSL, the accuracy is 50.44, which was compared with Matching Nets [24], Meta-Nets [35], Meta-Learn LSTM [9], MAML [36], and Prototypical Networks [25], and found to be the highest among all. We used this paper as a baseline paper and compared the accuracy of our proposed CNN model with this paper. We found that the accuracy of our proposed model is the highest among all reported accuracies for training and test sets.

Hui et al. [39] extended the concept of RN and proposed a Self-Attention Relation Network (SARN) for few-shot learning, introducing an attention module to enhance the learned features extracted by the embedding module. The relation module is used to compare the features of the query sample and support set to produce the relation score. Their work emphasized the importance of nonlocal information and allowed long-range dependency. They executed their experiments on benchmarks, i.e., Omniglot, miniImageNet, AwA, and CUB. Authors in [40] proposed Temporal Attention-based RN (TARN) for video segments to determine the relation scores between a query video and other sample videos. They also utilized attention mechanisms to perform temporal alignment and learned a deep-distance measure, and worked on the aligned representations at the video segment level. The network is trained entirely using an episode-based training scheme. The authors of [41] introduced a Position-Aware RN (PARN) that focuses on the characteristic of CNN-based RNs, which can be influenced by the spatial positional correlation of semantic objects in compared images. By introducing certain parameters, the authors enhanced the position awareness capabilities of the RN. The proposed approach was evaluated on Omniglot and miniImagenet datasets.

He et al. [42] modified the relation module by parameterizing to estimate the relationship between two samples. It produces close relationships for analogous samples and distant relationships for unlike samples. In their proposed Memory-Augmented RN (MRN), authors learned end-to-end transferable representations and distance metrics and enhanced representations by aggregating information from the neighborhood environment. The proposed approach was evaluated on miniImagenet and tieredImagenet benchmark datasets. Gao et al. [43] benefited from the relation network for Hyperspectral Image few-shot classification and acquired reasonable classification accuracy with the limited labeled sample.

Previous research on relation networks and one-shot learning has primarily focused on the Omniglot and miniImagenet datasets, and the accuracy of malware detection using these datasets has been relatively low, with reported accuracy rates of less than 60%.

V. PROPOSED METHODOLOGY

This paper primarily concentrates on the task of classifying new, uncommon instances of malware with only a limited amount of training samples, in a scenario called few-shot classification for malware datasets. This research is distinctive as it investigates the use of binary classification to distinguish between benign and malicious PE files, which is an understudied area. Our approach fills this gap by dividing PE files into two groups: benign and malicious, and using one-shot learning. Furthermore, we applied episode-based training to train the model in a few-shot way.

A dataset that is labeled with ''Benign'' and ''Malicious'' classes with few samples per class is divided into three parts: the meta-training set (Dtrain), meta-validation set (Dval), and the meta-testing set (Dtest), and the class labels in the Dtrain and Dtest sets are kept separate. The validation set is exclusively used to evaluate how well the model generalizes to new data. In each classification task, both the meta-training and meta-testing sets consist of a support set (S) that the model is familiar with, and a query set (Q) that the model has not seen before. The task requires the model to classify the samples in Q based on the information in S, where the target samples can vary in each task and have the same label space, as shown in Figure 4. During the training phase, only the training dataset (Dtrain) is utilized. In the testing phase, the performance of the model is evaluated using the test dataset (Dtest) to assess its ability to adapt to new classes

Our model uses a series of images, xi, for each class, during each episode. The model receives a point reward for each correct classification. The goal is to maximize the total rewards earned during the episode. The model uses a strategy of storing a set of class representations and their associated labels in memory. When presented with a new image xi, the model compares the representation of xi to the stored class representations, taking into account the potential uncertainty of a match and the cost of being correct or incorrect. The cost or loss is minimized during each Epoch with a specified

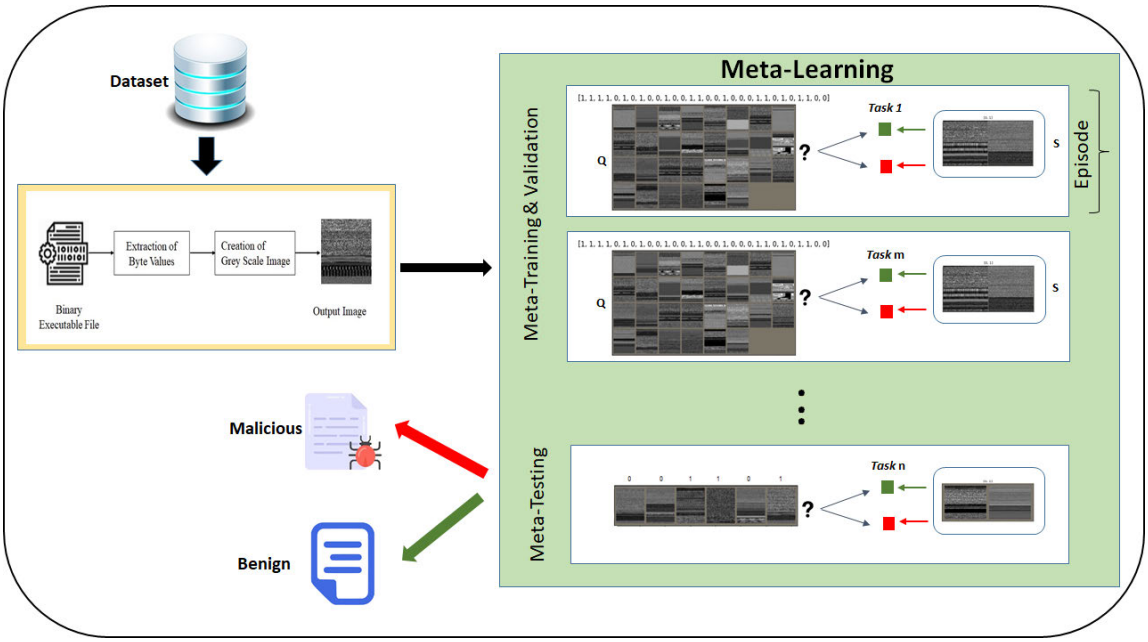

FIGURE 4. Detailed Methodology of the Proposed Architecture ResRelNet (RRN).

learning rate. By selecting the optimal learning rate and initial weights, the model can achieve higher accuracy.

A. MODEL ARCHITECTURE AND PARAMETERS CONFIGURATION

In this paper, a binary classification approach using one-shot learning is applied to distinguish between malicious and benign files. This specific task has not been previously addressed. A ResNet-based RN is utilized for the detection of malware files and is implemented in two stages. First, the VGG16 network is utilized as the backbone for feature extraction with default layers, and a non-parametric relation score learning module is used for the calculation of the relation score, resulting in an accuracy of up to 70%. Then, a ResNet18-based architecture is introduced for the embedding module, and the default RN relation module is modified for improved results. A modified version of ResNet-18 is used for training from scratch with randomly initialized network parameters, with the number of input channels reduced to one for training with greyscale images. PE files are transformed into images, scaled to 224 $\times$ 224 dimensions, and used as input for the Discriminative Feature Embedder (DFE) module, as shown in Figure 5.

A prior baseline study [26] used four convolution blocks for feature extraction in miniImageNet, but the model only achieved a 50% accuracy in malware detection. To improve the performance, we employ transfer learning and utilize a pre-trained network for weight initialization. The pre-trained model is adapted to fit in our specific problem by modifying it. The detailed architecture of our proposed RRN is illustrated in Figure 6.

The input is fed into the DFE module, which consists of a single convolutional layer followed by four residual blocks. Specifically, the convolutional block comprises a 2D convolution with 64 filters, followed by batch normalization and ReLU non-linearity layers. This initial convolutional operation helps to extract relevant features from the input data. After the first convolution, a 2D max-pooling layer is added to capture the most prominent features within local regions while reducing the spatial dimensions of the data. This pooling operation aids in identifying the most salient information at a coarser scale. The four residual blocks are then employed, each consisting of two convolutional layers with 3×3 filters, batch normalization, ReLU activations, and skip connections. These residual blocks play a crucial role in addressing the issue of vanishing gradients. By allowing the gradients to flow directly through the skip connections, the network is able to retain important information and facilitate better gradient propagation during training. Overall, the DFE module, with its convolutional block, max-pooling layer, and multiple residual blocks, aims to capture relevant features from the input while mitigating the vanishing gradients problem commonly encountered in deep neural networks.

The Relation Module consists of three layers of 64 output channels with 3×3 filter sizes, followed by two fully connected layers, and a final dense layer to match the number of classes for our task. The Rectified Linear Unit (ReLU) activation function is applied and the Sigmoid function is used for classification. The training is done using mini-batch Stochastic Gradient Descent with a learning rate of 0.001 for DFE and 0.01 for the RM. The initial value of Gamma is set to 0.1, while the momentum is set to 0.9 and the weight decay factor is set to 0.0005. Pretraining

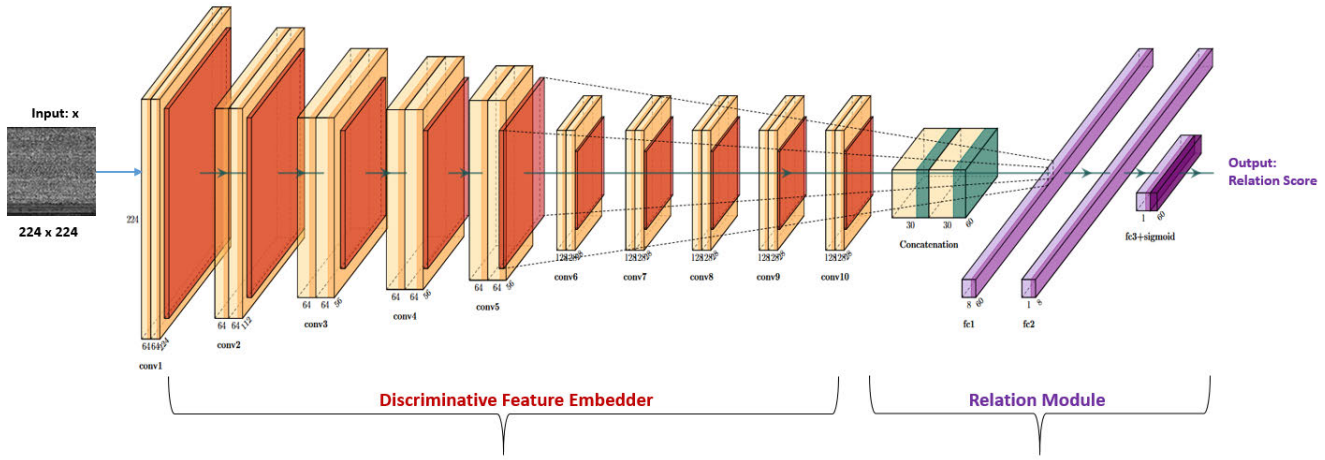

FIGURE 5. Structure of Proposed Architecture.

ResNet18 on a large dataset, such as ImageNet, and then fine-tuning it on our smaller, specialized dataset help to improve RNN feature extraction capabilities for our specific task.

B. LOSS FUNCTION OPTIMIZATION

During the meta-training phase, the optimization of the loss function involves iteratively updating the model's parameters or employing episodic training, as described in our proposed model. In our model training, we utilized the Mean Squared Error (MSE) as the loss function (Eq.3). The objective was to regress the relation score to its corresponding reference value. For matched pairs, where the similarity is 1, and mismatched pairs, where the similarity is 0, the MSE was used to quantify the discrepancy between the predicted scores and the ground truth values.

$$\sum_{i=1}^{D}(x_i - y_i)^2 \tag{3}$$

The meta-training process involves iteratively sampling a batch of tasks or episodes from a meta-training dataset. For each task or episode, the model is initialized with task-specific parameters and trained on a support set of labeled examples. The loss function is computed based on the model's predictions and the ground truth labels of the support set. The main goal is to update the model's parameters in a way that minimizes the loss function across the sampled tasks or episodes. We achieved this through gradient-based optimization method, i.e., stochastic gradient descent (SGD). The gradients of the loss function with respect to the model's parameters are computed using backpropagation, and the parameters are updated accordingly to minimize the loss.

The optimization process is repeated for multiple iterations or epochs, allowing the model to adapt and learn task-specific parameters that generalize well across different tasks.

VI. IMPLEMENTATION OF THE PROPOSED ARCHITECTURE

The use of meta-learning in the context of low-data malware detection faces certain limitations as it relies on the availability of diverse and representative data to effectively learn and generalize to new tasks. In the case of low-data malware detection, due to scarcity of labeled samples, it becomes challenging to build accurate meta-learning models. Moreover, it is also challenging to represent diverse malware families adequately.

During the implementation of experiments in one-shot learning, we have faced various limitations and encountered several challenges. One of the primary challenges revolved around the task of generalizing from a single or a small number of examples. The scarcity of labeled data for each class posed a significant obstacle in developing robust models and attaining high levels of accuracy. Additionally, in the context of one-shot learning, it is crucial to address the issue of overfitting, considering the limited data available for each class. Due to this limitation, one-shot learning models are more susceptible to overfitting. To mitigate this problem and enhance the generalization performance of the models, it became necessary to employ regularization techniques like weight decay in a careful manner. We successfully tackled the encountered challenges and achieved commendable performance.

As far as computational requirements of the meta-learning based model is concerned, our model uses training deep neural networks; therefore, it is supported by powerful GPUs or specialized hardware accelerators to speed up computations and reduce training time. Moreover, our model often requires significant memory to store the model parameters, intermediate activations, and gradients during the training process. Therefore, the available memory should be sufficient to accommodate the model and the training data to avoid memory-related issues during training.

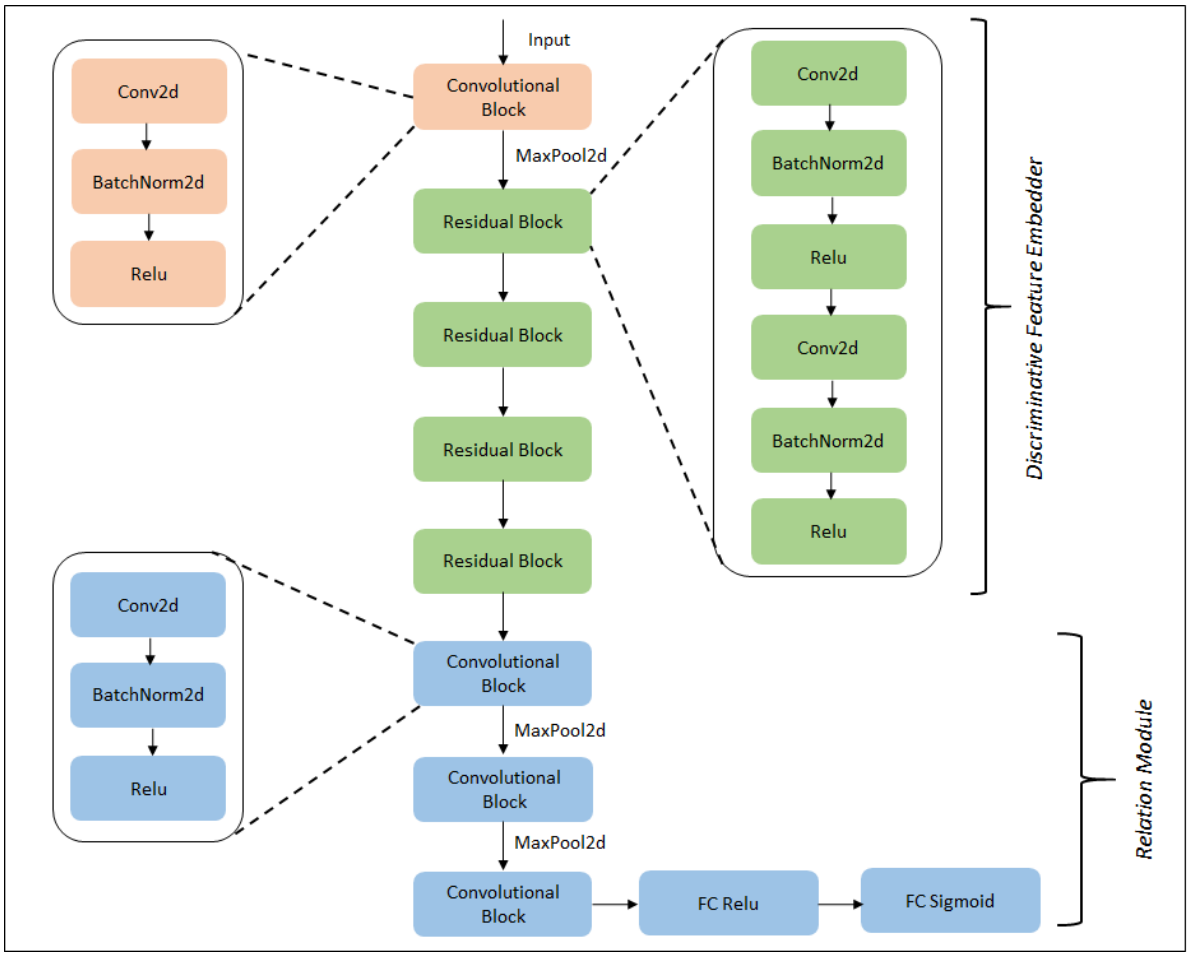

FIGURE 6. Detailed internal structure of proposed RRN.

The proposed framework is implemented using PyTorch in Anaconda Spyder 5.3.3.

A. DATASET AND PREPROCESSING

Virus PE files for the implementation purpose are collected from the publicly available Virusshare dataset and CIPMA-MW-PE2023-1 dataset collected from CIPMA lab. Benign files are acquired from Figshare dataset. We obtained a dataset consisting of two categories of files: benign and malicious. Specifically, we focused on studying and collecting PE files, which are potential candidates for our research work. PE files refer to executable files commonly found in Windows operating systems. By analyzing and including these files in our dataset, we aim to investigate and address the specific challenges and characteristics associated with malware detection and classification. The inclusion of both benign and malicious PE files enables us to develop robust models and algorithms that can accurately differentiate between safe and potentially harmful software. Our study focuses on malware types with limited data availability, and it is found that four PE types (OCX, ACM, COM, and SCR) are relatively scarce. The following provides information on each of these:

An OCX file is a type of file that holds a reusable software component, known as an ActiveX control. These controls can be used within various Windows software applications and can serve different purposes, such as creating user interfaces, webpage widgets, online games and multimedia viewers. OCX files can be found in both Windows and as part of software installations. Some of the common OCX files include mscomctl.ocx and Flash.ocx.

ACM files are typically associated with Adobe Photoshop. They typically include command button settings used by the software for raster graphics design and photo editing.

COM files are executable programs that can be run on MS-DOS and Windows. They are stored in a binary format, similar to an EXE file, but have a smaller maximum size of around 64KB, and lack a header or metadata. While COM files are used to execute a set of instructions, EXE files are used for more fully developed programs.

Files with the .scr extension are screen saver files used by the Microsoft Windows operating system. They can include animations, graphics, slideshows, or videos that can be set as a Windows screensaver. SCR files are typically located in the main directory of Microsoft Windows.

TABLE 3. Dataset Split Count.

Dataset	Benign	Malicious
Training	80	80
Validation	30	30
Test	110	110

The dataset with .ocx files is divided into three subsets: meta-train, meta-validation, and meta-test (Table 3). The meta-train set is used for learning the meta-learner's parameters, the meta-validation set is used for hyperparameter tuning, and the meta-test set is used for evaluating the generalization performance of the trained meta-learner. The meta-train subset includes 80 files, both benign and malicious. The meta-validation subset has 30 samples per class and the meta-test subset has 110 samples per class, as shown in Table 3.

We start the implementation task by utilizing the proposed framework for the miniImageNet dataset. The baseline accuracy recorded for miniImageNet is 50 to 55 %, but with ResRelNet, it improved up to 75-79 % for 1-shot and 5-shot schemes. As we are focused to improve the low data malware detection accuracy; therefore, we further investigate by using the figshare dataset containing benign and malicious executables with a total of 8,500 samples coming from different malware families (4GB of data). Initially, we conduct the experiments using the miniImageNet dataset. Firstly, we convert these executables into images. We extract byte values from binary executable files and store them in a list. Then, we create a greyscale image from binary data. We create square images of size 84×84 from binary data (see Figure 8). As the proposed model performs well for 224×224-sized images, the size is further scaled programmatically in Python. The experiments are performed in two phases. Firstly, we implement the baseline paper approach with our different models like conv4, VGG 16, ResNet 18, and ResRelNet using the dataset as in the baseline work. We use miniImageNet for one-shot and 5-shot learning approach. Results are improved for conv4, VGG 16, and ResNet18. Then using the framework ResRelNet on miniImageNet, results are improved by a good percentage. In the second phase, the implementation is accomplished for carrying out experiments using malware datasets, i.e., Figshare datasets. The evaluation shows that using ResRelNet, results improve remarkably with a percentage accuracy of over 93% for unseen instances.

B. EXPERIMENTAL SETTINGS

We conduct two types of experiments to demonstrate the effectiveness of our approach for classifying malware in the few-shot setting. The first experiment involves meta-training and meta-testing on the same dataset, i.e., .ocx dataset, while the second experiment involves meta-testing on different datasets (.acm, .com, .scr) without additional retraining. In each experiment, the dataset is divided into three parts: a meta-training set, a meta-validation set, and a meta-testing set. The meta-validation set is used to evaluate the model's ability to generalize, and the model with the best validation accuracy is selected for further testing. This process employs the early stopping technique, which is a common and effective method for preventing overfitting.

For all experiments, we use SGD with an initial learning rate of 0.01 and then set it to 0.001. With several passes, i.e., episodes, as the algorithm keeps stroking through the training set, the algorithm is converged. In our experimental setup, we trained our model using 500 episodes and observed that the problem reached convergence within this number of episodes. This indicates that the model successfully learned to adapt and generalize to new tasks or domains, and further training did not result in significant improvements. Each episode in the training process follows a specific sequence of steps, including task sampling, model adaptation, evaluation, and loss minimization. Initially, the one-shot learning task is sampled, consisting of a support set and a query set. The model is then reset to its initial state before each episode. Next, the model is adapted to the support set examples from the sampled task, which involves updating its parameters to acquire task-specific knowledge. The model's performance is evaluated on the query set to assess its effectiveness. Finally, the model's parameters are updated to minimize the evaluation loss, guiding it towards improved performance. These steps are repeated for multiple episodes to train the model and enhance its ability to adapt and generalize to new tasks or domains.

It is observed that 0.001 LR adapts to the problem well in 500 episodes, as shown in Figure 10. For 1-shot learning, we select 15 query images per class during meta-training and 3 images per class for the meta-testing scenario. So for the 2-way 1-shot scenario, there are $(15 \times 2) + (1 \times 2) = 32$ examples per mini-batch/training episode, as shown in Figure 7, and $(3 \times 2) + (1 \times 2) = 8$ examples per mini-batch for each meta-testing episode, as shown in Figure 9.

C. BASELINE/REFERENCE APPROACHES FOR COMPARISONS

Six models previously suggested for few-shot learning are used as reference points for comparison. These include Relation Network [26], Self-attention Relation Network (SARN) [39], Temporal Attentive Relation Network (TARN) [40], Position-Aware Relation Networks (PARN) [41], Memory-Augmented Relation Network (MRN) [42], and Prototype-Relation [44]. In order to increase the number of comparisons, we implement three other machine learning methods as reference points: Conv4, ResNet18 and VGG16.

D. PERFORMANCE EVALUATION METRICS

We have used different performance metrics to perform a comprehensive evaluation of the model's performance from different angles. Each metric highlights a specific aspect of

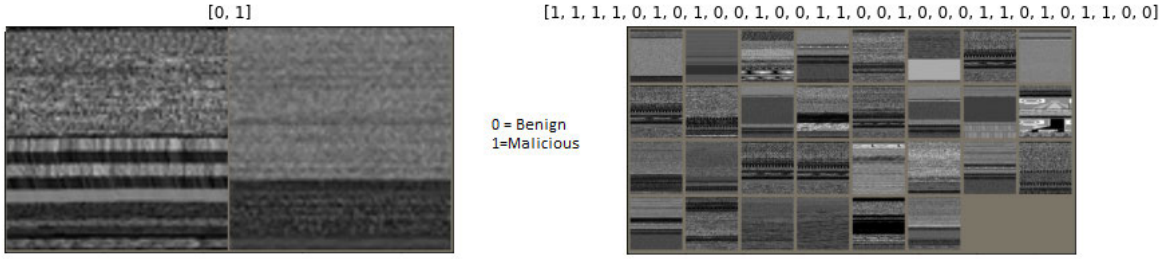

FIGURE 7. Support and Query sets during Meta-Training.

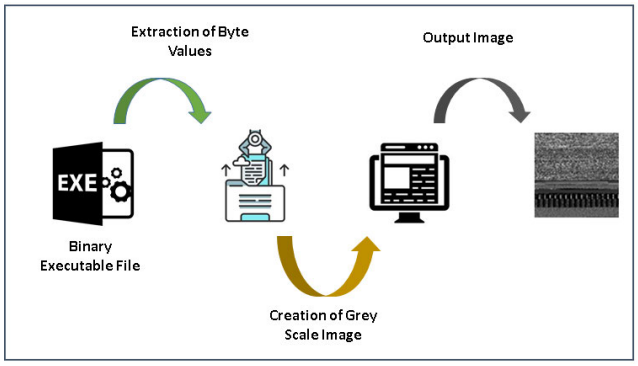

FIGURE 8. Process of Malware Conversion into Image.

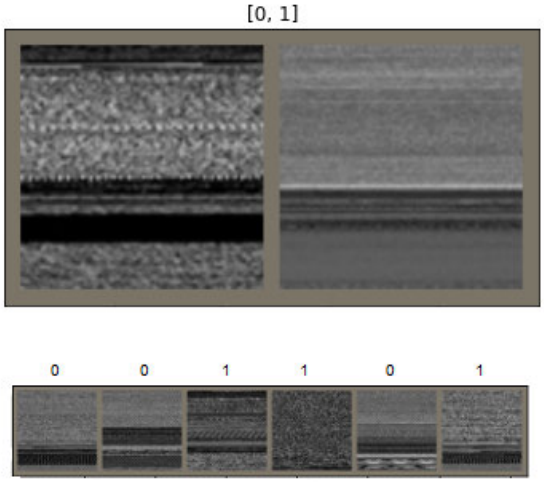

FIGURE 9. Support and Query sets during Meta-Testing.

TABLE 4. Detail of other Datasets for meta-testing.

Dataset	No. of Samples in meta-test set
.acm	50
.com	30
.scr	70

the model's behavior, and depending on the problem domain and specific goals, different metrics can provide valuable insights and guide on decision-making. This comprehensive evaluation helps in making informed decisions about the effectiveness and suitability of the meta-learning model for the given task. In order to accurately present the results obtained by the proposed model, we employ five evaluation metrics: accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (ROC-AUC).

Accuracy is the segment of predictions that our model made right. Accuracy can be defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Precision is a measure of a machine learning model's performance that indicates the accuracy of positive predictions made by the model.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall is the capability of a classification model to correctly identify only the pertinent data points.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F1-score is a measure of a classification model's accuracy. It is the harmonic mean of precision and recall, and it ranges from 0 to 1, where 1 is the best possible value.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{7}$$

An ROC curve is a graphical representation of a classification model's performance at various classification thresholds. AUC quantifies the entire two-dimensional region beneath the entire ROC curve.

E. MODEL EVALUATION ACROSS OTHER DATASETS

The focus of this study is to demonstrate that the model, trained on a single malware dataset, would exhibit good performance on the meta-testing set of a different dataset, as both sets include classes that the model is not exposed to during training. To confirm that our model has acquired transferable knowledge, we conduct validation experiments where the meta-training and meta-testing sets come from different datasets. We consider miniImagenet, and other discussed PE data-scarce formats for our model evaluation.

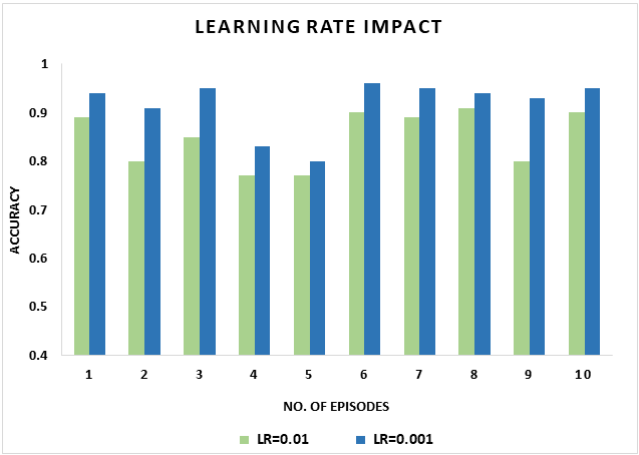

FIGURE 10. Analysis of the Learning Rate Impact on the Performance of Proposed Model.

VII. RESULTS AND ANALYSIS

We conducted experiments using the miniImagenet and malware datasets to evaluate the performance of our proposed model. The results of these experiments are presented in Tables 5, 6, and 7. The findings indicate that our proposed model achieved improved accuracy on the benchmark dataset. Specifically, compared to TARN, which achieved an accuracy of approximately 80%, our RRN model achieved an accuracy of up to 90% due to its enhanced feature extraction capabilities. Moreover, when evaluating the model on the malware dataset, our RRN model achieved an impressive accuracy of 94%. This demonstrates that our approach exhibits strong generalization capabilities even when dealing with limited instances. These results highlight the effectiveness and potential of our proposed RRN model in the domain of few-shot learning, showcasing its ability to outperform existing approaches and achieve high accuracy on diverse datasets. This section presents a comprehensive analysis of the results obtained from our experiments.

A. IMPACT OF LEARNING RATE

We conducted experiments using different numbers of images in mini-batches during training and testing. Specifically, we used one image (1-shot) and five images (5-shot). The training process employed a learning rate of 0.01 and 0.001, and consisted of 500 episodes. The results of the experiments indicate that the model achieved optimal performance when using a learning rate of 0.001, as it allowed for slow and effective training (see Figure 10).

B. MODEL ACCURACY AND LOSS PLOTS

The proposed model's accuracy and loss are illustrated in Figure 11. The accuracy plot indicates that the models have been well-trained as the curves do not continue to rise after a certain point. Furthermore, the model's performance is consistent on both datasets, indicating that it has not excessively learned from the training data. The loss graph demonstrates that the model achieves matching results on both the train and

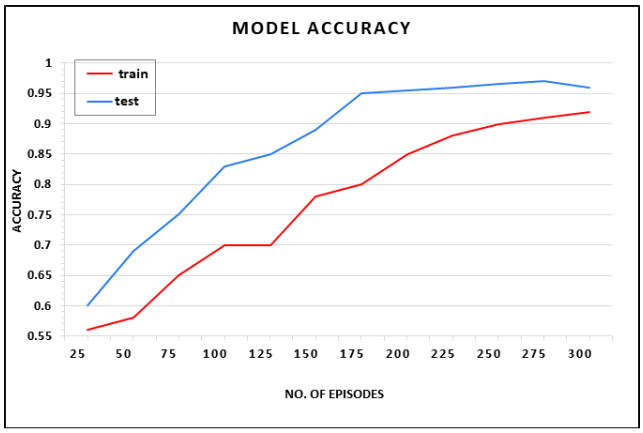

FIGURE 11. Model Accuracy Graph.

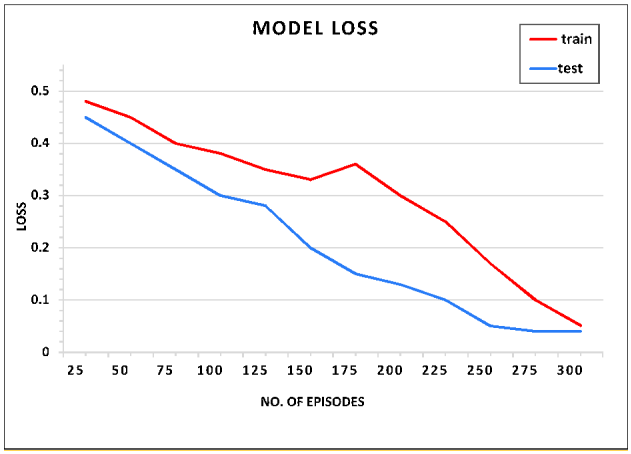

FIGURE 12. Model Loss.

test datasets, and the loss value decreases with an increase in the number of epochs. If these parallel curves begin to diverge, this is an indication that the training should be stopped, as shown in Figure 12.

C. MODEL PERFORMANCE ASSESSMENT WITH EXISTING FSL METHODS

We have conducted a comprehensive analysis of the advanced designs employed in ResRelNet and their enhancements over existing few-shot models. In order to ensure a fair comparison, we carefully selected the existing models that have been proposed for few-shot learning, making sure that they employ a similar base technique (see section VI-C). These models serve as appropriate reference points for evaluating the performance of our proposed method.

In most scenarios, proposed ResRelNet surpasses all the baseline models by a significant margin. TARN has the narrowest performance gap compared to ResRelNet among the baseline few-shot models, but still has an average accuracy deficit of around 10%, while other models have an average accuracy deficit of over 15%. Table 5 shows test accuracy and 95% confidence interval for malware 5-shot classification

TABLE 5. Performance Comparisons with Existing Methods for 5-shot problem on benchmark dataset miniImagenet.

Existing Models	Measures				
	Accuracy	Precision	Recall	F1-Score	AUC
Relation Network [26]	65.32 ± 0.70%	0.80	0.756	0.8	0.7
Self-Attention Relation Network (SARN) [39]	66.16 ± 0.51%	0.76	0.55	0.68	0.65
Temporal Attentive Relation Network (TARN) [40]	80.66 ± 0.54%	0.78	0.79	0.78	0.7
Position-Aware Relation Network (PARN) [41]	70.50 ± 0.64%	0.767	0.75	0.755	0.756
Memory-Augmented Relation Network (MRN) [42]	71.13 ± 0.50%	0.78	0.81	0.79	0.778
Prototype-Relation [44]	49.54 ± 0.09%	0.50	0.69	0.601	0.677
ResRelNet (RRN)	**90.13 ± 0.88%**	0.87	0.898	0.88	0.87

TABLE 6. Performance Comparisons with Existing Methods for 1-shot problem on benchmark dataset miniImagenet.

Existing Models	Measures				
	Accuracy	Precision	Recall	F1-Score	AUC
Relation Network [26]	50.44 ± 0.82%	0.550	0.656	0.58	0.57
Self-Attention Relation Network (SARN) [39]	51.62 ± 0.31%	0.56	0.55	0.58	0.546
Temporal Attentive Relation Network (TARN) [40]	66.55 ± 0.14%	0.68	0.669	0.678	0.687
Position-Aware Relation Network (PARN) [41]	54.36 ± 0.84% 0.567	0.55	0.555	0.656	
Memory-Augmented Relation Network (MRN) [42]	57.83 ± 0.69%	0.658	0.681	0.6679	0.678
Prototype-Relation [44]	68.34 ± 0.06%	0.650	0.69	0.661	0.697
ResRelNet (RRN)	**95.13 ± 0.54%**	0.98	0.89	0.91	0.95

TABLE 7. Performance Comparisons with Existing Methods for 1-shot problem with malware dataset.

Existing Models	Measures				
	Accuracy	Precision	Recall	F1-Score	AUC
Relation Network [26]	53.34 ± 0.82%	0.56	0.656	0.55	0.57
Self-Attention Relation Network (SARN) [39]	50.13 ± 0.41%	0.47	0.57	0.57	0.601
Temporal Attentive Relation Network (TARN) [40]	64.34 ± 0.23%	0.64	0.569	0.648	0.686
Position-Aware Relation Network (PARN) [41]	52.13 ± 0.74% 0.558	0.55	0.579	0.660	
Prototype-Relation [44]	65.13 ± 0.07%	0.641	0.59	0.681	0.677
ResRelNet (RRN)	**94.13 ± 0.34%**	**0.94**	**0.87**	**0.90**	**0.93**

on benchmark miniImagenet data set. Table 6 summarizes test accuracy with 95% interval for 1-shot classification on benchmark miniImagenet data set. Table 7 summarizes the test accuracy for 1-shot classification with a 95% confidence interval for the .ocx dataset. Our meta-learning framework exploits the potential of transfer learning from ResNet to improve adaptability and generalization when encountering new tasks or domains. By utilizing pre-trained ResNet models trained on extensive datasets like ImageNet, our framework gains access to learned feature representations that capture intricate and significant visual patterns. Through episodic training, our proposed RRN model becomes proficient in identifying shared visual patterns and features. Consequently, it can rapidly adapt and provide precise predictions for unseen examples. This transfer learning approach effectively addresses the constraints posed by limited training data in

TABLE 8. Performance Assessment with other Datasets.

Measures	Datasets			
	.ocx	.com	.acm	.scr
Accuracy	0.9513	0.8807	0.8559	0.7858
Precision	0.8299	0.7813	0.8666	0.7966
Recall	0.8305	0.8728	0.8545	0.7623
F1-Score	0.8286	0.7964	0.84	0.7494
AUC	0.9685	0.8956	0.9218	0.8891

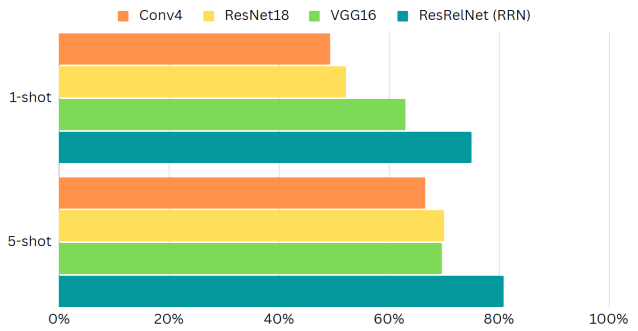

FIGURE 13. Observed Performance of Different Approaches for miniImagenet Dataset.

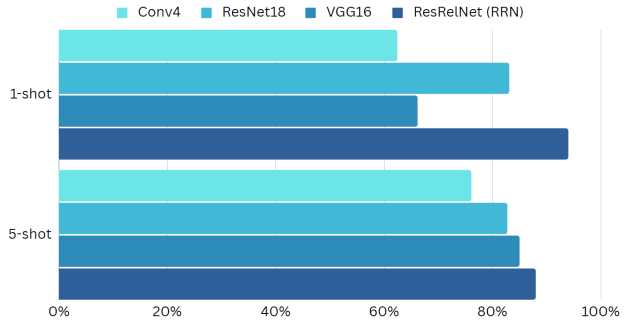

FIGURE 14. Observed Performance of Different Approaches for Malware Dataset.

few-shot learning scenarios, thereby augmenting the model's capacity to effectively handle novel tasks.

D. PERFORMANCE OBSERVED WITH OTHER MODELS

At the start of the implementation process, we utilize Conv4 and VGG-16 models for feature extraction. However, we observe that deeper networks often experience accuracy saturation and degradation during convergence. To enhance the performance, we apply the pre-trained ResNet-18 model to the miniImageNet and malware datasets, resulting in superior accuracy compared to VGG-16. We then implement our proposed ResRelNet for the image set and malware datasets, and due to its improved feature extraction capability, it achieves better results than previous approaches. The results of the different approaches adopted for the miniImagenet are displayed in Figure 13, and the results for the malware dataset with various approaches can be seen in Figure 14. These results demonstrate that our ResRelNet outperformed all other methods by a significant margin for both dataset types.

E. MODEL PERFORMANCE ASSESSMENT WITH OTHER DATASETS

Our experimental results using the .ocx PE dataset demonstrate that our approach is effective for detecting malware across different formats, even when only very small number of samples are used. The ResRelNet model trained using our meta-learning technique performs similar to models trained on other datasets without the need for additional retraining or fine-tuning. This indicates that our model possesses robust generalization abilities, and is capable of categorizing malware from different origins after training on a single dataset shown in Table 8.

We have conducted a detailed analysis of the advanced designs in ResRelNet and how they improve upon existing few-shot models. The shortcomings present in current models served as a motivation for the development of ResRelNet, which addresses issues such as vanishing gradient, complex feature extraction, and adoption of residual learning based structure for feature extraction.

This study demonstrates that combining malware image representation with residual learning techniques, along with pre-training and transfer learning approaches, results in improved performance. The results suggest that utilizing residual learning on malware images can effectively extract high-level features without the issue of vanishing gradients.

F. MODEL ASSESSMENT WITH PACKED AND POLYMORPHIC MALWARE

Polymorphic malware is a type of malicious software that uses various techniques to constantly modify its code or structure in order to avoid detection by the security software. This makes it difficult for traditional antivirus software to detect and block the malware, as it can appear as a completely new variant each time it is encountered. The constant changes to the malware's identifiable features make it challenging to identify patterns and signatures that can be used for detection, requiring more advanced security measures,

such as behavior-based detection and machine learning algorithms.

Our proposed model is well-suited for detecting polymorphic malware as it is designed to perform accurate malware detection with just one training instance available. Unlike traditional methods, our model does not rely on large amounts of labeled data, making it particularly effective in cases where new malware variants are constantly emerging and have no prior known labels. As a result, our model performs well in detecting new and unique variants of polymorphic malware, and achieves a high level of accuracy in this challenging context.

Packed malware is a type of malicious software that has been compressed or encrypted to hinder its detection and analysis by security tools such as antivirus software. For the purpose of this study, we converted the binary form of the malware into images. The process involved converting the binary code into 8-bit vectors, which were then utilized to create grayscale images. Each vector served as a pixel and was representative of the pixel's intensity level. To evaluate our model's effectiveness, we utilized unpacked malware to produce the images. As a result, our model achieved an accuracy rate of 94%, but due to the presence of packed malware, we experienced a 6% decrease in accuracy.

The overall success of a meta-learning model can be determined based on its generalization, task adaptation and computational efficiency. A successful meta-learning model should demonstrate strong generalization capabilities. It should be able to quickly adapt to new tasks or domains with a limited number of training examples and achieve high performance on unseen data. Our proposed model RRN has shown good generalization even with one training instance. Moreover, our model adapted to learn new task-specific parameters quickly during the adaptation phase. The model's ability to adapt to new tasks and improve its performance with limited task-specific data is an important measure of its success.

VIII. CONCLUSION AND FUTURE WORK

Malware has become a widespread problem in the Internet era. This refers to all programs that can lead to a computer crash, data breach, unauthorized access, resource damage, and other negative consequences. Traditional malware detection systems often rely on machine learning techniques such as decision trees, support vector machines, and logistic regression. However, these models require a significant amount of data and proper management to ensure accurate representation of information. With limited available data, these techniques can become insufficient. Malware can be categorized into various forms, including PE, ZIP, XHTML, RTF, JPEG image, and Windows registry, and each type is analyzed to identify malware and its variants. However, certain categories may have limited training samples, leading to overfitting and a reduction in the models' ability to generalize to rare forms of malware. The solution to this issue is the introduction of few-shot learning. This approach allows for classification with just a small number of samples and quickly adapting to new tasks with supervised guidance. Few-shot learning is characterized by its ability to generalize to new classes during training, even without prior exposure to them.

In this paper, we used One-shot Learning (OSL) to classify malicious and benign Portable Executable (PE) files. OSL is a method of classifying based on previous data and only requires a few labeled examples per category. When there is only one labeled example for each class, the method is suitable for one-shot techniques. In our study, we have achieved the classification of the two classes by learning their identification from a single labeled example for each class.

Our proposed architecture presents a new way to implement the relation network, a straightforward technique used for one-shot learning. We used this architecture to improve data-scarce malware detection. The Discriminative Feature Embedding module uses a deep residual learning based model, attached to the Relation Module for combining query and support feature vectors. The loss function used is Mean Squared Error (MSE). End-to-end episodic training was employed to optimize the feature embedding and similarity measures for effective one-shot learning. We used PE file formats, i.e., ocx, acm, com and scr, after transforming them into images. One-shot learning was employed, which produced 94% accuracy with only one training instance. It was observed that the proposed architecture outpaced the baseline method and enhanced accuracy by up to 94% with only one sample. The practical applications of our proposed model are evident in its ability to enhance few-shot learning performance, effectively handle novel tasks, reduce data requirements, and improve cross-domain generalization. By applying our model, users can expect improved performance in scenarios with limited training data, increased efficiency when faced with new tasks or domains, and enhanced generalization capabilities across different problem domains, e.g., rapid prototyping, adaptive systems, personalized recommendations, and anomaly detection.

In the future, we plan to enhance our model for multi-class malware classification to identify specific malware families. The challenge is that there is often limited data available for each malware family, which can lead to misclassification. One-shot learning helped train the model even with a single instance, avoiding inaccuracies caused by changes in malware variants. We also plan to expand our research to address the detection of packed malware. This involves modifying our proposed technique to effectively detect and analyze malware that has been compressed or obfuscated using packing techniques. Additionally, in future research, we can explore and harness the potential of vision transformers combined with meta-learning. By leveraging the self-attention mechanism and the ability to capture global contextual information, we can further improve the representation learning capabilities of our model. Integrating vision transformers into our framework can enhance feature extraction, enable better understanding of complex visual patterns,

and potentially lead to further advancements in areas such as few-shot learning, transfer learning, and domain adaptation.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number MoE-IF-UJ-22-04101171-2.

REFERENCES

[1] U.-E.-H. Tayyab, F. B. Khan, M. H. Durad, A. Khan, and Y. S. Lee, "A survey of the recent trends in deep learning based malware detection," *J. Cybersecurity Privacy*, vol. 2, no. 4, pp. 800–829, Sep. 2022.

[2] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[3] M. Belaoued, A. Derhab, S. Mazouzi, and F. A. Khan, "MACoMal: A multi-agent based collaborative mechanism for anti-malware assistance," *IEEE Access*, vol. 8, pp. 14329–14343, 2020.

[4] M. Belaoued, A. Boukellal, M. A. Koalal, A. Derhab, S. Mazouzi, and F. A. Khan, "Combined dynamic multi-feature and rule-based behavior for accurate malware detection," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 11, Nov. 2019, Art. no. 155014771988990.

[5] C. Kim, S. Chang, J. Kim, D. Lee, and J. Kim, "Automated, reliable zero-day malware detection based on autoencoding architecture," *IEEE Trans. Netw. Service Manag.*, early access, Mar. 1, 2023, doi: 10.1109/TNSM.2023.3251282.

[6] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, May 2021.

[7] N. Amjad, H. Afzal, M. F. Amjad, and F. A. Khan, "A multi-classifier framework for open source malware forensics," in *Proc. IEEE 27th Int. Conf. Enabling Technol., Infrastructure Collaborative Enterprises (WETICE)*, Jun. 2018, pp. 106–111.

[8] N. Tariq, M. Asim, and F. A. Khan, "Securing SCADA-based critical infrastructures: Challenges and open issues," *Proc. Comput. Sci.*, vol. 155, pp. 612–617, Jan. 2019.

[9] S. Ravi and H. Larochelle. (2016). *Optimization as a Model for Few-Shot Learning*. [Online]. Available: https://openreview.net/forum?id=rJY0-Kcll

[10] Y. Wang and Q. Yao, "Few-shot learning: A survey," 2019, *arXiv:1904.05046v1*.

[11] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.

[12] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-shot learning for semantic segmentation," 2017, *arXiv:1709.03410*.

[13] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low data drug discovery with one-shot learning," *ACS Central Sci.*, vol. 3, no. 4, pp. 283–293, Apr. 2017.

[14] Y. Pan, Y. Jing, T. Wu, and X. Kong, "Knowledge-based data augmentation of small samples for oil condition prediction," *Rel. Eng. Syst. Saf.*, vol. 217, Jan. 2022, Art. no. 108114.

[15] F. J. Moreno-Barea, J. M. Jerez, and L. Franco, "Improving classification accuracy using data augmentation on small data sets," *Exp. Syst. Appl.*, vol. 161, Dec. 2020, Art. no. 113696.

[16] A. Brodzicki, M. Piekarski, D. Kucharski, J. Jaworek-Korjakowska, and M. Gorgon, "Transfer learning methods as a new approach in computer vision tasks with small datasets," *Found. Comput. Decis. Sci.*, vol. 45, no. 3, pp. 179–193, Sep. 2020.

[17] H. Yamada, C. Liu, S. Wu, Y. Koyama, S. Ju, J. Shiomi, J. Morikawa, and R. Yoshida, "Predicting materials properties with little data using shotgun transfer learning," *ACS Central Sci.*, vol. 5, no. 10, pp. 1717–1730, Oct. 2019.

[18] S. Shen, M. Sadoughi, M. Li, Z. Wang, and C. Hu, "Deep convolutional neural networks with ensemble learning and transfer learning for capacity estimation of lithium-ion batteries," *Appl. Energy*, vol. 260, Feb. 2020, Art. no. 114296.

[19] F. A. Khan and A. Gumaei, "A comparative study of machine learning classifiers for network intrusion detection," in *Proc. 5th Int. Conf. Artif. Intell. Secur.* (Lecture Notes in Computer Science), vol. 11633. Cham, Switzerland: Springer, 2019, pp. 75–86.

[20] R. Upadhyay, R. Phlypo, R. Saini, and M. Liwicki, "Sharing to learn and learning to share—Fitting together meta-learning, multi-task learning, and transfer learning: A meta review," 2021, *arXiv:2111.12146*.

[21] J. Vanschoren, "Meta-learning: A survey," 2018, *arXiv:1810.03548*.

[22] M. Huisman, J. N. van Rijn, and A. Plaat, "A survey of deep meta-learning," *Artif. Intell. Rev.*, vol. 54, no. 6, pp. 4483–4541, Aug. 2021.

[23] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah, "Signature verification using a 'Siamese' time delay neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 6, 1993, pp. 1–8.

[24] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.

[25] J. Wang and Y. Zhai, "Prototypical Siamese networks for few-shot learning," in *Proc. IEEE 10th Int. Conf. Electron. Inf. Emergency Commun. (ICEIEC)*, Jul. 2020, pp. 178–181.

[26] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1199–1208.

[27] X. Wang, A. Liu, L. Wu, C. Li, Y. Liu, and X. Chen, "A generalized zero-shot learning scheme for SSVEP-based BCI system," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 863–874, 2023, doi: 10.1109/TNSRE.2023.3235804.

[28] S. Kadam and V. Vaidya, "Review and analysis of zero, one and few shot learning approaches," in *Proc. Int. Conf. Intell. Syst. Design Appl.*, vol. 940. Cham, Switzerland: Springer, 2018, pp. 100–112, doi: 10.1007/978-3-030-16657-1_10.

[29] P. Kumari and K. Seeja, "One shot learning approach for cross spectrum periocular verification," *Multimedia Tools Appl.*, vol. 82, pp. 20589–20604, Jan. 2023.

[30] W. Li, L. Wang, X. Zhang, L. Qi, J. Huo, Y. Gao, and J. Luo, "Defensive few-shot learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5649–5667, May 2023, doi: 10.1109/TPAMI.2022.3213755.

[31] B. Ermis, G. Zappella, and C. Archambeau, "Towards robust episodic meta-learning," in *Proc. PMLR*, 2021, pp. 1342–1351.

[32] S. Ritter, J. Wang, Z. Kurth-Nelson, S. Jayakumar, C. Blundell, R. Pascanu, and M. Botvinick, "Been there, done that: Meta-learning with episodic recall," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4354–4363.

[33] M. Woodward and C. Finn, "Active one-shot learning," 2017, *arXiv:1702.06559*.

[34] T. K. Tran, H. Sato, and M. Kubo, "Image-based unknown malware classification with few-shot learning models," in *Proc. 7th Int. Symp. Comput. Netw. Workshops (CANDARW)*, Nov. 2019, pp. 401–407.

[35] T. Munkhdalai and H. Yu, "Meta networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2554–2563.

[36] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[37] D. Das and C. S. G. Lee, "A two-stage approach to few-shot learning for image recognition," *IEEE Trans. Image Process.*, vol. 29, pp. 3336–3350, 2020.

[38] Z. Chen, Y. Fu, Y. Zhang, Y. Jiang, X. Xue, and L. Sigal, "Multi-level semantic feature augmentation for one-shot learning," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4594–4605, Sep. 2019.

[39] B. Hui, P. Zhu, Q. Hu, and Q. Wang, "Self-attention relation network for few-shot learning," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2019, pp. 198–203.

[40] M. Bishay, G. Zoumpourlis, and I. Patras, "TARN: Temporal attentive relation network for few-shot and zero-shot action recognition," 2019, *arXiv:1907.09021*.

[41] Z. Wu, Y. Li, L. Guo, and K. Jia, "PARN: Position-aware relation networks for few-shot learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6658–6666.

[42] J. He, R. Hong, X. Liu, M. Xu, Z.-J. Zha, and M. Wang, "Memory-augmented relation network for few-shot learning," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 1236–1244.

[43] K. Gao, B. Liu, X. Yu, J. Qin, P. Zhang, and X. Tan, "Deep relation network for hyperspectral image few-shot classification," *Remote Sens.*, vol. 12, no. 6, p. 923, Mar. 2020.

[44] X. Liu, F. Zhou, J. Liu, and L. Jiang, "Meta-learning based prototype-relation network for few-shot classification," *Neurocomputing*, vol. 383, pp. 224–234, Mar. 2020.

FAIZA BABAR KHAN received the M.S. degree in computer science with machine learning. She is currently pursuing the Ph.D. degree in computer science with the Pakistan Institute of Engineering and Applied Sciences, Islamabad. Her research interests include novel malware detection using few-shot learning techniques.

MUHAMMAD HANIF DURAD received the M.Sc. degree in physics from the University of Punjab, Pakistan, the M.Sc. degree in systems engineering from Quaid-i-Azam University, Pakistan, and the Ph.D. degree in computer engineering from the Beijing Institute of Technology (BIT), China. He is currently a Professor with the Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan. He has over 40 publications in refereed international journals and conferences. He has received research grants for ICT related projects from various national funding agencies. His research interests include computer networks, cyber security, cloud computing, parallel computing, grid computing, the Internet of Things (IoT), and computer architecture.

ASIFULLAH KHAN received the M.S. and Ph.D. degrees in computer systems engineering from the GIK Institute, Pakistan.

He is a highly accomplished professor with over 24 years of teaching and research experience. He is also leading the PIEAS Artificial Intelligence Center, PIEAS. He has made significant contributions to the field of artificial intelligence with more than 120 international journal publications with over 8,000 citations. He has also published 55 conference papers and eight book chapters. He has a proven track record of supervising Ph.D. scholars, with 24 successful Ph.D. supervisions to his credit. He is a highly respected researcher and has won eight research grants as a principal investigator. He has been actively involved in deep learning research and was the Head of the Department of Computer and Information Sciences, PIEAS, from 2016 to 2020.

Dr. Khan has been listed in the world's top 2% scientists by Stanford University in both career-long and last year categories, in 2020 and 2021. He has been awarded the President's Award for Pride of Performance, in 2018, and has received four HEC's Outstanding Research Awards and one Best University Teacher Award. He has also received the PAS-COMSTECH Prize, in 2011, in Computer Science and I. T. and Research Productivity Awards from Pakistan Council for Science and Technology (PCST), in 2012 and 2016. Additionally, he won the Youm-e-Takbir Performance Gold Medal by PAEC, in May 2017.

FARRUKH ASLAM KHAN (Senior Member, IEEE) received the M.S. degree in computer system engineering from the GIK Institute of Engineering Sciences and Technology, Pakistan, in 2003, and the Ph.D. degree in computer engineering from Jeju National University, South Korea, in 2007. He is currently a Professor in cybersecurity with the Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia. He also received professional training from the Massachusetts Institute of Technology, New York University, IBM, and other professional institutions. He has over 130 publications in refereed international journals and conferences. He has co-organized several international conferences and workshops. He has successfully supervised/co-supervised six Ph.D. students and over 20 M.S. thesis students. His research interests include cyber-security, body sensor networks and e-health, computational intelligence, bio-inspired and evolutionary computation, smart grids, and the Internet of Things. His name has been listed in the world's top 2% scientists in a study conducted by Stanford University, in 2022. He is on the panel of reviewers of over 40 reputed international journals and numerous international conferences. He serves/served as an Associate Editor for prestigious international journals, including IEEE Access, *PLOS One*, *Neurocomputing* (Elsevier), *Ad-Hoc & Sensor Wireless Networks*, *KSII Transactions on Internet and Information Systems*, *Human-Centric Computing and Information Sciences* (Springer), *PeerJ Computer Science*, and *Complex and Intelligent Systems* (Springer). He is a fellow of the British Computer Society (BCS).

SAJJAD HUSSAIN CHAUHDARY received the M.S. degree from Ajou University and the Ph.D. degree from Korea University, South Korea. He is currently an Associate Professor with the College of Computer Science and Engineering (CCSE), University of Jeddah. He was with the Advance Technology Research and Development Center, LG/LSIS Company Ltd., as a Senior Research Engineer for six years. His research interests include the Industrial Internet of Things, smart cities, smart grids, cyber security, power line communications, and industrial wired/wireless communication networks.

MOHAMMED ALQARNI received the bachelor's degree in computer science from King Khalid University, Saudi Arabia, in 2008, the M.Sc. degree in computational science from Laurentian University, Sudbury, Canada, in 2012, and the Ph.D. degree in computer science from McMaster University, Hamilton, Canada, in 2016. He is currently an Associate Professor with the College of Computer Science and Engineering, University of Jeddah, Saudi Arabia.