A Multi-label Fuzzy Relevance Clustering System for Malware Attack Attribution in the Edge Layer of Cyber-Physical Networks

MOHAMMADHADI ALAEIYAN, School of Computer Engineering, Iran University of Science and Technology, Iran
ALI DEHGHANTANHA*, University of Guelph, Canada
TOOSKA DARGAHI, University of Salford, UK
MAURO CONTI, Department of Mathematics, University of Padua, Italy
SAEED PARSA, School of Computer Engineering, Iran University of Science and Technology, Iran

The rapid increase in the number of malicious programs has made malware forensics a daunting task and caused users system to become on danger. Timely identification of malware characteristics including its origin and the malware sample family would significantly limit the potential damage of the malware. This is a more profound risk in Cyber-Physical Systems (CPS) where a malware attack may cause significant physical damage to the infrastructure. Due to limited on-device available memory and processing power in CPS devices, most of the efforts for protecting CPS networks are focused on the edge layer, where the majority of security mechanisms are deployed.

Since the majority of advanced and sophisticated malware programs are combining features from different families, these malicious programs are not similar enough to any existing malware family and easily evade binary classifiers detection. Therefore, in this paper, we propose a novel multi-label fuzzy clustering system for malware attack attribution. Our system is deployed on the edge layer to provide an insight into applicable malware threats to the CPS network. We leverage static analysis by utilizing Opcode frequencies as the feature space to classify malware families.

We observed that a multi-label classifier does not classify a part of samples. We named this problem as instance coverage problem. To overcome this problem, we developed an ensemble-based multi-label fuzzy classification method to suggest the relevance of a malware instance to the stricken families. This classifier identified samples of VirusShare, RansomwareTracker, and BIG2015 with an accuracy of, 94.66%, 94.26%, and 97.56%, respectively.

Additional Key Words and Phrases: Edge layer, Cyber-Physical Systems, CPS, Internet of Things, IoT, malware classification, Fuzzy classification, Instance coverage

ACM Reference Format:
Mohammadhadi Alaeiyan, Ali Dehghantanha, Tooska Dargahi, Mauro Conti, and Saeed Parsa. 2019. A Multi-label Fuzzy Relevance Clustering System for Malware Attack Attribution in the Edge Layer of Cyber-Physical Networks. *ACM Transactions on Cyber-Physical Systems* 1, 1, Article 1 (January 2019), 22 pages. https://doi.org/0000001.0000001

*This is the corresponding author

Authors' addresses: Mohammadhadi Alaeiyan, School of Computer Engineering, Iran University of Science and Technology, Narmak, Tehran, Tehran, 16844, Iran, hadi_alaeiyan@comp.iust.ac.ir; Ali Dehghantanha, University of Guelph, 3326, Reynolds Building, Guelph, Ontario, Canada, ali@cybersciencelab.org; Tooska Dargahi, University of Salford, The Crescent, Salford, Greater Manchester, UK, t.dargahi@salford.ac.uk; Mauro Conti, Department of Mathematics, University of Padua, 35122, Padua, Italy, conti@math.unipd.it; Saeed Parsa, School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran, parsa@iust.ac.ir.

XXXX-XXXX/2019/1-ART1 $15.00
https://doi.org/0000001.0000001

1 INTRODUCTION

Malware is becoming a serious threat to our smart critical infrastructure and Cyber-Physical Systems (CPS) [23]. In 2017, the number of new malware increased by 22.9% over 2016 to 8,400,058 samples [8]. Also, the number of new malware files, processed by Kaspersky lab, increased 11.5% over 2016 to 360,000 samples [15]. This denotes that the number of malicious files is increasing. This especially is a problem in the Internet of Things (IoT) and CPS networks where a compromise could actually endanger the safety of system user·s. Due to limited processing power and memory of CPS devices, most cyber defense and security mechanisms should be implemented in the edge layer of these networks. [30]. Therefore, edge layer devices like Content Delivery Network (CDN) routers or servers will use multi-label classification to detect and prevent the malicious programs from being spirited.

The majority of malicious programs are sharing a significant amount of codes and offer similar functionalities [10, 12]. These shared codes and services are key to the investigation of malware and attack attribution activities [5]. Early identification of a malware family, could save a lot of time and resources in malware analysis [22]. Many reverse engineers are using binary classifiers to identify if a malware belongs to a family [21]. They apply static or dynamic analysis to extract features which depict the characteristics of a family of malware. For instance, Opcode sequences [6, 9, 11, 14, 37] are among the most popular features for malware analysis. However, a binary classifier only determines the family of malware. In contrast, multi-label classifiers help us to understand the differences between various malware families. Multi-label classifiers attempt to categorize a sample into families in which the samples are similar.

A. Shalaginov et al. [27] proposed a deep Neuro-Fuzzy multi-label classifier which leveraged a variety of static and dynamic features to categorize malicious programs. They collected a list of malware labeled based on Virustotal [33] to evaluate their proposed deep Neuro-Fuzzy (NF) multi-label classifier which categorized samples with an accuracy of 69.44% with simple NF [28] and deep neural network classifiers classified these samples with an accuracy of 20.105% and 62.093%, respectively. In their work, however, `C4.5` binary classifier identified samples with an accuracy of 82.85%. From their results, one can see the low performance of multi-label classifiers compared to the performance of binary classifiers.

While the accuracy of multi-label fuzzy classification techniques is not high, they are best suited to detect the level of similarities between different malware opcodes. Therefore, we will use multi-label fuzzy relevance clustering techniques to identify similarities between a given sample and existing families of malicious programs. We leveraged the Opcode frequency of malicious families as a feature set for our Multi-label Fuzzy Relevance Classification technique [17] (MFRC).

To measure the probability of categorizing a sample by a multi-label classifier, we present a new measurement called *instance coverage*. Instance coverage is $P(z) = q/n$, where q is the number of samples which are not classified by the multi-label classifier as the member of any family and n is the total number of samples.

We proposed a new relevance fuzzy multi-label classification algorithm called Multi-label Fuzzy Selective Relevance classifier (MFSRC) which also addresses the instance coverage problem in multi-label classification. As part of our evaluation process, the classification performances of both algorithms, MFRC and MFSRC, were evaluated by *accuracy*, *precision*, *recall*, $F1$, breakeven point (*BEP*) [26], hamming loss (*Hloss*) [29] and *MCC* metrics which are calculated as shown in Table 1. In Table 1, true positive, TP_i, is the number of samples of c_i family that they are correctly classified as a member of c_i family. False positive, FP_i, is the number of samples of the c_i family that they are incorrectly classified as a member of other families. The false negative, FN_i, is the number of samples of non-c_i families which are incorrectly classified as a member of the c_i family. Also, true negative (TN_i) is the number of samples of non-c_i families that they are correctly classified as a member of the non-c_i family.

To overcome the instance coverage problem, we built a fuzzy multi-label classifier. For evaluating our work we created several malware datasets from VirusShare [32], RansomwareTracker [1] and utilized BIG2015 [20]

Table 1. Performance metrics

Metric	Formula
$Accuracy$	$\frac{TP+TN}{TP+FP+TN+FN}$
$Precision$	$\frac{\sum_{i=1}^{c} TP_i}{\sum_{i=1}^{c}(TP_i+FP_i)}$
$Recall$	$\frac{\sum_{i=1}^{c} TP_i}{\sum_{i=1}^{c}(TP_i+FN_i)}$
$F1$	$\frac{2\times Precision\times Recall}{Precision+Recall}$
$Hloss$	$\frac{\sum_{i=1}^{c}(FP_i+FN_i)}{c\times n}$
BEP	$\frac{Precision+Recall}{2}$
$P(z)$	q/n
MCC	$\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

samples. Our proposed system is best suited for deployment on the edge layer of CPS networks. Our results on BIG2015 indicate an accuracy of 97.56%, precision of 90.68%, and f-measure of 89.21%. Also, our results on with dataset created from RansomwareTracker reveal an accuracy of 94.26%, a precision of 87.21%, and f-measure of 83.52%. Moreover, our results with the dataset created by VirusShare demonstrate an accuracy of 94.66%, a precision of 86.46%, and f-measure of 84.37%.

Contributions. The main contributions of this paper are as follows:

- We leverage a relevance fuzzy multi-label classification algorithm to predict similarities between a given sample and known families of malicious programs. We use single labeled datasets and classify them using multi-label and binary classifiers. We utilize Opcode frequency as a feature set to segregate malware families.
- Existing multi-label fuzzy classifiers suffer from the instance coverage problem which leads to many samples being not classified in any family. To remedy this instance coverage problem, we propose a new multi-class fuzzy relevance classification algorithm in Section 3 and provide ensembles based on two relevance fuzzy multi-label classification algorithm.

Organization. The rest of this paper is organized as follows. Malware feature extraction and data preparation are presented in Section 2. Our proposed fuzzy clustering is presented in Section 3. The performance of the proposed method is presented in Section 4. Section 5 explains the comparison of our method with related works. Further, the conclusion is presented in Section 6.

2 DATA PREPARATION AND EXPERIMENT SETUP

Our multi-classifier identifier model has two engines, identifier, and analyzer. The identifier engine runs in an edge layer device and analyzer engine is executed within a centralized server. While a new malicious program is detected as any analyzed malicious family, the edge layer device blocks its network traffics and sends the program to the analyzer engine for future learning tasks. Thus, the analyzer engine provides updates on the identifier engine by analyzing collected datasets. The identifier engines on the edge layer are updated on a specific interval or upon detection of a major attack.

We used three datasets in this research. Our datasets include malware samples which have targeted IoT and CPS networks in the past. The samples which are not packed and are able to be disassembled is selected to be analyzed. The first dataset includes 555 Ransomware samples collected from RansomwareTracker [1] and categorized into 6 families as shown in Table 4. The second dataset contains 9601 disassembled malware samples categorized into 9 families as part of the Microsoft Malware Classification Challenge (BIG2015) [20]. The third

dataset includes 5449 malware samples of six APT families shown in Table 2 collected from VirusShare [32]. For the sake of brevity, we named these datasets as RT, BIG2015 and VS, respectively. RansomwareTracker categorized Ransomware based on the similarity of malicious network communication. It listed the MD5 of malware instances. We collected MD5 of all malware samples until 31-1-2018 and used the MD5s to collect malicious executable files from Virustotal [33] website. We have also leveraged `ExeInfo PE` [2] to determine whether a malicious program is packed or not. Additionally, we utilized `IDA pro` [7] version 6.5 to extract applications Opcodes. We also developed an auxiliary tool to extract the frequency of Opcodes from the disassembled programs.

Table 2. VisusShare (VS) dataset family [33].

#	Family name	Count
1.	APT1_293	271
2.	Citadel-Zeus_PE-Arc_20130113-20130712	661
3.	CryptoRansom_201607153	1640
4.	InstallCore_000	1700
5.	Locker_20150505	137
6.	Mediyes_000	1040

Table 3. Microsoft Malware Classification Challenge (BIG2015) dataset family [20].

#	Family name	Count
1.	Ramnit	1367
2.	Lollipop	2182
3.	Kelihos_ver3	2628
4.	Vundo	387
5.	Simda	37
6.	Tracur	662
7.	Kelihos_ver1	347
8.	Gatak.ACY	1076
9.	Obfuscator.ACY	915

Table 4. RansomwareTracker (RT) dataset family [1].

#	Family name	Count
1.	Cerber	54
2.	CryptoWall	107
3.	CTB-Locker	46
4.	Locky	140
5.	Sage	33
6.	TeslaCrypt	175

To resemble processing power that is commonly available on the edge layer, we run all our experiments on a PC with Intel Core i7-CPU @ 3.3 GHz and 8GB of physical memory running Windows 10 version 1803 (OS build 17134.165). We used Weka 3.8's [35] implementation of Random Forest, J48 and Naive Bayes for testing.

Moreover, we implemented our proposed multi-label fuzzy relevance classifier and the proposed fuzzy relevance classifier based on the algorithm suggested by [17], in Matlab [19] version R2016(9.1.0.441655).

3 PROPOSED FUZZY CLUSTERING SYSTEM

A task which groups similar objects into one set (cluster) is called clustering. Multi-label fuzzy clustering is a form of clustering in which each object may belong to more than one cluster. In this study, executable files are considered as objects. A triplet (E, T, C) is an input parameter, where

$$E = \{(e^{(1)}, \mathbf{y}^{(1)}), (e^{(2)}, \mathbf{y}^{(2)}), \ldots, (e^{(n)}, \mathbf{y}^{(n)})\}, \tag{1}$$

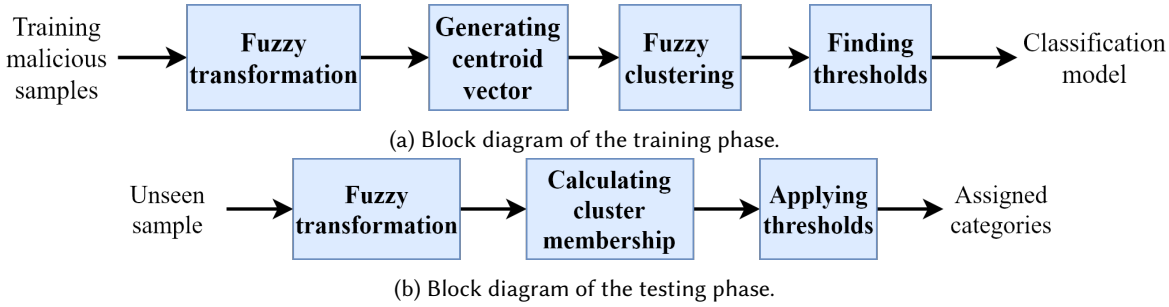
is a set of n training patterns, $T = \{t_1, t_2, \ldots, t_m\}$ is a set of m features, and $C = \{c_1, c_2, \ldots, c_p\}$ is a set of p families. A training pattern $e^{(i)}$, where $1 \leq i \leq n$, contains class label $\mathbf{y}^{(i)} = \{y_1, y_2, \ldots, y_p\}$. y_k, where $1 \leq k \leq p$, is equal to 1 if the executable files belongs to a given family c_k, otherwise it is equal to 0. In a binary classification, only one component of $\mathbf{y}^{(i)}$ is equal to 1. In a multi-label classification, however, several component of $\mathbf{y}^{(i)}$ may set to 1.

Symbols, which are used in this paper, are listed in Table 5.

Table 5. Symbols which are used in this paper.

Symbol	Discription
n	The number of samples.
p	The number of categories.
C	A set of categories, $C = \{c_1, c_2, \ldots, c_p\}$.
m	The number of features.
T	A set of features, $T = \{t_1, t_2, \ldots, t_m\}$.
$\omega_i^{(j)}$	The i^{th} attribute value of j^{th} sample.
t_j	j^{th} attribute, $t_j = \langle \omega_1^{(j)}, \omega_2^{(j)}, \cdots, \omega_m^{(j)} \rangle$.
$e^{(i)}$	The i^{th} sample.
$y_k^{(i)}$	Membership value of the k^{th} category of the i^{th} sample, $y_k^{(i)} \in \{0, 1\}$.
$\mathbf{y}^{(i)}$	A set of category labels of the i^{th} sample, $\mathbf{y}^{(i)} = \{y_1^{(i)}, y_2^{(i)}, \ldots, y_p^{(i)}\}$.
E	A set of samples and labels, $E = \{(e^{(1)}, \mathbf{y}^{(1)}), (e^{(2)}, \mathbf{y}^{(2)}), \ldots, (e^{(n)}, \mathbf{y}^{(n)})\}$.
x	Fuzzy relevance vector, $x = \{x_1, x_2, \ldots, x_p\}$.
d	The number of similar labeled groups or the number of families.
χ	A set of similar label fuzzy relevance vectors, $\chi = \{(X^{(1)}, \mathbf{y}^{(1)}), (X^{(2)}, \mathbf{y}^{(2)}), \ldots, (X^{(n)}, \mathbf{y}^{(q)})\}$ where q is $\lvert\chi\rvert$.
θ	A centroid vector, $\theta = \langle x_1, x_2, \ldots, x_p \rangle$.
ζ	A set of centroid vectors, $\zeta = \{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(d)}\}$.
A	A $n \times d$ cluster similarity matrix.
τ_j	Threshold of j^{th} category.
o_j	Output of j^{th} category.
λ	A set of candidate threshold vectors.
$O_{k,j}$	A vector of output value computed by k^{th} candidate threshold vector.
$\mathbf{Y}_j$	A set of all actual j^{th} category membership value of n samples.

Our proposed Multi-label Fuzzy Selective Relevance Clustering (MFSRC) system categorizes multi-labeled data in two phases, training, and testing. Training phase contains four steps, fuzzy transformation, generating centroid



(a) Block diagram of the training phase.

(b) Block diagram of the testing phase.

Fig. 1. Block diagram of the training and testing phases of our method.

vectors, fuzzy clustering and finding thresholds which are shown in Figure 1a, and testing phase contains three steps, fuzzy transformation, calculating cluster memberships, and applying thresholds as shown in Figure 1b.

Fuzzy transformation step transforms a high-dimensional fuzzy training data to a low-dimensional fuzzy relevance vector. Generating centroid vectors step selects one fuzzy relevance vector in each family by looking for fuzzy relevance vectors which have the minimum Euclidean distance from the median fuzzy relevance vectors. Next, in the fuzzy clustering, we leverage Cosine similarities between selected vectors and fuzzy relevance vectors to determine membership of a sample to a fuzzy cluster. Finally, in finding thresholds step, a set of thresholds are obtained to be used in the testing phase.

In the testing phase, the fuzzy transformation step transforms a high-dimensional fuzzy testing data to a low-dimensional fuzzy relevance vector. Next, in calculating cluster membership step cosine similarities between centroid vectors and fuzzy relevance vector is calculated to identify cluster membership vector. Finally, in applying the thresholds step, assigned families are determined by comparing cluster membership vector with the threshold vector.

Fuzzy transformation, finding thresholds and applying thresholds steps are similar to the fuzzy relevance clustering method for multi-label text classification [17]. Thus, generating centroid vectors, fuzzy clustering, and cluster memberships steps are our contributions to this research.

3.1 Fuzzy Transformation

During fuzzy transformation step we reduced the number of features, m, in T, to p features and transfer feature space to a fuzzy relevance vector. p is the number of families in C which usually is smaller than m. We utilized the fuzzy transformation approach suggested by S. Lee et al. [17] for feature reduction. Three fuzzy transformations μ_{R_1}, μ_{R_2}, and μ_S provide fuzzy relevance vectors of samples.

$R_1 : T \times C \rightarrow [0, 1]$ is the first fuzzy relation which reveals the relevance of features to families. $\mu_{R_1}(t_i, c_j)$, shown in Equation 2, specifies the degree of relevance of features $t_i \in T$ to category $c_j \in C$.

$$\mu_{R_1}(t_i, c_j) = \frac{\sum_{v=1}^{n} \omega_i^{(v)} y_j^{(v)}}{\sum_{v=1}^{n} \omega_i^{(v)}} \frac{\sum_{v=1}^{n} h_0(\omega_i^{(v)}) y_j^{(v)}}{\sum_{v=1}^{n} y_j^{(v)}}, \tag{2}$$

where $1 \leq i \leq m$, $1 \leq j \leq p$ and $h_0(x)$ is shown in Equation 3.

$$h_0(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0. \end{cases} \tag{3}$$

$R_2 : T \times E \rightarrow [0, 1]$ is the second fuzzy relation with notation $\mu_{R_2}(t_i, e)$. It specifies the degree of relevance of feature t_i to file $e = \langle \omega_1, \omega_2, \ldots \omega_v \rangle$ and is defined in Equation 4 for $1 \leq i \leq m$.

$$\mu_{R_2}(t_i, e) = \frac{\omega_i}{\max_{1 \leq v \leq m} \omega_v}. \tag{4}$$

Finally, $S : D \times C \rightarrow [0, 1]$ is the third fuzzy relation with notation $\mu_s(e, c_j)$. It specifies the degree of relevance of feature between file e to category c_j and is defined in Equation 5 for $1 \leq j \leq p$.

$$\mu_S(e, c_j) = \frac{\sum_{i=1}^{m} \top(\mu_{R_1}(t_i, c_j), \mu_{R_2}(t_i, e))}{\sum_{i=1}^{m} \bot(\mu_{R_1}(t_i, c_j), \mu_{R_2}(t_i, e))}. \tag{5}$$

where $\top$ and $\bot$ are shown in Equation 6 and Equation 7, respectively.

$$\top(x, y) = x \times y. \tag{6}$$

$$\bot(x, y) = x + y - x \times y. \tag{7}$$

In conclusion, a fuzzy relevance vector, which is the reduced feature space of a file e, is defined by

$$x = \langle \mu_S(e, c_1), \mu_S(e, c_2), \ldots, \mu_S(e, c_p) \rangle. \tag{8}$$

3.2 Generating Centroid Vectors

Centroid vectors are obtained to determine the difference between the samples. To obtain the centroid vectors, suppose d is the number of similar labelled groups that the member of groups are fuzzy relevance vectors with similar labels (i.e., d is the number of families). In this step, we compute d vectors to be used in fuzzy clustering step. d vectors are the average attribute of the families. Suppose $\chi_l = \{(x^{(1)}, \mathbf{y}^{(1)}), (x^{(2)}, \mathbf{y}^{(2)}), \ldots, (x^{(q)}, \mathbf{y}^{(q)})\}$, where $1 \leq l \leq d$ and q is $|\chi_l|$, is a group of fuzzy relevance vectors with similar labels, $\mathbf{y}^{(i)}$ where $1 \leq i \leq n$. Equation 9 calculates the l^{th} centroid vector, $\theta^{(l)} = \langle x_1^{(l)}, x_2^{(l)}, \ldots, x_p^{(l)} \rangle$ to collect a set of centroid vectors $\zeta = \{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(d)}\}$.

$$x_j^{(l)} = \frac{\sum_{i=0}^{n} x_j^{(i)}}{q}, \tag{9}$$

where $1 \leq j \leq p$ and $x_j^{(i)}$ is the j^{th} fuzzy relevance value of i^{th} which is member of χ_l.

3.3 Fuzzy Clustering

We cluster the fuzzy relevance vectors by comparing them with ζ. As shown in Equation 10, we use Cosine similarity and invert cosine to compare fuzzy relevance vectors with centroid vectors. We use invert cosine to scale the output of Cosine similarity and increase the accuracy of clustering. We keep the similarity results into a matrix, A. A is a $n \times d$ matrix where each row shows the similarity of a fuzzy relevance vector and centroid vectors. Then, rows show fuzzy relevance vectors and columns show centroid vectors. Finally, those fuzzy relevance vectors which have equal row can be categorized in the same cluster. However, we just need A to find suitable thresholds for the classification.

$$A_{ij} = ArcCosine(CosineSimilarity(X^{(i)}, \theta^{(j)})). \tag{10}$$

where $1 \leq i \leq d$ and $1 \leq j \leq d$. Also Equation 11 shows the $CosineSimilarity$ function and Equation 12 shows the invert cosine function, $ArcCosine$.

$$CosineSimilarity(X, \theta) = \frac{X.\theta}{\|X\|\|\theta\|}. \tag{11}$$

$$ArcCosine(b) = \frac{\pi}{2} - \sum_{l=0}^{\infty} \frac{(\frac{1}{2})_{l-1}}{(l-1)!(2l-1)} b^{2l-1}. \tag{12}$$

3.4 Finding Thresholds

To determine the family of a sample, an output vector, $\langle o_1, o_2, \ldots, o_p \rangle$, is provided by applying threshold on A. The values of output vector are member of $\{0, 1\}$ and they determine whether a sample is belong to a family or not. As shown in Equation 13, a vector of threshold, $\langle \tau_1, \tau_2, \ldots, \tau_p \rangle$, maps A to the o_j where $1 \leq j \leq p$. In this equation, τ_j is the threshold of c_j where $1 \leq j \leq p$.

$$o_j = \begin{cases} 1, & \text{if } a_j > \tau_j \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The threshold vector, $\langle \tau_1, \tau_2, \ldots, \tau_p \rangle$, is calculated by using A and $\mathbf{y}$ as follow. Let $\psi_j^{(1)}, \psi_j^{(2)}, \ldots, \psi_j^{(n)}$ be similarity of all samples with centroid vector which is member of c_j. It means, $\psi_j^{(1)}, \psi_j^{(2)}, \ldots, \psi_j^{(n)}$ are j^{th} column of A. Next, they are sorted in ascending order to provide candidate threshold vectors. These candidate threshold vectors are computed by Equation 14. Then, Hamming loss [25] determines the performance of each candidate threshold vectors.

$$\lambda_{k,j} = \begin{cases} -P, & \text{if } k = 0 \\ \frac{\psi_j^{(k)} + \psi_j^{(k+1)}}{2}, & \text{if } 0 < k < n \\ P, & \text{if } k = n. \end{cases} \tag{14}$$

By applying candidate threshold vectors on A, $O_{k,j}$ is calculated such that $o_{k,j}^{(i)} \in 0, 1$ where $1 \leq i \leq n$, $1 \leq j \leq p$ and $1 \leq k \leq n + 1$.

$$O_{k,j} = \{o_{k,j}^{(1)}, o_{k,j}^{(2)}, \ldots, o_{k,j}^{(n)}\}. \tag{15}$$

Finally, $\tau_j = \lambda_{b,j}$ where $1 \leq j \leq p$. Equation 16 computes b.

$$b = \arg \min_{k=0 \sim n} hloss(O_{k,j}, \mathbf{y}_j). \tag{16}$$

Let $S = \{s_1, s_2, \ldots, s_q\}$ and $T = \{t_1, t_2, \ldots, t_q\}$. The $hloss(S, T)$ is defined in Equation 17 that $\oplus$ is XOR boolean logic operator.

$$hloss(S, T) = \frac{1}{q} \sum_{i=1}^{q} s_i \oplus t_i. \tag{17}$$

3.5 Training and Testing Phases

As shown in Figure 1, there are training and testing phases to predict a new instance. Four steps which are shown in Figure 1a are performed in the training phase. Suppose there is a set of samples that features of all samples are extracted, then:

(1) Fuzzy transformation. Equation 2 calculates $\mu_{R_1}(t_i, c_j)$ where $1 \leq i \leq m$ and $1 \leq j \leq p$. Also, Equation 4 computes $\mu_{R_2}(t_i, e^{(j)})$ where $1 \leq i \leq m$ and $1 \leq j \leq n$. Then, fuzzy relevance vectors $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ are provided by Equation 5 and Equation 8.
(2) Generating centroid vectors. The similar label fuzzy relevance vectors are grouped and their centroid vectors, $\zeta = \{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(d)}\}$, are computed by Equation 9.

(3) Fuzzy clustering. Cluster similarity vectors are computed by Equation 10. Those similar cluster similarity vectors are member of same clusters.
(4) Finding thresholds. The threshold vector, $\langle \tau_1, \tau_2, \ldots, \tau_p \rangle$, is obtained by selecting optimal thresholds for each category c_j where $1 \leq j \leq p$.

The testing phrase of an unseen sample e^t is provided by applying steps of Figure 1b. Three steps, fuzzy transformation, cluster memberships, and applying threshold, are described here:

(1) Fuzzy transformation. μ_{R_1} was obtained in training phase. Also, Equation 4 computes $\mu_{R_2}(t_i, e^{(t)})$ where $1 \leq i \leq m$ and $1 \leq j \leq n$. Then, fuzzy relevance vectors $x^{(t)}$ are provided by Equation 5 and Equation 8.
(2) Cluster memberships. Cluster similarity vector of $x^{(t)}$ are computed by Equation 10.
(3) Applying thresholds. The output vector $\langle o_1, o_2, \ldots, o_p \rangle$ is provided by applying thresholds $\langle \tau_1, \tau_2, \ldots, \tau_p \rangle$ on Cluster similarity vector of $x^{(t)}$ according to the Equation 13.

Therefore, o_j, where $1 \leq j \leq p$, determines whether e^t is member of c_j or not. If $o_j = 1$ then e^t is member of c_j. In contrast, if $o_j = 0$ then e^t is not member of c_j.

The computational cost of this method depends on the steps that should be taken in the method. Fuzzy transformation cost is $O(nmp)$, cost of generating centroid vectors is $O(n)$ because it needs to group similar labeled fuzzy relevance vectors. In addition, cost of finding thresholds step is $O(np)$. Therefore, the total cost is $O(nmp + n + np)$ while the total cost of MFRC [17] is $O(nmp + nJp + nJ^2p + np)$. Also, the cost of testing phase is $O(mp + p)$.

3.6 Example

In the following example, we show how our method works. Let $T = \{t_1, t_2, t_3, t_4, t_5, \}$, $C = \{c_1, c_2\}$ and

$$E : \left\{ \begin{array}{ll} (e^{(1)} = \langle 5, 37, 19, 5, 44 \rangle, & y^{(1)} = \langle 1, 0 \rangle) \\ (e^{(2)} = \langle 13, 21, 28, 11, 50 \rangle, & y^{(2)} = \langle 1, 0 \rangle) \\ (e^{(3)} = \langle 1, 45, 38, 19, 44 \rangle, & y^{(3)} = \langle 1, 0 \rangle) \\ (e^{(4)} = \langle 36, 21, 45, 27, 19 \rangle, & y^{(4)} = \langle 0, 1 \rangle) \\ (e^{(5)} = \langle 22, 34, 32, 11, 46 \rangle, & y^{(5)} = \langle 0, 1 \rangle) \\ (e^{(6)} = \langle 44, 39, 25, 38, 25 \rangle, & y^{(6)} = \langle 0, 1 \rangle) \end{array} \right\}$$

with $n = 6$, $m = 5$, and $p = 2$. We want to classify $e^{t_1} = \langle 3, 15, 12, 3, 19 \rangle$, $y^{(t_1)} = \langle 1, 0 \rangle$ and $e^{t_2} = \langle 31, 1, 84, 61, 59 \rangle$, $y^{(t_2)} = \langle 0, 1 \rangle$

According to the Equation 2, we calculate $\mu_{R_1}(t_i, c_j)$ where $1 \leq i \leq 5$ and $1 \leq j \leq 2$. The results of $\mu_{R_1}(t_i, c_j)$ are shown in Equation 18. Therefore, $\mu_{R_1}(1, 1) = 0.1570$, $\mu_{R_1}(1, 2) = 0.8429$, $\mu_{R_1}(2, 1) = 0.5228$, $\cdots$, $\mu_{R_1}(5, 2) = 0.3947$. Also, we calculate $\mu_{R_2}(t_i, e_k)$ according to Equation 4, where $1 \leq i \leq 5$ and $1 \leq k \leq 6$. The results of $\mu_{R_2}(t_i, e_k)$ are shown in Equation 19. Therefore, $\mu_{R_2}(1, 1) = 0.1136$, $\cdots$, $\mu_{R_2}(5, 5) = 1$, $\mu_{R_2}(5, 6) = 0.5681$.

$$\mu_{R_1}(t_i, c_j) = \begin{bmatrix} 0.1570 & 0.8429 \\ 0.5228 & 0.4771 \\ 0.4545 & 0.5454 \\ 0.3153 & 0.6846 \\ 0.6052 & 0.3947 \end{bmatrix} \tag{18}$$

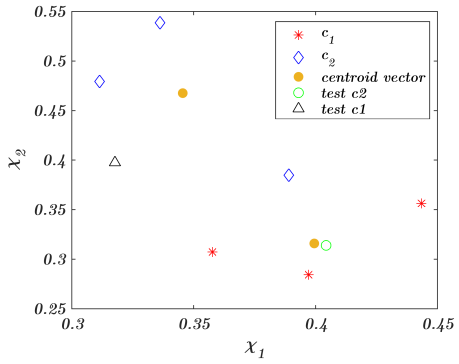


Fig. 2. Distribution of fuzzy relevance vectors and centroid vectors.

$$\mu_{R_2}(t_i, e_k) = \begin{bmatrix} 0.1136 & 0.26 & 0.0222 & 0.8 & 0.4782 & 1 \\ 0.8409 & 0.42 & 1 & 0.4666 & 0.7391 & 0.8863 \\ 0.4318 & 0.56 & 0.8444 & 1 & 0.6956 & 0.5681 \\ 0.1136 & 0.22 & 0.4222 & 0.6000 & 0.2391 & 0.8636 \\ 1 & 1 & 0.9777 & 0.4222 & 1 & 0.5681 \end{bmatrix} \tag{19}$$

Thus, we calculate $\mu_S(e_k, c_j)$ according to Equation 5, where $1 \leq k \leq 6$ and $1 \leq j \leq 2$. The results of $\mu_S(e_k, c_j)$ are shown in Equation 20. Then, $\mu_S(1,1) = 0.3971, \cdots, \mu_S(6,1) = 0.2726$, $\mu_S(6,2) = 0.529$. Consequently, fuzzy relevance vectors, $x^{(k)}$ where $1 \leq k \leq 6$, of the training files are computed. For example, $x^{(3)} = \langle 0.4435, 0.3562 \rangle$ and $x^{(5)} = \langle 0.389, 0.3846 \rangle$. They are plotted in Figure 2. These fuzzy relevance vectors are reduce dimension of features. In this example, a five dimension feature space is reduced to a two dimension feature space.

$$\mu_S(e_k, c_j) = \begin{bmatrix} 0.3971 & 0.2842 \\ 0.3577 & 0.3072 \\ 0.4435 & 0.3562 \\ 0.3113 & 0.4793 \\ 0.389 & 0.3846 \\ 0.2726 & 0.529 \end{bmatrix} \tag{20}$$

Referring to the example, χ_1 and χ_2 are two groups which their labels are equal where $\chi_1 = \{(x^{(1)}, \mathbf{y}^{(1)}), (x^{(2)}, \mathbf{y}^{(2)}), (x^{(3)}, \mathbf{y}^{(3)})\}$ and $\chi_2 = \{(x^{(4)}, \mathbf{y}^{(4)}), (x^{(5)}, \mathbf{y}^{(5)}), (x^{(6)}, \mathbf{y}^{(6)})\}$. Thus, $\zeta = \{\theta^{(1)}, \theta^{(2)}\}$ where $\theta^{(1)} = \langle 0.3994, 0.3159 \rangle$ and $\theta^{(2)} = \langle 0.3455, 0.4675 \rangle$. $\theta^{(1)}$ and $\theta^{(2)}$ are two centroid vectors and are shown in Figure 2.

Afterwards, we computed A by Equation 10. The computed A contains $A(1,1) = 2.7483, \cdots, A(6,1) = 19.6908$, $A(6,2) = 4.4942$.

$$A = \begin{bmatrix} 2.7483 & 17.9449 \\ 2.3234 & 12.8731 \\ 0.4332 & 14.7633 \\ 18.6563 & 3.4597 \\ 6.3381 & 8.8584 \\ 19.6908 & 4.4942 \end{bmatrix} \tag{21}$$

To follow the example, we computed thresholds which are $T = \{3.6458, 9.8621\}$. We will use T to classify instances, $e^{(t_1)}$ and $e^{(t_2)}$.

To continue the example, we applied fuzzy transformation on $e^{(t_1)}$ and $e^{(t_2)}$. Then, $x_{e^{(t_2)}} = \langle 0.4043, 0.3138 \rangle$ and $x_{e^{(t_1)}} = \langle 0.3176, 0.3974 \rangle$. The cluster memberships of $e^{(t_2)}$ and $e^{(t_2)}$ are $\langle 0.5214, 15.718 \rangle$ and $\langle 13.0285, 2.168 \rangle$, respectively. By applying threshold vector, T, on cluster memberships of $e^{(t_1)}$ and $e^{(t_2)}$, $y^{(t_1)} = \langle 1, 0 \rangle$ and $y^{(t_2)} = \langle 0, 1 \rangle$ are achieved.

4 RESULTS AND DISCUSSION

4.1 Experiment I: Binary Classification

Contrary to the multi-label classification which classifies a sample into a set of categories, binary classification categorizes samples into one category only. To evaluate the performance of binary classification, we used machine learning classifiers [16] to classify samples of the three datasets, VS, BIG2015 and RT.

Cross-validation [16] is used to estimate the quality of a binary classification model. We employed $k - fold$ cross-validation where samples are randomly divided into k subsets. Training is ascertained with $k - 1$ instances and prediction is conducted on the last subset. The process is repeated k times and the average evaluation metrics is reported as the outcome of the entire process. Therefore, the entire subset of data appears in both the training and testing phases. Ideally the use of cross-validation assists in determining if a classification model is generalized and it is not over-fitted. In this experiment, we employed $10 - fold$ cross-validation, which is widely used in different application domains.

Results of four classifiers, RandomForest, NaiveBayes, SVM, and J45 are shown in Table 6. RandomForest is an ensemble of binary tree classifier and it achieved the highest accuracy; 88.46% for RT and 98.85% for BIG2015 and 96.7% for VS. However, NaiveBayes could classify VS, RT, and BIG2015 with an accuracy of 48.15%, 62.34%, and 63.51%, respectively. ROC curve of datasets are shown in Figure 3. RandomForest has better performance than NaiveBayes and J45. Also, NaiveBayes has the worst classification results.

4.2 Experiment II: Multi-label Classification

In this section, we evaluate the proposed fuzzy clustering system with multi-label classification techniques. There is a list of abstract names mentioned in Table 7. Table 8 shows the performance of both our proposed method (MFSRC) and multi-label Fuzzy Relevance Clustering (MFRC) [17] on VS, RT, and BIG2015. Similar to the first experiment, we have established $10 - fold$ cross-validation on VS, BIG2015 and RT which their identification performances are presented in Table 9.

ρ and ϵ are two input parameters of multi-label fuzzy relevance clustering [17]. $\rho \in [0, 1]$ is a predefined threshold which the smaller the ρ, the larger the number of clusters. Also, $\epsilon \in [0, 1]$ is a predefined threshold to limit the maximum label similarity of clusters. In our experiments, $\rho = 0.95$ and $\epsilon = 0.0001$.

As shown in Table 8 and Table 9, the accuracy of MFRC is higher than MFSRC in all datasets, but MFRC has lower precision. In fuzzy classification, a sample could be classified correctly, incorrectly, or misclassified. If a sample is correctly classified as a member of a category, it is classified correctly. If a sample is incorrectly classified

Table 6. The performances of three datasets of VS, BIG2015 and RT based on four binary classifiers, RandomForest, NaiveBayes, SVM, and J48.

Dataset	Classifier Name	Accur.%	Preci.%	Recall%	F1%	MCC%	BEP%	Hloss%
RT	RandomForest	88.46	88.5	88.5	88.5	85.4	88.5	1.92
RT	J48	78.37	78.4	78.4	78.3	72.3	78.4	3.6
RT	SVM	74.41	74.6	74.4	74.1	67	74.5	2.83
RT	NaiveBayes	62.34	69.5	62.3	63.7	56.3	65.9	6.27
BIG2015	RandomForest	98.85	98.9	98.9	98.9	98.7	98.9	0.12
BIG2015	J48	97.41	97.4	97.4	97.4	97.1	97.4	0.28
BIG2015	SVM	80.31	83.4	80.3	78.4	76	81.85	2.08
BIG2015	NaiveBayes	63.51	72.8	63.5	65	62.2	68.15	4.05
VS	RandomForest	96.7	96.7	96.7	96.6	95.8	96.7	0.54
VS	J48	94.34	94.2	94.3	94.3	92.8	94.25	0.94
VS	SVM	83.89	86.9	83.9	81.2	79.4	85.4	2.53
VS	NaiveBayes	48.15	73.6	48.2	48.1	46.2	60.9	8.64

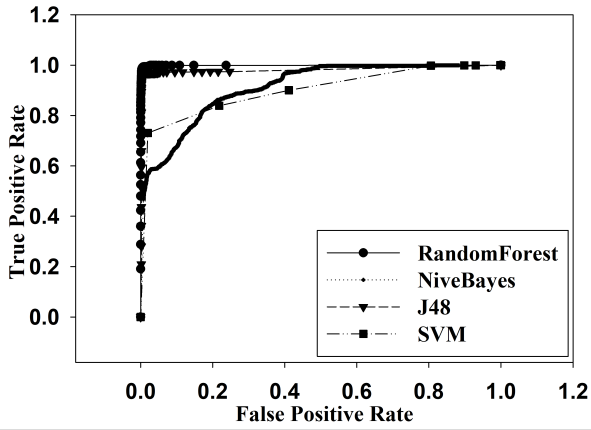
Table 7. Symbols for abstraction

Symbols	**Information**
MFRC	Fuzzy classification based on relevance fuzzy clustering [17]
MFSRC	Our proposed fuzzy method.
MFRC-MFRC	MFRC classifies samples which are not classified by MFRC.
MFRC-MFSRC	MFSRC classifies samples which are not classified by MFRC.
MFRC, MFRC-MFSRC	Map the results of MFRC and MFRC-MFSRC. Classification result of samples, which are not classified by MFRC, are replace by the classification result of MFRC-MFSRC.
MFRC-MFRC-MFSRC	MFSRC classifies samples, which are not classified by (MFRC, MFRC-MFRC)

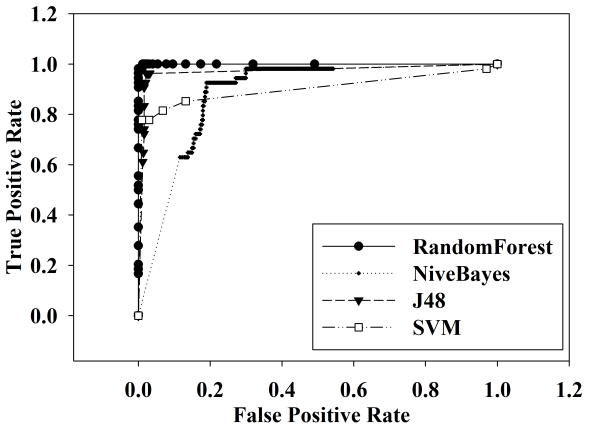
Table 8. The performances of MFSRC and MFRC on VS, BIG2015, and RT.

Dataset	Method	*Accur.%*	*Prec.%*	*Recall%*	*F1%*	*BEP%*	*Hloss%*	$P(z)$
RT	MFSRC	64.8	83.78	30.06	44.24	56.92	35.2	0.082
RT	MFRC	91.71	56.40	90.20	69.4	73.3	8.29	0.374
BIG2015	MFSRC	42.88	92.31	15.42	26.42	53.86	57.12	0.030
BIG2015	MFRC	96.9	86.24	85.9	86.07	86.07	3.1	0.377
VS	MFSRC	33.95	98.72	19.99	33.25	59.36	66.05	0.001
VS	MFRC	93.57	77.24	82.99	80.01	80.11	6.43	0.099

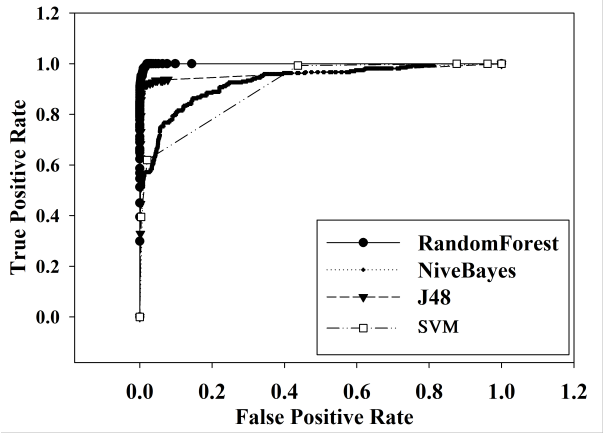
as a member of another category, it is classified incorrectly. Also, if a sample is not classified as a member of any category, it is misclassified. According to $P(z)$, MFRC misclassifies 37.4% and 37.7% of RT and BIG2015 samples, respectively. In contrast, $P(z)$ of MFSRC for RT and BIG2015 datasets are 8.2% and 3%, respectively. Therefore, MFSRC can classify more samples than MFRC. Both techniques have disadvantages as well, MFRC has lower



(a) ROC of binary classifiers on BIG2015.



(b) ROC of binary classifiers on RT.



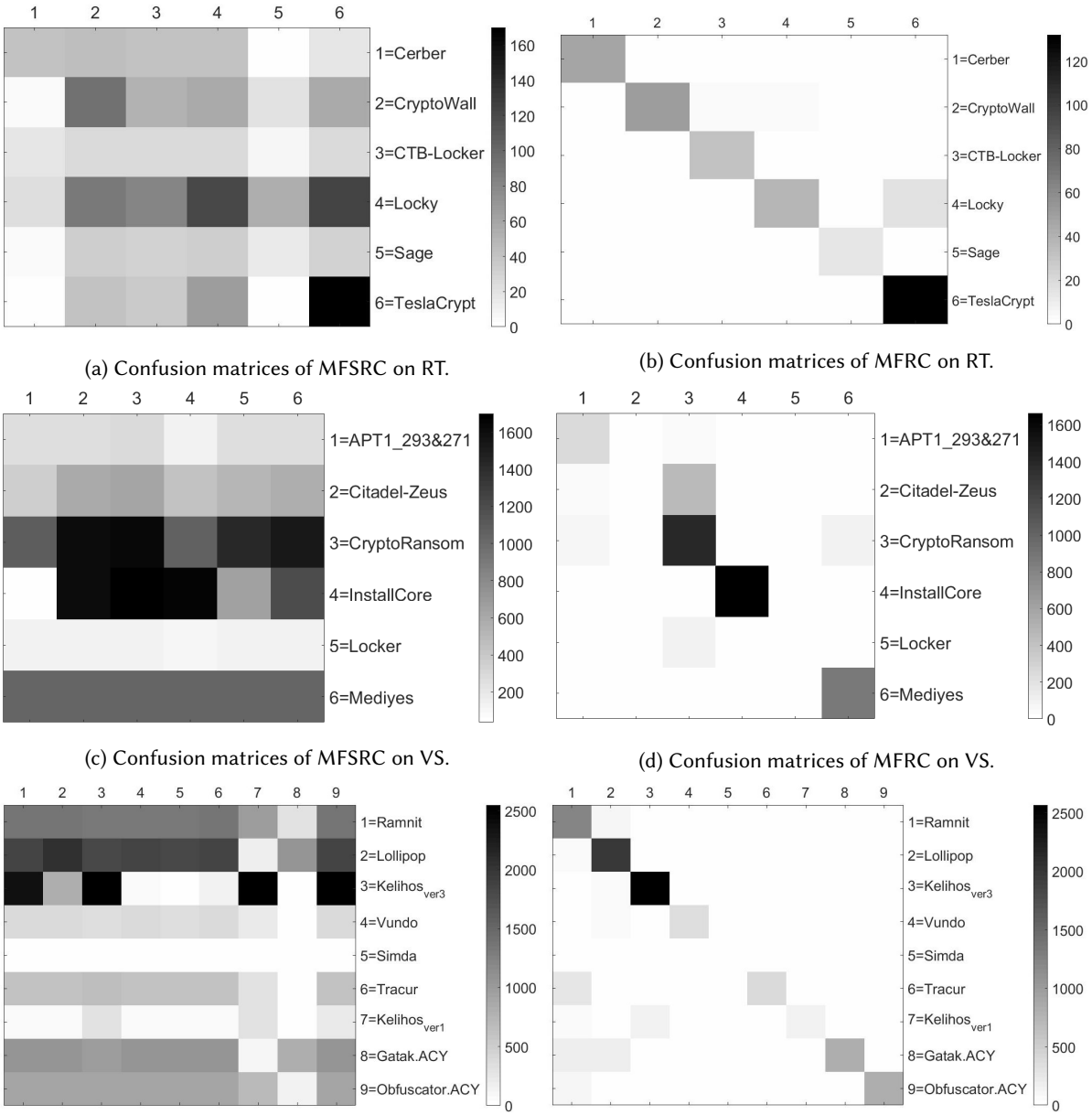
(c) ROC of binary classifiers on VS.

Fig. 3. RandomForest, J48 and NaiveBayes ROCs of VS, BIG2015, and RT datasets.

precision and MFSRC has lower recall (see Table 9 and Figure 4). To overcome this problem, the ensembles of these two techniques are evaluated in Table 12, Table 14, and Table 16.

Confusion matrices of MFRC and MFSRC classification results for the databases of VS, RT and BIG2015 are shown in Figure 4. Since samples can be members of each of the families, the confusion matrix is not diagonal. As shown in Figure 4a, Figure 4c and Figure 4e, MFSRC confusion matrices are less diagonal and strive to increase the precision values. In contrast, confusion matrices of MFRC are shown in Figure 4b, Figure 4d and Figure 4f are diagonal and strive to increase the recall values.

Figure 4a disputes the confusion matrix resulted by MFSRC on RT. 40 of 54 Cerber samples are detected correctly. Also, 45, 42, 40, 2, 17 of these samples are categorized into CryptoWall, CTB-Locker, Locky, Sage, and TeslaCrypt, respectively. In contrast, 3, 17, 22, 5 samples of CryptoWall, CTB-Locker, Locky, and Sage, are



(a) Confusion matrices of MFSRC on RT.

(b) Confusion matrices of MFRC on RT.

(c) Confusion matrices of MFSRC on VS.

(d) Confusion matrices of MFRC on VS.

(e) Confusion matrices of MFSRC on BIG2015.

(f) Confusion matrices of MFRC on BIG2015.

Fig. 4. Confusion matrices of MFSRC and MFRC on VS, RT and BIG2015 datasets.

Table 9. The performances of $10-fold$ classification of Deep neural Network (DNN), MFSRC and MFRC on VS, BIG2015, and RT.

Dataset	Method	*Accur.%*	*Prec.%*	*Recall%*	*F1%*	*BEP%*	*Hloss%*	$P(z)$
RT	MFSRC	69.81	83.02	33.88	48.03	58.45	30.19	0.052
RT	MFRC	91.04	57.74	83.67	68.02	70.70	8.96	0.32
RT	DNN	62.34	69.5	62.3	63.7	65.9	10.2	0.06
BIG2015	MFSRC	40.37	93.78	15.02	25.9	54.40	59.63	0.025
BIG2015	MFRC	97.04	87.28	86.27	86.77	86.78	2.96	0.4
BIG2015	DNN	63.51	72.8	63.5	86.77	68.15	9.43	0.061
VS	MFSRC	38.2	98.69	21.1	34.76	59.89	61.8	0.001
VS	MFRC	94.06	77.89	85.22	81.39	81.55	5.94	0.111
VS	DNN	48.15	73.6	48.2	48.1	60.9	49.34	0.521

categorized as Cerber. From Table 10, we see that MFSRC wrongly categorizes RT samples as three families CryptoWall, Locky, and TeslaCrypt. In addition, 9, 7, 19, 8, and 2 of Cerber, CryptoWall, CTB-Locker, Locker, and TeslaCrypt samples did not categorize as any category.

Table 10. Incorrectly classified samples by MFSRC on RT. Here, malware families in the first column exhibit similarity with one or more families in the second column. The families in the second column are arranged in descending order of the similarity.

Name	Incorrectly classified as
Cerber	**CryptoWall**, CTB-Locker, Locky, TeslaCrypt, Sage.
CryptoWall	**TeslaCrypt**, **Locky**, CTB-Locker, Sage, Cerber.
CTB-Locker	**CryptoWall**, Locky, TeslaCrypt, Cerber, Sage.
Locky	**TeslaCrypt**, CryptoWall, CTB-Locker, Sage, Cerber.
Sage	**Locky**, **CryptoWall**, CTB-Locker, TeslaCrypt, Cerber.
TeslaCrypt	**Locky**, CryptoWall, CTB-Locker, Sage.

Figure 4f disputes the confusion matrix resulted by MFRC on RT. 132 of 175 TeslaCrypt samples are detected correctly. Also, 2, 1, and 1 of these samples are categorized into CryptoWall, CBT-Locker, and Sage, respectively. In contrast, 16 of Locky samples are categorized as TeslaCrypt. From Table 11 we see that MFRC wrongly categorizes RT samples as three families Locky, CryptoWall, and TeslaCrypt. In addition, respectively, 7, 50, 12, 104, 17 and 39 of Cerber, CryptoWall, CBT-Locker, Locker, Sage, and TeslaCrypt samples did not categorize as any category.

The intersection of mis-classifications of both algorithms, MFSRC and MFRC, on RT is 4 samples which are members of CryptoWall. This particular outcome has led us to ensemble the classifiers.

Table 12 shows three ensembles (MFRC, MFRC-MFSRC), (MFRC, MFRC-MFRC), and (MFRC, MFRC-MFRC-MFSRC). As shown in Table 8, MFRC classified RT with an accuracy of 91.71% but 37.4% of samples are misclassified and again they are classified by MFSRC with an accuracy of 64% and $P(z) = 0.048$. Therefore, the total accuracy and mis-classification probability of (MFRC, MFRC-MFSRC) are 84.65% and 0.018, respectively. Also, another ensemble, (MFRC, MFRC-MFRC), classified RT samples with an accuracy of 95.08% and $P(z) = 0.05$ which are the results of mapping MFRC and (MFRC-MFRC). Since the value of $P(z)$ of (MFRC, MFRC-MFSRC) is less than the value of $P(z)$ of (MFRC, MFRC-MFRC), the ensemble of (MFRC, MFRC-MFRC-MFSRC) is provided with $P(z) = 0.0036$. Also, Table 13 shows the result of $10-fold$ and three aforementioned ensembles on RT, which shows that (MFRC, MFRC-MFRC-MFSRC) with $P(z) = 0$ has the best performance.

Table 11. Incorrectly classified samples by MFRC on RT. Here, malware families in the first column exhibit similarity with one or more families in the second column. The families in the second column are arranged in descending order of the similarity.

Name	Incorrectly classified as
Cerber	-
CryptoWall	**Locky**, **CTB-Locker**.
CTB-Locker	**Locky**.
Locky	**TeslaCrypt**, CTB-Locker.
Sage	**Locky**.
TeslaCrypt	**CryptoWall**, CTB-Locker, Sage.

Table 12. Ensemble performances of RT dataset.

#	Method	*Accur.%*	*Prec.%*	*Recall%*	*F1%*	*BEP%*	*Hloss%*	$P(z)$
1	MFRC-MFSRC	64.5	82.69	29.71	43.71	56.2	35.5	0.048
2	MFRC, MFRC-MFSRC	84.65	87.39	52.38	65.5	69.88	15.35	0.0180
3	MFRC-MFRC	92.31	75.96	77.45	76.7	76.71	7.69	0.1346
4	MFRC, MFRC-MFRC	95.08	84.86	85.48	85.17	85.17	4.92	0.0504
5	MFRC-MFRC-MFSRC	67.26	46.43	24.53	32.1	35.48	32.74	0.714
6	MFRC, MFRC-MFRC-MFSRC	94.26	87.21	80.13	83.52	83.67	5.74	0.0036

Table 13. Ensemble performances of $10-fold$ classification of RT dataset.

#	Method	*Accur.%*	*Prec.%*	*Recall%*	*F1%*	*BEP%*	*Hloss%*	$P(z)$
1	MFRC, MFRC-MFSRC	89.81	74.69	69.32	71.49	72.01	10.19	0.065
2	MFRC, MFRC-MFRC	95.57	88.49	85.50	86.96	86.99	4.43	0.056
3	MFRC, MFRC-MFRC-MFSRC	95.53	88.44	85.40	86.87	86.92	4.47	0

Also, similar results are achieved by classifying the BIG2015 based on three ensembles (MFRC, MFRC-MFSRC), (MFRC, MFRC-MFRC), and (MFRC, MFRC-MFRC-MFSRC), which are shown in Table 14. Although the $P(z)$ of (MFRC, MFRC-MFSRC) is 0, the accuracy of (MFRC, MFRC-MFRC-MFSRC) is higher. Moreover, Table 15 shows the results of $10-fold$ and these three ensembles on BIG2015, which clearly shows that (MFRC, MFRC-MFRC-MFSRC) with $P(z) = 0.001$ has the best performance.

Moreover, Table 16 shows the results of these three ensembles on VS, which 0.4068, 0.0149 and 0.003 are $P(z)$ of (MFRC, MFRC-MFSRC), (MFRC, MFRC-MFRC), and (MFRC, MFRC-MFRC-MFSRC), respectively. Furthermore, Table 17 shows the results of $10-fold$ and these three ensembles on BIG2015, which may show that (MFRC, MFRC-MFRC-MFSRC) with $P(z) = 0.002$ has the best performance. According to these experiments, it is safe to conclude that the (MFRC, MFRC-MFRC-MFSRC) ensemble is the best ensemble to classify malware datasets.

Figure 5a disputes the confusion matrix resulted by (MFRC, MFRC-MFRC) on RT. 53 of 54 Cerber samples are detected correctly. 4 and 1 of these samples are categorized into CryptoWall, and TeslaCrypt, respectively. In contrast, 2 samples of CTB-Locker are categorized as Cerber. From Table 18, we can see that (MFRC, MFRC-MFRC) wrongly categorizes RT samples as three families Locky, CryptoWall, and TeslaCrypt. In addition, 9, 4, 9, 3, and 3 of CryptoWall, CTB-Locker, Locker, Sage, and TeslaCrypt samples did not categorize as any category. As we had predicted, the number of samples not classified by MFRC was reduced.

Table 14. Ensemble performances of BIG2015 dataset.

#	Method	*Accur.%*	*Prec.%*	*Recall%*	*F1%*	*BEP%*	*Hloss%*	$P(z)$
1	MFRC-MFSRC	67.18	87.09	23.57	37.09	55.33	32.82	0
2	MFRC, MFRC-MFSRC	96.58	90.46	80.98	85.46	85.72	3.42	0
3	MFRC-MFRC	98.27	91.06	93.22	92.13	92.14	1.73	0.6291
4	MFRC, MFRC-MFRC	97.56	90.58	87.85	89.19	89.22	2.44	0.0019
5	MFRC-MFRC-MFSRC	90.06	47.37	56.25	51.43	51.81	9.94	0.2105
6	MFRC, MFRC-MFRC-MFSRC	97.56	90.68	87.79	89.21	89.24	2.44	0.0004

Table 15. Ensemble performances of $10-fold$ classification of BIG2015 dataset.

#	Method	*Accur.%*	*Prec.%*	*Recall%*	*F1%*	*BEP%*	*Hloss%*	$P(z)$
1	MFRC, MFRC-MFSRC	97.01	89.4	84.58	86.91	86.99	2.99	0.0086
2	MFRC, MFRC-MFRC	97.47	91.14	86.76	88.89	88.95	2.53	0.18
3	MFRC, MFRC-MFRC-MFSRC	97.49	91.3	86.84	89.01	89.07	2.51	0.001

Table 16. Ensemble performances of VS dataset.

#	Method	*Accur.%*	*Prec.%*	*Recall%*	*F1%*	*BEP%*	*Hloss%*	$P(z)$
1	MFRC-MFSRC	60.2	81.53	27.01	40.58	54.27	3.98	0.041
2	MFRC, MFRC-MFSRC	91.28	85.32	69.37	76.52	77.35	8.72	0.4068
3	MFRC-MFRC	95.09	79.85	89.54	84.42	84.7	4.91	0.1511
4	MFRC, MFRC-MFRC	94.73	85.15	83.56	84.35	84.36	5.27	0.0149
5	MFRC-MFRC-MFSRC	78.6	87.65	43.03	57.72	65.34	21.40	0.0246
6	MFRC, MFRC-MFRC-MFSRC	94.66	86.46	82.38	84.37	84.42	5.34	0.0003

Table 17. Ensemble performances of $10-fold$ classification of VS dataset.

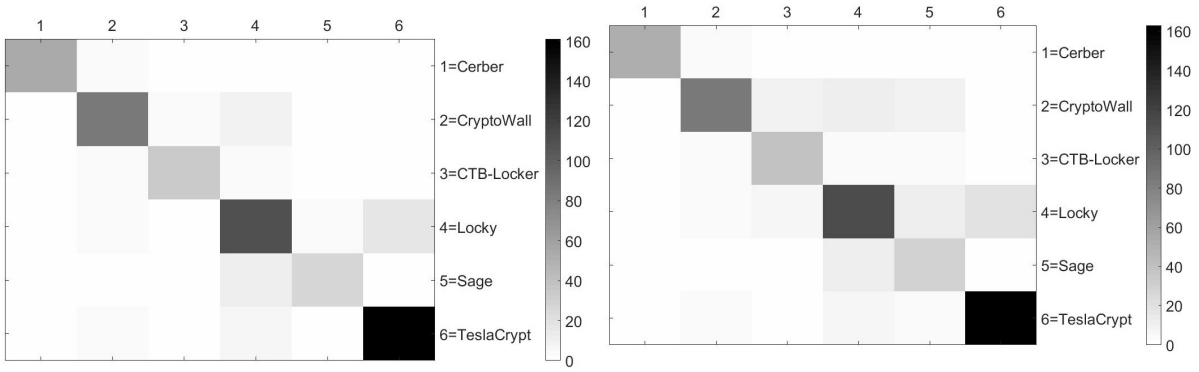
#	Method	*Accur.%*	*Prec.%*	*Recall%*	*F1%*	*BEP%*	*Hloss%*	$P(z)$
1	MFRC, MFRC-MFSRC	92.57	86.27	73.82	79.5	80.04	7.43	0.009
2	MFRC, MFRC-MFRC	95.64	88.11	86.06	87.07	87.08	4.36	0.0075
3	MFRC, MFRC-MFRC-MFSRC	95.67	88.59	85.88	87.21	87.23	4.33	0.002

Figure 5b depicts the confusion matrix resulted by (MFRC, MFRC-MFRC-MFSRC) on RT. 53 out of 54 Cerber samples were detected correctly. 4 and 1 of the samples are categorized into CryptoWall, and TeslaCrypt, respectively. In contrast, 2 samples of CTB-Locker are categorized as Cerber. From Table 19 we see that (MFRC, MFRC-MFRC-MFSRC) wrongly categorizes RT samples into four families Locky, CryptoWall, Sage, and TeslaCrypt. In addition, 2 of the Locker samples was not categorized in any category. As expected, the number of samples that are not classified by (MFRC, MFRC-MFRC) was reduced.

MFRC classifies BIG2015 with an accuracy of 97.04%. To improve the identification accuracy, the ensemble (MFRC, MFRC-MFRC) classified BIG2015 with an accuracy of 97.47%. Next, to reduce the error value, we have applied (MFRC, MFRC-MFRC-MFSRC) with an accuracy of 97.49%. Tabel 20 and Tabel 21 show the order of

Table 18. Incorrectly classified samples based on (MFRC, MFRC-MFRC) multi-class classification on RT. Here, malware families in the first column exhibit similarity with one or more families in the second column. The families in the second column are arranged in descending order of the similarity.

Name	Incorrectly classified as
Cerber	**CryptoWall**, TeslaCrypt.
CryptoWall	**Locky**, CTB-Locker, Sage.
CTB-Locker	**CryptoWall**, **Locky**, Cerber.
Locky	**TeslaCrypt**, Sage, CryptoWall, CTB-Locker.
Sage	**Locky**.
TeslaCrypt	**Locky**, CryptoWall, Sage, CTB-Locker.



(a) Confusion matrices of (MFRC, MFRC-MFRC) on RT.

(b) Confusion matrices of (MFRC, MFRC-MFRC-MFSRC) on RT.

Fig. 5. Confusion matrices of (MFRC, MFRC-MFRC) and (MFRC, MFRC-MFRC-MFSRC) on RT.

Table 19. Incorrectly classified samples based on (MFRC, MFRC-MFRC-MFSRC) on RT. Malware families in the first column exhibit similarity with one or more families in the second column. The families in the second column are arranged in descending order of the similarity.

Name	Incorrectly classified as
Cerber	**CryptoWall**,TeslaCrypt.
CryptoWall	**Locky**, Sage, CTB-Locker, TeslaCrypt.
CTB-Locker	**Sage**, Locky, CryptoWall, Cerber.
Locky	**TeslaCrypt**, Sage, CTB-Locker, CryptoWall.
Sage	**Locky**, TeslaCrypt, CTB-Locker.
TeslaCrypt	**CryptoWall**, Locky, Sage, CTB-Locker.

incorrect classification of MFRC and (MFRC, MFRC-MFRC-MFSRC) on BIG2015, respectively. (MFRC, MFRC-MFRC-MFSRC) tries to reduce the value of instance coverage, P(z). Then, incorrectly identification has not reduced in Table 21.

Therefore, a growth of 2.55% on accuracy and 30.81% on precision were achieved by comparing the second row of Table 8 with six rows in Table 12. MFRC classifies BIG2015 with an accuracy of 96.9%, as shown in the

Table 20. Incorrectly classified samples based on MFRC on BIG2015. Malware families in the first column exhibit similarity with one or more families in the second column. The families in the second column are arranged in descending order of the similarity.

Name	Incorrectly classified as
Ramnit	**Lollipop**, Gatak.ACY, Kelihos_ver1, Kelihos_ver3, Tracur, Vundo.
Lollipop	**Ramnit**, Gatak.ACY., Tracur, Kelihos_ver1, Kelihos_ver3, Obfuscator.ACY, Vundo.
Kelihos_ver3	**Obfuscator.ACY**, Kelihos_ver1, Ramnit, Gatak.ACY, Lollipop, Tracur.
Vundo	**Lollipop**, Ramnit, Kelihos_ver3, Kelihos_ver1, Gatak.ACY, Simda, Tracur, Obfuscator.ACY.
Simda	**Lollipop, Ramnit**.
Tracur	**Ramnit**, Kelihos_ver3, Lollipop, Vundo, Simda, Kelihos_ver1.
Kelihos_ver1	**Kelihos_ver3**, Ramnit, Gatak.ACY, Tracur, Obfuscator.ACY.
Gatak.ACY	**Ramnit**, Lollipop, Kelihos_ver3, Tracur, Kelihos_ver1, Obfuscator.ACY.
Obfuscator.ACY	**Ramnit**, Gatak.ACY, Lollipop, Kelihos_ver3, Tracur.

Table 21. Incorrectly classified samples based on (MFRC, MFRC-MFRC-MFSRC) on BIG2015. Malware families in the first column exhibit similarity with one or more families in the second column. The families in the second column are arranged in descending order of the similarity.

Name	Incorrectly classified as
Ramnit	**Lollipop**, Gatak.ACY, Kelihos_ver1, Kelihos_ver3, Tracur, Vundo, Obfuscator.ACY.
Lollipop	**Ramnit**, Gatak.ACY., Tracur, Kelihos_ver1, Kelihos_ver3, Obfuscator.ACY, Vundo.
Kelihos_ver3	**Obfuscator.ACY**, Kelihos_ver1, Ramnit, Gatak.ACY, Lollipop, Tracur.
Vundo	**Lollipop**, Ramnit, Kelihos_ver3, Kelihos_ver1, Gatak.ACY, Simda, Tracur, Obfuscator.ACY.
Simda	**Lollipop, Ramnit**, Tracur, Gatak.ACY, Obfuscator.ACY.
Tracur	**Ramnit**, Kelihos_ver3, Lollipop, Vundo, Simda, Kelihos_ver1, Gatak.ACY, Obfuscator.ACY.
Kelihos_ver1	**Kelihos_ver3**, Ramnit, Gatak.ACY, Tracur, Obfuscator.ACY, Vundo.
Gatak.ACY	**Ramnit**, Lollipop, Kelihos_ver3, Tracur, Kelihos_ver1, Obfuscator.ACY.
Obfuscator.ACY	**Ramnit**, Gatak.ACY, Lollipop, Kelihos_ver3, Tracur.

fourth row of Table 8. Despite the marginal improvement, MFRC did not classify 37.7% of samples while (MFRC, MFRC-MFRC-MFSRC) did not classify 0.36% of samples. Also, MRFC classified VS with an accuracy of 93.57 and $P(z) = 0.099$ while (MFRC, MFRC-MFRC-MFSRC) classified it with an accuracy 94.66 and $P(z) = 0.0003$.

In Section 4.2, machine learning techniques classified VS, RT, and BIG2015 datasets. The resultant classification of the RandomForest method was more accurate than the other tested machine learning methods. Since learning methods are often single labeled and are more accurate than multi-label classifiers. However, the ensemble-base method has desirable accuracy. RandomForest classified VS, RT, and BIG2015 with an accuracy of 96.7%, 88.46%, and 98.85%, respectively. In contrast, as shown in Table 13, Table 15, and Table 17, (MFRC, MFRC-MFRC-MFSRC) ensemble classified RT, BIG2015, and VS with accuracy of 95.53%, 97.49% and 95.67%, respectively. Therefore, comparing to the accuracy of RandomForest, (MFRC, MFRC-MFRC-MFSRC) classified RT with higher accuracy.

5 RELATED WORKS

Most of the prior studies collected a set of features and utilized similarity measurements or leveraged machine learning techniques to classify malware families or segregate malware from benign samples [4, 13, 21, 31, 34]. However, previous works were mainly focused on binary classification to divide instances into a single class.

Table 22. Deep Neural Network 10 − $fold$ classification of VS, RT and BIG2015 datasets.

#	Dataset	$Accur.\%$	$Prec.\%$	$Recall\%$	$F1\%$	$BEP\%$	$Hloss\%$	$P(z)$
1	VS	95.67	88.59	85.88	87.21	87.23	4.33	0.002
2	RT	95.67	88.59	85.88	87.21	87.23	4.33	0.002
3	BIG2015	95.67	88.59	85.88	87.21	87.23	4.33	0.002

For instance, H. HaddadPajouh et al. [10] and A. Azmoodeh et al. [3] leveraged a deep recurrent neural network and a deep Eigenspace learning method to detect Internet of things malware via Opcode sequence, respectively. Also, I. Santos et al. [24] presented variants of known malware families detection method based on Opcode sequence frequencies and leveraged cosine similarity to detect variants of known malware families. Similarly, Wong and Stamp [36] utilized a hidden Markov model to detect metamorphic malware by using Opcode sequences.

Y. Li et al. [18], presented a fuzzy hash function which extracts *5-grams* from code section of Windows Portable Executable (PE) files and maps to a bit array to provide a fingerprint. In their approach, the bitwise Jaccard distance function is the fingerprint cooperator. Their proposed method was tested based on a set of malware families collected from *VirusTotal* and the authors have leveraged a binary classifier to classify these samples. As turns out, the samples were classified by a binary classifier with a precision of 0.919 and recall of 0.914.

More recently, A. Shalaginov et al. [28] presented a multi-label malware family classification based on PE file format. They initially focused on generating fuzzy rules based on Neuro-Fuzzy (NF) techniques. Later on, they improved their model accuracy by adding malware dynamic behavioral features [27].

In an extended work by A. Shalaginov et al. [27], they leveraged static and dynamic features extracted from PE files to classify malware instances by a multi-label classifier. They have proposed deep NF multi-label classifier to classify samples collected from *VirusShare* and archive and labeled by *VirusTotal* API [33]. While `C4.5` binary classifier identifies samples with an accuracy of 82.85%, their proposed deep NF categorized samples with an accuracy of 69.44%. This method improved the accuracy of simple NF and deep neural network to 49.335% and 7.347%, respectively.

The method presented in [27] is a good multi-label classier for malware identification, but this method leverages both static and dynamic features. Extracting dynamic features in edge layer devices is expensive. To remedy this problem, we have utilized static analysis methods to identify malicious programs. In the experiment, we have understood that fuzzy multi-label classifiers [17, 28] suffer from instance coverage problem. To solve this problem, we present a new fuzzy multi-label classifier and provide ensembles to improve malicious program identification.

6 CONCLUSION AND FUTURE WORKS

Since the number of unknown malware is increasing rapidly, malware analysis is becoming a time and resource consuming task. Malicious actors are using this gap to cause significant damage. The risks of these attacks are further increased in CPS and IoT networks were a cyber attack could cause real harm to the end user. Moreover, the majority of previous malware targeted IoT networks shared a substantial amount of codes, functionalities, and even malware files properties. A multi-label classifier would significantly reduce analysis space by identifying similarities between new instances and previously known malware. Therefore, in this paper, we built a multi-label malware family classification engine using relevance fuzzy classification techniques that can be implemented in the edge layer of CPS networks. We leveraged MFRC to classify malicious instances.

As many multi-label classifiers failed to classify a significant number of samples to any family, we built a new relevance fuzzy method, called MFSRC. MFSRC uses fuzzy transformation, generation of centroid vectors, and fuzzy clustering with specified thresholds for training. In testing, fuzzy transformation and cluster membership

functions in regard to set thresholds were used to classify samples. MFSRC classified three datasets, VS, BIG, and RT, with an accuracy of 33.95%, 91.71%, and 64.8% while MFRC classified VS, BIG, and RT with an accuracy of 93.57%, 96.9%, and 42.88% which was better than previous fuzzy classifiers. However, as of instance coverage, resultant recall and precision of our classification method were still not convincing (i.e. sometimes worst than a random guess), we built an ensemble by combining MFSRC and MFRC classifiers. Our ensemble could classify malware in VS, BIG and RT datasets with an accuracy of 94.66%, 97.56%, and 94.26%, respectively which shows a significant improvement.

In the future, we will extend this work by improving the generation of centroid vectors step and we will evaluate our classifiers on bigger data sets. Furthermore, we will improve the performance of our system to be used as an inline, real-time system for protecting CPS networks.

REFERENCES

[1] Abuse.ch. [n.d.]. *Ransomware Tracker.* https://ransomwaretracker.abuse.ch/

[2] A.S.L. [n.d.]. *Exeinfo PE.* http://exeinfo.atwebpages.com

[3] Amin Azmoodeh, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2018. Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning. *IEEE Transactions on Sustainable Computing* (2018).

[4] Amin Azmoodeh, Ali Dehghantanha, Mauro Conti, and Kim-Kwang Raymond Choo. 2017. Detecting crypto-ransomware in IoT networks based on energy consumption footprint. *Journal of Ambient Intelligence and Humanized Computing* (2017), 1–12.

[5] Mauro Conti, Ali Dehghantanha, Katrin Franke, and Steve Watson. 2018. Internet of Things security and forensics: Challenges and opportunities. *Future Generation Computer Systems* 78 (2018), 544 – 546. https://doi.org/10.1016/j.future.2017.07.060

[6] Yuxin Ding, Wei Dai, Shengli Yan, and Yumei Zhang. 2014. Control flow-based opcode behavior analysis for Malware detection. *Computers & Security* 44 (2014), 65–74.

[7] Chris Eagle. 2011. *The IDA pro book: the unofficial guide to the world's most popular disassembler.* No Starch Press.

[8] GDATA. [n.d.]. *G-DATA.* https://www.gdatasoftware.com/blog/2018/03/30610-malware-number-2017

[9] N. George and Vinod P. 2015. Opcode position aware metamorphic malware detection: Signature vs histogram approach. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom).* 1011–1017.

[10] Hamed HaddadPajouh, Ali Dehghantanha, Raouf Khayami, and Kim-Kwang Raymond Choo. 2018. A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting. *Future Generation Computer Systems* 85 (2018), 88–96.

[11] Hashem Hashemi and Ali Hamzeh. 2018. Visual malware detection using local malicious pattern. *Journal of Computer Virology and Hacking Techniques* (2018), 1–14.

[12] Sajad Homayoun, Ali Dehghantanha, Marzieh Ahmadzadeh, Sattar Hashemi, and Raouf Khayami. 2017. Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence. *IEEE Transactions on Emerging Topics in Computing* (2017).

[13] Sajad Homayoun, Ali Dehghantanha, Marzieh Ahmadzadeh, Sattar Hashemi, Raouf Khayami, Kim-Kwang Raymond Choo, and David Ellis Newton. 2019. DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer. *Future Generation Computer Systems* 90 (2019), 94–104.

[14] Shamsul Huda, Rafiqul Islam, Jemal Abawajy, John Yearwood, Mohammad Mehedi Hassan, and Giancarlo Fortino. 2018. A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection. *Future Generation Computer Systems* (2018).

[15] Kaspersky. [n.d.]. *Kaspersky Lab Number of the Year: 360,000 Malicious Files Detected Daily in 2017.* https://usa.kaspersky.com/about/press-releases/2017_kaspersky-lab-number-of-the-year

[16] Ludmila I Kuncheva. 2004. *Combining pattern classifiers: methods and algorithms.* John Wiley & Sons.

[17] Shie-Jue Lee and Jung-Yi Jiang. 2014. Multilabel text categorization based on fuzzy relevance clustering. *IEEE Transactions on Fuzzy Systems* 22, 6 (2014), 1457–1471.

[18] Yuping Li, Sathya Chandran Sundaramurthy, Alexandru G Bardas, Xinming Ou, Doina Caragea, Xin Hu, and Jiyong Jang. 2015. Experimental study of fuzzy hashing in malware clustering analysis. In *8th workshop on cyber security experimentation and test (cset 15)*, Vol. 5. 52.

[19] MATLAB. 2016. *version 9.1.0.441655 (R2016a).* The MathWorks Inc., Natick, Massachusetts.

[20] Microsoft. [n.d.]. *Microsoft Malware Classification Challenge (BIG 2015).* https://www.kaggle.com/c/malware-classification

[21] Nikola Milosevic, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2017. Machine learning aided android malware classification. *Computers & Electrical Engineering* 61 (2017), 266–274.

[22] Hamed Haddad Pajouh, Ali Dehghantanha, Raouf Khayami, and Kim-Kwang Raymond Choo. 2018. Intelligent OS X malware threat detection with code inspection. *Journal of Computer Virology and Hacking Techniques* 14, 3 (01 Aug 2018), 213–223. https://doi.org/10.1007/s11416-017-0307-5

[23] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. 2018. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems* 78 (2018), 680–698.
[24] Igor Santos, Felix Brezo, Javier Nieves, Yoseba K Penya, Borja Sanz, Carlos Laorden, and Pablo G Bringas. 2010. Idea: Opcode-sequence-based malware detection. In *International Symposium on Engineering Secure Software and Systems*. Springer, 35–43.
[25] Robert E Schapire and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine learning* 39, 2-3 (2000), 135–168.
[26] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)* 34, 1 (2002), 1–47.
[27] Andrii Shalaginov and Katrin Franke. 2017. A deep neuro-fuzzy method for multi-label malware classification and fuzzy rules extraction. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*. IEEE, 1–8.
[28] Andrii Shalaginov, Lars Strande Grini, and Katrin Franke. 2016. Understanding Neuro-Fuzzy on a class of multinomial malware detection problems. In *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 684–691.
[29] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. In *Data mining and knowledge discovery handbook*. Springer, 667–685.
[30] Umit Deniz Ulusar, Erdinc Turk, Ahmet Sefa Oztas, Alp Erkan Savli, Guner Ogunc, and Murat Canpolat. 2019. *IoT and Edge Computing as a Tool for Bowel Activity Monitoring*. Springer International Publishing, Cham, 133–144. https://doi.org/10.1007/978-3-319-99061-3_8
[31] P Vinod, Akka Zemmari, and Mauro Conti. 2018. A machine learning based approach to detect malicious android apps using discriminant system calls. *Future Generation Computer Systems* (2018).
[32] virusshare. [n.d.]. *virusshare*. www.virusshare.com
[33] virustotal. [n.d.]. *virustotal*. www.virustotal.com
[34] Shanshan Wang, Qiben Yan, Zhenxiang Chen, Bo Yang, Chuan Zhao, and Mauro Conti. 2018. Detecting android malware leveraging text semantics of network flows. *IEEE Transactions on Information Forensics and Security* 13, 5 (2018), 1096–1109.
[35] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.
[36] Wing Wong and Mark Stamp. 2006. Hunting for metamorphic engines. *Journal in Computer Virology* 2, 3 (2006), 211–229.
[37] Ding Yuxin and Zhu Siyi. 2017. Malware detection based on deep learning algorithm. *Neural Computing and Applications* (2017), 1–12.

Received February 2007; revised March 2009; accepted June 2009