IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 5, 2019, accepted November 29, 2019, date of publication December 3, 2019, date of current version December 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957429

Feature Selection for Malware Detection Based on Reinforcement Learning

ZHIYANG FANG, JUNFENG WANG, JIAXUAN GENG, AND XUAN KAN
College of Computer Science, Sichuan University, Chengdu 610065, China

Corresponding author: Junfeng Wang (wangjf@scu.edu.cn)

This work was supported in part by the National Key Research and Development Program under Grant 2019QY1404 and Grant 2018YFB0804503, in part by the National Natural Science Foundation of China under Grant U1836103, and in part by the Technology Research and Development Program of Sichuan, China, under Grant 2017GZDZX0002 and Grant 19ZDZX0024.

ABSTRACT Machine learning based malware detection has been proved great success in the past few years. Most of the conventional methods are based on supervised learning, which relies on static features with labels. While selecting static features requires both human expertise and labor. New selections, which fix features from a wide range, are handcrafted by careful manual experimentation or modified from existing methods. Despite their success, the static features are still hard to be determined. In this paper, a Deep Q-learning based Feature Selection Architecture (DQFSA) is introduced to cover the deficiencies of traditional methods. The proposed architecture automatically selects a small set of highly differentiated features for malware detection task without human intervention. DQFSA trains an agent through Q-learning to maximize the expected accuracy of the classifiers on a validation dataset by sequentially interacting with the features space. The agent, based on an ϵ-greedy exploration strategy and experience replay, explores a large but finite space of possible actions and iteratively discovers selections with improved performance on the learning task. Actions are a set of reasonable choices, which indicate whether a feature is chosen or not. Extensive experimental results indicate that the proposed DQFSA outperforms existing baseline approaches for feature selection on malware detection with minimum features, improves the generalization performance of the learning model and reduces human intervention. More specifically, the proposed architecture's underlying representation is robust enough for re-calibrating models to other domains of information security.

INDEX TERMS Feature selection, malware detection, deep reinforcement learning, Q-learning.

I. INTRODUCTION

Malware is malicious code which designed to compromise information security by gathering sensitive information from computer system or making unauthorized access to computer system. The proliferation of malware variants means that detecting malware is one of the biggest challenges in information security [1]–[4]. To tackle the malware detection problem, many researchers invented new methods to protect computer system. Generally, they characterized training samples by extracting diverse features from programs files. Classification algorithms use these features and trained to complete this task.

Typically, common feature extraction methods mainly depend on signature [5], format structure [6], raw binary features [7], multi-view features [8] and so on. These aforementioned methods partially capture the distinguishable information between benign and malicious programs. While selecting features, a detector constructor may confront the following challenges:

- the features extracting process can rarely done automatically and mainly based on human experience;
- the extracted features can not comprehensively cover the key distinguishing characteristics of samples;
- a wide range of indicators can gain better detection achievement, but lead to redundancy and slow down the training speed.

The number of possible choices makes the combination space of features extremely large and hence, infeasible for an exhaustive manual search.

In this work, a general architecture (DQFSA) using deep Q-learning to automatically select features for classifying

The associate editor coordinating the review of this manuscript and approving it for publication was Zhen Qin.

malware is proposed. The novel Q-learning agent whose goal is to discover an optimal feature set that performs well on malware classification without human intervention. The task of learning agent is to sequentially pick features for a machine learning model. Furthermore, the classification accuracy of the given machine learning model is served as reward to feed to agent for determining proper features. By using the ϵ-greedy strategy, agent learns through random exploration and slowly begins to exploit its findings to select optimal indicators. Based on priority experience replay [9], the learning process is accelerated by repeated memory sampling. This architecture discovers less features automatically for detection. Detection accuracy is mainly used to access the performance of this architecture. The results demonstrate that proposed method competitive against the means that leverage more features. In this study, the static Windows portable executable (PE) malicious software are used as the training samples to validate the proposed DQFSA.

Moreover, it worth mentioning that the proposed method in this paper is not limited to the software with portable executable format, and furthermore it can be extended to other sorts of malware. Considering that the advantage—trial-and-error search—of the reinforcement learning, this DQFSA can be applied to other selection tasks.

The rest of this paper is organized as follows: Section II discusses the related work. Section III gives a glance at the overall framework. Sections IV describes the proposed DQFSA in details. Section V presents the experimental setup and numerical simulation results. Section VI concludes the paper.

II. RELATED WORK

A. FEATURE SELECTION METHODS

In the traditional static malware detection, feature extraction methods can be roughly divided into signature feature extraction, format information extraction, unstructured feature extraction and multiple abstract feature extraction [10]. The signature-based methods had achieved admirable results as the earliest malware detection technique [5], [11]. These methods extract some field of the header or calculate an unique number, which similar to hash code, from a known malware file. The extracted attributes, usually called signature features, are stored to the database to verify a new file malicious or not. However, this technique can never detect new malware for the novel signature features are the only basis of discrimination.

According to Peter [12], malware is different from benign software in static structure characteristics. Some heuristic detection methods used static structure characteristics as a part of the features. However, some structural features could not distinguish benign software and malware, or even affected the detection results and speed. Some scholars had proposed a series methods for the extracting process of malware features. In the studies [13], the static structured features and some statistical features of PE files were used as the basic feature set, and several dimensionality reduction technique were applied to filter the original features. The detection rate exceeded 99%. Raman *et al.* [6] divided the characteristics of PE file into 7 parts and select 7 features from each of these part. The training was carried out with IBk, J48, J48 Graft, PART, Random Forest and Ridor classifier respectively. Finally, under the J48 classify algorithm, the true positive rate (TP Rate) reached 98.56% and the false positive rate (FP Rate) reached 5.68%. Recently, Kim *et al.* [14] extracted totally 87 features from the header of PE file. After using a variety of different classifiers for comparison, a considerable result was achieved. However, the domain knowledge is needed when extracting format features for a sample.

Some researchers introduced the unstructured features, namely the n-grams features, to the malware detection. In the context of malware detection, n-grams are all substrings of a larger string. Generally, a string is simply split into substrings of fixed length N. Many studies [15]–[18] based on n-grams features shared the same procedure. A file scanned by slide window to generate the original n-grams features and the most relevant n-grams features were chosen as the input of the machine learning algorithms. Numerous experiments were done to determine a combination of fixed length N, the number of n-grams features and classification algorithm. The results of these experiments indicated that the detection based on this kind of features are not likely to obtain outstanding performance. Different from the studies mentioned above, Raff *et al.* [19] combined the deep learning method and the malware detection task, and proposed a LSTM-based model. They simply extracted n-grams features from the header of PE file, which latterly acted as the input of the deep learning model. Finally, the highest accuracy of this model reached 90%.

Due to the drawbacks of single view features, some researchers intuitively proposed multi-view features. A reasonable explanation is that the n-grams features and format features partially capture the distinguishable information between benign software and malware. Each types of features has its own inherent strength and weakness. The more feature views are used, the comprehensive information of a software can be presented. In studies [7], [8], [20], the features used in classifier were extracted from different views. Truth is that this feature extraction method can obviously improve detection level.

B. REINFORCEMENT LEARNING

In recent years, reinforcement learning technology has demonstrated its outstanding capabilities in certain fields. For instance, methods using deep Q-learning network have been successful in automatically game-playing [21] and human level control [22]. In addition, some research [23], [24] creatively applied reinforcement learning methods to the field of information security. Inspired by the research [25], which trained a reinforcement learning model to design the architecture of CNN for the task of image classification, we previously proposed a DQEAF framework using reinforcement learning to evade anti-malware engines [26].

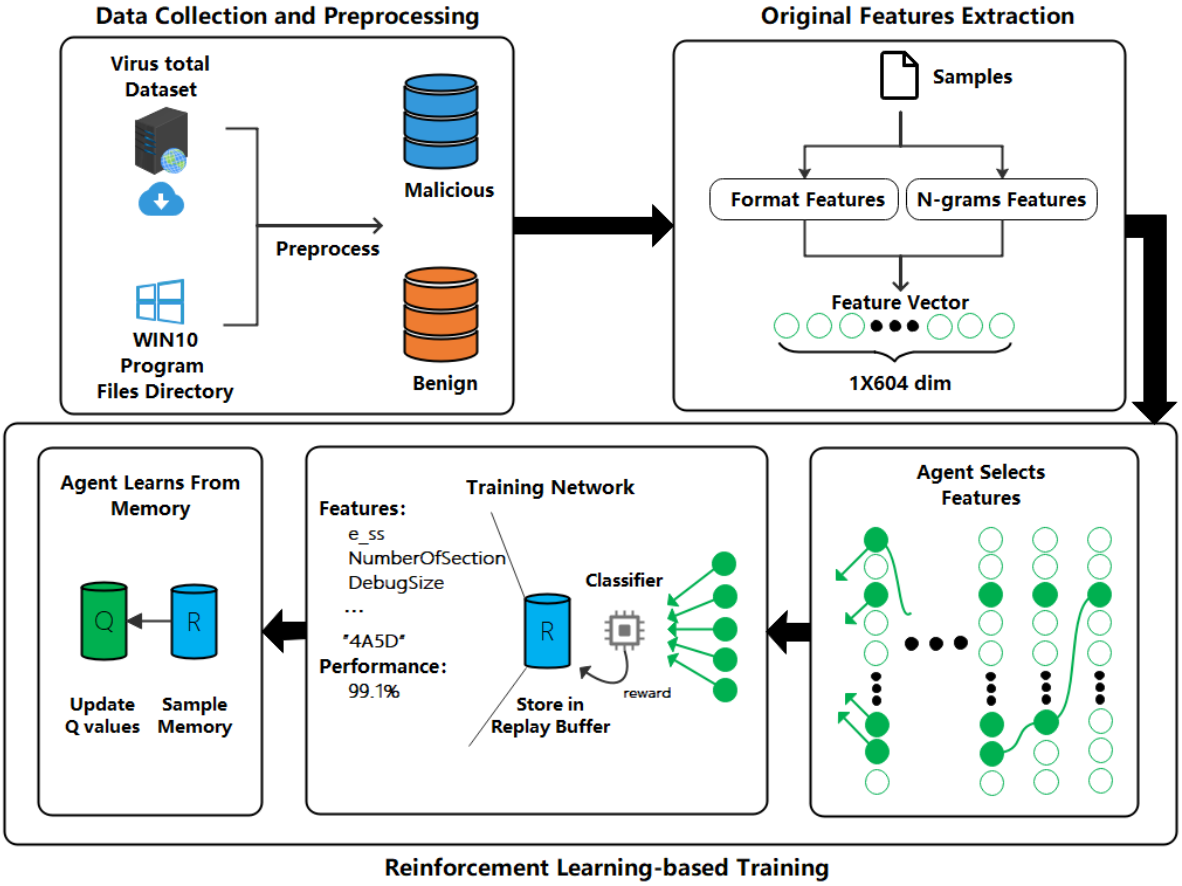

FIGURE 1. DQFSA model structure.

Considering that rarely researches has previously used reinforcement learning to select features for malware classification, a reinforcement learning-based model is proposed. This work focuses on the following points and has made some improvements: (1) The introduced method automatically selects features for malware detection which greatly reduces manual intervention; (2) Less features are determined to fulfill the detection task compared with other similar work;

III. FRAMEWORK OVERVIEW

As is shown in Figure 1, the proposed DQFSA embodies three important parts, data set collection and preprocessing part, original feature extraction part and reinforcement learning-based training part. In the data set collection and preprocessing part, the malware data comes from VirusTotal data set. Our benign samples are collected from the program files directory of Windows 10 operating system. The raw samples are proprocessed by data processing module to exclude the PE file which can not be analyzed. The original feature extraction method mainly focuses on two kinds of features, the format features of PE file and n-grams features of PE file. Finally, the feature vectors are used as the input of the reinforcement learning-based model. The agent chooses some features to determine a sample malicious or not. Then the environment feeds the reward, which is the accuracy of classification under the features selected by agent, back to the agent. After multiple rounds of training, the agent have the ability to choose an optimal subset of original features that makes the accuracy of detection task highest.

The reinforcement learning-based model is the key part of our malware detection model. For each iteration, the agent begins by sampling features conditioned on a predefined behavior distribution and the agent's prior experience. Subsequently, the features are selected to fed to the classifier. That classifier is then trained on a specific task. The features and performance, e.g., validation accuracy, are then stored in the agent's replay buffer. Finally, the agent uses its memories to learn about the space of associated features through Q-learning network. Our goal is to make the proposed reinforcement learning-based model select the best feature subset easily and detect the malware efficiently.

IV. FEATURE SELECTION WITH REINFORCEMENT LEARNING

For this model, the primary task is to train a learning agent to sequentially choose features for classification. With the assumption that a feature performs well in one classification task should also attribute to the result of another classification

TABLE 1. List of format features.

Feature Sources	Number	Description
DOS Header	15	All properties except **e_magic** and reserved words are selected.
File Header	7	All properties except **TimeDateStamp** field are selected. One more boolean attribute is added, the value is set 0 if the number of sections does not equal to the value of **NumberOfSection** field.
Optional Header	29	All properties are selected.
Data Directory	32	The virtual address and size of 15 directories, the value is set 0 if the directory does not exist.
Resource	22	The total number of resources, and the entry number of resources whose resource ID belongs to {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 17, 19, 20, 21, 22, 23, 24}.
DLL imported	21	The DLL with high information gain and relate to information security are selected.
API called	30	The API with high information gain and relate to information security are selected.
The number of DLL imported	1	—
The number of API called	1	—
The header of .text section	9	All properties are selected.
The header of .data section	9	All properties are selected.
The header of .rsrc section	9	All properties are selected.
The header of .rdata section	9	All properties are selected.
The header of .reloc section	9	All properties are selected.
The number of other sections	1	The number of sections exclude the common sections mentioned above.
Total	**204**	

task, so the feature selection process can be modeled as a Markov Decision Process. Under the ϵ-greedy strategy, the agent sequentially chooses features until it arrives a termination state. The specifics of DQFSA will be presented in the rest of this section.

A. THE ENVIRONMENT

The environment is a key element in reinforcement learning, it represents the external observed by the agent [27]. In DQFSA, the environment is responsible for observing the selected features from original features and sensitive to each step of action. The original features are mainly derived from The format features and bytes n-grams features.

1) FORMAT FEATURES

The PE files under Windows platform share an unified structure [28]. Generally, the file contains many headers and several sections so that the Windows OS knows how to load and run the executable code. The malware have the same format, but they do exist some differences in format information. Therefore, the format information attributes to the accuracy of detection. The studies [13], [19], [29] used different format information of PE file to fulfill the detection task. Based on the research [13], a total of 204 format features are determined. The details of the selected format features are shown in Table 1.

2) N-GRAMS FEATURES

For the PE file can also be considered as byte sequences, some scholars intuitively thought that the malware may have some similarity in the byte form. In researches [16], [18], the n-grams features were employed to malware detection. Inspired by text classification, some researchers thought some n-grams features have high frequencies in malware, while it can be rarely detected in the benign software. According to the study [8], the malware detector showed the best performance when the value of N is set to 4. In addition, the TF (term frequency) – IDF (inverse document frequency) measure is commonly used to weight n-grams features and obtained by multiplying TF and IDF. The definition of TF according to Equation (1).

$$TF = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

where $n_{i,j}$ is the number of occurrences of specific n-grams in jth sample and $\sum_k n_{k,j}$ represents the number of occurrences of all n-grams in jth sample. The IDF is defined as

$$IDF_i = \log \frac{|D|}{|j : t_i \in d_j|} \tag{2}$$

where $|D|$ is the total number of samples and $|j : t_i \in d_j|$ is the number of samples which contain a specific n-grams. The intuition of IDF lies on that if the number of samples which contain feature t_i is small, i.e., the value of IDF is quite large, the feature t_i may contribute to classification result. In order to prevent the excessive features from affecting the efficiency of training, we counted the DF (document frequency) of each feature. The DF is defined as the number of samples which contain a specific n-grams. We filtered a specific n-gram feature out if its DF value is quite small, for it does not contribute to the detection result. The proposed method selects 400 n-grams features with the highest DF value.

Figure 2 shows the entire process of original feature extraction intuitively.

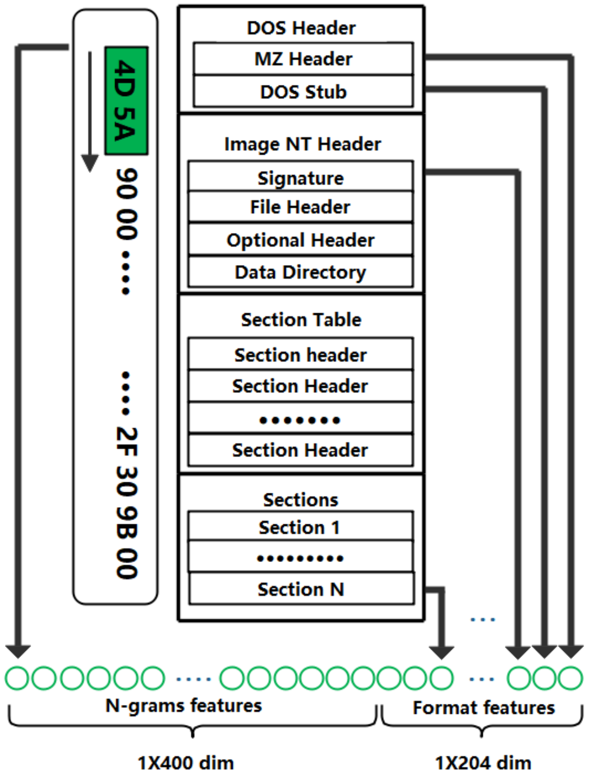

FIGURE 2. The process of original features.

B. THE ACTION SPACE

The action space represents a set of actions that the agent can choose from, which is similar to all the responses reacted by human to a certain stimulus from the outside world [30]. The action space in the proposed framework can be defined as:

$$\{a : a \in N \wedge a \leq |F|\} \tag{3}$$

where N represents all the alternative original features to be selected and a termination state to be selected. $|F|$ represent the index of the action which selects the final terminal action. When $a < |F|$, action a means to set the element with index a in the environment state to 1. When $a = |F|$, action a means that the agent stops the selection procedure. In other words, the environment is instructed to enter the termination state.

Each time the agent observes the state from the current environment, it immediately takes an allowable action from the action space based on a given strategy.

- (1) We restrict the agent from taking certain actions to both limit the state-action space and make learning feasible.
- (2) We allow the agent to terminate a path at any point, i.e., it may choose a termination state from any non-termination state.
- (3) If the agent takes action a before, the action a is not allowed to be taken any longer, i.e., one feature can not be selected twice.

By restricting the range of features (consisting of 1 × 604 dimensional), the agent is left with a finite but large space of associated features to search from. The agent continuously searches for actions based on the strategy by means of loop until it finds a legal action and then indicates to the environment.

C. THE REWARD

The reward mechanism of reinforcement learning distinguishes it from supervised learning. It will only give feedback to the agent a positive or negative evaluation, rather than tell the agent what the correct action is [27]. It must represent the changes that the environment has made after the agent performs the corresponding action in the current state, and the gap between the target. In this case, the target can be specifically defined as selecting a set of features which maximizes the detection accuracy.

When the agent performs an action and acts on the environment, the environment gets a new state. According to the current state, the environment picks the features up and feeds to the classifier. Consequently, the classifier screens the corresponding features from the samples to fulfill the classification task. As with most classification problems, the evaluation criterion is the classification accuracy of each model. Accordingly, the accuracy of classifier is served as reward. During the training phase of classifier, the 10-fold cross-validation is used.

D. THE TRAINING PROCEDURE

The training process is illustrated in Training Algorithm (Algorithm 1) integrated with Testing Algorithm (Algorithm 2) to generate models with an efficient selection of discriminative indicators.

Exploration and exploitation policy are common used in the reinforcement learning algorithm, which allows the model to gain more from the training process [31]. Exploration refers to random choice of features to explore more possibilities. Exploitation is the choice of the best feature that has been selected to improve the model. Here $\epsilon - greedy$ policy is employed, which practices a random action with a probability ϵ and selects the most valuable action with a probability $1-\epsilon$ otherwise. The value of ϵ decreases from 1.0 to 0.3 in steps during training epochs progress according to Equation (4). Where n is the current training step and N is the total number of training steps.

$$\epsilon_n = 1.0 - \frac{n}{N} \tag{4}$$

In DQFSA, the agent is set to explore half of max epochs in order to make agent explore as much as possible and promote the performance of agent. Equation (5) is introduced for enabling the agent to explore according to the strategy mentioned above

$$\epsilon_{decay\_steps} = \digamma * N/2 \tag{5}$$

where ϵ_{decay_steps} represents how many steps it takes for epsilon to decrease and $\digamma$ is the number of maximum features.

Algorithm 1 Training Algorithm

```
Initialize Memory $M$ to capacity $S$
Initialize two identical networks, action-value $Q$ and target-value $\hat{Q}$,
with random weights $\theta$ and weights $\hat{\theta} = \theta$ respectively
for $episode = 1$ to $N$ do
    Initialize $s_0$ state array as zero vector of length 604
    for $t = 1$ to $\digamma$ do
        With probability $\epsilon$ using equation (4) select a random action $a_t$ or choose $a_t = \arg\max_a Q(s_t, a; \theta)$
        Transform $s_t$ by action $a_t$ to $s_{t+1}$
        observe reward $r_{t+1}$
        Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ with priority
        Every $U$ step reset $\hat{\theta} = \theta$
        if done then
            break
        end if
    end for
    if episode mod EVALUATE_INTERVAL == 0 then
        Run Testing Algorithm and get classification accuracy $CA$
        Store Q and $\hat{Q}$ to a new model $m$
        if $CA >$ MAX_ACC then
            break
        end if
    end if
end for
```

Algorithm 2 Testing Algorithm

```
Initialize classification accuracy $CA = 0$
for $i = 1$ to $\digamma$ do
    Compute $\hat{Q}$ and choose $a_t = \arg\max_a \hat{Q}(s_t, a; \hat{\theta})$
    Transform $s_t$ by action $a_t$ to $s_{t+1}$
    observe reward $r_{t+1}$
    if done then
        break
    end if
end for
Compute the value of $CA$
return $CA$
```

During the entire training process (starting at $\epsilon = 1.0$), we maintain a replay dictionary which stores (1) the selected features and (2) classification accuracy, for all of the sampled features. If a set of features that has already been trained is re-sampled, it is not re-trained, but instead the previously found validation accuracy is presented to the agent. Inspired by prioritized experience replay [9], our framework takes the priority of transitions into consideration and transitions with high classification accuracy can be replayed more frequently. This strategy gives the agent more opportunities to learn to select better features, rather than just limits to the initial learning experience. Figure 3 shows the Markov Decision Process for feature selection.

V. PERFORMANCE EVALUATION

A. EXPERIMENT SETUP

The Double Deep Q-learning Network used in proposed method have less training parameters, while it has other specific parameters to set. The detail description and setting of parameters are shown in Table 2. During training phase, the discount factor is defined as 1, because the same actions always lead to the same reward, i.e., the order of feature selection does not affect the accuracy.

All the experiments carried out in NVIDIA DGX Station with an Intel Xeon E5-2698 (2.2GHz), GPU is Tesla V100. We used python 3.7.3 to preprocess the sample, Scikit-learn 0.21.2 to build machine learning based classifier, Chainerrl 0.7.0 to build double deep Q-learning network. The evaluation criterion of the model is accuracy (AUC).

Source codes of the Q-learning neural network, the malware binary feature extraction, framework details and successfully trained model based on chainerrl are available at https://github.com/fanmcgrady/select-features.

B. DATASET DESCRIPTION AND PREPROCESSING

The PE files used in this experiment are divided into benign and malicious categories. Benign PE files come from the Program Files directories in Windows 10 operating system, totaling 10152, and malicious files from the VirusTotal dataset for a total of 12371. Considering the stability and robustness of subsequent experiments, all the samples are parsed and those who can not be analyzed will be filtered out. Finally, about 12146 malicious samples and 9057 benign samples are remained after this procedure.

The data used in the experiment are characterized by the original features and their labels (benign or malicious), so the samples need to convert to vectors before sent to the reinforcement learning model. A python-based pefile toolkit is used for analyzing PE files. We used this toolkit to implement a python script. The job of this script is to extract the original format features of samples. For the parts that are not included in the file (such as some files lack some sections), all the features of this part are denoted by 0. Once a sample are screened, one 1 × 204 dimensional vector will be created.

Another python script is used for extracting byte 4-grams features. The steps of extraction process are listed as followed. Firstly, the PE files are converted to hexadecimal files which could be used for generate byte subsequence. Secondly, the slide window method is introduced to handle hexadecimal files to produce 4-grams features, e.g., if the hexadecimal sequence are "4D5A90", the 4-grams byte features screened by slide window are "4D5A", "D5A9" and "5A90". The *DF* criterion is used to select 400 4-grams features which have highest *DF* value. The script eventually generate a 1 × 400 dimensional vector for each sample, and it is extended to the format feature vector. Finally, the label

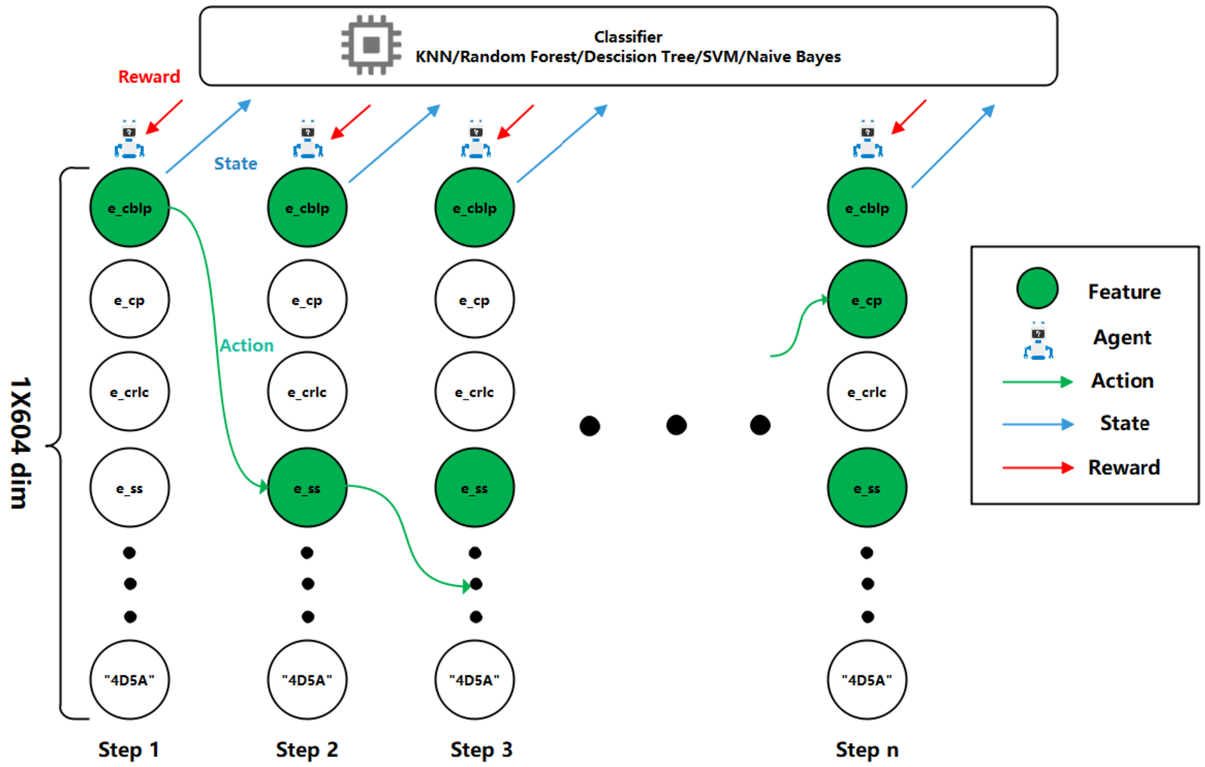

FIGURE 3. Markov decision process for feature selection.

TABLE 2. Major parameter setting in the training algorithm.

Parameter	Value	Description
replay start size	20	the agent will carry out an uniform random policy for this number of steps before learning
replay buffer size	5×10^5	the capacity of replay buffer memory
minibatch size	16	the number of training cases over which each stochastic gradient descent (SGD) update is computed
target update interval	10	the frequency with which the target network is updated
discount factor	1	discount factor γ used in the Q-learning
initial ϵ	1	the initial value of ϵ in ϵ-greedy exploration strategy
final ϵ	0.3	the final value of ϵ in ϵ-greedy exploration strategy
test interval	10	the number of steps we carried out the evaluation to the agent

of sample is appended to the joint vector, 1 for malware and 0 for benign software. For the convenience of training, a CSV format file is created to save the original features and the label of each sample. In this CSV file, one line represent a sample, which is a 1×605 dimensional vector.

C. RESULTS

1) COMPARISON WITH DIFFERENT CLASSIFIERS

In order to demonstrate the performance of DQFSA convincingly, we conducted a series of comparative experiments to find the combination of least possible features and classifiers. As is shown in Figure 4 to 8, the vertical axis represents the classification accuracy, the horizontal axis represents the max number of features selected by reinforcement learning-based algorithm for the detection. More precisely, for the purpose of making learning procedure efficient and meaningful, the number of features selected by agent is limited from 5 to 15. The accuracy of using all features is also denoted as a reference.

It can be seen from Figure 4 to 8, when the number of selected feature is small, the accuracy is only about 96%. As the number of selected features increases, the accuracy can be gradually improved. Finally, each classifier can achieve the highest accuracy, exceeds 99%, when using about 11 features.

In addition, on the given dataset, the combination of KNN classifier with 11 features selected was the best combination out of the other combinations tried. Compared with the

TABLE 3. Evaluation details with different classifiers.

Accuracy / Features Used / Classifier	5	6	7	8	9	10	11	12	13	14	15
KNN	95.21%	97.69%	96.20%	98.80%	99.32%	99.48%	**99.53%**	98.63%	98.70%	99.27%	99.06%
Decision Tree	96.48%	96.76%	96.03%	97.66%	96.60%	98.09%	98.99%	**99.03%**	98.13%	98.23%	98.34%
Random Forest	96.04%	95.96%	98.16%	97.17%	98.59%	98.61%	**99.46%**	98.68%	99.01%	98.87%	99.34%
Naive Bayes	95.90%	94.31%	96.44%	99.25%	98.26%	97.71%	99.17%	99.20%	**99.41%**	98.82%	99.50%
SVM	95.33%	96.86%	**99.29%**	99.15%	97.74%	97.50%	99.20%	98.89%	98.89%	98.89%	98.89%

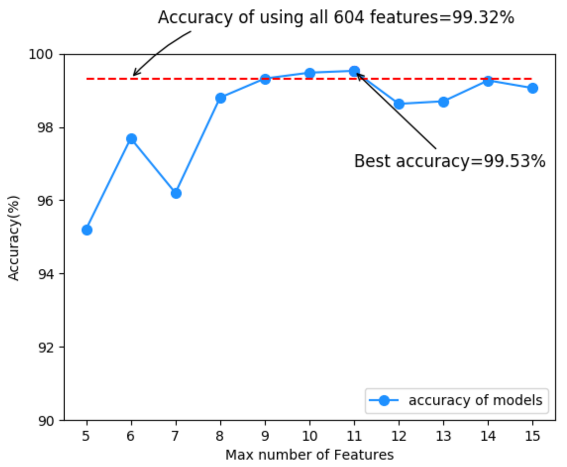

FIGURE 4. Accuracy of models with KNN.

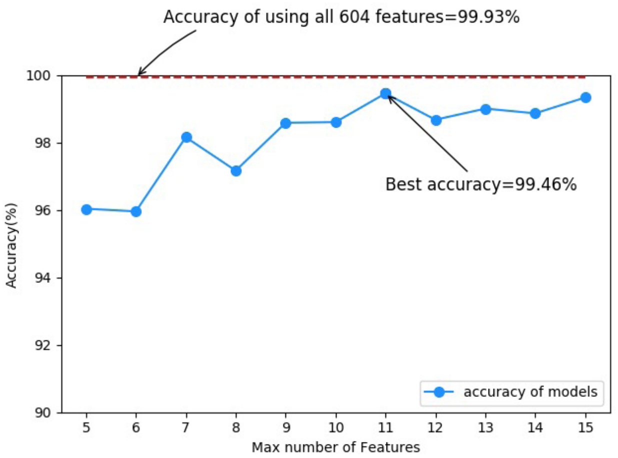

FIGURE 6. Accuracy of models with RandomForest.

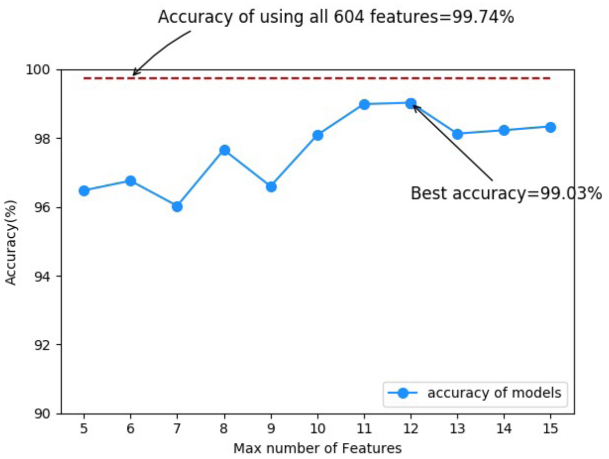

FIGURE 5. Accuracy of models with DecisionTree.

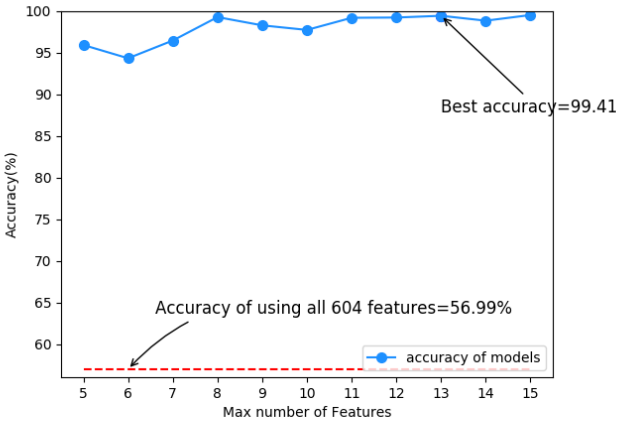

FIGURE 7. Accuracy of models with Naive Bayes.

experiments base on using all features, an optimal minimum feature set is automatically determined by DQFSA to fulfill the detection task and achieve a competitive performance. Namely, the proposed method has certain practicability and feasibility. The details of evaluation are shown in Table 3. In Table 5, we present the top five models' features details selected by DQFSA, along with their prediction accuracy gained on the validation dataset.

2) COMPARISON WITH RELATED WORK

In the experiments mentioned above, our method shows better capacity on different classifiers. In order to make a comprehensive comparison between the proposed DQFSA and traditional methods, this work conduct a comparative experiment base on the same dataset. According to Raman *et al.* [6], Bai *et al.* [13], Kim *et al.* [14], researchers have leveraged relevant detection features for the task of classification.

The number of features and detection performance corresponding to each methodology are shown in Table 4.

TABLE 4. Comparison with related work.

Paper	Features Used	KNN	Decision Tree	Random Forest	Naive Bayes	SVM
Raman et al. [6]	7	94.51%	99.69%	99.83%	44.02%	79.49%
Bai et al. [13]	197	99.32%	99.79%	99.93%	56.99%	57.65%
Kim et al. [14]	87	99.32%	99.91%	99.36%	56.61%	57.44%
DQFSA(Features Used)	-	**99.53%(11)**	**99.03%(12)**	**99.46%(11)**	**99.41%(13)**	**99.29%(7)**

TABLE 5. Feature selection details of our top 5 models.

Classifier	Features Used	Details	Accuracy
KNN	11	1. **Magic** field 2. **MajorOperatingSystemVersion** field 3. **MajorImageVersion** field 4. **DllCharacteristics** field 5. **PointerToLinenumbers** in .reloc section 6. '**3006**' byte sequences 7. '**0062**' byte sequences 8. '**0018**' byte sequences 9. '**002B**' byte sequences 10. '**002F**' byte sequences 11. '**0360**' byte sequences	99.53%
Decision Tree	12	1. **MajorLinkerVersion** field 2. **Size** of **EXPORT** DATA_DIRECTORY 3. **Size** of **IMPORT** DATA_DIRECTORY 4. **Characteristics** in .data section 5. whether the API function '**_vsnwprintf**' is called 6. '**6500**' byte sequences 7. '**1FFF**' byte sequences 8. '**0640**' byte sequences 9. '**4E00**' byte sequences 10. '**1001**' byte sequences 11. '**3746**' byte sequences 12. '**4701**' byte sequences	99.03%
Random Forest	11	1. whether the number of sections equals to the value of **NumberOfSection** 2. **VirtualAddress** of **RESERVED** DATA_DIRECTORY 3. the number of **RESOURCE** whose id is 16 used 4. whether the API function '**_amsg_exit**' is called 5. '**0909**' byte sequences 6. '**2000**' byte sequences 7. '**CCCC**' byte sequences 8. '**A000**' byte sequences 9. '**7500**' byte sequences 10. '**B450**' byte sequences 11. '**1100**' byte sequences	99.46%
Naive Bayes	13	1. **NumberOfSections** field 2. **Characteristics** field 3. **MajorImageVersion** field 4. whether the DLL lib '**comctl32.dll**' is imported 5. '**0610**' byte sequences 6. '**4040**' byte sequences 7. '**6800**' byte sequences 8. '**4696**' byte sequences 9. '**002E**' byte sequences 10. '**4010**' byte sequences 11. '**7006**' byte sequences 12. '**0202**' byte sequences 13. '**7374**' byte sequences	99.41%
SVM	7	1. **Characteristics** field 2. **MinorOperatingSystemVersion** field 3. '**008B**' byte sequences 4. '**004C**' byte sequences 5. '**00E0**' byte sequences 6. '**6200**' byte sequences 7. '**3900**' byte sequences	99.29%

As it can be seen in Table 4, some classifiers achieve a higher accuracy with fewer features. Further more, due to the reduction of features, the time for extracting features and training is shorter.

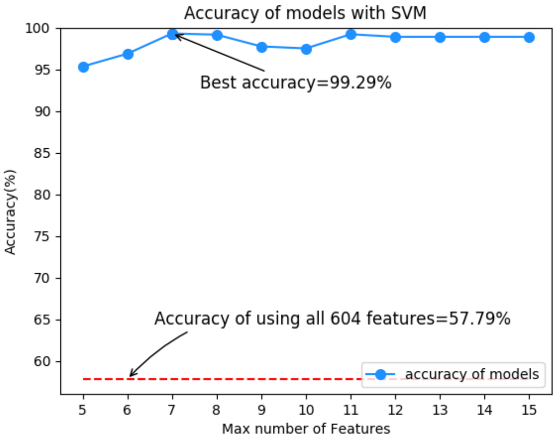

FIGURE 8. Accuracy of models with SVM.

Our framework is compared against the related work mentioned above and enhance the accuracy of validation dataset by using KNN, SVM and Naive Bayes classifiers. The results demonstrate that reinforcement learning can be applied to malware detection with further improved performance compared to traditional methods.

VI. CONCLUSION

This paper proposed an architecture named DQFSA using reinforcement learning to implement feature selection task. The highly differentiated features selected can be fed to supervised-learning algorithms to classify malware. The core component of DQFSA is an AI agent, which is constantly interacting with samples feature space without human intervention. The agent could choose optimal sequences of reasonable features deliberately by deep reinforcement learning, which aims to maximize the accuracy of detection. Experiments show that the proposed DQFSA discovers features for malware detection competitive against the baseline means that use more features. In the future work, we will apply our framework to other selection tasks, for the advantage—trial-and-error search—of the reinforcement learning.

REFERENCES

[1] D. Ucci, L. Aniello, and R. Baldoni, "Survey on the usage of machine learning techniques for malware analysis," *Comput. Secur.*, vol. 81, 2017.

[2] A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, "Machine-learning aided static malware analysis: A survey and tutorial," in *Cyber Threat Intelligence*. Berlin, Germany: Springer, 2018, pp. 7–45.

[3] P. Vinod, R. Jaipur, V. Laxmi, and M. Gaur, "Survey on malware detection methods," in *Proc. 3rd Hackers' Workshop Comput. Internet Secur. (IITKHACK)*, Mar. 2009, pp. 74–79.

[4] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *Proc. 5th Conf. Inf. Knowl. Technol.*, May 2013, pp. 113–120.

[5] R. W. Lo, K. N. Levitt, and R. A. Olsson, "MCF: A malicious code filter," *Comput. Secur.*, vol. 14, no. 6, pp. 541–566, 1995.

[6] K. Raman, "Selecting features to classify malware," *InfoSec Southwest*, to be published.

[7] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015, pp. 11–20.

[8] J. Bai and J. Wang, "Improving malware detection using multi-view ensemble learning," *Secur. Commun. Netw.*, vol. 9, no. 17, pp. 4227–4241, 2016.

[9] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*. [Online]. Available: https://arxiv.org/abs/1511.05952

[10] H. El Merabet and A. Hajraoui, "A survey of malware detection techniques based on machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 1, pp. 366–373, 2019.

[11] J. O. Kephart, "Automatic extraction of computer virus signatures," in *Proc. 4th Virus Bull. Int. Conf.*, Abingdon, England, 1994, pp. 178–184.

[12] P. Szőr, *The Art of Computer Virus Research and Defense*. Upper Saddle River, NJ, USA: Pearson Education, 2005.

[13] J. Bai, J. Wang, and G. Zou, "A malware detection scheme based on mining format information," *Sci. World J.*, vol. 2014, Jun. 2014, Art. no. 260905.

[14] S. Kim, "PE header analysis for malware detection," M.S. thesis, 2018, vol. 624.

[15] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2004, pp. 470–478.

[16] D. K. S. Reddy and A. K. Pujari, "N-gram analysis for computer virus detection," *J. Comput. Virology*, vol. 2, no. 3, pp. 231–239, Dec. 2006.

[17] Y. Elovici, A. Shabtai, R. Moskovitch, G. Tahan, and C. Glezer, "Applyingmachine learning techniques for detection of malicious code in networktraffic," in *Proc. Annu. Conf. Artif. Intell.* Berlin, Germany: Springer, 2007, pp. 44–50.

[18] I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, "N-grams-based file signatures for malware detection," in *Proc. ICEIS*, vol. 2, May 2009, pp. 317–320.

[19] E. Raff, J. Sylvester, and C. Nicholas, "Learning the pe header, malware detection with minimal domain knowledge," in *Proc. 10th ACM Workshop Artif. Intell. Secur. AISec*, Nov. 2017, pp. 121–132.

[20] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proc. IEEE Symp. Secur. Privacy. S&P*, May 2000, pp. 38–49.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: https://arxiv.org/abs/1312.5602

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[23] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to evade static PE machine learning malware models via reinforcement learning," 2018, *arXiv:1801.08917*. [Online]. Available: https://arxiv.org/abs/1801.08917

[24] C. Wu, J. Shi, Y. Yang, and W. Li, "Enhancing machine learning based malware detection model by reinforcement learning," in *Proc. 8th Int. Conf. Commun. Netw. Secur. ICCNS*, Nov. 2018, pp. 74–78.

[25] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2016*arXiv:1611.02167*. [Online]. Available: https://arxiv.org/abs/1611.02167

[26] Z. Fang, J. Wang, B. Li, S. Wu, Y. Zhou, and H. Huang, "Evading anti-malware engines with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 48867–48879, 2019.

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*. Cambridge, MA, USA: MIT Press, 2018.

[28] M. Pietrek, "Inside windows-an in-depth look into the win32 portable executable file format," *MSDN Mag.*, vol. 17, no. 2, pp. 1–4, 2002.

[29] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "PE-Miner: Mining structural information to detect malicious executables in realtime," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, 2009, pp. 121–141.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 2, no. 4. Cambridge, MA, USA: MIT Press, 1998.

[31] M. Coggan, "Exploration and exploitation in reinforcement learning," CRA-W DMP Project at Comput. Sci., McGill Univ., Montreal, QC, Canada, Tech. Rep., 2004.

ZHIYANG FANG received the M.S. degree in computer science and technology from Sichuan University, Chengdu, Sichuan, in 2009, where he is currently pursuing the Ph.D. degree in computer science and technology. He is currently involved in research work on information security. His research interest includes software security.

JUNFENG WANG received the M.S. degree in computer application technology from the Chongqing University of Posts and Telecommunications, Chongqing, in 2001, and the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, in 2004. From July 2004 to August 2006, he held a postdoctoral position at the Institute of Software, Chinese Academy of Sciences. Since August 2006, he has been a Professor with the College of Computer Science and the School of Aeronautics and Astronautics, Sichuan University. His recent research interests include network and information security, spatial information networks, and data mining. He is currently serving as an Associate Editor for IEEE Access, the IEEE Internet of Things, and *Security and Communication Networks*.

JIAXUAN GENG received the bachelor's degree in computer science and technology from the Chongqing University of Posts and Telecommunications, Chongqing, in 2018. He is currently pursuing the M.S. degree in computer science and technology with Sichuan University, China. He is expected to get the degree, in 2021. His research interests include software security and cybersecurity.

XUAN KAN received the M.S. degree in computer science and technology from Sichuan University, Sichuan, in 2017. She is currently involved in research work on information security. Her research interest includes software security.