This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

MADMAX: Browser-Based Malicious Domain Detection through Extreme Learning Machine

KAZUKI IWAHANA[1], TATSUYA TAKEMURA[1], CHENG JU CHIEN[1], NAMI ASHIZAWA[1], NAOKI UMEDA[1], KODAI SATO[1], RYOTA KAWAKAMI[1], REI SHIMIZU[1], YUICHIRO CHINEN[1], (Non-Member, IEEE), and NAOTO YANAI[1], (Member, IEEE),

[1]Osaka University, 1-5 Yamadaoka, Suita, Osaka, Japan.

Corresponding author: Kazuki Iwahana (e-mail: k-iwahana@ist.osaka-u.ac.jp).

A part of this work is supported by Innovation Platform for Society 5.0 at MEXT.

ABSTRACT Fast and accurate malicious domain detection is an essential research theme to prevent cybercrime, and machine learning is an attractive approach for detecting unseen malicious domains in the past decade. In this paper, we present *MADMAX (MAchine learning-baseD MAlicious domain eXhauster)*, a browser-based application leveraging extreme learning machine (ELM) for malicious domain detection. In contrast to the existing work of ELM-based domain detection, MADMAX newly introduces two methods, i.e., *selection of optimized features* to provide higher accuracy and throughput based on permutation importance and *real-time training* to retrain a model with an updated malicious dataset for continuous malicious domain detection. We demonstrate that MADMAX fairly outperforms the existing work with respect to accuracy and throughput by virtue of the selection of optimized features. Moreover, we also confirm a model with real-time training stably detects even unseen malicious domains, whereas accuracy of a model without the real-time training decreases due to the unseen domains. The source codes of MADMAX is publicly available via GitHub.

INDEX TERMS Browser application, extreme learning machine, feature selection, malicious domain detection, machine learning, real-time training

I. INTRODUCTION

A. BACKGROUNDS

The use of malicious domains rapidly increases in cybercrime in recent years. For example, an adversary often utilizes his/her generated malicious domains to operate command and control (C&C) servers or sets up phishing sites. A typical countermeasure against such malicious domains is to prepare for a deny list of these domains. Nevertheless, new domains continuously appear since domain generation algorithms (DGAs), which automatically generate new domains, are often utilized. Consequently, countermeasures based on a deny list are insufficient, and hence a framework to cover even unseen domains is crucial.

Based on the background described above, domain detection based on machine learning has attracted attention in the past years [1]. Meanwhile, a machine learning-based domain detection tool on a browser, which is the closest interface for a user, has never been proposed so far, to the best of our knowledge. Although there is VT4Browsers[1] which is an add-on of Google Chrome to detect malicious domains on a browser automatically, it is just a deny list-based approach. Namely, existing browser-based applications may be ineffective against unseen domains. In contrast, existing works [2]–[6] on (browser-independent) malicious domain detection methods based on machine learning often utilizes a complex and large-sized architecture. The training time becomes longer in proportion to the architecture complexity despite providing a high inference accuracy. In other words, they are somewhat tough to introduce in a browser environment such that a user utilizes them in real-time.

In this paper, we propose *MADMAX (MAchine learning-baseD MAlicious domain eXhauster)*, a novel application of malicious domain detection based on machine learning together with browsing. Loosely speaking, MADMAX enables

[1]https://chrome.google.com/webstore/detail/vt4browsers/efbjojhplkelaegfbieplglfidafgoka

a user to detect malicious domains automatically by installing the application as an add-on in his/her browser. MADMAX is able to provide substantial advantages because it can cover even unseen malicious domains that a deny list does not contain.

In general, there are two technical problems for incorporating machine learning into an environment requiring the real-time process, e.g., browsing. First, domain detection based on machine learning may consume a long time compared to a standard deny list-based tool since the machine learning needs additional overheads such as feature extraction about domains. Second, the trade-off between throughput and accuracy of machine learning should be considered carefully. Generally speaking, the throughput of a model decreases due to the complicated computation of machine learning, whereas a complicated machine learning model is often necessary for providing high accuracy.

Besides, machine learning for cybersecurity may cause concept drift [7], which is a change of essential features because of the appearance of new malicious domains. Hence, re-training to catch up with new domains, i.e., *real-time training*, is necessary as well. For real-time training, the throughput of the training is an essential factor. However, complicated and large-sized architectures utilized in the existing works require the training time to be overlong, and hence are unsuitable for real-time training. Whereas machine learning research does often not take the training time into account, the training time is crucial when introducing machine learning in a browser is considered. Thus, it is considered that machine learning is no longer introduced into a browser as far as we know.

A potential solution for the aforementioned problems is malicious domain detection [8] based on *extreme learning machine (ELM)* [9]. Informally, ELM is a neural network with a single hidden layer, which can learn features without backpropagation whose computational cost is heavy. Namely, both learning and inferring domains can be run quite fast, and thus the above problems can be solved potentially. In recent years, ELM has been pointed out as the renaming of early neural networks with random weights, and there are many extensions so far [10].

Nonetheless, the existing work [8] just evaluated whether ELM can infer malicious domains or not. In other words, the existing work did not implement domain detection as an application, and thus throughput and accuracy of malicious domain detection with respect to features on the application level are uncertain. More concretely, features should often be extracted in real-time when an application is deployed in the real world. Hence, the performance, including feature extraction on an application-level implementation, needs to be evaluated. Furthermore, a model should be updated following the update of malicious domains. In doing so, we need to confirm if the updated model can continuously and precisely detect unseen malicious domains, which are newly generated and rapid training. The results shown in the existing work are thus insufficient for the feasibility of a web browser.

In contrast, we develop MADMAX as an application-level implementation. Furthermore, through the use of more features described in our previous works [11], we rigorously select features such that a model providing a fast inference and high accuracy is constructed. Consequently, MADMAX outperforms the existing work [8] in terms of accuracy and throughput. Moreover, via discussion on real-time training in accordance with the update of a model, we confirm that MADMAX can cover the update of malicious domains. In comparison with the existing malicious domain detection methods based on neural networks [2]–[6], MADMAX is expected to reduce the training and inference time by virtue of ELM.

To sum up, we make the following contributions:

- We present MADMAX, a browser-based application for malicious domain detection. Notably, a prototype is implemented as a browser add-on. The implementation is publicly available via GitHub (https://github.com/kzk-IS/MADMAX).
- We shed light on optimized features for accuracy and throughput of domain detection by rigorously selecting the features. As a result, we demonstrate that MADMAX outperforms existing work [8] with respect to the accuracy of malicious domain detection.
- We confirm that a model can learn features in real-time; that is, it can continuously detect even unseen malicious domains by virtue of introducing real-time training. By contrast, we also show that accuracy of a model without the real-time training decreases due to a drift of malicious domains.

B. PAPER ORGANIZATION

The rest of this paper is organized as follows. Section II describes domain names and a machine learning-based malicious domain detection, including its formulation. Section III presents the system requirements of MADMAX and then explains the key questions to realize MADMAX. Section IV describes the methodology for the design of MADMAX to tackle the key questions. Section V evaluates the performance of MADMAX, and then the discussions and limitations are described in Section VI. Section VII describes related works, and finally, the conclusion and future directions are presented in Section VIII.

II. PRELIMINARIES

In this section, we describe domain names and malicious domain detection based on machine learning as background knowledge to understand this work.

A. DOMAIN NAMES

Domain names are texts correlated to network hosts and are operated via the domain name system (DNS). In general, domain names are hierarchically managed under namespaces called a zone, and the highest domain is called root. The most popular domains are `.com`, `.us`, and `.jp`, and such

domains are called top-level domains (TLDs). There are plural domains under each TLD, and these domains are managed hierarchically and distributively through their zones.

B. MALICIOUS DOMAIN DETECTION BASED ON MACHINE LEARNING

Domain detection model based on machine learning makes inferences to determine if the given domains are malicious or not. Informally, the objective model is obtained whereby a machine learning model learns features of domains and their labels that represent the domains as benign or malicious. Afterward, the model takes features of a target domain as inputs and then infers its label. A typical approach for domain detection in recent years is based on neural networks.

Problem Formulation: We formalize our approach against the problem of domain detection based on machine learning as follows. Let $\mathcal{F} = \{f_1, \cdots, f_l\}$ be a set of features. Each domain $d_i \in D$ has features $F_i = \{f_{i,1}, \cdots, f_{i,l}\}$, where D denotes a set of domains, and $l \in \mathbb{N}$ denotes the size of F_i, i.e., the number of features of each domain. In addition, each $d_i \in D$ has a label $L_i \in \{0, 1\} \subseteq L$, where 0 of each label denotes a benign domain and 1 of each label denotes a malicious domain. Given the size of D, i.e., the number of domains, $DFL = \{(d_1, F_1, L_1), \cdots (d_n, F_n, L_n)\}$ denotes the combinations with domains, features of each domain, and labels of each domain. Let $Model = M(DFL)$ denotes a trained model, where M denotes a learning algorithm. Our goal is to get the results of inference, $L_t = Model(F_t)$, by extracting features $F_t = \{f_{t,1}, \cdots, f_{t,l}\}$ from unlearned domain d_t.

C. EXTREME LEARNING MACHINE

We describe the development of extreme learning machine (ELM) [9] which is a fast machine learning algorithm to train single hidden layer feedforward neural networks [12]. Roughly speaking, ELM can obtain generalization performance efficiently and, according to the paper [12], is able to compute a global optimization equal to or better than support vector machine (SVM) [13]. Also, ELM is much faster and easier to implement than most state-of-the-art machine learning approaches [14]. Consequently, there are many applications of ELM in various areas, e.g., life science or computer vision. Meanwhile, ELM has never been utilized in the context of malicious domain detection except for Shi et al. [8], to the best of our knowledge. As described in Section I, concept drift is essential for malicious domain detection, and hence the real-time training discussed in this paper is a different problem from the above works.

The original ELM [9] is used in the context of supervised learning for both classification tasks and regression tasks. Although we focus on the original ELM in this paper, there are many extensions of ELM to deal with imbalanced data and errors in the real world. For instance, Xiao et al. [15] proposed a class-specific cost regulation extreme learning machine (CCR-ELM) to deal with imbalanced classification data introducing a class-specific regulation cost into the classification. Also, Zhang et al. [16] proposed residual compensation ELM (RC-ELM) to deal with the prediction error of ELM due to causes, e.g., non-linearity. The main idea of RC-ELM is to introduce a multilayer structure to build a feature mapping between input and output. Similarly, Zhang et al. [17] proposed robust ELM (R-ELM) to deal with highly complicated noise. According to Zhang et al., an objective function of R-ELM is constructed to fit the noise using a mixture of Gaussian distribution to approximate any continuous distribution between Gaussian noise and non-Gaussian noise.

Furthermore, online sequential ELM (OS-ELM) [18] learns data that is continuously obtained instead of preparing for the data in advance. OS-ELM is utilized in actual applications such as stock price and weather forecasting. Afterward, Zhao et al. [19] proposed FOS-ELM, a method of retraining a model by sliding window [20] for each data block, which is similar to the real-time training in this paper. Matias et al. [21] also proposed an extension of OS-ELM to deal with time-series data by introducing a forgetting factor. New data is then biased rather than the old data. The methods described above are beneficial because both past and future data should be considered for the concept drift. Although we utilize the original ELM [9] in this paper, the existing ELM algorithms described above can be used for MADMAX.

III. MADMAX

MADMAX (MAchine learning-baseD MAlicious domain eXhauster) is a browser-based application for malicious domain detection based on machine learning. In this section, we first present the system requirements of MADMAX. Next, malicious domain detection based on ELM is described as a potential approach, and then the key questions in this paper are described.

A. SYSTEM REQUIREMENTS

MADMAX is a malicious domain detection application based on machine learning, which runs on a web browser with high accuracy in a short time. Concretely, MADMAX is a client-server application, where a machine learning model deployed on a server infers domains sent from a browser extension. The overview of MADMAX is shown in Figure 1. Preventing threats on web browsers is quite important because users are commonly accessing websites through web browsers. Meanwhile, add-ons are widely available on various web browsers. Therefore, the use of add-ons is an easily deployable approach for malicious domain detection.

Functions that MADMAX provides to users and servers are shown as follows:

a: User-Side Functions

A user benefits from a function that automatically outputs a warning by deploying the add-on when he/she accesses a malicious site. Concretely, at first, MADMAX extracts a domain from a URL of a website and then sends the domain data to the server. Afterward, MADMAX displays a warning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

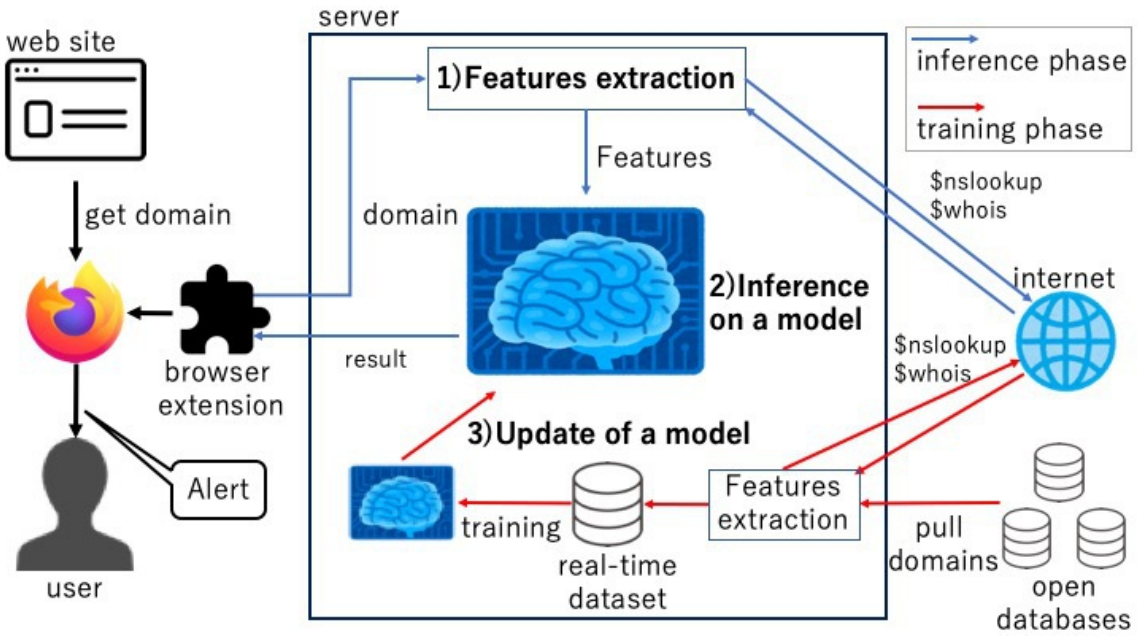


FIGURE 1: The overview of MADMAX

if the domain data represents a malicious domain. On the other hand, MADMAX allows access to the website when the domain data indicates that the website is benign.

b: Server-Side Functions

A server provides the following functions as a part of MADMAX. The server first receives a domain as input from a user and then returns whether it is malicious by utilizing a trained model controlled by the server. At that time, the server executes domain detection through the following three functions:

1) Features extraction from a domain: A domain itself has few features generally. Consequently, extracting features based on DNS records for a domain is expected since the information itself is insufficient. However, in real-time domain detection through an add-on, the throughput of feature extraction should be considered because the above feature extraction is a time-consuming task in general.
2) Inference on a model: The server infers $Model(F_i)$ from features F_i of a domain d_i with a trained model $Model$ and then identifies if d_i is benign $L_i = 0$ or malicious $L_i = 1$. The throughput of inference on the trained model should be high because the latency of a user is influenced by the throughput of the inference itself as well as the feature extraction.
3) Update of a model: The server needs to continuously detect unseen malicious domains through retraining a model to be stored in the server with the latest malicious domains. If the update based on retraining takes a long time, a user may be exposed to threats of new malicious domains. Consequently, a real-time trained model should continuously detect malicious domains with high accuracy, and hence the update of the model should be finished within a short time.

B. MALICIOUS DOMAIN DETECTION BY ELM

We focus on the malicious domain detection [8] based on *extreme learning machine (ELM)* [9] for the design of MADMAX. ELM has been proposed for training single hidden layer feedforward neural networks. The training process of neural networks is roughly divided into three kinds of layers, i.e., an input layer, one or more hidden layers, and an output layer. The performance of neural networks is commonly improved by increasing the number of hidden layers and neurons in each layer. However, the training and inference processes consume time in proportion to the increase of the hidden layers. In contrast, ELM incorporates a pseudo-inverse matrix for computing weight matrices in a single hidden layer. Therefore, ELM realizes significantly fast learning and is suitable for malicious domain detection on a web browser that needs the real-time process because ELM provides fast inference.

The inference process of ELM is formalized as follows. Given features $F_i = \{f_{i,1}, \cdots, f_{i,l}\}$ of a domain d_i, ELM calculates the following:

$$\sum_{j=1}^{N} \beta_j A(W_j \cdot F_i + b_j) = O_i, \tag{1}$$

where N denotes the number of nodes in a hidden layer, $A(\cdot)$ denotes the activation function, W_j denotes the weight vector on the input layer for the j-th node, b_j the j-th bias term, and O_i denotes the output of the inference. In addition, β_j denotes the parameter given by training ELM.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

The training process of ELM is to solve the minimization problem defined as $||H\beta - L||$, where $\beta = (\beta_1^T, \cdots, \beta_N^T)^T$, and H is defined as follows:

$$H = \begin{pmatrix} A(W_1 \cdot F_1 + b_1) & \dots & A(W_N \cdot F_1 + b_N) \\ \vdots & \ddots & \vdots \\ A(W_1 \cdot F_n + b_1) & \dots & A(W_N \cdot F_n + b_N) \end{pmatrix} \tag{2}$$

ELM randomly generates W_j and b_j as the beginning step of the training process, and then the minimization problem is regarded as the problem of finding β that satisfies $||H\beta - L|| = 0$. Namely, ELM can find β by computing $\beta = H^{\dagger}L$, where ELM utilizes a pseudo-inverse matrix to calculate $H^{\dagger}$. ELM then computes inverse matrices just one time and finishes all the training process. Therefore, the learning algorithm of ELM can obtain W, b, β in a shorter time than backpropagation.

C. KEY QUESTIONS

The primary motivation for the design of MADMAX is to allow users to distinguish benign domains from malicious domains through an add-on on web browsers. To do this, MADMAX needs to detect malicious domains in a short time and continuously detect unseen malicious domains. Concretely, we have two main key questions for the design of MADMAX as follows:

The first key question is a trade-off between the accuracy of the trained model and throughput on the server, including feature extraction. Higher accuracy is generally obtained from extracting more features. It also means that the throughput of feature extraction becomes worthy in proportion to the number of features. Although early literature [8] empirically showed a high throughput of inference on their model, the throughput of the inference for application-level, including the feature extraction, was not explicitly shown. Namely, a set $\mathcal{F}' \subseteq \mathcal{F}$ of features to provide inference with high accuracy and high throughput are still not evident.

The second key question is whether the inference model can detect unseen malicious domains with high accuracy. More specifically, a model should be trained and updated in real-time because malicious domains are constantly generated, e.g., by domain generation algorithms (DGAs) as described in Section I. Despite that ELM can provide a high throughput of the training according to the early literature [8], the model should also be continuously trained with the latest malicious domains. Existing works do not clarify if the continuously training and detecting unseen malicious detection is feasible.

The goal of this paper is to answer the two key questions described above.

IV. METHODOLOGY

In this section, we describe the concrete methodology of MADMAX. First, we describe how we tackle the key questions described in the previous section and then describe *selection of optimized features* and *real-time training* as concrete methods.

A. OVERVIEW OF METHODOLOGY

To realize MADMAX, we describe two methodologies to answer the key questions described in Section III-C. At the beginning step of the discussion, we answer the first question, i.e., a trade-off between accuracy and throughput, by selecting an optimized set of features. Next, we answer the second question, i.e., continuous detection for malicious domains, by incorporating the optimized features into an ELM inside MADMAX. We describe more details below.

To answer the first key question, we focus on a set of features presented in our previous work [11] and then, for each subset of the features in [11], evaluate a trade-off between throughput, including feature extraction, and accuracy of inference on ELM with the subset. More specifically, we select the optimized features by leveraging the permutation importance algorithm [22] for each subset. We call the above process the selection of optimized features. Intuitively, the permutation importance algorithm can clarify which feature is practical to increase the accuracy of domain detection, i.e., providing an importance ranking of features. Therefore, we can decide the set $\mathcal{F}'$ of optimized features that provide domain detection with high accuracy and high throughput by determining the relationship between features, accuracy, and throughput with respect to inference from the ranking. (See Section IV-B for detail.)

Next, for the second key question, we introduce the real-time training that enables MADMAX to continuously maintain high accuracy in detecting unseen malicious domains generated in the future. In particular, a server updates malicious domain dataset DFL and then retrains the model $Model = M(DFL)$. Intuitively, the real-time training is expected to enable the model to detect the unseen malicious domains, e.g., the concept drift [7]. For the update of the malicious domain dataset, the domains are continuously pulled in real-time from the open databases utilized in our previous work [11]. Then the optimized features $\mathcal{F}' \subseteq \mathcal{F}$ obtained from the first methodology, the selection of optimized features, are extracted for each newcomer malicious domain d_i. In doing so, the model $Model$ is retrained using the updated dataset. (See Section IV-C for detail.)

B. SELECTION OF OPTIMIZED FEATURES

We describe how to select the optimized features by utilizing the importance of the features below.

1) List of Features

We follow the features utilized in our previous work [11]. These features are defined from three perspectives, i.e., text-based features, DNS-based features, and web-based features. We describe these features in detail below.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

a: Text-Based Features

Text-based features represent information obtained from strings of domain names and discuss whether malicious domains can be detected from the domain names. In particular, there are 14 features, i.e., the length of a domain name, rate of vowels, the number of vowel characters, the number of consonants, the number of consonant characters, conversions of vowels and consonants, the number of numeric characters, rate of numeric characters, conversions of numerals and alphabets, the number of other characters, the length of max consecutive characters, the entropy [23], and the reputation value [24].

b: DNS-Based Features

DNS-based features represent information obtained from DNS records of their corresponding domains and discuss the difference of DNS records between malicious domains and benign domains. In particular, there are eight features, i.e., the number of different IP addresses, the number of distinct PTR records, the number of name servers, the number of MX servers, the similarity between name servers, the number of countries, the mean of TTL, and the standard deviation of TTL.

c: Web-Based Features

Web-based features represent information obtained from contents on domains and discuss characteristics of the contents provided by malicious domains. In particular, there are three features, i.e., the number of HTML tags of contents in the corresponding web pages, the WHOIS lifetime to represent the difference of expiration date and creation date, the WHOIS active time to represent the difference of update date and creation date.

2) Optimized Features

We describe how to optimize the features described above. First, we utilize the dataset in our previous work [11] including the above features and then adopt the permutation importance [22] as a computation for the importance of features. Hence, the importance for each feature f_j among a set $\mathcal{F} = \{f_1, f_2, \cdots, f_l\}$ of features is computed. Their resulting features are represented under the importance, and we call it the feature importance ranking.

Next, we choose a threshold T to decide features to be utilized through the feature importance ranking. The top T features are then utilized from the ranking, and the accuracy of a model trained with the features and the computational overhead for detecting domains are measured. In other words, through computing the permutation importance for each feature in advance, combinations of beneficial features for malicious domain detection can be discovered. The aforementioned process is shown in Figure. 2.

C. REAL-TIME TRAINING

This section describes the real-time update of the dataset and the method of retraining with the updated dataset. The flow of the real-time training is shown in Figure 3.

1) Real-Time Data Collection

We describe how to update the dataset for the retraining in real-time. Loosely speaking, the training dataset is updated by pulling the latest malicious domain data from the public databases utilized in our previous work [11]. In particular, benign domains are pulled from the top of Tranco[2], and malicious domains are continuously pulled from URLhaus[3], CyberCrime Tracker[4], and PhishTank[5] in real-time, respectively. The trained dataset DFL is continuously updated in real-time by extracting a set $\mathcal{F}'$ of the optimized feature shown in Section IV-B with respect to an unseen domain d_i.

2) Retraining

We describe the proposed retraining method with an updated dataset. The retraining with the dataset updated in real-time will enable a machine learning model to detect even unseen malicious domains, which are newly generated in the future, with high accuracy as soon as possible. In the design of the real-time training for MADMAX, n benign domains and n malicious domains for some $n \in \mathbb{N}$ are given to an ELM as input. More specifically, n benign domains are given only at the initial process. In other words, the benign domains are no longer updated. On the other hand, malicious domains are constantly updated, and the ELM learns the latest n domains in the updated dataset as inputs following some regular schedule, e.g., for every ten minutes.

Although one might think of a retraining method such that an ELM is retrained with only the latest n domains when n unseen domains appear. The retraining in MADMAX is fast even with such the retraining method.

D. ETHICAL CONSIDERATION

In this section, we discuss cybersecurity ethics for the design of MADMAX.

First, we utilize Tranco [25], a pubic list for cybersecurity research, which is based on commercial services such as Alexa. According to the paper [11], Tranco is utilized as benign domains. Domains included in both Tranco and the public databases of malicious domains are dealt with as both benign and malicious. Since services such as Alexa are for the commercial purpose, MADMAX may potentially degrade the effectiveness of those products by giving their domains malicious labels. However, MADMAX may still bring merits to their service providers. In particular, via analysis on domains that are detected as malicious, the providers may be able to find potential malicious services that were undetected. Likewise, MADMAX will be beneficial for improving the ranking of related services as well.

Besides, the features used in this paper are selected since we focus on the general characteristics of malicious domains.

[2]https://tranco-list.eu/
[3]https://urlhaus.abuse.ch/
[4]https://cybercrime-tracker.net/
[5]https://www.phishtank.com/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

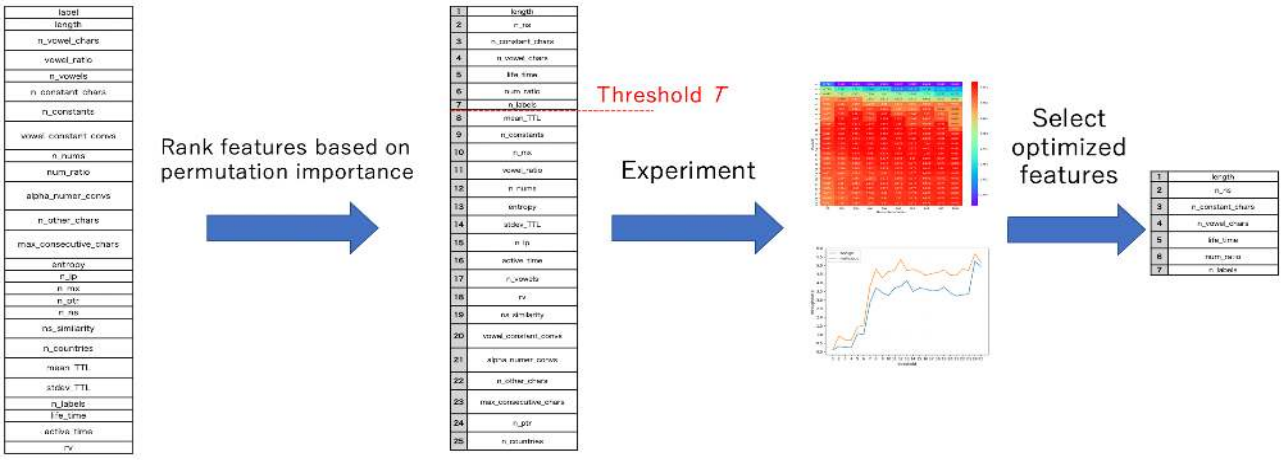


FIGURE 2: The flow of the selection of optimized features in MADMAX: First, the features are ranked through the permutation importance. Second, the accuracy and throughput of a model are measured by utilizing the features chosen by a threshold T. An appropriate threshold T is then decided based on the above experiments.

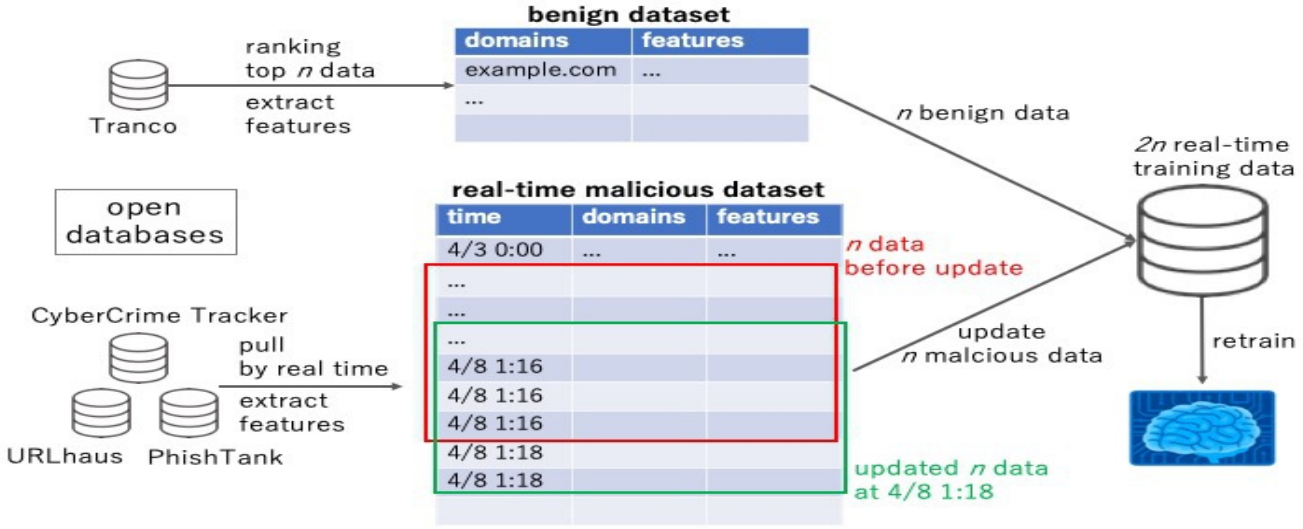


FIGURE 3: The flow of the real-time training in MADMAX: The real-time malicious domain dataset is updated from the databases, and then a model is retrained by utilizing the latest n malicious data, i.e., domains, and the n benign data. The red box and the green box represent datasets to update a model in each period, respectively.

From this perspective, we recommend providing feedback to the owners or organizations whose domains are unfortunately detected as malicious to encourage the update of their configurations. As described above, we aim to support the detection of potentially malicious services. Namely, to prevent benign domains from unfortunately detected as malicious domains in the future, we strongly suggest a reconsideration of configurations of domains.

V. EXPERIMENTS

In this section, we conduct two experiments to evaluate the performance of MADMAX from the perspective of the key questions described in Section III-C. In particular, we discuss the selection of optimized features and the real-time training. We first describe the experimental purposes and experimental settings, including the implementation of MADMAX, and then show the experimental results.

A. EXPERIMENTAL PURPOSES

We describe the experimental purposes.

First, we aim to evaluate the trade-off between throughput and well-known metrics for detection, i.e., *F1 score, G-mean, accuracy, precision*, and *recall*, and then find an optimized set $\mathcal{F}'$ of features whereby malicious domains can be detected

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

with high throughput and high scores of the metrics for inference. In particular, the permutation importance [22] is utilized to find $\mathcal{F}^{'}$ from results of malicious domain detection.

Second, we confirm that MADMAX can continuously detect unseen domains by utilizing the real-time training described in Section IV-C. More concretely, by using the set $\mathcal{F}^{'}$ of features obtained in the first experiment, we compare the scores of the metrics described above of a retrained model with those of a model without the retraining. Hence, we confirm that the real-time training supports the detection of unseen malicious domains for MADMAX as well as providing high scores of the metrics. Hereafter, we denote by *normal model* a model without the real-time training, and by *retrained model* a model with the real-time training for the sake of convenience.

B. EXPERIMENTAL SETTINGS

1) Implementation

Implementation of MADMAX is shown below. All the user-side functions of MADMAX are implemented as a Firefox (version 81.0.2) add-on with JavaScript[6]. On the other hand, all the server-side functions of MADMAX are implemented on Amazon EC2 c4.8xlarge with Python[7] and Flask[8] library. Especially, an ELM is implemented by utilizing the NumPy[9] library. The permutation importance to compute the feature importance is implemented using the scikit-learn[10] library.

2) Evaluation Metrics

In this section, we define evaluation metrics in this paper below. To do this, we first describe the four terms, i.e., true positive (TP), true negative (TN), false positive (FP), false negative (FN), to evaluate the detection performance. TP is the number of malicious domains that are correctly detected as malicious. TN is the number of benign domains that are correctly detected as benign. FP is the number of benign domains that are wrongly detected as malicious, and FN is the number of malicious domains that are wrongly detected as benign.

We then define the following evaluation metrics based on the above four terms.

Accuracy: It is the ratio of correctly detected domains to the total number of domains. The accuracy is defined as follows:

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

Precision: It is the ratio of the number of correctly detected domains as malicious to the total number of detected domains as malicious. The precision is defined as follows:

$$Precision = \frac{TP}{TP + FP}.$$

Recall: It is the ratio of the number of correctly detected domains as malicious to the total number of malicious domains. The recall is defined as follows:

$$Recall = \frac{TP}{TP + FN}.$$

F1 score: It is the harmonic mean of the precision and the recall. The F1 score is defined as follows:

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}.$$

G-mean: It is the geometric mean of the precision and the recall. The G-mean is defined as follows:

$$G - mean = \sqrt{Precision \times Recall}.$$

3) Process of Experiments

We describe the process and settings of experiments for the selection of optimized features and the real-time training below.

a: Selection of Optimized Features

This experiment has three steps below.

First, we rank the features in a high order of importance based on permutation importance. Then, we utilize a dataset in our previous work [11], which includes 24,126 benign domains and the 24,126 malicious domains, i.e., the 48,252 domains in total.

Second, we evaluate the accuracy with respect to the threshold T and the number N of nodes in a hidden layer. Then, we use the five-fold cross-validation, which provides numerical stability for the evaluation.

Finally, we evaluate the throughput of MADMAX for T based on the optimized value N. Then, 100 benign domains and 100 malicious domains are randomly chosen from the dataset. We measure the average time to receive results from the server for benign domains and malicious domains by utilizing these domains. Based on the above process, the throughput of MADMAX from the user's perspective can be evaluated.

b: Real-Time Training

The retained model is trained with the real-time training using an updated dataset, and then scores of the metrics by the retrained model are compared with those by the normal model. In this experiment, whereas a dataset for malicious domains is updated as time goes on, benign domains are no longer updated.

At the beginning of the experiment, we first collect time-series data from the open databases shown in Section IV-C1 similarly to our previous work [11]. In particular, we collect top 25,000 domains from Tranco [25] on November 25 as

[6]https://developer.mozilla.org/ja/docs/Web/JavaScript
[7]https://www.python.org/
[8]https://flask.palletsprojects.com/en/1.1.x/
[9]https://numpy.org/
[10]https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html

benigns, and extract the twenty-five features shown in Section IV-B. We then shuffle the benign dataset, and 20,000 domains are used as the training data, and the remaining 5,000 domains are used as the test data, respectively. Likewise, we collect the 35,000 malicious domains from URLhaus, CyberCrime Tracker, and PhishTank, including the time of malicious domain, observed from January 29 to November 25. The domain data from the three databases described above are merged and then sorted in chronological order. Finally, the twenty-five features are extracted in the same manner as the benign domains. Hereinafter, we define the dataset of 35,000 malicious domains as $d_1 - d_{35,000}$ for the sake of convenience. Using the time-series data described above, we evaluate how much the F1 score, G-mean, and the accuracy of the retrained model are different from those of the normal model.

More specifically, both the normal model and the retrained model are trained with 20,000 benign domains and 20,000 malicious domains, i.e., (d_1-$d_{20,000}$) as the training data, where the malicious domains were generated from 22:48:07 29, Jan. and 15:10:52, 25 Aug. The real-time training is then performed with malicious domains observed from 15:15:05, 25 Aug. to 12:01:15, 28 Oct. following the time series. In doing so, d_i is corresponding to the latest malicious domain data then, where a domain on the last row is used if multiple malicious domains exist simultaneously.

Whereas the normal model is kept until the end of the experiments, the retrained model is retrained iteratively by shifting to the latest 20,000 malicious domains, i.e., ($d_{i-20,000}$-d_i), for every thirty seconds. Then, we evaluate the scores of metrics for detection, i.e., F1 score, G-mean, and accuracy, of the two models for every thirty seconds by utilizing 5,000 benign domains and the future 5,000 malicious domains, i.e., d_i-d_{i+5000}, as the test data. We also measure the training time on a server to evaluate the throughput for MADMAX, where throughput is computed as the average of 10,000 iterations.

C. RESULTS

The experimental results are shown below.

1) Selection of Optimized Features

First, we measure the permutation importance [22] for each model with respect to the number N of nodes in a hidden layer. The result on the feature importance is shown in Table 1. The rankings are almost stable. For instance, the top three important features are common for each number of nodes except for $N = 100$.

Next, based on the feature importance ranking in Table 5, the F1 score, the G-mean, the accuracy, the recall, and the precision for T and the number N of nodes in a hidden layer are shown in Figure 4, where the value of T means the use of the top T features in Table 5. The score of the precision is higher than that of the recall. It means that the number of malicious domains wrongly detected as benign is fewer than that of benign domains wrongly detected as malicious. Meanwhile, the F1 score, the G-mean, and the accuracy have numerical stability, and then we focus on the F1 score. The highest F1 score is 0.885 for $N = 600$ and $T = 10$. Moreover, the F1 score, the G-mean, and the accuracy are improved by selecting the important features rather than using all the features. According to Cao et al. [26], more hidden layer nodes on ELM cannot guarantee the best performance of the ELM. Our result described above is identical to the finding by Cao et al. [26].

Following the above results, the model with 600 nodes is utilized for evaluating the throughput of MADMAX because the highest F1 score is achieved. The result of the throughput is shown in Figure 5, where throughput is measured with respect to the threshold T[11]. Entirely, the throughput decreases in proportion to the number of the threshold T, i.e., the number of features increases. For instance, while the detection time is 1.0 seconds for benign domains and 1.5 seconds for malicious domains for $T = 6$, those are 3.3 seconds for benign domains and 4.6 seconds for malicious domains for $T = 10$.

As a result, we first choose $N = 600$ and $T = 10$ as the selection of optimized features from the perspective of the F1 score. Meanwhile, also throughput should be considered as a browser-based application. In the case of throughput, the values of $5 \leq T \leq 10$ in the column of $N = 600$ seem to provide high F1 scores. For these values of T, the highest throughput is provided for $T = 5$ according to Figure 5. Therefore, we measure the performance of the real-time training for $N = 600$ and $5 \leq T \leq 10$ in the next paragraph. (See Table 1 for the selected features.)

2) Real-time Training

We show the results of the F1 score, G-mean, and the accuracy of the two models, i.e., the normal model and the retrained model, for each set in Figures 6, 7, and 8, where the retraining is conducted 6,136 times totally during the real-time training. Note that, as described in the previous paragraph, we adopt $N = 600$ as the number of nodes in a hidden layer and $5 \leq T \leq 10$ as the number of optimized features in this experiment. According to the figures, the F1 score, G-mean, and the accuracy for each model increase entirely. Also, the results of the F1 score and the G-mean are almost the same as each other.

Meanwhile, from early October, the F1 score, G-mean, and accuracy of the normal model become lower than those of the retrained model for each result. Namely, the retrained model can provide a better F1 score, G-mean, and accuracy than the normal model.

Meanwhile, the training time of the ELM model inside MADMAX is shown in Figure 9. The training times are stable for every threshold T and are 1.6 seconds.

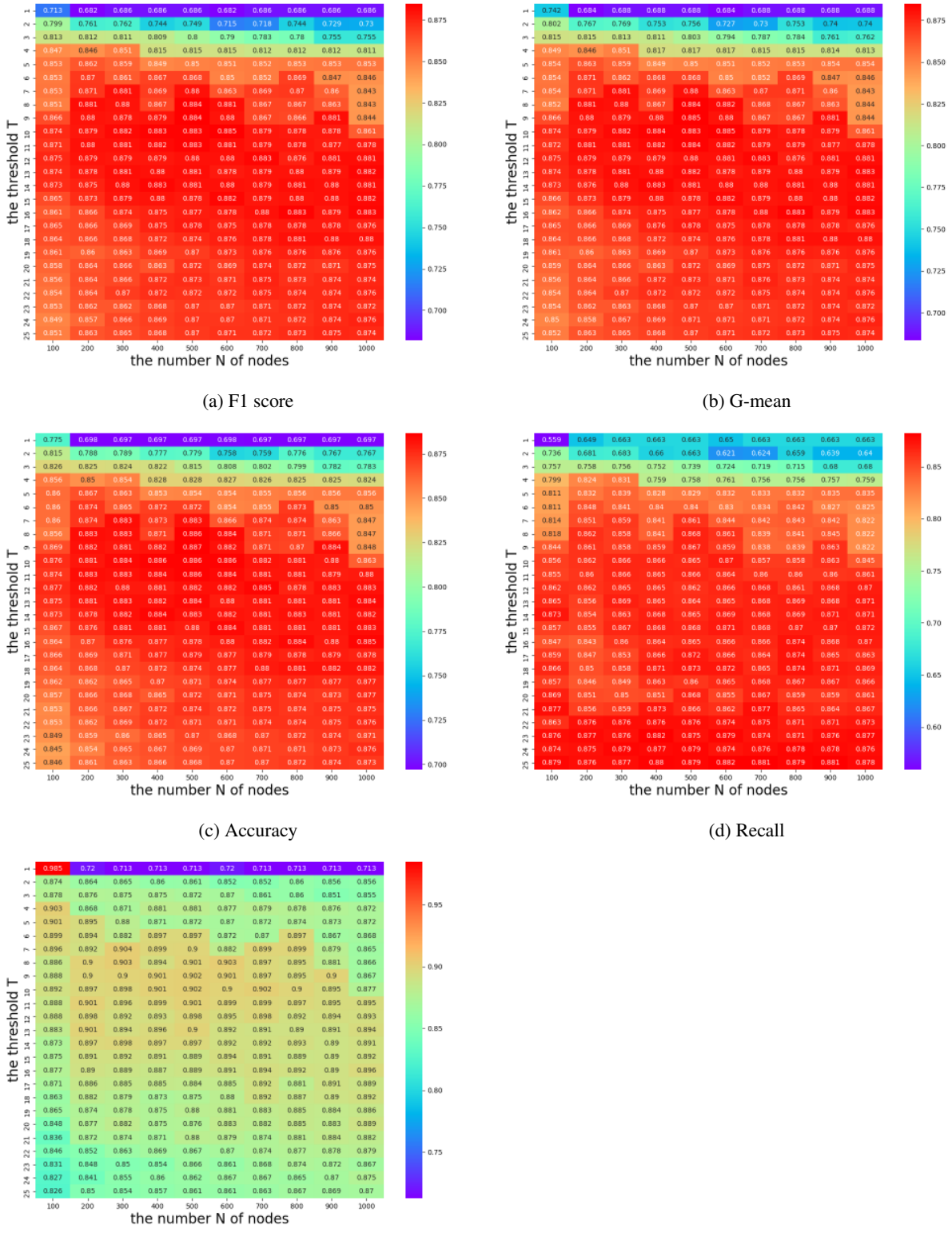
[11] We do not take into account changing the number N of nodes for $T = 10$ because the primary key question in this paper is to find a set of optimized "features".

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

TABLE 1: Feature importance ranking based on the permutation importance with respect to the number N of nodes: The rankings are sorted in accordance with the permutation importance. A higher position represents more important for malicious domain detection. Each column represents the ranking based on the number N of nodes.

	100	200	300	400	500	600	700	800	900	1000
1	Number of NS	Length of domain	Length of domain	Length of domain	Length of domain	Length of domain	Length of domain	Length of domain	Length of domain	Length of domain
2	Length of domain	Number of NS	Number of NS	Number of NS	Number of NS	Number of NS	Number of NS	Number of NS	Number of NS	Number of NS
3	Number of numeric characters	Number of consonants	Number of consonants	Number of consonants	Number of consonants	Number of consonants	Number of consonants	Number of consonants	Number of consonants	Number of consonants
4	Mean of TTL	WHOIS life time	WHOIS life time	Number of vowels	Number of vowels	Number of vowels	Number of vowels	Number of vowels	Number of vowels	Number of vowels
5	Number of MX	Mean of TTL	Number of HTML elements	WHOIS life time	WHOIS life time	WHOIS life time	WHOIS life time	WHOIS life time	WHOIS life time	WHOIS life time
6	Number of vowels	Number of MX	Number of vowels	Mean of TTL	Mean of TTL	Rate of numeric characters	Rate of numeric characters	Mean of TTL	Number of consonant characters	Number of consonant characters
7	Rate of numeric characters	Rate of numeric characters	Mean of TTL	Rate of numeric characters	Number of HTML elements	Number of HTML elements	Mean of TTL	Rate of numeric characters	Number of HTML elements	Entropy
8	Rate of vowel	Number of HTML elements	Rate of numeric characters	Number of numeric characters	Number of MX	Mean of TTL	Rate of vowel	Number of consonant characters	Rate of numeric characters	Rate of vowel
9	Number of HTML elements	Number of numeric characters	Number of numeric characters	Number of HTML elements	Rate of numeric characters	Number of consonant characters	Number of consonant characters	Rate of vowel	Mean of TTL	Rate of numeric characters
10	WHOIS life time	Rate of vowel	Number of MX	Number of MX	Number of numeric characters	Number of MX	Number of HTML elements	Number of HTML elements	Entropy	Number of HTML elements
11	Number of consonants	Standard deviation of TTL	Number of consonant characters	Rate of vowel	WHOIS active time	Rate of vowel	Number of numeric characters	Number of numeric characters	Rate of vowel	Mean of TTL
12	Standard deviation of TTL	Number of vowels	Rate of vowel	Number of consonant characters	Rate of vowel	Number of numeric characters	Number of MX	Entropy	Number of MX	Number of MX
13	Number of consonant characters	Number of IP	Standard deviation of TTL	Number of IP	Standard deviation of TTL	Entropy	Entropy	Number of MX	Number of numeric characters	Reputation Value
14	Number of other characters	Conversions of vowels and consonants	Number of IP	Standard deviation of TTL	Number of consonant characters	Standard deviation of TTL	Number of IP	Number of IP	Reputation Value	Number of numeric characters
15	WHOIS active time	Number of consonant characters	Entropy	Entropy	Entropy	Number of IP	WHOIS active time	Reputation Value	Number of IP	Number of IP
16	Number of IP	WHOIS active time	NS similarity	WHOIS active time	Number of IP	WHOIS active time	Standard deviation of TTL	Standard deviation of TTL	WHOIS active time	NS similarity
17	Conversions of numerals and alphabets	Conversions of numerals and alphabets	WHOIS active time	Reputation Value	Reputation Value	Number of vowel characters	NS similarity	Number of vowel characters	NS similarity	WHOIS active time
18	Entropy	Entropy	Reputation Value	Number of vowel characters	Number of vowel characters	Reputation Value	Reputation Value	NS similarity	Standard deviation of TTL	Standard deviation of TTL
19	Conversions of vowels and consonants	NS similarity	Conversions of vowels and consonants	Conversions of vowels and consonants	NS similarity	NS similarity	Number of vowel characters	WHOIS active time	Number of vowel characters	Number of vowel characters
20	Number of vowel characters	Number of other characters	Conversions of numerals and alphabets	NS similarity	Conversions of numerals and alphabets	Conversions of vowels and consonants	Conversions of numerals and alphabets	Conversions of vowels and consonants	Conversions of vowels and consonants	Conversions of vowels and consonants
21	Reputation Value	Number of vowel characters	Number of vowel characters	Number of other characters	Conversions of vowels and consonants	Conversions of numerals and alphabets	Number of other characters	Conversions of numerals and alphabets	Conversions of numerals and alphabets	Conversions of numerals and alphabets
22	NS similarity	Reputation Value	Number of other characters	Conversions of numerals and alphabets	Number of other characters	Number of other characters	Conversions of vowels and consonants	Number of other characters	Number of other characters	Number of other characters
23	Length of max consecutive character	Length of max consecutive character	Length of max consecutive character	Length of max consecutive character	Length of max consecutive character	Length of max consecutive character	Length of max consecutive character	Length of max consecutive character	Length of max consecutive character	Length of max consecutive character
24	Number of countries	Number of PTR	Number of PTR	Number of PTR	Number of PTR	Number of PTR	Number of PTR	Number of PTR	Number of PTR	Number of PTR
25	Number of PTR	Number of countries	Number of countries	Number of countries	Number of countries	Number of countries	Number of countries	Number of countries	Number of countries	Number of countries

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access



(a) F1 score

(b) G-mean

(c) Accuracy

(d) Recall

(e) Precision

FIGURE 4: F1 score, G-mean, accuracy, recall, and precision for T and N: The heatmaps are introduced in measuring the five evaluation metrics described in Section V-B2. The above maps are colored in accordance with the feature importance measured by the permutation importance. The bar on the right side for each map represents colors based on the feature importance. The red color and blue color mean high score and low score, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

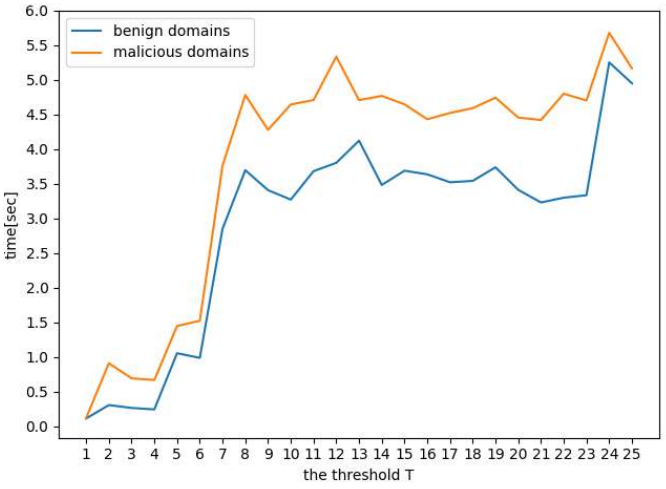


FIGURE 5: The detection time with respect to the threshold: Features are sorted in high order in accordance with the feature importance.

VI. DISCUSSION

In this section, we discuss the results for each experiment and then compare the performance of MADMAX with the existing work [8] of the ELM-based malicious domain detection. Next, we discuss results if an imbalanced dataset is utilized as a more real-world application. We then discuss the initialization of weight matrices of ELM inside MADMAX. Finally, we explain the limitations of MADMAX.

A. SELECTION OF OPTIMIZED FEATURES

We discuss the results on the selection of optimized features in terms of the feature importance, the improvement of accuracy, the throughput on malicious domains, and the trade-off between accuracy and throughput.

a: Feature Importance

First, we discuss the feature importance. Our feature importance based on the permutation importance for MADMAX is different from that based on the LightGBM [27] shown in our previous work work [11]. In particular, in comparison with the top 13 important features presented in [11], there are only seven common features for MADMAX, i.e., the length of a domain, the number of NS, the WHOIS lifetime, the number of HTML elements, the mean of TTL, the number of MX, and the entropy [23]. The difference indicates several considerations for malicious domain detection as described below.

According to our previous work [11], the length of domains and the entropy are essential as the text-based features because they represent the characteristics of malicious domain names generated by DGA. More specifically, names of benign domains are usually composed of one or two words, the second-level domain (SLD) and the top-level domain (TLD). On the other hand, malicious domains generated by DGA are often composed of long-and-random strings to avoid collision of domain names [28]. In other words, the generated domains are no longer registered as long as they have a collision with existing domains. Consequently, malicious domains tend to provide longer names and higher entropy than benign domains.

Meanwhile, for DNS-based features, the number of NS and the number of MX for malicious domains are less than those for benign domains. The reason is that malicious domains provide fewer functions compared to benign domains. Although we omit the detail of domains from the perspective of ethics, many multinational organizations are top benign domains.

Likewise, for the web-based features, the number of HTML elements in malicious domains tends to be fewer than that in benign domains. Indeed, when we investigate the average number of HTML elements in the 24,126 benign domains and the 24,126 malicious domains, the average numbers are 724 for the former and 136 for the latter. It is considered that several malicious domains, which only distribute malware, do not care about the provided web contents' designs. Therefore, the number of HTML elements is less for malicious domains than benign domains.

b: Improvement of Accuracy

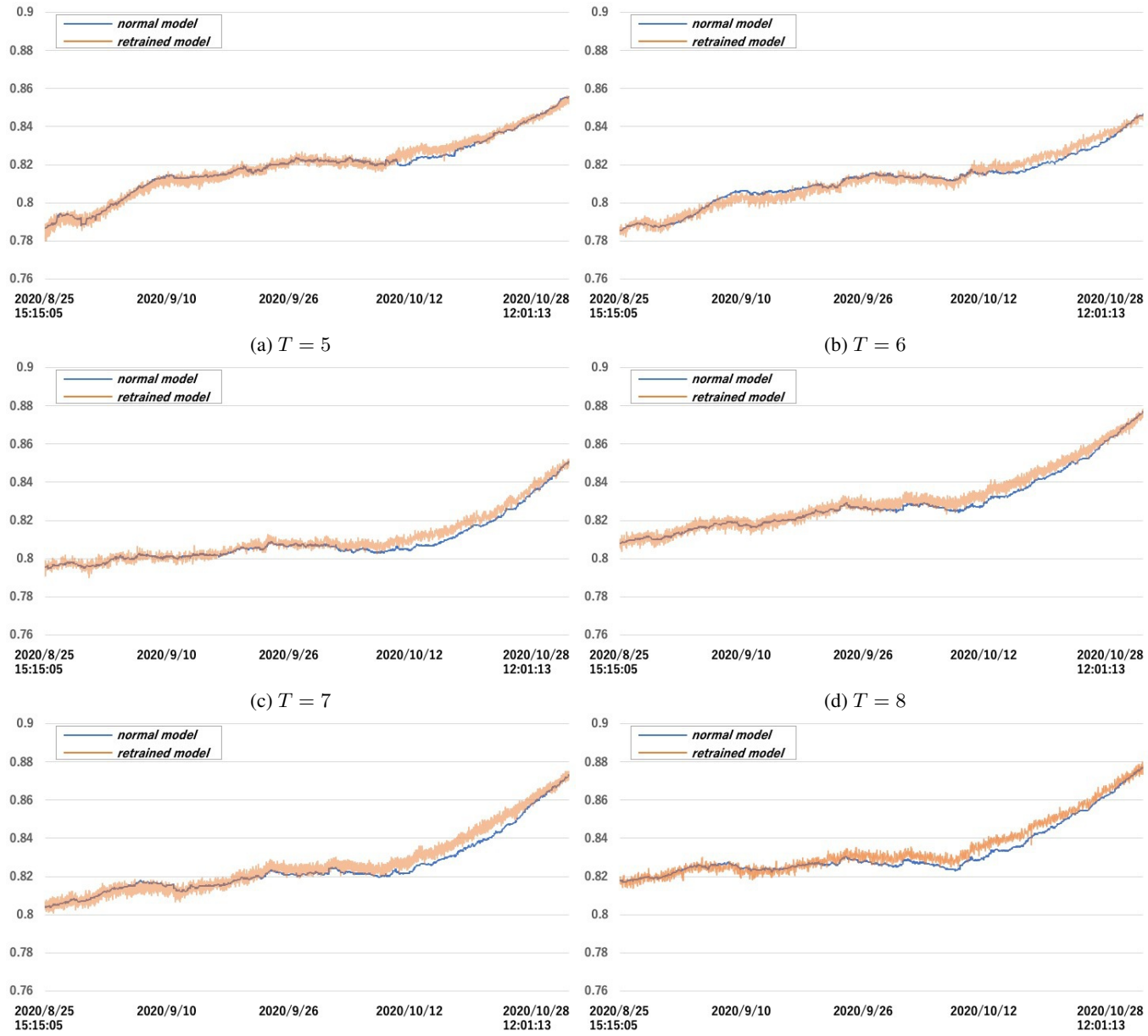
Second, MADMAX provides better scores for both the F1 score and the accuracy by selecting several important features, rather than utilizing all the features shown in Figure 4. It means that unnecessary features, which are removed by the selection of optimized features, perturb essential characteristics of malicious domains. Accordingly, the F1 score and the accuracy of MADMAX can be improved by virtue of the selection of optimized features.

c: Throughput on Malicious Domains

Third, Figure 5 shows that the detection time for malicious domains is longer than that for benign domains. The reason is that malicious domains are often unregistered because they are newly generated. Namely, DNS records for malicious domains should be requested from a browser with MADMAX to DNS servers and hence consumes the time for extracting the DNS-based features. Accordingly, Figure 5 shows that the detection time for malicious domains much increases from $T = 1$ to $T = 2$ and from $T = 9$ to $T = 10$ than those of benign domains. The above insight is identical to our previous work [11].

Meanwhile, the throughput decreases as the threshold T increases according to Figure 5. For instance, the throughput for both malicious and benign domains decreases from $T = 6$ to $T = 7$. Indeed, the reason is different for benign domains and malicious domains. Generally speaking, benign domains provide well-designed web content with numerous HTML tags, so the time is consumed to obtain the contents. On the other hand, HTML tags for several malicious domains cannot be gained because they provide no HTML contents as described above.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access



(a) $T = 5$ (b) $T = 6$

(c) $T = 7$ (d) $T = 8$

(e) $T = 9$ (f) $T = 10$

FIGURE 6: F1 score of the normal model and the retrained model for each threshold T: The blue lines represent the precisions for the normal model, and the orange line represents the precisions for the retrained model, respectively.

d: Trade-off

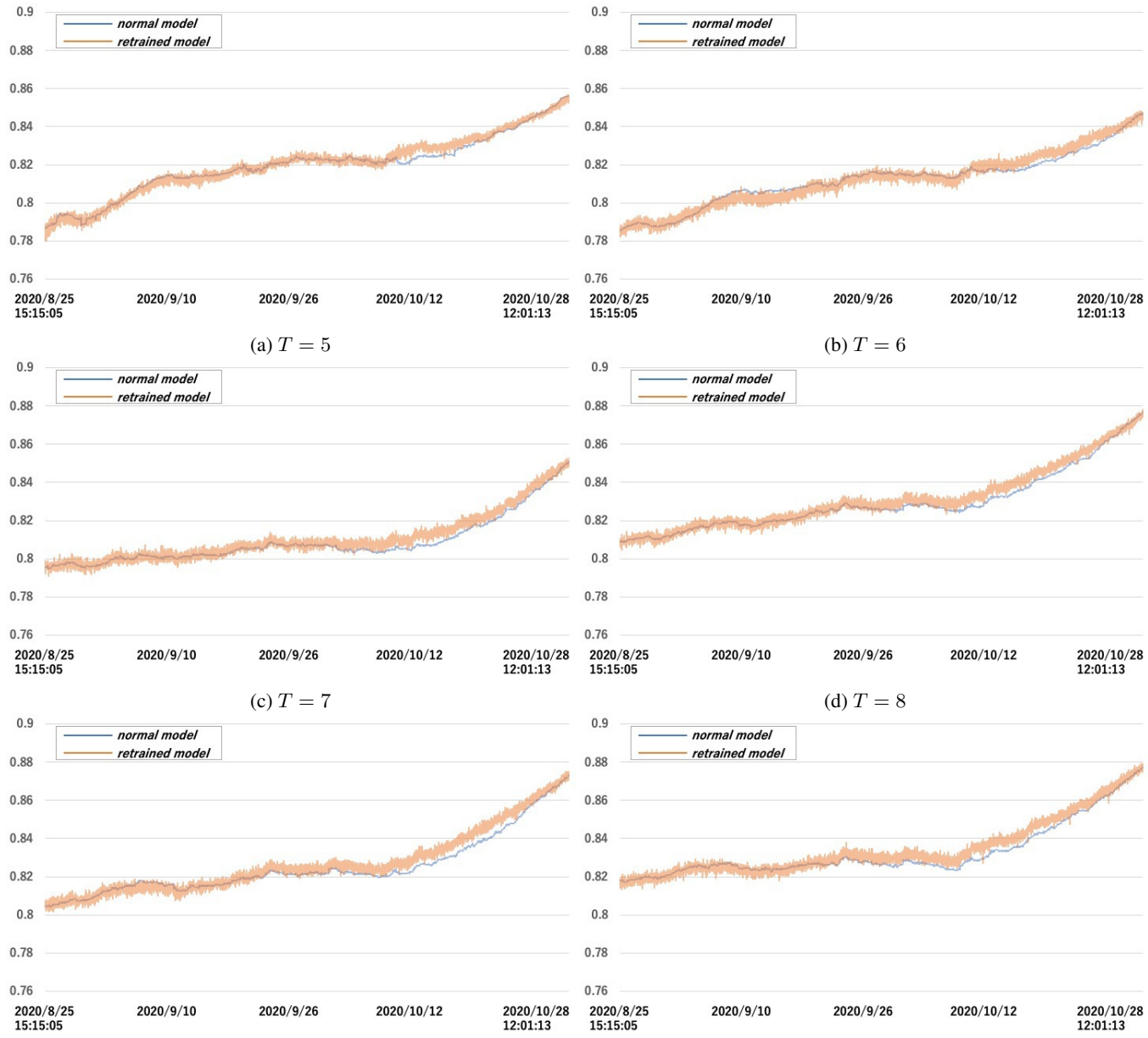
Finally, we conclude the trade-off between accuracy and throughput for MADMAX. According to Figures 4 and 5, the setting with $T = 6$ and $N = 600$, i.e., the length of domains, the number of NS, the number of consonants, the number of vowels, WHOIS lifetime, and the rate of numeric characters, is optimal for MADMAX. Besides, if the highest detection rate is necessary, the setting with $T = 7$ and $N = 600$, i.e., the number of HTML elements in addition to the six features described above, is the best for MADMAX.

B. REAL-TIME TRAINING

We discuss the effect of the real-time training on detecting unseen malicious domains below. According to the experimental results shown in Section V-C2, since some concept drifts occurred during the period, the accuracy of the normal model decreased after early October for all the patterns. In contrast, the retrained model was able to learn features of newly appeared malicious domains.

For further discussion, we measure the precision and the recall on the experiments and the results are shown in Figures 10 and 11. According to these figures, the normal

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access



(a) $T = 5$ (b) $T = 6$

(c) $T = 7$ (d) $T = 8$

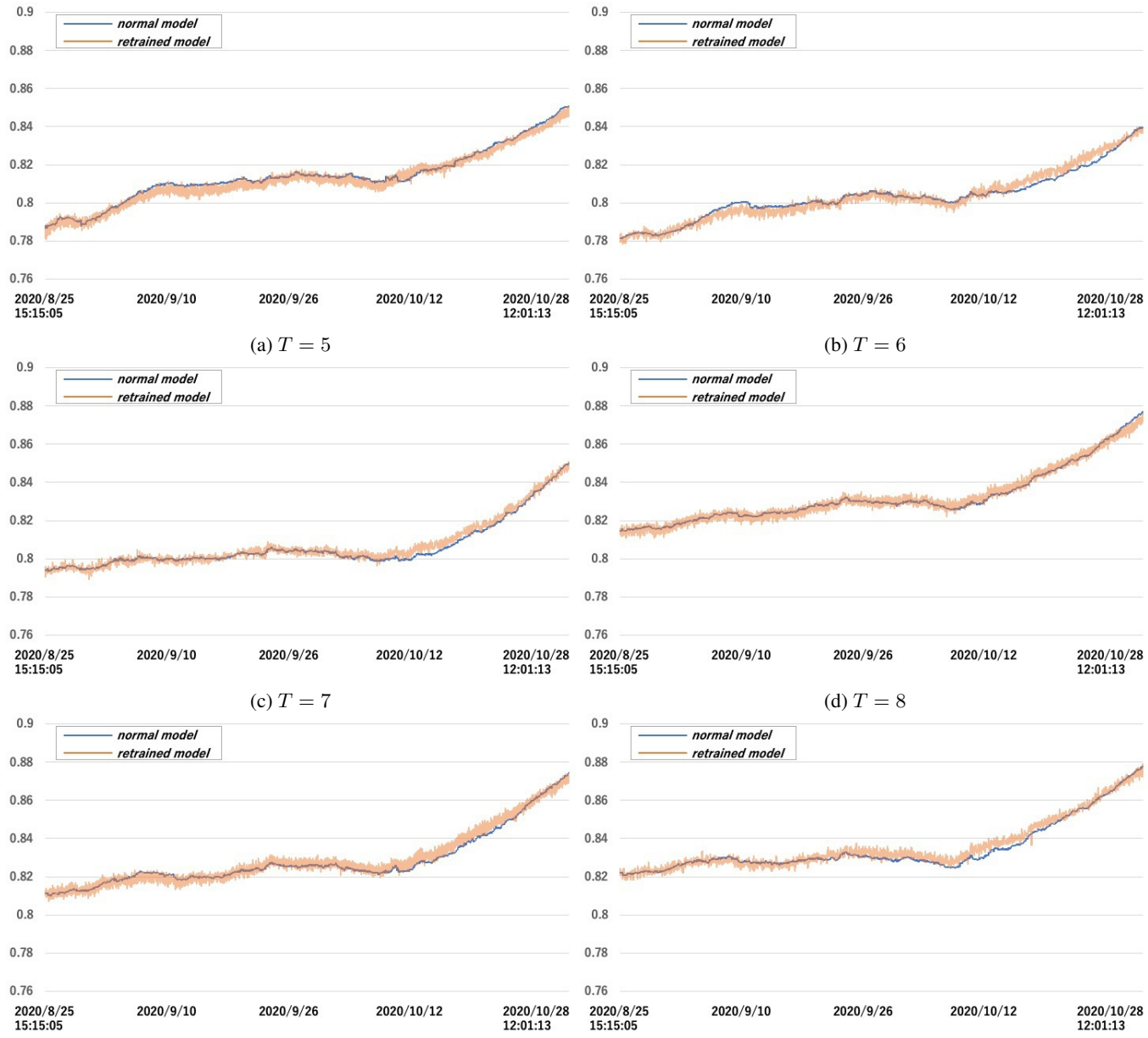
(e) $T = 9$ (f) $T = 10$

FIGURE 7: G-mean of the normal model and the retrained model for each threshold T: The blue lines and the orange lines are defined in the same manner in Figure 7.

model has higher precision but lower recall than those of the retrained model. The above fact means that the normal model has fewer false positives but more false negatives about malicious domains than the retrained model. In other words, the normal model missed unseen malicious domains while the retrained model could detect them. Tables 2- 7 show the metrics by each model at the time when the F1 score of the normal model and the retrained model is the most distant for each threshold T. Each time for any T is within the period after early October, i.e., the concept drift has occurred. The F1 score and G-mean of the retrained model are at most 0.011 higher than those of the normal model. At that time, the recall of the retrained model is at most 0.025 higher than that of the normal model as a ratio of missing malicious domains. In the experiments, the dataset of 2,500 malicious domains is used as the test data. Then, the difference in the recall values between the retrained model and the normal model implies that the normal model misses 625 malicious domains more than the retrained model.

Next, we discuss an update of the benign domain dataset. In our real-time training experiment, only the data of malicious domains were updated from August 25 to October

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access



(a) $T = 5$ (b) $T = 6$

(c) $T = 7$ (d) $T = 8$

(e) $T = 9$ (f) $T = 10$

FIGURE 8: Accuracy of the normal model and the retrained model for each threshold T: The blue lines and the orange lines are defined in the same manner in Figure 6.

TABLE 2: The difference of metrics between the retrained model and the normal model for $T = 5$.

Metrics	F1 score	G-mean	Accuracy	Precision	Recall
The normal model	0.819	0.82	0.811	0.785	0.857
The retrained model	0.829	0.831	0.818	0.781	0.884
difference	0.00976	0.01057	0.0067	-0.00441	0.0272

28, while the data of benign domains were used from Tranco [25] on November 25 statically, i.e., without the update of the 24,126 domains. Thus, it is considered that both the F1 score and the accuracy for each model increase entirely toward November 25. This result indicates that the data of benign domains should be updated for the real-time training.

Next, we discuss the throughput of the real-time training. Although the retrained model was trained for every thirty seconds in the experiment, we believe that the model would be retrained only when a new malicious domain is pulled

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

TABLE 3: The difference of metrics between the retrained model and the normal model for $T = 6$.

Metrics	F1 score	G-mean	Accuracy	Precision	Recall
The normal model	0.828	0.817	0.82	0.792	0.868
The retrained model	0.837	0.826	0.83	0.802	0.875
difference	0.00855	0.00878	0.0095	0.01048	0.0062

TABLE 4: The difference of metrics between the retrained model and the normal model for $T = 7$.

Metrics	F1 score	G-mean	Accuracy	Precision	Recall
The normal model	0.808	0.808	0.804	0.791	0.8256
The retrained model	0.818	0.819	0.812	0.790	0.849
difference	0.01058	0.0109	0.0082	-0.00016	0.0234

TABLE 5: The difference of metrics between the retrained model and the normal model for $T = 8$.

Metrics	F1 score	G-mean	Accuracy	Precision	Recall
The normal model	0.825	0.825	0.826	0.831	0.818
The retrained model	0.834	0.834	0.832	0.823	0.845
difference	0.00931	0.00935	0.0057	-0.0082	0.027

TABLE 6: The difference of metrics between the retrained model and the normal model for $T = 9$.

Metrics	F1 score	G-mean	Accuracy	Precision	Recall
The normal model	0.838	0.838	0.841	0.853	0.824
The retrained model	0.849	0.849	0.849	0.849	0.85
difference	0.01104	0.01092	0.0082	-0.00421	0.0258

TABLE 7: The difference of metrics between the retrained model and the normal model for $T = 10$.

Metrics	F1 score	G-mean	Accuracy	Precision	Recall
The normal model	0.829	0.829	0.83	0.832	0.826
The retrained model	0.839	0.839	0.838	0.835	0.844
difference	0.01026	0.01027	0.0087	0.00236	0.0182

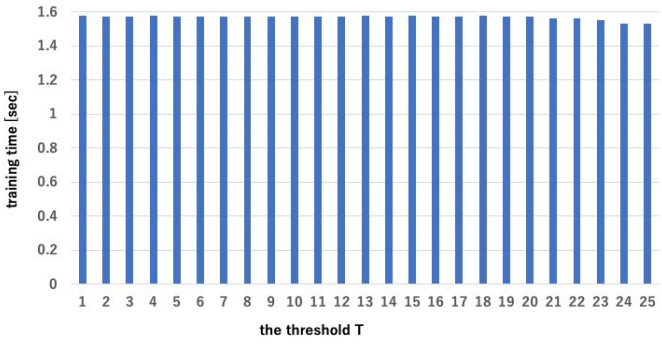


FIGURE 9: The training time for the real-time training of MADMAX: The training time for the threshold T is measured by that of the ELM model on a server. Here, the number N of nodes in the ELM model is 600.

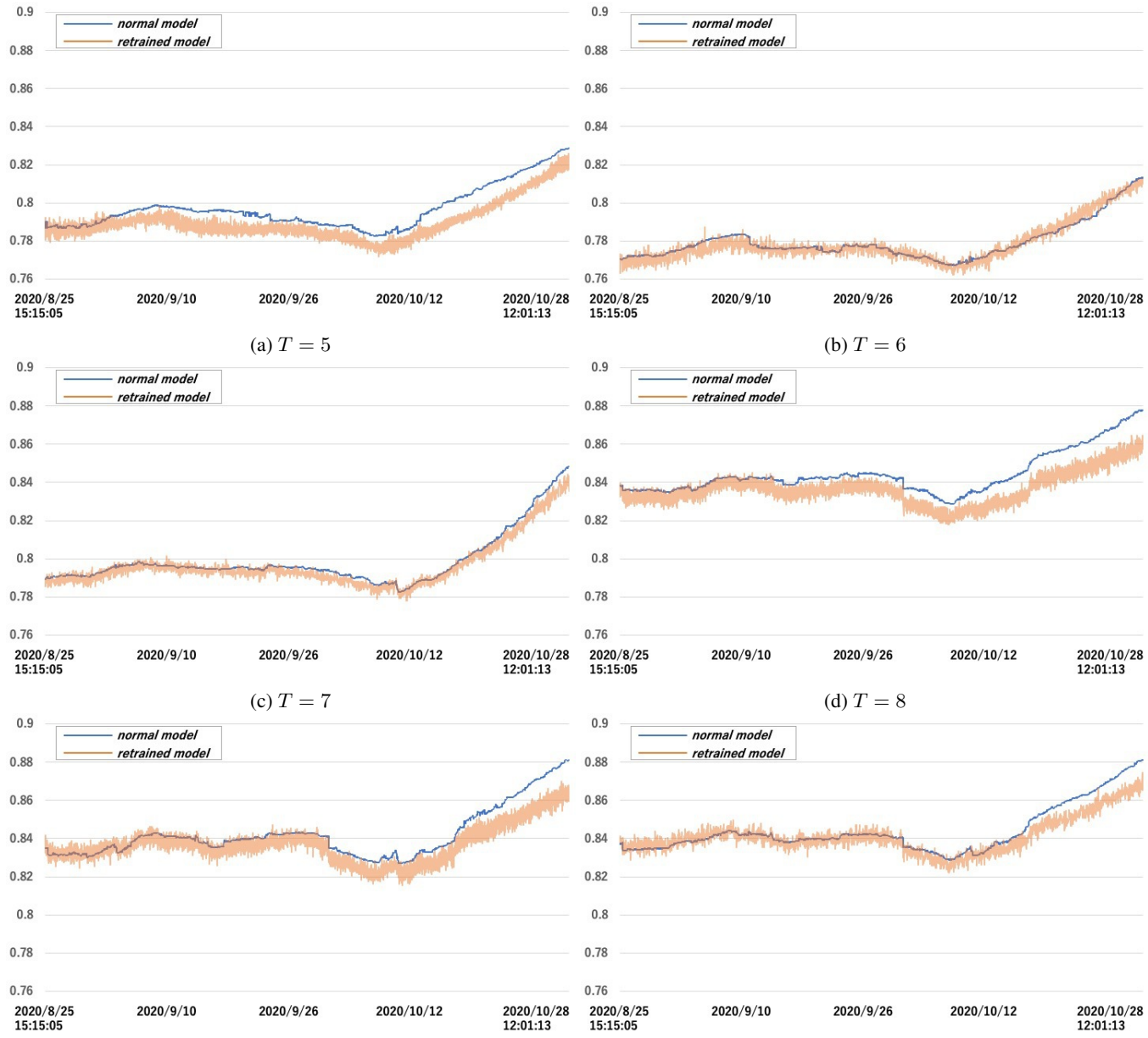
from a malicious database in an actual use case. Indeed, in the three databases for malicious domains used in our experiments, domains are updated every twelve minutes on average. In doing so, there are many cases where two or three domains are newly given, and the variance is considerably significant, e.g., in the maximized case, 64 domains are given in just one update. The frequency of new domains in one update is shown in Figure 12.

More concretely, we discuss the time required for pulling data of the new domains from the databases to update the retrained model, i.e., we call the time *vulnerable slot against unseen domains* for the sake of convenience. As shown in Figure 9, the training time of ELM with 600 nodes is generally within 1.6 seconds. We also measure the training time with typical backpropagation for neural networks, which is the same architecture as ELM used in the experiment above, i.e., a neural network with 600 nodes in a hidden layer in the same server setting. The time required for updating an ELM model and a neural network model is then shown in Table 8, where the time includes feature extraction for domains and the training with its resulting features. According to Table 8, MADMAX has a significantly higher throughput and drastically shortens the vulnerable slot against unseen domains than those of neural networks.

C. COMPARISON WITH EXISTING WORK

We discuss the performance of MADMAX in comparison with the existing work [8] of the ELM-based malicious domain detection. Table 9 shows the results of comparing each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access



(a) $T = 5$ (b) $T = 6$

(c) $T = 7$ (d) $T = 8$

(e) $T = 9$ (f) $T = 10$

FIGURE 10: Precision for the normal model and the retrained model for each threshold T: The blue lines and the orange lines are defined in the same manner in Figure 6.

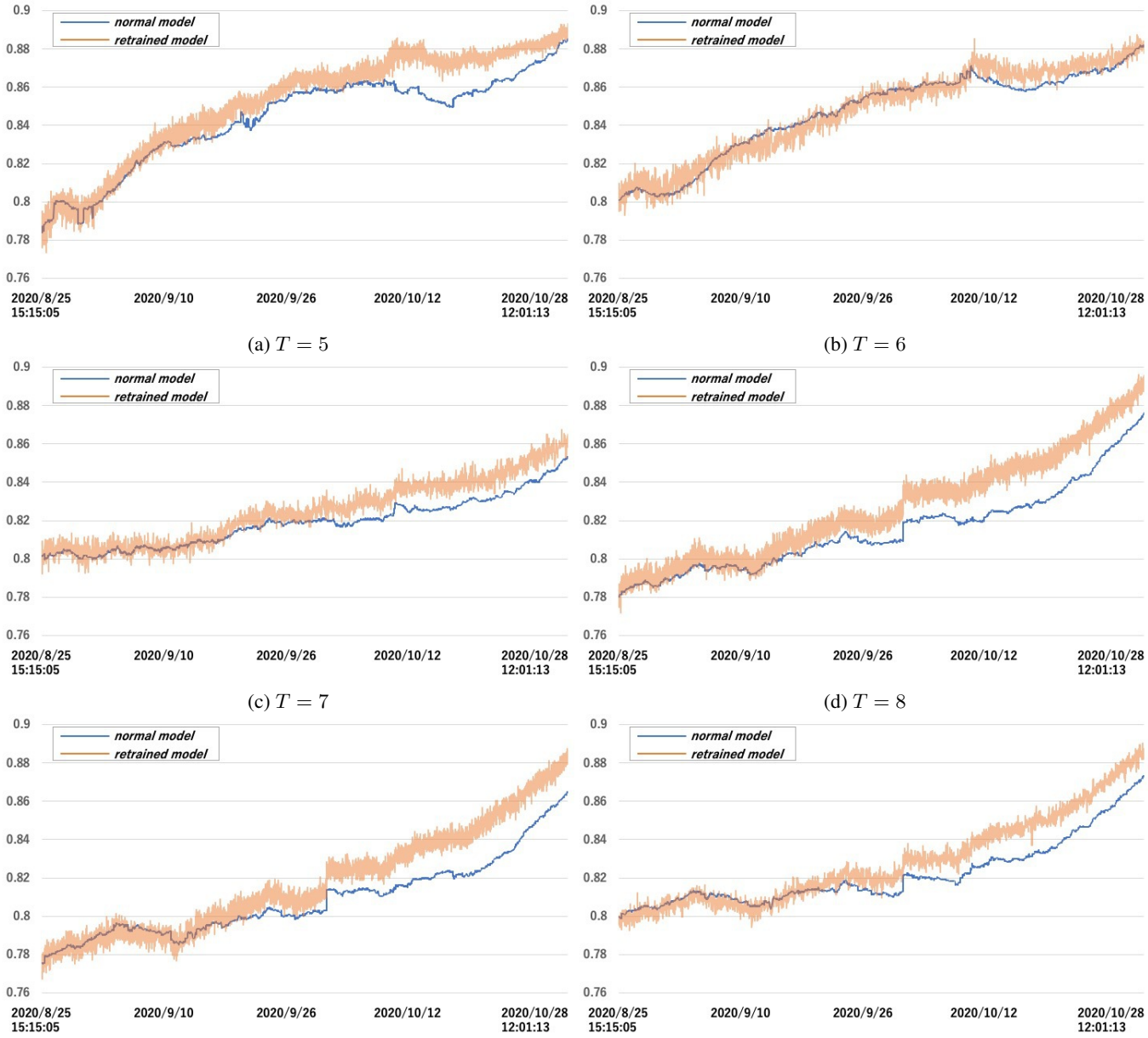
evaluation index with the throughput using the features used in existing work [8] and the features selected by MADMAX. Note that since a different dataset is used in [8], the comparison here is conducted using the dataset in our previous work [11]. In the model of 600 nodes, the detection time of the malicious domains is 0.7 seconds faster when utilizing the 6 features in MADMAX compared to the features of the existing work [8]. On the other hand, in the model of 500 nodes, the F1 score is 0.2 higher with the same throughput when utilizing the 6 features in MADMAX compared to the features of the existing work [8].

Consequently, MADMAX can provide better performance than trivially introducing the existing work [8] by virtue of our selection of the optimized features as a browser-based application.

D. EVALUATION BY USING IMBALANCED DATASET

We discuss the performance of MADMAX with a balanced dataset and an imbalanced dataset. Table 10 shows the results of comparing the F1 scores in training a model with each of the balanced dataset and the imbalanced dataset in which the number of benign and malicious domains is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access



(a) $T = 5$ (b) $T = 6$

(c) $T = 7$ (d) $T = 8$

(e) $T = 9$ (f) $T = 10$

FIGURE 11: Recall of the normal model and the retrained model for each threshold T: The blue lines and the orange lines are defined in the same manner in Figure 6.

biased. Concretely, the experiment is conducted in a total of three datasets, i.e., the balanced dataset with the 24,126 benign domains and the 24,126 malicious domains, the imbalanced dataset with the 6,050 benign domains and the 19,500 malicious domains, and the imbalanced dataset with the 19,500 benign domains and the 6,050 malicious domains. In addition, all twenty-five features are used, and the F1 score is shown.

From Table. 10, in any model, the F1 score is higher utilizing the UMD than the UBD and BD. It is likely to capture malicious domains more accurately due to the small number of benign domains. In other words, more accurate detection is possible by constructing a dataset that collects more malicious domains.

E. INITIALIZATION OF WEIGHT MATRICES

We discuss the initialization of weight matrices of ELM inside MADMAX. We assumed that weight matrices of ELM are randomly chosen in the experiments in Section V but did not concern the impact of the initialization of weight matrices on model performance. Stochastic parameters, i.e., weight matrices, of ELM may potentially affect the instability of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

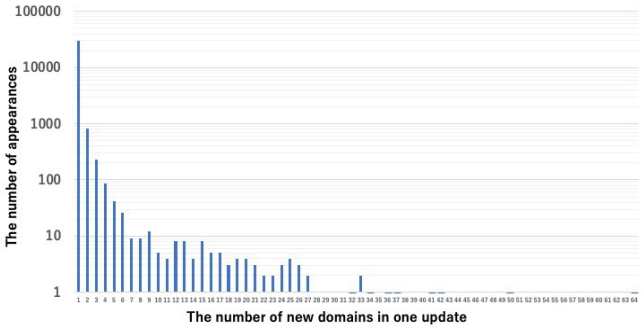


FIGURE 12: The frequency of new domains in one update: The y-axis is represented as a log-scale graph. For 32, 34, 36, 37, 41, 42, 50, and 64 on the x-axis, only a new single domain appears. Meanwhile, for more than 65 on the x-axis, no new domain appears in one update.

TABLE 8: Time to update the ELM and neural network model: As the number of newly pulled domains in one update, we show for frequent one and two, and up to sixty-four, respectively. We denote by ELM(X) the time to update an ELM for X new domains in one update for any X. Similarly, we denote by NN(X) the time to update a neural network in the same manner.

Scheme	$T = 5$	$T = 6$	$T = 7$	$T = 8$	$T = 9$	$T = 10$
ELM (1)	3.02	3.09	5.34	6.35	5.85	6.22
ELM (2)	4.47	4.61	9.11	11.13	10.13	10.87
ELM (3)	5.92	6.13	12.88	15.91	14.41	15.52
ELM (64)	94.37	98.85	242.85	307.49	275.49	299.17
NN (1)	101.45	90.52	90.77	171.78	199.28	283.65
NN (2)	102.9	92.04	94.54	176.56	203.56	288.3
NN (3)	104.35	93.56	98.31	181.34	207.84	292.95
NN (64)	192.8	186.28	328.28	472.92	468.92	576.6

malicious domain detection in general.

Fortunately, the above concern can be overcome by leveraging existing techniques [29], [30]. In particular, initializing ELM with the Gaussian distribution can help the model have a faster convergence rate than the uniform distribution. Besides, both a convergence rate and generalization performance become better as long as a distribution with more minor variances is provided. Thus, when MADMAX is deployed in a real-world environment, the performance will become stable by initializing weight matrices with the Gaussian distribution suitable for the environment.

F. LIMITATIONS

We describe several limitations of the current specification of MADMAX below.

1) Feature Importance

MADMAX leverages the permutation importance to determine the feature importance. However, the permutation importance has a limitation, whereby particular feature importance degrades due to variance on features. For example, the number of countries is one of the most important features for detecting malicious domains [11]. Nonetheless, the feature importance of the number of countries is relatively low for MADMAX under the permutation importance. This reason is that most of the malicious domains have "0" as the value about the number of countries, and therefore the values are stable even if they are shuffled randomly. In other words, the number of countries does not have an impact on accuracy as long as the permutation importance is utilized. Future work will thus explore more possibilities by other methods for the feature importance.

2) Server Managements

MADMAX is a client-server application, so it is necessary to prepare and deploy a server that detects malicious domains to use MADMAX in advance. Furthermore, since a user needs to send domain data to the server through the browser add-on, the server should be maintained continuously to provide the functions of MADMAX. If a user wants to utilize MADMAX local, he/she needs to set up a localhost environment and then manage the server by him-/herself to detect malicious domains.

3) Impossibility of Extracting Features

According to the experimental results in Section V-C2, in the data collection process for the real-time training, several features cannot be obtained, e.g., WHOIS lifetime due to the failure of WHOIS search or the number of HTML elements due to certificates with unavailable language. We call such features *missing values* for the sake of convenience.

We then replace such features with zeros in the experiments. Although we have not discussed that the performance of MADMAX is influenced rigorously, generally speaking, a model may learn the missing values themselves in proportion to the number of the missing values. Namely, the malicious domain detection by MADMAX may have an unexpected influence caused by the missing values. As another way to handle the missing values, when the missing values are found, the corresponding data can be excluded, or the missing values can be replaced with a numerical value other than 0. Further studies, which take the missing values into account, will need to be undertaken.

4) Accuracy Unstability on Nonlinear Data

The accuracy of the original ELM utilized in MADMAX is often unstable, especially for nonlinear data. The dataset utilized in the current experiments has linearity, and hence accuracy is stable. However, nonlinear data may appear when MADMAX is utilized in more various domains. To capture nonlinear data, more advanced ELMs such as RC-ELM [16] or robust-model ELM [17] can be utilized. We recommend that further research should be undertaken on the topic mentioned above.

VII. RELATED WORK

In this section, we describe related works of malicious domain detection. Malicious domain detection is classified into

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

TABLE 9: Comparison with the existing work [8]: Time of benign domains and time of malicious domains indicate the average time to send each domain to the server and receive its result for repeating 100 times.

Scheme	F1 score	Accuracy	Precision	Recall	Time of benign domains(s)	Time of malicious domains(s)
Shi et al. [8]($N = 500$)	0.848	0.85	0.855	0.841	0.8	2.4
Shi et al. [8]($N = 600$)	0.85	0.851	0.856	0.843	0.9	2.3
MADMAX($T = 6, N = 500$)	0.868	0.872	0.897	0.84	1.1	2.2
MADMAX($T = 10, N = 500$)	0.883	0.886	0.902	0.865	3.5	4.5
MADMAX($T = 6, N = 600$)	0.85	0.854	0.872	0.83	1.0	1.5
MADMAX($T = 10, N = 600$)	0.885	0.885	0.9	0.867	3.3	4.6

TABLE 10: F1 score with respect to the number N of nodes and three datasets. UBD means the imbalanced dataset with 19,500 benign domains and 6,050 malicious domains, UMD means the imbalanced dataset with 6,050 benign domains and the 19,500 malicious domains, and BD means the balanced dataset with 24,126 benign domains and malicious domains.

N	UBD	UMD	BD
100	0.674	0.928	0.851
200	0.691	0.936	0.863
300	0.709	0.939	0.865
400	0.730	0.940	0.868
500	0.735	0.940	0.870
600	0.742	0.943	0.871
700	0.747	0.943	0.872
800	0.746	0.944	0.873
900	0.752	0.944	0.875
1000	0.753	0.945	0.874

two types, i.e., a domain-based approach focusing on text strings of domain names and a behavior-based approach focusing on other information in addition to the text strings. Roughly speaking, the domain-based approach utilizes only domain names as texts, and the accuracy can be improved by utilizing a complex deep neural network. On the other hand, the behavior-based approach can improve accuracy by increasing the kind of inputs. We describe the details of each approach below.

A. DOMAIN-BASED APPROACH

The major way on the domain-based approach is to leverage a complex and enriched model to detect malicious domains only by domain names. Woodbridge et al. [2] took the limitation of a deny list into account and utilized a long short-term memory (LSTM) for the malicious domain detection. Next, Bin et al. [31] utilized both convolution neural networks (CNN) and LSTM to leverage a large amount of actual traffic. Bin et al. [32] also compared the accuracy of five models described below: a single LSTM layer model by Woodbridge et al. [2], a combination model of forwarding LSTM layers and backward LSTM layers by Dhingra et al. [33], parallel CNN layers by Saxe et al. [34], stacked CNN layers by Zhang et al. [35], and a hybrid model of stacked CNN layers and a single LSTM layer by Vosoughi et al. [3]. Next, Berman et al. [4] showed the first work based on CapsNet and, as a result, a better performance than the conventional RNN and CNN. Yanchen et al. [5] introduced an attention mechanism that ignores parts of domain names under some designated conditions. Furthermore, Luhui et al. [6] proposed a method based on heterogeneous Deep neural networks which combine parallel CNN and multiple LSTMs.

The works described above have utilized complicated architectures and hence are unsuitable for use in a browser environment. In contrast, MADMAX focuses on introducing into a browser, and hence ELM is utilized.

B. BEHAVIOR-BASED APPROACH

The following four types of data are mainly used in most research of malicious domain detection [1].

- DNS information
- Certificate information
- Structures of web pages
- Auxiliary information

We briefly describe each type below.

a: DNS Information

First, we describe research utilizing DNS information to detect malicious domains. It is shown that hosts controlled by botnets often have similar query content and time-series patterns [36], [37]. In other words, malicious domains tend to have intercommunication between clients and DNS servers [36], [38]–[43]. In several works, such information is converted into a data form such as a matrix or graph. For instance, Grill et al. [44] proposed a knowledge-based malware detection algorithm for domain generation algorithms (DGAs). Concurrently, Chiba et al. [45] utilized random forests to infer domains based on the data registration date and its reason on the list of trusted sites or denylist. Likewise, machine learning models based on DNS traffic information have been studied well [39], [46]–[48]. Nevertheless, MADMAX is the first browser-based application by incorporating machine learning into a browser, to the best of our knowledge.

b: Certificate Information

There are several works that utilize certificate information for malicious domain detection [49], [50]. Torroledo et al. [49] focused on data such as issuer information whereby certificates are self-signed, and successfully detected malware with high accuracy. Meanwhile, Anderson and McGrew [51] built a classification model based on logistic regression by extracting features from TLS and HTTP communications that contain certificate information. Several works [49], [50] showed that machine learning models such as deep neural

networks (DNN) and support vector machines (SVM) could detect malware and phishing sites by leveraging only on certificate information. However, detection based on certificates cannot determine whether a site is malicious unless it supports HTTPS, and thus providing comprehensive detection is difficult.

c: Structures of Web Pages

Structures of web pages can be utilized to detect malicious sites. For example, Huang et al. [52] focused on texts, fonts, and colors of the page, and Mao et al. [53] utilized cascading style sheets (CSS), respectively. However, these attempts are ineffective against code obfuscation techniques [54], [55]. Meanwhile, there is research [56]–[60] using images of web pages displayed in a browser as a practical approach against the code obfuscation. CNN is useful for providing high accuracy because the above research is about image processing [61], [62]. Hence, Abdelnabi et al. [63] built a model for detecting phishing sites on CNN based on image information from benign and malicious sites. However, the computational cost of CNN is huge in general and thus is unsuitable for a browser-based application.

d: Auxiliary Information

Finally, external information publicly available is often utilized for malicious domain detection [51], [64], [65]. In particular, denylists such as URLhaus and lists of popular sites such as Tranco [25] are utilized similarly to our work. Furthermore, geographic information obtained from IP addresses using the MaxMind Database[12] [66]–[70] can also be utilized. The potential performance of MADMAX may be increased by incorporating geographic information into its features.

VIII. CONCLUSION

This paper presented MADMAX, a browser-based application for malicious domain detection by leveraging extreme learning machine (ELM) [9]. The key insights for MADMAX were the selection of optimized features and the real-time training, and MADMAX discussed these functions as an application-level implementation for the first time, as far as we know. We released the implementation of MADMAX via GitHub as well.

In the selection of optimized features, we showed that the accuracy of malicious domain detection could be improved by selecting important features rather than the use of all 25 features. Notably, throughput for detection of malicious domains and benign domains was different due to extracting DNS records. We also demonstrated that MADMAX outperformed the ELM-based existing work [8] by virtue of the selection of optimized features. On the other hand, in the real-time training, we demonstrated that the retrained model could continuously detect unseen malicious domains while the accuracy of the normal model decreases because of missing a concept drift of malicious domains.

We also found a new problem for malicious domain detection through the design of MADMAX via the experiments, i.e., the permutation importance, which are unavailable features for several domains. Research into investigating the influence of the permutation importance on domain detection and its improvement is already underway. Likewise, we adopted the classic ELM to realize the real-time training in this paper, but the current specification of MADMAX cannot learn the dependencies between past and future domains. Further studies, which take the online sequential ELM [18] into account, will need to be undertaken to continuously learn the stream information of malicious domains as time-series data.

CODE AVAILABILITY

Our implementation of MADMAX is publicly available via GitHub (https://github.com/kzk-IS/MADMAX) for reproducibility and as reference for future works.

REFERENCES

[1] T. Yu, Y. Zhauniarovich, I. Khalil, and M. Dacier, "A survey on malicious domains detection through dns data analysis," *ACM Computing Surveys*, vol. 51, no. 4, 2018.

[2] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," *arXiv preprint arXiv:1611.00791*, 2016.

[3] S. Vosoughi, P. Vijayaraghavan, and D. Roy, "Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder," in *Proc. of SIGIR 2016*. ACM, 2016, pp. 1041–1044.

[4] D. S. Berman, "Dga capsnet: 1d application of capsule networks to dga detection," *Information*, vol. 10, no. 5, p. 157, 2019.

[5] Y. Qiao, B. Zhang, W. Zhang, A. K. Sangaiah, and H. Wu, "Dga domain name classification method based on long short-term memory with attention mechanism," *Applied Sciences*, vol. 9, no. 20, p. 4205, 2019.

[6] L. Yang, G. Liu, Y. Dai, J. Wang, and J. Zhai, "Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework," *IEEE Access*, vol. 8, pp. 82 876–82 889, 2020.

[7] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "Tesseract: Eliminating experimental bias in malware classification across space and time," in *Proc. of USENIX Security 2019*. USENIX Association, 2019, pp. 729–746.

[8] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," *Neural Processing Letters*, vol. 48, no. 3, pp. 1347–1357, 2018.

[9] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[10] W. Cao, X. Wang, Z. Ming, and J. Gao, "A review on neural networks with random weights," *Neurocomputing*, vol. 275, pp. 278–287, 2018.

[11] C.-J. Chien, N. Yanai, and S. Okamura, "Design of malicious domain detection dataset for network security," 2021. [Online]. Available: http://www-infosec.ist.osaka-u.ac.jp/~yanai/dataset.pdf

[12] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[13] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[14] J. Zhang, Y. Li, W. Xiao, and Z. Zhang, "Non-iterative and fast deep learning: Multilayer extreme learning machines," *Journal of the Franklin Institute*, vol. 357, no. 13, pp. 8925–8955, 2020.

[15] W. Xiao, J. Zhang, Y. Li, S. Zhang, and W. Yang, "Class-specific cost regulation extreme learning machine for imbalanced classification," *Neurocomputing*, vol. 261, pp. 70–82, 2017.

[16] J. Zhang, W. Xiao, Y. Li, and S. Zhang, "Residual compensation extreme learning machine for regression," *Neurocomputing*, vol. 311, pp. 126–136, 2018.

[12]https://www.maxmind.com/

[17] J. Zhang, Y. Li, W. Xiao, and Z. Zhang, "Robust extreme learning machine for modeling with unknown noise," *Journal of the Franklin Institute*, vol. 357, no. 14, pp. 9885–9908, 2020.

[18] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on neural networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[19] J. Zhao, Z. Wang, and D. S. Park, "Online sequential extreme learning machine with forgetting mechanism," *Neurocomputing*, vol. 87, pp. 79–89, 2012.

[20] H. Zhang, S. Zhang, and Y. Yin, "Online sequential elm algorithm with forgetting factor for real applications," *Neurocomputing*, vol. 261, pp. 144–152, 2017.

[21] T. Matias, F. Souza, R. Araújo, N. Gonçalves, and J. P. Barreto, "Online sequential extreme learning machine based on recursive partial least squares," *Journal of Process Control*, vol. 27, pp. 15–21, 2015.

[22] L. Breiman, "Random forests," Machine learning, vol. 45, pp. 5–32, 2001.

[23] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

[24] H. Zhao, Z. Chang, G. Bao, and X. Zeng, "Malicious domain names detection algorithm based on N-gram," *Journal of Computer Networks and Communications*, vol. 2019, pp. 1–9, 2019.

[25] V. L. Pochat, T. van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proc. of NDSS 2019*. Internet Society, 2019. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/tranco-a-research-oriented-top-sites-ranking-hardened-against-manipulation/

[26] W. Cao, J. Gao, Z. Ming, and S. Cai, "Some tricks in parameter selection for extreme learning machine," *IOP Conference Series: Materials Science and Engineering*, vol. 261, p. 012002, 2017.

[27] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proc. of NIPS 2017*, vol. 30. Curran Associates, Inc., 2017, pp. 3146–3154.

[28] A. K. Sood and S. Zeadally, "A taxonomy of domain-generation algorithms," *IEEE Security & Privacy*, vol. 14, no. 4, pp. 46–53, 2016.

[29] W. Cao, J. Gao, Z. Ming, S. Cai, and H. Zheng, "Impact of probability distribution selection on rvfl performance," in Proc. of SmartCom 2018, ser. LNCS, vol. 10699. Springer, 2018, pp. 114–124.

[30] W. Cao, M. J. A. Patwary, P. Yang, X. Wang, and Z. Ming, "An initial study on the relationship between meta features of dataset and the initialization of nnrw," in *Proc. of IJCNN 2019*. IEEE, 2019, pp. 1–8.

[31] B. Yu, D. L. Gray, J. Pan, M. De Cock, and A. C. Nascimento, "Inline dga detection with deep networks," in *Proc. of ICDMW 2017*. IEEE, 2017, pp. 683–692.

[32] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of dga domain names," in *Proc. of IJCNN 2018*. IEEE, 2018, pp. 1–8.

[33] B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen, "Tweet2vec: Character-based distributed representations for social media," *arXiv preprint arXiv:1605.03481*, 2016.

[34] J. Saxe and K. Berlin, "expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys," *arXiv preprint arXiv:1702.08568*, 2017.

[35] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *arXiv preprint arXiv:1509.01626*, 2015.

[36] H. Choi and H. Lee, "Identifying botnets by capturing group activities in DNS traffic," *Computer Networks*, vol. 56, no. 1, pp. 20–33, 2012.

[37] H. Choi, H. Lee, and H. Kim, "Botgad: detecting botnets by capturing group activities in network traffic," in *Proc. of COMSWARE 2009*. ACM, 2009, pp. 1–8.

[38] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in dns traffic," in *Proc. of ICCIT 2007*. IEEE, 2007, pp. 715–720.

[39] P. K. Manadhata, S. Yadav, P. Rao, and W. Horne, "Detecting malicious domains via graph inference," in *Proc. of ESORICS 2014*. Springer, 2014, pp. 1–18.

[40] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *Proc. of DSN 2015*. IEEE, 2015, pp. 45–56.

[41] I. Prieto, E. Magaña, D. Morató, and M. Izal, "Botnet detection based on DNS records and active probing," in *Proc. of SECRYPT 2011*. IEEE, 2011, pp. 307–316.

[42] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Segugio: Efficient behavior-based tracking of malware-control domains in large isp networks," in *Proc. of DSN 2015*. IEEE, 2015, pp. 403–414.

[43] ——, "Efficient and accurate behavior-based tracking of malware-control domains in large isp networks," *ACM Transactions on Privacy and Security*, vol. 19, no. 2, pp. 1–31, 2016.

[44] M. Grill, I. Nikolaev, V. Valeros, and M. Rehak, "Detecting DGA malware using NetFlow," in *Proc. of IM 2015*. IEEE, 2015, pp. 1304–1309.

[45] D. Chiba, T. Yagi, M. Akiyama, T. Shibahara, T. Yada, T. Mori, and S. Goto, "Domainprofiler: Discovering domain names abused in future," in *Proc. of DSN 2016*. IEEE, 2016, pp. 491–502.

[46] X. Sun, J. Yang, Z. Wang, and H. Liu, "Hgdom: Heterogeneous graph convolutional networks for malicious domain detection," in *Proc. of NOMS 2020*. IEEE, 2020, pp. 1–9.

[47] I. Khalil, T. Yu, and B. Guan, "Discovering malicious domains through passive dns data graph analysis," in *Proc. of ASIACCS 2016*. ACM, 2016, pp. 663–674.

[48] X. Sun, M. Tong, J. Yang, L. Xinran, and L. Heng, "Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification," in *Proc. of RAID 2019*. USENIX Association, 2019, pp. 399–412.

[49] I. Torroledo, L. D. Camacho, and A. C. Bahnsen, "Hunting malicious TLS certificates with deep neural networks," in *Proc. of AISec 2018*. ACM, 2018, pp. 64–73.

[50] D. Herrald and R. Kovar, "The "hidden empires" of malware." 2018, https://www.slideshare.net/RyanKovar/the-hidden-empires-of-malware-with-tls-certified-hypotheses-and-machine-learning.

[51] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proc. of AISec 2016*. ACM, 2016, pp. 35–46.

[52] C.-Y. Huang, S.-P. Ma, W.-L. Yeh, C.-Y. Lin, and C.-T. Liu, "Mitigate web phishing using site signatures," in *Proc. of TENCON 2010*. IEEE, 2010, pp. 803–808.

[53] J. Mao, W. Tian, P. Li, T. Wei, and Z. Liang, "Phishing-alarm: Robust and efficient phishing detection via page component similarity," *IEEE Access*, vol. 5, pp. 17 020–17 030, 2017.

[54] A. Y. Fu, L. Wenyin, and X. Deng, "Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD)," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 301–311, 2006.

[55] I.-F. Lam, W.-C. Xiao, S.-C. Wang, and K.-T. Chen, "Counteracting phishing page polymorphism: An image layout analysis approach," in *Proc. of ISA 2009*, ser. Lecture Notes in Computer Science, vol. 5576. Springer, 2009, pp. 270–279.

[56] R. S. Rao and S. T. Ali, "A computer vision technique to detect phishing attacks," in *Proc. of CSNT 2015*. IEEE, 2015, pp. 596–601.

[57] K.-T. Chen, J.-Y. Chen, C.-R. Huang, and C.-S. Chen, "Fighting phishing with discriminative keypoint features," *Internet Computing*, vol. 13, no. 3, pp. 56–63, 2009.

[58] A. S. Bozkir and E. A. Sezer, "Use of hog descriptors in phishing detection," in *Proc. of ISDFS 2016*. IEEE, 2016, pp. 148–153.

[59] L. Malisa, K. Kostiainen, and S. Capkun, "Detecting mobile application spoofing attacks by leveraging user visual similarity perception," in *Proc. of CODASPY 2017*. ACM, 2017, pp. 289–300.

[60] S. Afroz and R. Greenstadt, "Phishzoo: Detecting phishing websites by looking at them," in *Proc. of ICSC 2011*. IEEE, 2011.

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. of NIPS 2012*, vol. 1. Curran Associates Inc., 2012, pp. 1097–1105.

[62] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *Proc. of CVF 2014*. IEEE, 2014, pp. 512–519.

[63] S. Abdelnabi, K. Krombholz, and M. Fritz, "Visualphishnet: Zero-day phishing website detection by visual similarity," in *Proc. of CCS 2020*. ACM, 2020, pp. 1681–1698.

[64] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for dns," in *Proc. of USENIX Security 2010*. USENIX Association, 2010, pp. 273–290.

[65] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting malware domains at the upper dns hierarchy," in *Proc. of USENIX Security 2011*. USENIX Association, 2011, pp. 1–16.

[66] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "EXPOSURE: Finding malicious domains using passive DNS analysis," in *Proc. of NDSS 2011*. The Internet Society,

2011. [Online]. Available: https://www.ndss-symposium.org/ndss2011/exposure-finding-malicious-domains-using-passive-dns-analysis/

[67] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive DNS analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security*, vol. 16, no. 4, pp. 1–28, 2014.

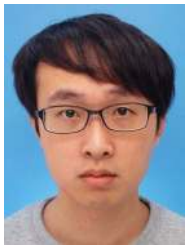
[68] I. Mishsky, N. Gal-Oz, and E. Gudes, "A topology based flow model for computing domain reputation," in *Proc. of DBSec 2015*, ser. Lecture Notes in Computer Science, vol. 9149. Springer, 2015, pp. 277–292.

[69] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "Fluxor: Detecting and monitoring fast-flux service networks," in *Proc. of DIMVA 2006*, ser. Lecture Notes in Computer Science, vol. 5137. Springer, 2008, pp. 186–206.

[70] M. Stevanovic, J. M. Pedersen, A. D'Alconzo, S. Ruehrup, and A. Berger, "On the ground truth problem of malicious dns traffic analysis," *Computers & Security*, vol. 55, pp. 142–158, 2015.

KAZUKI IWAHANA received B.Eng degree in Engineering Science from Osaka University, Japan, in 2020. He has recently joined M.S. course in Graduate School of Information Science and Technology in Osaka University, Japan. His research interests include machine learning security.

TATSUYA TAKEMURA received the B. Eng degree in Engineering Science from Osaka University, Japan, in 2019. He has recently joined the M.S. course in the Graduate School of Information Science and Technology in Osaka University, Japan. His research interests include machine learning and network security.



JU CHIEN CHENG received a B.S. in Computer Science and Information Engineering from National Taiwan University, Taiwan, in 2017. He has recently joined the M.S. course in the Graduate School of Information Science and Technology in Osaka University, Japan. His research interests include network security and machine learning.

NAMI ASHIZAWA has joined M.S. course in Graduate School of Information Science and Technology in Osaka University, Japan, in 2019. Her research interests include information security and blockchains.

NAOKI UMEDA received the B. Eng degree in Engineering Science from Osaka University, Japan, in 2020. He has recently joined the M.S. course in the Graduate School of Information Science and Technology in Osaka University, Japan. His research interests include network security and machine learning.

KODAI SATO received the B. Eng degree in Engineering Science from Osaka University, Japan, in 2019. He has recently joined the M.S. course in the Graduate School of Information Science and Technology in Osaka University, Japan. His research interests include cryptography.

RYOTA KAWAKAMI received B.Eng degree in Engineering Science from Osaka University, Japan, in 2020. He has recently joined M.S. course in Graduate School of Information Science and Technology in Osaka University, Japan. His research interests include information security and social engineering.

REI SHIMIZU has recently joined I.S. course in Engineering Science from Osaka University, Japan. His research interests include blockchains and network security.

YUICHIRO CHINEN received M.S. degree in Graduate School of Information Science and Technology in Osaka University, Japan, in 2020. His research interests include blockchains.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3080456, IEEE Access

NAOTO YANAI received the B.Eng. degree from The National Institution of Academic Degrees and University Evaluation, Japan, in 2009, the M.S.Eng. from the Graduate School of Systems and Information Engineering, the University of Tsukuba, Japan, in 2011, and the Dr.E. degree from the Graduate School of Systems and Information Engineering, the University of Tsukuba, Japan, in 2014. He is an assistant professor at Osaka University, Japan. His research area is information security.

.