MADONNA: Browser-Based MAlicious Domain Detection through Optimized Neural Network with Feature Analysis

Janaka Senanayake[1][0000−0003−2278−8671], Sampath Rajapaksha[1][0000−0001−7772−3774], Naoto Yanai[2][0000−0002−0817−6188], Chika Komiya[2][0009−0001−2253−5407], and Harsha Kalutarage[1][0000−0001−6430−9558]

[1] School of Computing, Robert Gordon University, Aberdeen, UK
{j.senanayake,s.rajapaksha,h.kalutarage}@rgu.ac.uk
[2] Department of Information Security Engineering, Osaka University, Japan
{yanai,c-komiya}@ist.osaka-u.ac.jp

Abstract. The detection of malicious domains often relies on machine learning (ML), and proposals for browser-based detection of malicious domains with high throughput have been put forward in recent years. However, existing methods suffer from limited accuracy. In this paper, we present MADONNA, a novel browser-based detector for malicious domains that surpasses the current state-of-the-art in both accuracy and throughput. Our technical contributions include optimized feature selection through correlation analysis, and the incorporation of various model optimization techniques like pruning and quantization, to enhance MADONNA's throughput while maintaining accuracy. We conducted extensive experiments and found that our optimized architecture, the Shallow Neural Network (SNN), achieved higher accuracy than standard architectures. Furthermore, we developed and evaluated MADONNA's Google Chrome extension, which outperformed existing methods in terms of accuracy and F1-score by six points (achieving 0.94) and four points (achieving 0.92), respectively, while maintaining a higher throughput improvement of 0.87 seconds. Our evaluation demonstrates that MADONNA is capable of precisely detecting malicious domains, even in real-world deployments.

Keywords: malicious domain detection · machine learning · feature engineering · browser extension.

1 Introduction

The incidence of cybercrime has significantly increased in recent years, with the use of rogue domains being a common tactic employed by adversaries for various purposes such as running command and control (C&C) servers or setting up phishing websites. Shockingly, the creation of about 40,000 malicious domains per day has been reported, and this leads to an average loss of $17,700 per minute [15]. To circumvent blacklist blocking, attackers often use short-lived malicious domains generated by domain generation algorithms (DGAs). Therefore,

the use of machine learning (ML) for detecting malicious domains has received significant attention in recent years [29].

A browser is the closest interface for a user, and hence users can be alerted to ongoing malicious attempts, such as phishing, via the browser in real-time. However, the inference throughput and accuracy of the existing work need to be increased further to build an efficient malicious domain detection model. In this paper, we aim to design an artificial intelligence-based domain detection application for web browsers with higher accuracy and lower computation overhead (throughput). We note that such a design is *non-trivial*.

Indeed, the reason for the low accuracy of previous work is caused by the use of simple neural network models or traditional ML models to improve the throughput. If an enriched ML model is trivially introduced in the application instead of a simple neural network, the throughput will be downgraded drastically [9]. It might be unsuitable for a browser-based deployment due to the high throughput. Hence, we need to address a trade-off problem between the throughput and accuracy to develop a practically deployable browser-based malicious domain detection application.

Our approach to addressing the aforementioned problems involves two standpoints. Firstly, we identify the most relevant features for detecting malicious domains. In general, choosing such features is a technical matter [23], and one potential approach is to analyze feature correlations [2, 10]. Through an in-depth examination of feature correlations, we can eliminate redundant features, thereby improving throughput without compromising accuracy. Secondly, we employ model optimization techniques, such as pruning and parameter quantization, in deep learning models. By implementing these techniques and eliminating unnecessary neurons in the models, we can significantly improve throughput while maintaining accuracy. Based on the above viewpoints, we propose *MADONNA (MAlicious Domain detection through Optimized Neural Network with feature Analysis)*. We demonstrate that MADONNA outperforms other state-of-the-art models in terms of both accuracy and throughput by virtue of the analysis of feature correlations and neural network-based learning techniques while achieving 94% accuracy.

To sum up, we make the following contributions:

- We present MADONNA, an open-source browser-based extension (plug-in) that runs AI in the backbend to detect malicious domains in near real-time.
- We analyze feature correlations for malicious domain detection by removing highly correlated features to improve both throughput and accuracy.
- We show that parameter quantization and pruning in a deep learning model can strikingly improve throughput by keeping the same-level accuracy for malicious domain detection.
- We conduct a real-world experiment to distinguish benign and malicious domains in the real world and show that MADONNA can detect these domains precisely.
- We demonstrate that MADONNA outperforms the benchmarked models with respect to the accuracy and throughput of malicious domain detection.

The rest of the paper is organized as follows: Section 2 contains preliminaries. Section 3 explains the methodology, and Section 4 discusses the results. Finally, the conclusions and future work directions are discussed in Section 5.

2 Preliminaries

This section provides background knowledge on domain names and malicious domain detection using machine learning (ML) to aid in the understanding of our work.

2.1 Domain Names

Domain names are texts correlated to network hosts and are operated via the Domain Name System (DNS). Generally, domain names are hierarchically managed under namespaces called a zone, and the highest domain is called root. The most popular domains are `.com`, `.org`, `.us`, `.uk`, and `.jp`, and such domains are called top-level domains (TLDs). There are multiple domains under each TLD, which are managed hierarchically and distributively through their zones. In a normal URL, protocol, subdomains (optional), domain name, TLD, and subdirectories can be linked as shown in this example: https://ifipsec2023.psnc.pl/program/.

2.2 Malicious Domain Detection

There are two main approaches to detect malicious domains, namely knowledge-based and machine learning-based methods [29]. The former approach is based on expertise and heuristics to distinguish benign and malicious domains. However, these methods often fail to detect novel attacks, leading to zero-day exploits. In contrast, the latter approach can effectively infer unknown domains as benign or malicious [12]. Especially, supervised learning is a common setting because of the efficiency and the selection of the most relevant features from raw data [13, 14, 16, 17, 19, 25]. It can infer new domains after the training with labeled domains with high accuracy. Hence, we focus on malicious domain detection based on ML.

Malicious domain detection based on ML provides inferences to determine whether the given domains are malicious. Informally, an ML 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 in the inference phase and then infers its label as benign or malicious. A typical approach for domain detection in recent years is based on deep neural networks.

Problem Formulation: We formalize the problem of domain detection based on ML below. 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 $l \in \mathbf{N}$ 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 each label denotes a benign domain by 0 and a malicious domain by 1. For the size n 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, and labels. Let $Model = M(DFL)$ denote a trained model, where M denotes a learning algorithm. If d_t is a test domain (test case) unseen by M during its training time, our goal is then to obtain an inference result, $L_t = Model(F_t)$, by extracting features $F_t = \{f_{t,1}, \cdots, f_{t,l}\}$ for the unlearned domain d_t.

2.3 Related Works

We describe related works in three aspects: feature selection, feature engineering, and browser-based applications.

For the feature selection, the major way of malicious domain detection is to utilize an enriched model only with domain names [4, 27, 28]. While domain names are dealt with text data, malicious domain detection often needs more information, such as DNS information [21, 22] and web contents [1, 3]. We follow features in [9], which include domain names, DNS information, and web content.

For feature engineering, a typical way to improve accuracy is to evaluate the feature importance. To this end, features can be analyzed by principal component analysis [31] or decision trees [26]. Redundant features can also be removed by computing zero scores [18] or the equality of data points [30]. We adopt feature correlations [10] to remove redundant features for the design of MADONNA because training prediction models with too many correlated features reduces their accuracy and increases the model's computational overhead.

For browser-based applications, a web browser for detecting a phishing site [6] was developed in recent years. However, it is not a common browser such as Firefox or Google Chrome. As plug-ins for existing browsers, several works [3] have developed phishing site detection, and there are several products according to the recent survey [23]. They utilize whitelists and blacklists of domains as a part of detection. We discuss more general malicious domains, including phishing sites, only with ML as browser-based applications. The closest work to ours is MADMAX [9], which is a browser-based application for detecting malicious domains and includes feature selection. Another important related work in [2] provided four feature importance evaluations. Our goal is to design a high-performing browser-based extension that outperforms the benchmarked and state-of-the-art model. Therefore, we introduce MADONNA, which will be compared to MADMAX in the benchmarking process.

Various libraries [11, 20] exist for implementing deep learning in web browsers, which have the potential to deliver performance equivalent to JavaScript. Additionally, MADONNA functions as a distributed platform [7]. These libraries can be used to build a browser-based system for detecting malicious domains. While these libraries focus on creating generalized ML platforms, MADONNA is designed specifically for detecting malicious domains.

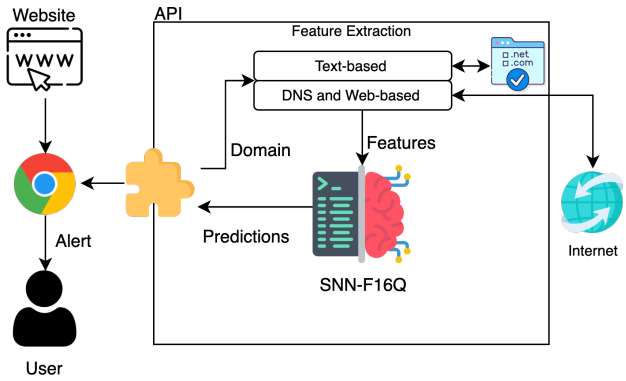


Fig. 1: The Overview of MADONNA

3 Methodology

This section outlines the specific methodology used for detecting malicious domains, which is divided into three subsections: Feature Extraction, Model Training and Optimization, and Browser-based Deployment. Figure 1 provides an overview of the MADONNA system and explains each of the steps involved. When a user clicks on the MADONNA extension to check the malicious status of a domain, the system's Application Programming Interface (API) is called. The API extracts the required features and uses the trained SNN model to generate prediction results, which are then displayed to the user through the extension.

3.1 Feature Extraction

This work utilized the dataset introduced in [5, 9]. This dataset consists of 25 features, including text-based features, DNS-based features, and web-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. 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. Web-based features represent information obtained from contents on domains and discuss characteristics of the contents provided by malicious domains.

Throughput is one critical criterion of the proposed model, as real-time or near-real-time malicious domain detection is highly important. Therefore, a minimum number of features should be selected whilst achieving the highest level of detection rate. Figure 2 shows the feature correlation metrics for the 25 features.

Label feature includes the ground truth 1 and 0 for malicious and benign domains, respectively. Based on this, features n_ns and ns_similarity have the

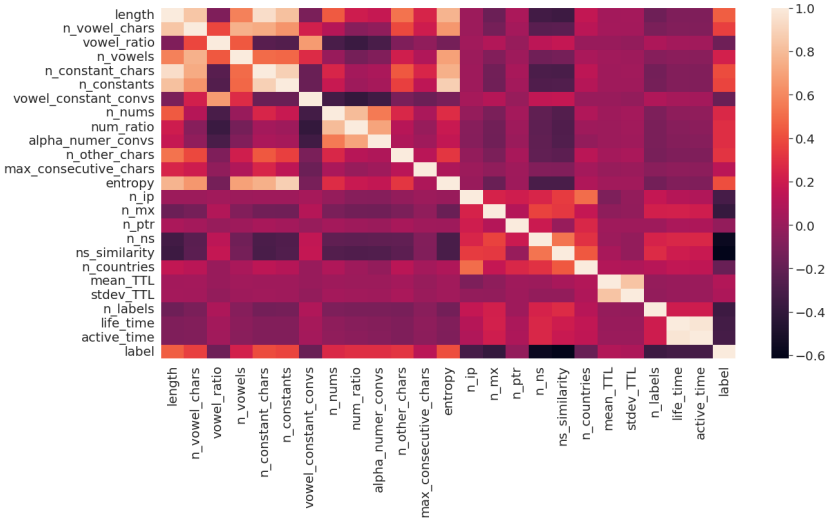


Fig. 2: Feature Correlations

highest Pearson correlations with the label, whereas mean_TTL and stdev_TTL have the lowest Pearson correlations. Some features have higher correlations with some other features. For example, active_time and life_time have a correlation of 0.97. Therefore, we can remove one of these highly correlated features from the ML model.

In their study [9] the authors employed the permutation importance algorithm to choose seven features. The backward selection was utilized to take into account feature correlation and distribution. The chosen features include: length, n_ns, n_vowels, n_vowel_chars, life_time, n_constant_chars, n_nums, ns_similarity, n_other_chars, entropy, n_countries, n_mx, and n_labels. Table 1 provides a summary of the feature description and their behavior in both benign and malicious scenarios based on our analysis.

Feature distributions for a sample of features are depicted in Figure 3. X-axis represents malicious and benign class labels, while the y-axis represents value ranges for the respective variable. None of the features create 100% class separability. For example, the highest correlated variable ns_similarity, which is shown in Figure 3c, has shared values between 0.71 and 1.00 for both benign and malicious domains. Further, this clearly shows that outliers are available in the benign dataset. Therefore, these outliers are removed using Z-score to achieve higher generalization capability. Z-score greater than +3 or less than -3 is considered as the threshold to identify the outliers.

3.2 Model Training and Optimization

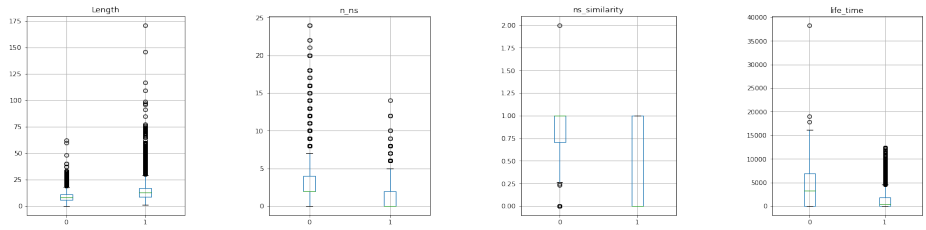
A supervised learning model was trained by utilizing the dataset proposed in [5] and selecting the optimized features only. To evaluate the performance of the

Table 1: Selected Features

Feature Name	Description
length	The length of the domain. The average length of malicious domains is about two times that of benign domains.
n_ns	The number of distinct name servers. n_ns values tend to be low for malicious domains.
n_vowels	The number of vowels in the domain. These values tend to be high for malicious domains.
life_time	The difference of expiration date and creation date of WHOIS data, in days. Generally, life_time is low for malicious domains.
n_vowel_chars	The number of vowel characters in the domain. n_vowel_chars has similar characteristics as n_vowels.
n_constant_chars	The number of constant characters in the domain. Malicious domains include more constant characters.
n_nums	The number of numeric characters in the domain. This is typically high in malicious domains.
n_other_chars	Number of characters other than digits and alphabets in the domain. This is comparatively high in malicious domains.
entropy	The entropy of the domain. High values can be observed for malicious domains.
ns_similarity	The similarity between name servers. This is significantly low for malicious domains.
n_countries	The number of countries obtained from GeoLite2 service queried using each of the distinct IP addresses. This tends to be greater than 1 for malicious domains.
n_mx	The number of distinct mail exchange records. Low values can be observed for malicious domains.
n_labels	The number of HTML elements of the content. This is significantly low in malicious domains.

models, a variety of experiments were conducted, which involved training them with different machine learning algorithms, such as Logistic Regression (LR) and Random Forest (RF), boosting algorithms like Gradient Boosting (GB), eXtreme Gradient Boosting (XGB), Light Gradient Boosting (LGB), and Extreme Learning Machine (ELM). In addition, Multilayer Perceptron (MLP) and SNN were trained and assessed the performance in terms of the accuracy, F1-Score, and throughput of the models, and it was found that the SNN delivered a good performance. Therefore, the optimal artificial neural network model was selected for further experiments. A simple model architecture was selected considering the detection latency of the model. To this end, grid search was used to select the optimum hyperparameters of the SNN model.

Pruning and Quantization The throughput of a Neural Network can be improved by eliminating the least significant weight parameters. This aims to keep the model's accuracy while improving its efficiency. Magnitude-based pruning [24] is a simple but effective approach that eliminates the weights whilst keeping the same level of accuracy. Magnitude-based pruning gradually removes the insignificant weights by assigning value zeros during the model training process. Model accuracy depends on the level of sparsity and therefore, sparsity level should be selected carefully to achieve the same level of accuracy. The TensorFlow model optimization toolkit was used to apply the magnitude-based



(a) length (b) n_ns (c) ns_similarity (d) life_time

Fig. 3: Feature Selection

model pruning. First, the model was trained with all parameters and then applied pruning to achieve 50% of parameter sparsity starting from 0% sparsity. The pruned model is referred to as SNN-P in this work.

Quantization [8] is another optimization technique that reduces the precision of the numbers used for model parameters. Typically, Tensorflow uses 32-bit floating point numbers. Quantization leads to achieving a better throughput by moving 32-bit numbers into 16 or 8-bit numbers. However, this might reduce the model accuracy slightly due to the loss of precision. The TensorFlow optimization toolkit provides different quantization options. Accordingly, we used non-optimized quantization (SNN-NOQ), dynamic range quantization (SNN-DRQ), float16 quantization (SNN-F16Q), and int8 quantization (SNN-I8Q). The TensorFlow quantization also converts the model into a more lightweight TFLite version. Therefore, SNN-NOQ, SNN-DRQ, SNN-F16Q, and SNN-I8Q models are TFLite versions.

3.3 Browser Deployment

The browser extension named MADONNA, which was produced in this work, can be used to detect malicious domains near real-time when visiting websites. Due to the popularity of web browsers, Google Chrome was selected as the target browser to develop the extension. The extension can be downloaded from GitHub[3] and can be easily installed in Google Chrome.

The MADONNA extension has been connected with a Python Flask web API. The API can be started by executing the start_api.bat file (for Windows) or start_api.sh file (for Linux). To execute the API, Python 3 runtime should be available. Once the API is running in the backend, when a user uses the browser extension, it sends the URL user attempts to visit, to the API, and the API extracts the required text-based, DNS-based, and web-based features of the given URL. These features are passed to the trained SNN model. Then the model predicts whether the domain is malicious or benign. Based on the results user will be prompted by the browser extension whether the URL is safe to visit.

[3] https://github.com/softwaresec-labs/MADONNA

4 Results and Discussion

This section presents an evaluation of the performance of MADONNA using an existing dataset [5]. The analysis includes several key metrics, such as accuracy, model size, and throughput. Additionally, we benchmark the MADONNA extension in Google Chrome against MADMAX [9] to evaluate its performance.

4.1 Experimental Setting

The dataset introduced in [5], consisting of 48,252 domains (24,126 benign and 24,126 malicious), was utilized to train the model using the 13 identified important features. The SNN model was trained with 28 nodes in the hidden layer for 50 epochs and a batch size of 128, using the Adam optimizer with a learning rate of 0.001. To prevent overfitting, early stopping was applied. ReLU was used as the activation function for the hidden layer, while softmax was used for the classification layer. The model architecture was simple, with only 533 trainable parameters, which helped to achieve low latency. TensorFlow and Keras libraries were used to implement the SNN model, and Google Colab standard environment with 12GB of RAM was utilized for model training and inference.

4.2 Accuracy and Throughput of the Model

To evaluate the accuracy and F1-score of MADONNA's SNN model, traditional machine learning algorithms and boosting algorithms were trained using the same dataset and set of features. 5-fold cross-validation was performed for all models. The SNN model was further optimized for throughput by implementing pruning and quantization techniques. As the initialization time of the API is dependent on the model size, it is crucial to have a smaller model size to achieve faster performance. This is particularly important for detecting malicious domains since web users prefer fast browsing without additional delays. Therefore, it is necessary to ensure that domain analysis is performed within a reasonable timeframe to predict whether a domain is malicious or not. The accuracy, precision, recall and F1-Score, model size, and inference time of these models are compared in Table 2.

According to Table 2, it was identified that when applying boosting algorithms (XGB, GB, LGB), higher accuracies and F1-Scores can be obtained compared with the other ML algorithms (LR, RF, ELM, and MLP). However, the SNN model outperformed all the traditional machine learning models with 94% accuracy and 0.92 F1-Score. It was also identified that the optimized SNN variants achieve the same level of performance except for a slight performance drop in the SNN-I8Q model.

The best-performed ML-based model was XGB. It required the largest model size, whereas the ELM model showed the longest inference time. This is because, even if ELM requires a small training time, it still requires a large number of model parameters to learn the benign and malicious domain patterns. Therefore, it takes much time for the inference. On the other hand, due to the simple

Table 2: Comparison of Accuracy, Precision, Recall, F1-Score, Throughput

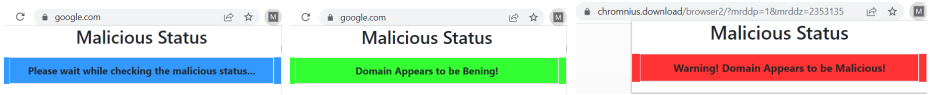
Model	Accuracy	Precision	Recall	F1-Score	Model size(KB)	Inference time(μs)
LR	87%	0.87	0.86	0.87	443	151
RF	88%	0.92	0.83	0.87	480	41
GB	89%	0.91	0.89	0.89	422	112
MLP	83%	0.88	0.78	0.82	78	69
LGB	89%	0.91	0.89	0.89	482	98
XGB	90%	0.92	0.89	0.90	526	27
ELM	87%	0.88	0.86	0.87	312	198
SNN	94%	0.96	0.89	0.92	33	64
SNN-P	94%	0.96	0.90	0.92	19	36
SNN-NOQ	94%	0.96	0.90	0.92	4	19
SNN-DRQ	94%	0.96	0.90	0.92	4	16
SNN-F16Q	94%	0.96	0.90	0.92	3	10
SNN-I8Q	93%	0.95	0.89	0.91	3	12

model architecture of the SNN model, it only takes 33KB of memory and 64μs inference time. Pruned model (SNN-P) optimized these values due to the removal of insignificant weights. As expected, quantized models further optimized both model size and inference time. However, despite low precision, SNN-I8Q took small additional time compared to the SNN-F16Q model. This is likely because quantized int requires an arm device such as Raspberry Pi to get the optimum inference.

By considering all the evaluation matrices, including the accuracy, precision, recall, F1-Score, model size, and inference time, the SNN-F16Q model was selected to integrate with the browser extension to detect malicious domains.

4.3 Performance of Browser Extension

Users receive notifications reflecting predictions of the malicious status of a domain upon clicking on the MADONNA extension icon in Google Chrome. Examples of these notifications are shown in Figure 4. The extension displays a notification similar to Figure 4b if a domain is benign (e.g., https://www.google.com). On the other hand, if the checked URL contains a malicious domain (e.g., https://chromnius. download/browser2/?mrddp=1 &mrddz=2353135), a notification resembling Figure 4c appears. Figure 4a illustrates the notification displayed when checking a domain's malicious status.



(a) Checking Domain (b) Benign Domain (c) Malicious Domain

Fig. 4: Chrome Browser Extension Notifications

Table 3: Comparison of MADMAX and MADONNA

Aspect	**MADMAX**	**MADONNA**
Used Text-based features	length, n_constant_chars, n_vowel_chars, num_ratio	length, n_vowels, n_vowel_chars, n_constant_chars, n_nums, entropy, n_other_chars
Used DNS-based Features	n_ns	n_ns, ns_similarity, n_mx, n_countries
Used Web-based Features	life_time, n_labels	life_time, n_labels
Model Inference Time	198μs	10μs
Supported Browser	Firefox	Chrome
Avg. Prediction time in Browser	3.3s	2.43s
Accuracy	88%	94%
F1-Score	0.88	0.92
Precision	0.90	0.96
Recall	0.86	0.90
Connectivity	Externally-hosted Sever	Internally-hosted API

4.4 Comparison with Existing Works

A comparison between MADONNA and MADMAX was conducted as MADMAX is the closest high-accuracy AI-based malicious domain detection work. The comparison results are presented in Table 3, indicating that MADONNA includes more text-based and DNS-based features than MADMAX, which were selected based on their significance in feature analysis. By combining these features with an optimized and quantized SNN model, MADONNA achieved better accuracy, F1-Score, precision, and recall than MADMAX. Specifically, MADONNA outperformed MADMAX by 6% in accuracy and 4% in F1-Score, representing a significant improvement.

MADONNA supports the widely used Google Chrome browser and its backend model achieves significantly faster inference times (10μs compared to MADMAX's 198μs) which should be considered as notable advantages. Additionally, the MADONNA browser extension predicts the malicious status of domains more quickly, with an average prediction time of 2.43s compared to MADMAX's 3.3s. MADONNA's extension connects to an internally hosted web API, prioritizing user privacy, while MADMAX's extension relies on an external server for predictions. In summary, MADONNA outperforms MADMAX in all aspects.

We also conducted real-world experiments to evaluate the performance of MADONNA, validating the performance of the MADONNA Chrome extension by visiting well-known benign sites and malicious sites listed in CyberCrime, PhishTank, and Tranco websites [9]. The experimental machine used had a Core i5 processor with 16GB RAM and 66.6 Mbps fiber broadband internet connectivity. The MADONNA extension can predict whether the URL is malicious or benign on average in 2.43 seconds, which is reasonable for practical use. Although the SNN model can predict the malicious status of a given feature set in just 10μs, the MADONNA extension takes more time to extract some of the DNS-based and web-based features, which is why it takes 2.43s on average to provide a notification. The detailed experiment results are available in MADONNA's

GitHub repository[4], although they are not presented in the paper due to space constraints.

4.5 Limitations

The misclassification results suggest that certain malicious domains exhibit benign feature values while some benign domains exhibit malicious feature values for the selected features. This observation implies the need for more sophisticated and distinguishable web-based features to further minimize misclassifications. Features such as pop-up messages, alert boxes, a high percentage of advertisements, and site redirection are examples of malicious web-based features that can be taken into account during the feature analysis stage. However, these features must be extracted after the page has been loaded in the browser, which could potentially reduce the throughput and real-time usability of the solution.

The MADONNA model has a very fast inference time of just 10μs, but the prediction time through the browser extension takes on average 2.43s. This delay is mainly due to internet connectivity and cannot be easily solved with existing web-engineering techniques. The authors explored the possibility of using Pyscript[5] to remove the API execution step and convert it to a fully browser-based model, but this was not feasible due to limited library support for extracting web and DNS-based features. Therefore, MADONNA still requires a connection with a hosted API on the local machine, which adds computational overhead to the overall process.

5 Conclusion and Future Work

Overall, MADONNA is a promising approach to detecting malicious domains and demonstrates the potential of leveraging machine learning and browser-based applications for malicious domain detection. The authors' contributions in optimizing feature extraction, applying ML methods and optimization techniques, and introducing the SNN architecture are significant and demonstrate the effectiveness of MADONNA compared to state-of-the-art methods. However, there are also some limitations to consider, such as the computational overhead of connecting with a hosted API on the local machine and the dependence on internet connectivity for timely predictions. Further research could explore ways to minimize prediction time and improve accuracy while integrating web-based features to enhance MADONNA's capabilities. Nonetheless, MADONNA represents a significant step forward in the field of cybersecurity (malicious domain detection) and shows promise for future development and improvement.

References

1. Abdelnabi, S., Krombholz, K., Fritz, M.: Visualphishnet: Zero-day phishing website detection by visual similarity. In: *Proc. of CCS 2020. pp. 1681–1698. ACM (2020)*

[4] https://github.com/softwaresec-labs/MADONNA
[5] https://pyscript.net/

2. *Alhogail, A.A., Al-Turaiki, I.: Improved detection of malicious domain names using gradient boosted machines and feature engineering. Information Technology and Control* ***51****(2), 313–331 (2022)*
3. *Ariyadasa, S., Fernando, S., Fernando, S.: Combining long-term recurrent convolutional and graph convolutional networks to detect phishing sites using url and html. IEEE Access* ***10****, 82355–82375 (2022). https://doi.org/10.1109/ACCESS.2022.3196018*
4. *Berman, D.S.: Dga capsnet: 1d application of capsule networks to dga detection. Information* ***10****(5), 157 (2019)*
5. *Chien, C.J., Yanai, N., Okamura, S.: Design of malicious domain detection dataset for network security (2021), http://www-infosec.ist.osaka-u.ac.jp/ yanai/dataset.pdf*
6. *HR, M.G., MV, A., S, G.P., S, V.: Development of anti-phishing browser based on random forest and rule of extraction framework. Cybersecurity* ***3****(1), 1–20 (2020)*
7. *Huang, Y., Qiao, X., Dustdar, S., Li, Y.: Aodnn: An auto-offloading approach to optimize deep inference for fostering mobile web. In: Proc. of INFOCOM 2022. pp. 2198–2207 (2022)*
8. *Idelbayev, Y., Carreira-Perpinan, M.A.: An empirical comparison of quantization, pruning and low-rank neural network compression using the lc toolkit. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2021). https://doi.org/10.1109/IJCNN52387.2021.9533730*
9. *Iwahana, K., Takemura, T., Cheng, J.C., Ashizawa, N., Umeda, N., Sato, K., Kawakami, R., Shimizu, R., Chinen, Y., Yanai, N.: Madmax: Browser-based malicious domain detection through extreme learning machine. IEEE Access* ***9****, 78293–78314 (2021)*
10. *Li, T., Kou, G., Peng, Y.: Improving malicious urls detection via feature engineering: Linear and nonlinear space transformation methods. Information Systems* ***91****, 101494 (2020)*
11. *Morell, J.A., Camero, A., Alba, E.: Jsdoop and tensorflow.js: Volunteer distributed web browser-based neural network training. IEEE Access* ***7****, 158671–158684 (2019)*
12. *Palaniappan, G., S, S., Rajendran, B., Sanjay, Goyal, S., B S, B.: Malicious domain detection using machine learning on domain name features, host-based features and web-based features. Procedia Computer Science* ***171****, 654–661 (2020)*
13. *Rajapaksha, S., Kalutarage, H., Al-Kadri, M.O., Petrovski, A., Madzudzo, G., Cheah, M.: Ai-based intrusion detection systems for in-vehicle networks: A survey. ACM Comput. Surv.* ***55****(11) (feb 2023). https://doi.org/10.1145/3570954, https://doi.org/10.1145/3570954*
14. *Rupa, C., Srivastava, G., Bhattacharya, S., Reddy, P., Gadekallu, T.R.: A machine learning driven threat intelligence system for malicious url detection. In: Proc. of ARES 2021. pp. 1–7. ACM (2021)*
15. *Saleem Raja, A., Vinodini, R., Kavitha, A.: Lexical features based malicious url detection using machine learning techniques. Materials Today: Proceedings* ***47****, 163–166 (2021)*
16. *Senanayake, J., Kalutarage, H., Al-Kadri, M.O.: Android mobile malware detection using machine learning: A systematic review. Electronics* ***10****(13), 1606 (2021). https://doi.org/10.3390/electronics10131606, https://www.mdpi.com/2079-9292/10/13/1606*
17. *Senanayake, J., Kalutarage, H., Al-Kadri, M.O., Petrovski, A., Piras, L.: Android source code vulnerability detection: A systematic literature review. ACM Comput. Surv.* ***55****(9) (jan 2023). https://doi.org/10.1145/3556974, https://doi.org/10.1145/3556974*

18. Shabudin, S., Sani, N.S., Ariffin, K.A.Z., Aliff, M.: Feature selection for phishing website classification. International Journal of Advanced Computer Science and Applications **11**(4) (2020)
19. Shi, Y., Chen, G., Li, J.: Malicious domain name detection based on extreme machine learning. Neural Processing Letters **48**(3), 1347–1357 (2018)
20. Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., Bileschi, S., Terry, M., Nicholson, C., Gupta, S.N., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viégas, F.B., Wattenberg, M.: Tensorflow.js: Machine learning for the web and beyond (2019). https://doi.org/10.48550/ARXIV.1901.05350, https://arxiv.org/abs/1901.05350
21. Sun, X., Tong, M., Yang, J., Xinran, L., Heng, L.: Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification. In: Proc. of RAID 2019. pp. 399–412. USENIX Association (2019)
22. Sun, X., Yang, J., Wang, Z., Liu, H.: Hgdom: Heterogeneous graph convolutional networks for malicious domain detection. In: Proc. of NOMS 2020. pp. 1–9. IEEE (2020)
23. Tang, L., Mahmoud, Q.H.: A survey of machine learning-based solutions for phishing website detection. Machine Learning and Knowledge Extraction **3**(3), 672–694 (2021)
24. Vadera, S., Ameen, S.: Methods for pruning deep neural networks. IEEE Access **10**, 63280–63300 (2022). https://doi.org/10.1109/ACCESS.2022.3182659
25. Vinayakumar, R., Soman, K., Poornachandran, P.: Detecting malicious domain names using deep learning approaches at scale. Journal of Intelligent and Fuzzy Systems **34**(3), 1355–1367 (2018)
26. Yahya, F., W Mahibol, R.I., Ying, C.K., Anai, M.B., Frankie, S.A., Nin Wei, E.L., Utomo, R.G.: Detection of phising websites using machine learning approaches. In: Proc. of ICoDSA 2021. pp. 40–47. IEEE (2021)
27. Yang, L., Liu, G., Dai, Y., Wang, J., Zhai, J.: Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework. IEEE Access **8**, 82876–82889 (2020)
28. Yu, B., Pan, J., Hu, J., Nascimento, A., De Cock, M.: Character level based detection of dga domain names. In: Proc. of IJCNN 2018. pp. 1–8. IEEE (2018)
29. Yu, T., Zhauniarovich, Y., Khalil, I., Dacier, M.: A survey on malicious domains detection through dns data analysis. ACM Computing Surveys **51**(4) (2018)
30. Zabihimayvan, M., Doran, D.: Fuzzy rough set feature selection to enhance phishing attack detection. In: Proc. of FUZZ-IEEE 2019. pp. 1–6. IEEE (2019). https://doi.org/10.1109/FUZZ-IEEE.2019.8858884
31. Zamir, A., Khan, H.U., Iqbal, T., Yousaf, N., Aslam, F., Anjum, A., Hamdani, M.: Phishing web site detection using diverse machine learning algorithms. The Electronic Library **38**(1), 65–80 (2020)