APPLYING THE AUTOENCODER MODEL FOR URL PHISHING DETECTION

Dang Thi Mai
University of Transport and Communications, Hanoi, Vietnam

ARTICLE INFORMATION

Journal: *Vinh University Journal of Science Natural Science, Engineering and Technology*
p-ISSN*: 3030-4563*
e-ISSN: *3030-4180*

Volume: 53
Issue: 4A
Correspondence:
dtmai@utc.edu.vn

Received: *17 October 2024*
Accepted: *12 December 2024*
Published: *20 December 2024*

Citation:
Dang Thi Mai (2024). Applying the Autoencoder model for URL phishing detection. ***Vinh Uni. J. Sci.*** *Vol. 53 (4A), pp. 124-132 doi: 10.56824/vujs.2024a143a*

ABSTRACT

In the era of digital transformation, alongside the rapid development of the Internet and online applications, phishing attacks targeting users through malicious URLs have been increasingly prevalent. Traditional methods for detecting malicious URLs often rely on blacklist-based techniques. However, these techniques have significant limitations as they cannot identify new URLs. Many machine learning-based approaches have been researched and implemented to overcome these shortcomings. This paper proposes an improved two-stage deep learning model using an Autoencoder network for phishing URL detection. The proposed approach will be evaluated and tested on the standard UCI's URL Phishing dataset, achieving better results in most measure metrics.

Keywords: URL Phishing detection; Autoencoder; latent space.

1. Introduction

The rise of phishing attacks has become a significant cybersecurity threat, targeting users through deceptive URLs designed to steal sensitive information such as login credentials, financial data, and personal details. According to a 2023 report by the Anti-Phishing Working Group (APWG), the number of phishing attacks surged to an unprecedented level, with over 4.7 million attacks recorded globally, marking a 30% increase compared to the previous year [1]. Similarly, Google reported that phishing accounts for over 90% of all data breaches, underscoring its pivotal role in modern cybercrime [2]. A 2022 study by Verizon found that phishing remains the leading cause of breaches, accounting for 36% of reported incidents, while IBM's 2022 Cost of a Data Breach report estimated that the average cost of a phishing-related data breach has risen to $4.91 million, reflecting both the direct and indirect consequences of such incidents [3], [4]. Furthermore, recent statistics from Kaspersky show that phishing emails now account for over 45% of all malicious email traffic, demonstrating the sheer volume of these attacks [5]. Prominent security

firms like Symantec and Palo Alto Networks have highlighted the increasing sophistication of phishing campaigns, which now use machine-generated URLs, advanced social engineering techniques, and domain spoofing to evade detection and manipulate victims [6], [7].

Traditional phishing URL detection methods rely on blocklisting, heuristic-based techniques, and DNS analysis to identify malicious URLs. Blocklists, such as Google Safe Browsing and PhishTank, store databases of previously identified phishing URLs and block access [1], [2]. Although effective for known threats, blacklists face limitations when dealing with newly created phishing URLs or URLs that have been altered to evade detection, as studies show that 75% of phishing URLs remain active for less than 24 hours [8].

Heuristic-based approaches attempt to overcome these limitations by using manually defined rules to identify suspicious characteristics in URLs. These rules may include analyzing URL length, the use of uncommon domain extensions, the presence of numeric characters or suspicious keywords (e.g., "secure," "verify"), and deviations in URL structure [3], [9]. While these methods can detect previously unseen URLs, they often result in high false favourable rates due to their reliance on static rules that cannot adapt to evolving phishing strategies.

DNS-based analysis is another non-ML method that examines domain-related features, such as the age of the domain, registration details, and anomalies in DNS traffic patterns. Attackers frequently use newly registered domains or rapidly change DNS records to host phishing sites temporarily, which can be detected through these methods. However, obfuscated or compromised domains that appear legitimate pose significant challenges for DNS-based systems [4], [10].

Machine learning (ML) models have been employed for phishing URL detection to address the limitations of static detection methods. These models leverage handcrafted features extracted from URLs to classify them as phishing or legitimate. Features commonly used include URL length, the entropy of the URL string, the ratio of numeric characters, domain age, and the frequency of suspicious substrings (e.g., "login", "bank") [11].

This domain has widely used algorithms such as decision trees, random forests, support vector machines (SVMs), and logistic regression. For instance, Marchal et al. demonstrated that random forest models achieve high accuracy in phishing detection by combining multiple URL features. At the same time, SVMs effectively handle high-dimensional data when properly tuned [12]. These models, however, rely heavily on feature engineering, requiring domain expertise to identify and design meaningful attributes. Traditional ML methods also face challenges with imbalanced datasets. Since legitimate URLs vastly outnumber phishing URLs, models often become biased toward the majority class, resulting in poor performance in detecting rare phishing URLs. Techniques such as oversampling, undersampling, and synthetic data generation (e.g., SMOTE) are frequently used to address this issue [13]. Another challenge for traditional ML models is their inability to adapt to novel phishing tactics. Attackers often use typosquatting (e.g., replacing "paypal.com" with "paypa1.com"), homoglyph substitutions (e.g., swapping "o" with "0"), or embedding malicious content into legitimate platforms. These evolving strategies can bypass static feature-based models, necessitating frequent retraining with updated datasets to maintain detection performance [14].

To address the limitations of traditional ML methods, deep learning models have emerged as a powerful alternative for phishing URL detection. Unlike traditional ML models, deep learning approaches automatically learn feature representations from raw data, eliminating the need for extensive feature engineering. In this paper, we proposed a method that uses Autoencoders for URL phishing detection. The autoencoder model has gained attention for their ability to model complex patterns and detect anomalies in data. Our main contributions are as follows:

- Proposing a new method combining Autoencoders and traditional machine learning models for URL phishing detection.

- Conduct experimental evaluations of the proposed method and compare them with other machine learning models.

2. Related works

Over the past years, significant progress has been made in phishing URL detection, with various methods proposed to combat the increasing sophistication of phishing attacks. These methods have evolved from traditional heuristic-based approaches to machine learning and deep learning techniques.

Heuristic-based detection methods, which identify suspicious URLs based on patterns such as excessive length, obfuscated characters, or domain similarity, offer greater adaptability. While effective against URLs that deviate from standard patterns, heuristic methods are prone to high false-positive rates and can be bypassed by attackers employing subtle modifications to their URLs [15], [16].

ML approaches have addressed many of these limitations by leveraging features extracted from URL structure, domain metadata, and textual content. Algorithms such as decision trees, random forests, and support vector machines (SVMs) have demonstrated significant success, with random forests achieving accuracy rates exceeding 95% in some studies [17], [18]. In [19], Arathi Krishna et al. analyzed the effectiveness of machine learning for phishing URL detection. The authors extracted 48 features from URLs, including lexical, domain-based, and content-based attributes, without relying on third-party services. Models like Random Forest and Gradient Boosting demonstrated substantial accuracy due to feature diversity. Key performance metrics included accuracy, precision, and recall. The study demonstrated high efficiency in distinguishing phishing from legitimate URLs.

Subasi *et al.* [20] proposed an intelligent phishing detection system utilizing the UCI dataset, leveraging various machine learning algorithms for classification. These included Artificial Neural Networks (ANN), K-Nearest Neighbors (K-NN), Support Vector Machines (SVM), C4.5 Decision Trees, Random Forests (RF), and Rotated Forests (RoF). Among these methods, the Random Forest classifier outperformed the others, demonstrating superior accuracy, F-measure, and Area Under the Curve (AUC) results. The study highlighted Random Forest's advantages, noting its speed, computational efficiency, and high predictive accuracy, making it a more robust choice for phishing website detection.

ML models, however, depend on high-quality, diverse datasets to generalize effectively across different phishing scenarios. Challenges such as data imbalance, where legitimate URLs significantly outnumber phishing examples, remain a critical issue

affecting the performance of these models [21]. On the other hand, deep learning (DL) automates feature extraction and uncovers complex patterns in data. Techniques like convolutional neural networks (CNNs) excel in detecting visual mimicry through website screenshots, while recurrent neural networks (RNNs) effectively analyze sequential URL patterns. Although DL models improve detection capabilities, they face challenges such as high computational costs, limited real-time applicability, and the lack of transparency in decision-making processes [22], [23].

3. Autoencoder in URL phishing detection

3.1. Autoencoder

Recently, deep learning models have been successfully applied in various fields, such as image recognition, speech processing, natural language processing, and big data analysis. An autoencoder (AE) is a multilayer neural network designed to encode input data into latent representations and then decode it to attempt to reconstruct the input using the encoded data. The architecture of an autoencoder is divided into three parts: the Encoder, the Decoder, and the Latent Space, also known as the "bottleneck," as shown in Figure 1.

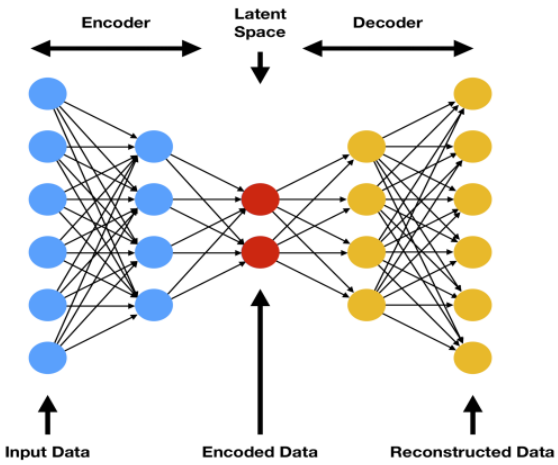

Figure 1: *Autoencoder model*

To learn the data representations of the input, the network is trained using unsupervised data (without labels). The input data will be transformed (encoded) through the neural network layers into a space with fewer dimensions (latent space). The data representations in this latent space will then undergo a decoding process to reconstruct the input.

Encoder: This consists of a standard feedforward neural network built to predict the latent representation of the input data. Where f_θ is the Encoder, and $X = \{x_1, x_2, ..., x_n\}$ is it a dataset? The Encoder f_θ aims to map the input $x_i \in X$ into a latent representation.

$$z_i = f_\theta(x_i) \quad (1)$$

Decoder (g_ϕ): This consists of a feedforward neural network, but the sizes of the layers increase in reverse order (symmetric) compared to the layers of the Encoder. The Decoder's purpose is to map the latent representation back to the input space; thus, the reconstruction is computed according to Formula (2):

$$\hat{x}_i = g_\phi(z_i) \tag{2}$$

The Encoder and Decoder are represented as activation functions of linear functions related to weights and biases as follows:

$$f_\theta(x) = \psi_f(\mathrm{W}x + b) \tag{3}$$

$$g_\phi(x) = \psi_g(\mathrm{W}'z + b') \tag{4}$$

Where $\theta = (W, b)$ and $\phi = (W', b')$ represent the weights and biases for the Encoder and Decoder, respectively. ψ_f and ψ_g are the corresponding activation functions.

The reconstruction loss over training samples can be written as follows:

$$\mathcal{L}_{AE}(\theta; \emptyset; x) = \frac{1}{n}\sum_{i=0}^{n} l(x_i, \hat{x}_i) \tag{5}$$

Where $l(x_i, \hat{x}_i)$ is the difference between the input x_i and its reconstruction $\hat{x}_i$, with $\hat{x}_i = g_\phi(f_\theta(x_i))$, n is the number of data samples in the dataset. The Autoencoder uses the backpropagation learning method to optimize the objective function in (5), which involves the parameters θ and ϕ.

The AE network is widely used in data science for applications such as dimensionality reduction, anomaly detection, image denoising, image compression, image generation, feature extraction, and recommendation systems.

3.2. Proposed method

We propose a two-stage model applied to the detection and classification of URL phishing detection using an AE network combined with traditional machine learning algorithms (AE_Decision Tree, AE_RandomForest, AE_kNN, AE_Naive Bayes, AE_MLP, AE_SVM) (Figure 2). In this model:

• Stage 1 involves using the AE network to train unsupervised data for feature extraction.

• Stage 2 is a supervised learning phase, where data from the latent space of the AE network is used as input for machine learning algorithms to detect and classify attacks.

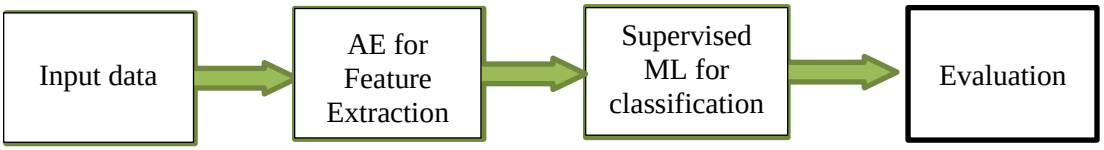

Figure 2: *A general flowchart of a two-stage model method*

The idea behind this proposal is that, instead of inputting all the dimensions of the original data into traditional classification algorithms, the AE network is used to reduce the data dimensions (feature extraction). Then, the reduced data is fed into classification algorithms. The goal is to improve the performance of traditional data classification models.

4. Results and discussion

4.1. Dataset

In this paper, we experiment using a widely used dataset for URL phishing detection - the UCI Repository URL phishing dataset. The UCI Repository dataset contains 11,054 links, of which 6,157 are labelled as legitimate and 4,897 as malicious. URLs are classified with a label of 1 for legitimate and -1 for phishing. Our approach divides the data into training and testing sets, following an 80/20 split.

To thoroughly assess the proposed method, we apply six different machine learning models: Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), K-Neighbors Classifier (KNN), Multiple Layer Perceptron (MLP), Support Vector Machine (SVM).

4.2. Evaluation measure

The classifier's results are typically evaluated using a confusion matrix and several metrics derived from it. The confusion matrix is described in Table 1, where Class 0 represents a standard URL and Class 1 represents a phishing URL.

The following metrics will be used to evaluate the performance of the models.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6}$$

$$Precision(\pi) = \frac{TP}{TP + FP} \tag{7}$$

$$Recall(\rho) = \frac{TP}{TP + FN} \tag{8}$$

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{9}$$

4.3. Result of evaluation

The experimental programs were implemented in Python using the PyTorch library. Experiments were conducted on a PC with an Intel Xeon Core i5-3.6 GHz processor and 24GB of RAM. The AE model has seven layers: the input layer, two hidden encoded layers, one latent layer, two hidden decoded layers, and the output layer. The input vector of the network has a size equal to the number of attributes of the dataset. The final configuration of the AE model was selected through grid search, and the numbers of neurons on two hidden layers are 28 and 18. The latent space dimension is 12.

The network training parameters are learning_rate = 0.01, batch_size = 100, display_step = 10, and epoch = 200. The weights of the Autoencoder (AE) are initialized using the Xavier initialization method to accelerate convergence. The optimization algorithm used is Adadelta, and the activation function is the TANH function.

The experimental result is shown in Table 1, in that the two-stage method has the prefix AE_. For example, AE_DT means that Autoencoder is used first for extracting the latent vector, then is applied to the Decision tree to classify standard or phishing URLs.

Table 1: *Results of URL phishing detection using different AI methods*

Classifier	Model	Accuracy	Precision	Recall	F_1
Decision Tree (DT)	AE_DT	**0.91180**	0.91234	**0.90919**	**0.91053**
	DT	0.90050	**0.91725**	0.89061	0.89686
Random Forest (RF)	AE_RF	**0.95477**	**0.95579**	**0.95286**	**0.95414**
	RF	0.92085	0.93693	0.91180	0.91814
Naïve Bayes (NB)	AE_NB	0.83265	0.83102	0.83397	0.83174
	NB	**0.83446**	**0.85065**	**0.84612**	**0.83436**
K-Neighbors Classifier (KNN)	AE_KNN	**0.94075**	**0.94249**	**0.93798**	**0.93983**
	KNN	0.93261	0.93286	0.93079	0.93172
MLP	AE_MLP	**0.94301**	0.94371	**0.94107**	**0.94223**
	MLP	0.93939	**0.95023**	0.93249	0.93773
Support Vector Machine (SVM)	AE_SVM	**0.91045**	0.91285	**0.90645**	**0.90885**
	SVM	0.90005	**0.92171**	0.88888	0.89588

The results indicate that the proposed model outperforms most algorithms. Specifically, five models achieve higher accuracy: AE_DT, AE_RF, AE_kNN, AE_MLP, and AE_SVM. Among them, models AE_RF and AE_kNN demonstrate superior performance across all metrics. Only with the Naïve Bayes classifier does the proposed method have a slightly lower accuracy.

5. Conclusion

The paper presented a technique combining AE's deep learning model with classifiers to detect phishing URLs. Experiments on the UCI URL Phishing dataset show that the proposed model achieves a high phishing URL detection rate while maintaining a lower false alarm rate than previous machine learning algorithms. Most of the proposed models demonstrated superior classification results, highlighting the effectiveness of AE in learning data features. However, training autoencoders involves optimizing both the Encoder and Decoder simultaneously, which is computationally intensive. This can lead to longer processing times, making them unsuitable for real-time or low-latency applications. Based on the obtained research results, future work will involve experimenting with online network data to evaluate the proposed model's performance in real-time phishing URL detection systems.

Acknowledgement: This research is funded by University of Transport and Communications (UTC) under grant number T2024-CB-010.

REFERENCES

[1] APWG, “Phishing Activity Trends Report,” 2023. [Online]. Available: https://apwg.org/trendsreports

[2] Google, “Data Breaches and the Role of Phishing,” *Google Security Blog*, 2023. [Online]. Available: https://security.googleblog.com

[3] Verizon, "2022 Data Breach Investigations Report," 2022. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir

[4] IBM, "Cost of a Data Breach Report 2022," 2022. [Online]. Available: https://www.ibm.com/security/data-breach

[5] Kaspersky, "Kaspersky Security Bulletin 2023," 2023. [Online]. Available: https://www.kaspersky.com/security-bulletin

[6] Symantec, "Internet Security Threat Report 2023," 2023. [Online]. Available: https://www.symantec.com/security-center

[7] Palo Alto Networks, "Threat Intelligence Report, 2023," 2023. [Online]. Available: https://www.paloaltonetworks.com/threat-intelligence

[8] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting Phishing With Streaming Analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458-471, 2014, DOI:10.1109/TNSM.2014.2377295

[9] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Soundsquatting: Uncovering the Use of Homophones in Domain Squatting," In *Proceedings of the 24th USENIX Security Symposium (USENIX Security '15)*, 2015, pp. 993-1008.

[10] U. Zara, K. Ayub, H. U. Khan, and A. Daud, "Phishing website detection Using Deep Learning models," *IEEE Access*, 7, pp. (99): 1-1, 01/2024, DOI: 10.1109/ACCESS.2024.3486462

[11] PhishTank, "Phishing Data Feed," 2023. [Online]. Available: https://www.phishtank.com

[12] Cisco, *Umbrella 2023 Threat Report*, 2023. [Online]. Available: https://umbrella.cisco.com

[13] R. Verma and A. Das, "What's in a URL? Fast Feature Extraction and Malicious URL Detection," In *Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics (IWSPA '17)*, 2017, pp. 55-63. DOI:10.1145/3041008.3041016

[14] A. K. Jain and B. B. Gupta, "Phishing Detection: Analysis of Visual Similarity-Based Approaches," *IEEE Access*, vol. 5, pp. 18521-18530, 8/2017. DOI:10.1155/2017/5421046

[15] Yue Zhang, Jason I. Hong and Lorrie F. Cranor, "CANTINA: A Content-Based Approach to Detecting Phishing Websites," In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*, 2007, pp. 639-648, DOI:10.1145/1242572.1242659

[16] C. Jackson, A. Barth, A. Bortz, W. Shao and D. Boneh, "Protecting Browsers from DNS Rebinding Attacks," *ACM Transactions on the Web (TWEB*), vol. 3, iss.1, pp. 1-26, 2009. DOI:10.1145/1462148.1462150

[17] K. Haynes, H. Shirazi and I. Ray, "Lightweight URL - based phishing detection using natural language processing transformers for mobile devices" *Procedia Computer Science*, vol. 191, pp. 127-134, 8/2021, DOI:10.1016/J.PROCS.2021.07.040

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002, DOI:10.1613/jair.953

[19] Arathi Krishna V, Anusree A, Blessy Jose, Karthika Anilkumarand and Ojus Thomas Lee, "Phishing Detection using Machine Learning based URL Analysis: A Survey," *International Journal Of Engineering Research & Technology (IJERT) NCREIS*, vol. 09, Issue. 13, 2021.
[20] A. Subasi, E. Molah, F. Almkallawi and T. J. Chaudhery, "Intelligent phishing website detection using random forest classifier," In *Proceedings of the International Conference on Electrical and Computing Technologies and Applications (ICECTA*), Phuket, Thailand, 10/2017, pp. 1-5.
[21] Y. Das et al., "Deep Learning Approaches for Phishing Website Detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 45-57, 2023.
[22] R. Zieni, L. Massari and M. C. Calzarossa, "Phishing or Not Phishing? A Survey on the Detection of Phishing Websites," *IEEE Access*, vol. 11, pp. 18496-18510, 2023.
[23] P. Sahoo et al., "Machine Learning for URL-Based Phishing Detection," *Journal of Cybersecurity Research*, vol. 29, pp. 12-25, 2022.

ABSTRACT

ỨNG DỤNG MÔ HÌNH AUTOENCODER TRONG PHÁT HIỆN URL PHISHING

Đặng Thị Mai
Trường Đại học Giao thông Vận tải, Hà Nội, Việt Nam
Ngày nhận bài 17/10/2024, ngày nhận đăng 12/12/2024

Trong công cuộc chuyển đổi số, cùng với sự phát triển nhanh chóng của Internet và các ứng dụng trực tuyến, ngày càng xuất hiện gia tăng các cuộc tấn công lừa đảo người dùng qua các đường URL độc hại. Các phương pháp phát hiện URL độc hại thường dùng các kỹ thuật chặn blacklist. Kỹ thuật này còn nhiều hạn chế do không phát hiện được các URL mới. Nhằm khắc phục hạn chế này, nhiều kỹ thuật ứng dụng máy học đã được nghiên cứu và triển khai. Trong bài báo này, chúng tôi đề xuất hướng tiếp cận cải tiến mô hình mạng học sâu hai giai đoạn với mạng học sâu Autoencoder ứng dụng trong phát hiện URL lừa đảo. Đề xuất sẽ được đánh giá, thử nghiệm với bộ dữ liệu tiêu chuẩn URL lừa đảo của Đại học UCI, cho kết quả tốt hơn ở hầu hết các độ đo đánh giá.

Từ khóa: Phát hiện URL lừa đảo; Mạng Autoencoder; không gian tiềm ẩn.