Received: 1 May 2022 | Revised: 21 September 2022 | Accepted: 24 December 2022

IET Information Security

DOI: 10.1049/ise2.12106

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

An enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders

Manoj Kumar Prabakaran[1] | **Parvathy Meenakshi Sundaram**[2] | **Abinaya Devi Chandrasekar**[3]

[1]Department of Information Technology, Thiagarajar College of Engineering, Madurai, India

[2]Department of Computer Science and Engineering, Sethu Institute of Technology, Virudhunagar, India

[3]Department of Computer Science and Engineering, Mepco Schlenk Engineering College, Sivakasi, India

Correspondence

Manoj Kumar Prabakaran, Department of Information Technology, Thiagarajar College of Engineering, Madurai, India.
Email: mano.btechme@gmail.com

Abstract

Phishing attacks have become one of the powerful sources for cyber criminals to impose various forms of security attacks in which fake website Uniform Resource Locators (URL) are circulated around the Internet community in the form of email, messages etc., in order to deceive users, resulting in the loss of their valuable assets. The phishing URLs are predicted using several blacklist-based traditional phishing website detection techniques. However, numerous phishing websites are frequently constructed and launched on the Internet over time; these blacklist-based traditional methods do not accurately predict most phishing websites. In order to effectively identify malicious URLs, an enhanced deep learning-based phishing detection approach has been proposed by integrating the strength of Variational Autoencoders (VAE) and deep neural networks (DNN). In the proposed framework, the inherent features of a raw URL are automatically extracted by the VAE model by reconstructing the original input URL to enhance phishing URL detection. For experimentation, around 1 lakh URLs were crawled from two publicly available datasets, namely ISCX-URL-2016 dataset and Kaggle dataset. The experimental results suggested that the proposed model has reached a maximum accuracy of 97.45% and exhibits a quicker response time of 1.9 s, which is better when compared to all the other experimented models.

KEYWORDS

deep learning, deep neural network, machine learning, phishing, uniform resource locator, variational autoencoders

1 | INTRODUCTION

The cybercrime referred to as 'phishing' is the process of enticing people into visiting fraudulent websites and persuading them to enter identifying information such as usernames, passwords, addresses, social security numbers, personal identification numbers and anything else that can be made to appear to be plausible [1]. These websites have similar webpage content and closely associated Uniform Resource Locator (URL) with the legitimate websites which in turn deceives the users into entering inside the website and providing their valuable credentials.

Taking into account the website of one of the largest banks in India—HDFC bank-as an example, the cyber criminal may term the official website URL, 'https://www.hdfcbank.com/' as 'https://www.hbfcbank.com/', with the aid of few unethical techniques such as URL hijacking, in order to distract the users to mistakenly login to the website since the URL and the web page content are close enough to the legitimate website [2]. Once the user is logged in, the hacker tries to collect the sensitive data of the user which can be used in identity theft, to remove funds from a customer account and in the theft of online resources [3].

[Correction added on 9 March 2023, after first online publication. The below correction needs to be noted: The order of the authors should be as follows: Manoj Kumar Prabakaran, Parvathy Meenakshi Sundaram and Abinaya Devi Chandrasekar.]

The numbers of phishing attacks have grown exponentially in the recent years, and according to the reports published in APWG [4], an international coalition unifying the global response to cybercrime across industry, government and law-enforcement sectors and NGO communities, between 68,000 and 94,000 phishing attacks per month have been observed since the early 2020s. By 2021, APWG recorded 260,642 malicious website URLs, the highest monthly attack count in the organizsation's history.

Conditions specifically during the COVID-19 pandemic led to an increase in the number of cyberattacks and cyber security problems. Phishing was by far the most prevalent form of attack that had distinctive features and was intended to take advantage of COVID-19's quirks in order to maximise its likelihood of success. [5] The majority of prominent phishing attacks that took place during COVID 2019 focussed on posing as government agencies (such as the WHO or the US Centres for disease Control and Prevention (CDC)) by either creating a fake email/website to lure the users to enter their sensitive information.

Research on this subject has been significantly prompted by the rise in phishing attempts over the recent few years, particularly during the COVID-19 pandemic that resulted in the loss of valuable human assets and serious real-world consequences.

Since most of the phishing-based crimes involves malicious website URLs to perform various form of cyberattacks, it is indeed mandatory to deploy an effective strategy to detect malicious URLs in a real-time environment. Hence to mitigate such attacks, various anti-phishing tools have been developed by many organisations in the recent past [6, 7]. Most of these tools use a technique referred to as Blacklists to analyse the legitimacy of the URL website. [8] Blacklists are databases of URLs for known phishing websites that are created and maintained by Internet communities like Phish tank. Blacklists are used by most of the lookup-based anti-phishing security toolbars found in web browsers. However, blacklists require time to update their most recent listings, leading to victimise lot of people [9].

While these methods are rapid and are intended to have low False Positive (FP) rates, one major drawback is that they cannot be entirely comprehensive, and they particularly fail when confronted with newly generated URLs. Since new URLs are generated on a daily basis, this is a serious constraint.

Hence to cope up with newly generated URLs, few researchers have come up with the idea of adopting machine learning algorithms to detect malicious URLs [10–12]. These ML-based techniques, in general, make use of statistical/Lexical-based features of URLs, namely Length of the URL, Number of periods in the URL, Top level domain etc. to classify URLs into two broad categories, namely benign and phishing. The main advantage of ML-based phishing detection approach is that it eliminates the dependency of collecting blacklists and its ability to detect new kind of malicious URLs.

Although ML-based approaches results in better detection accuracy, these techniques suffer from few considerable drawbacks: (a) Inability to extract semantic patterns that is, not all of the phishing website's qualities are extracted since URL is evaluated from a particular perspective (b) The feature extraction process involves manual feature engineering that requires the assistance of engineering experts. They must manually identify the features needed for classifier training from URLs and HTML text, which would lead to errors. (c) Inability to cope with URLs that do not conform to manually crafted features, for example, the effect of lexical features in categorisation has been inadvertently reduced due to the shift in the structure of today's modern-day URLs, which are substantially shorter in length.

Though many technical issues disrupt the effective performance of machine learning approaches, one way to reduce the cost of developing and maintaining machine learning approaches is to move beyond manual feature engineering, which is often considered as the most time-consuming attribute of machine learning.

In recent years, Deep learning (DL) algorithms have become popular in the field of phishing URL detection. Deep learning algorithms such as Convolutional Neural network (CNN), Auto-Encoder (Auto encoders (AE)) etc., have been used by various researchers [13, 14] to automatically extract abstract higher-level features from the raw URL. These algorithms have the aptitude to detect representations (features) that match the classifier's requirement from raw data. This helps in eliminating the dependency of experts for manual feature engineering and overcome the problem of lexical-based feature extraction. Also, DL models are scalable to a huge volume of data, and in fact its ability to classify increases as the number of the input data becomes higher. In contrast, DL models require the input data to be converted into numerical vectors and hence data needs to be preprocessed before being fed into the model. Also, the time taken to train a neural network model is a significant overhead since there are lots of training parameters involved.

Although, DL models have quite a few concerns, its ability to automatically extract higher-level features from the URL and precisely classify the input data fits it appropriately for malicious URL detection. Following its success and also in consideration with the complexities involved, we have proposed a hybrid DL model named VAE-DNN that combines Variational Autoencoders (VAE) (a special form of Autoencoders model) and Deep Neural network (DNN) for effectively classifying malicious URLs. By integrating these two neural network models, the detection accuracy shall be significantly improved.

Particularly, VAE-DNN receives raw URL input in the form of string and forwards it to a preprocessing stage where the input data is converted into a numerical matrix by means of a One Hot Encoding (OHE) mechanism. Once the URL is converted into numerical vectors, they are fed into the VAE architecture for the process of feature extraction/dimensionality reduction. After this stage, the VAE model produces a dimensionally reduced abstract representation of the input URL features from its latent space layer. Those extracted features are then fed into the DNN classifier for validation. Finally, the DNN model is trained and tested with the higher level extracted features produced from the latent space/bottleneck layer of the VAE model. After thorough experimentation, it has been observed that the proposed VAE-DNN model is able to effectively classify malicious URLs. The model is being

experimented with the data obtained from various resources such as Kaggle and ISCX-URL-2016 dataset. Experimental results suggest that our model reaches a maximum accuracy of 97.45% which is significantly better when compared with all the other experimented DL models. Following are the main contributions of our research. (1) We have incorporated the OHE mechanism to preprocess the input URL into a numerical matrix. This process embeds every URL into a fixed length numerical matrix with N × M dimension. We have adopted a strategy in setting the threshold value for N and M, such that the model shall be exposed to all kinds of possible characters that could probably form a URL for evaluation. (2) Also, we have adopted the VAE model for automatically extracting higher-level features from the input URL that eliminates the dependency of manual feature engineering. Since the VAE model has the ability to effectively reconstruct the original input, it is possible to obtain inherent information from the input URL and at the same time reduce the dimensionality of the input. (3) Since the VAE model produces dimensionally reduced inherent features of the URL, those features shall be used for effectively training the classifier. This significantly reduces the training time overhead of the classifier since the DNN model is trained with lesser number of features extracted from the latent space of the VAE model. (4) Since our model does not employ expert-assisted manually crafted features, the model eliminates the impediment of fixed human cognition to identify relevant features. Our architecture possesses the ability to discover a correlation between input features, implying that malicious URLs have more inherent properties. As a result, our approach can detect previously unknown malicious URL samples efficiently.

The rest of the paper is organised as follows. Section 2 discusses the related work undergone in the field of malicious URL detection. Section 3 provides a detailed overview of the proposed VAE-DNN framework. Section 4 demonstrates the various experiments being conducted and the appropriate results obtained. Finally in Section 5, discussions and conclusions are presented.

2 | RELATED WORK

In this section, we investigate the recent studies that have been conducted in the field of Information security. Numerous smart security solutions have been proposed in the recent past to detect the various intrusive attacks in a real-world networking environment. Probabilistic graphical models [15], Big Data-based Hierarchical DL System [16], stepping-stone intrusion detection based on clustering mechanisms [17], Deep Neural Network (DNN), CNN and other neural network models [18] have been effectively utilised to tackle the various network security issues.

However, the number of intrusive attacks based on phishing-based mechanisms has been growing exponentially. To effectively classify the phishing URLs from benign ones, many researchers have contributed in various ways to identify an optimal mechanism to generate a novel phishing detection model. Earlier studies reveal that rule-based phishing solutions were used to identify fake URL websites that adopted blacklist-based techniques that intimate the end user about the legitimacy of a particular website based on a fixed set of rules. Although these static approaches may work faster and respond quickly, still it is vulnerable to new URLs that are not available in the list which makes these techniques not so effective for a real-time environment. Followed by list-based approaches, many of the researchers focussed on incorporating machine learning and DL-based models for malicious URL detection. We have done an extensive survey on the various ML and DL-based approaches for phishing URL detection, where we have analysed the performance of the classifier with respect to manually crafted and automatically extracted features.

2.1 | Malicious URL detection using machine learning

In Ref. [19], the authors proposed an effective machine learning approach for phishing detection based on lexical/statistical features of the URL and the web page content. In this work, the researchers have considered 30 features based on address bar, anomaly, domain and webpage response for determining whether a website is phishing or not. For experimentation, data were collected from the publicly available UCI repository. Various Machine learning models were deployed for the purpose of classification. Among those models, the Random Forest (RF) classifier outperformed all the other models with the maximum accuracy of 96.87%. Although this approach seemed to be effective, this kind of technique suffers from the exploitation of 0-day attacks since the model is trained with fixed statistical/lexical-based features which can be easily decoded.

Saleem et al. [20] proposed a malicious URL detection mechanism based on lexical features by using machine learning algorithms. Similar to the work conducted by the authors in Ref. [19], in this work, fixed lexical features of the URL were considered for classification. Exactly 27 URL features were crafted based on the URL length, domain length, alphabets in URL etc., that represents the lexical behaviour of the URL. After extracting those features, few of the irrelevant features were reduced from the feature set and only 20 features were selected. For experimentation, URLs were collected from the UNB dataset that is composed of around 66,000 URLs. The results suggested that the RF classifier produced the maximum accuracy result of 99.5%. In comparison to the previously discussed methodology, this is a lightweight approach that offers better performance since it employs a minimum number of features for training the model.

Gupta et al. [21] devised an ML approach for phishing URL detection based on lexical properties of the URL. In this work, only nine lexical features of the URL were fixed based on the lexical characteristics of the URL. This work used ISCX URL-2016 dataset for collecting both benign and phishing URLs. Around 11,000 instances of the URL were composed and experimented among four different machine learning classifiers. A maximum accuracy of 99.7% was achieved by the

RF classifier. The uniqueness of this anti-phishing approach is that this model does not rely on third party services for feature extraction.

Ekta and Deepak [22] conducted a study on spoofed website detection by using a machine learning approach. In this study, the authors constructed 20 statistical features relevant to websites based on page, HTML and URL data. For experimentation, six machine learning classifiers such as NB, Decision Table, KNN, SVM, RF and Adaboost were used. The dataset was collected from the phish tank and open phish website that contains both legitimate and phishing websites. Around 5000 website URLs were collected from the mentioned resources and used for experimentation. Experimental results suggested that both RF and Adaboost algorithms produced better classification accuracy of more than 99% and minimal error in detecting appropriate URLs. Although this seems to be an effective approach for spoofed website detection, the model was only being trained with a minimal number of URL data, which makes it vulnerable to new kinds of URLs that are not available in the trained dataset.

Based on the study conducted, although it can be presumed that ML-based approaches for detecting malicious URLs seem to be an optimal approach, these techniques are still vulnerable to various issues such as reliability of third-party services for feature extraction, manual feature engineering, inability to cope up with a huge volume of URLs, dependency of fixed number of features for prediction etc. Since the modern-day hackers are generating URLs of which quite a few are shorter in length and certain other URLs have a unique structuring which are difficult to interpret. To overcome such issues, many researchers [23–26] have adopted the DL-based mechanism to classify the URL which does not require features to be manually extracted.

2.2 | Malicious URL detection using deep learning

Wang et al. [23] proposed PDRCNN (Precise Phishing Detection with Recurrent Convolutional Neural Networks, a quick phishing website detection method that entirely depends on the website's URL. This was the first way of detecting phishing in the context of cyber security issues using a DL model. To extract global features from the input URL, a bidirectional Long Short Term Memory network was used, and CNN was used to capture the significant components of the extracted features. PDRCNN had a detection accuracy of 97% and an Area under curve (AUC) value of 99%, which is much better than other approaches that employed artificial features for classification, although it took a long time to train the model. Furthermore, the model is oblivious about whether the relevant website to the URL being evaluated for training is active or not.

To enhance phishing website prediction, the authors at Ref. [24] presented an integrated intelligent phishing website prediction by using deep neural networks integrating evolutionary algorithm-based feature selection and weighting approaches. The genetic algorithm was used to intuitively determine the most influential features and the optimal weights of website features. The research's main contribution is the adoption of evolutionary algorithms to select suitable weights and attributes for URLs. The detection accuracy of the model was 89.5%. On the other hand, the proposed model has the following deficiencies: (i) Limitation of data sources (ii) Selecting and weighing features by using GAs considerably took a longer duration.

Liqun et al. [25] developed a unique approach based on a non-inverse matrix online sequence extreme learning machine (NIOSELM). To reduce the detection model's dependency on the majority class, an Adaptive Synthetic Sampling (ADASYN) approach was used. In addition, an enhanced denoising autoencoder (SDAE) was built to reduce the size of the experimental dataset. To balance the dataset and reduce dimensionality, the suggested model utilises unique preprocessing procedures. The detection accuracy, on the other hand, may not be as good as the existing approaches.

In order to overcome the issues prevailing in supervised classification in terms of robustness against 0-day phishing attacks, the authors at Ref. [26] proposed a hybrid model combining the convolution operation to model the character level feature of the URL and a deep Convolutional Autoencoder (CAE) to consider the nature of the 0-day attacks. Around 1.5 lakh URL data were collected from the three real-world datasets, namely Phish storm, Phish tank and ISCX-URL-2016 datasets for experimentation. The results demonstrate that the proposed method showed an accuracy of 96.42%. In comparison with the supervised approaches, the proposed model exhibited better performance. Incorporating autoencoders for phishing detection impacted heavily on the performance of the classification results. However, the proposed method tends to misclassify several phishing URLs as benign, since only character-level features were considered for individual URLs. Hence there is a need to identify a suitable preprocessing and feature extraction mechanism that reduces the FP rate of the model.

The models adopted in Refs. [25, 26] used various forms of Auto-encoder (AE) to effectively classify phishing URLs, and the findings reveal that adoption of AE was highly efficient in inheriting higher level abstract features from URLs. Taking advantage of the efficacies of the AE model, we have proposed a hybrid framework combining VAE and deep neural network (DNN) for effective classification of malicious URLs.

3 | METHODOLOGY

Various machine learning classifiers and hybrid algorithms for phishing detection models have been proposed in the literature, with most authors claiming to achieve the maximum accuracy in detecting phishing websites. Machine learning algorithms combine feature selection approaches with classification algorithms to provide a quick way to identify various forms of attacks. Despite the fact that ML-based algorithms have shown promising results in malicious URL detection, however these algorithms are susceptible to the following

issues: (a) Requires features of the URL to be manually extracted which depends on third-party services to obtain certain important features. (b) Inability to handle huge volumes of data. (c) Inability to cope with URLs that do not conform to the manually constructed features.

Deep learning algorithms, on the other hand, have grown in prominence as a result of their benefits over traditional machine learning classifiers. Deep learning algorithms possess the ability to automatically learn features from an input through successive forward and backward propagation of data. Few researchers in recent years have adopted neural network-based models for malicious URL detection. Various neural network-based approaches were used for automatically extracting inherent features from raw URLs, facilitating the unsupervised learning methodology. Despite its success, DL-based models have the following constraints: (a) The URL data should be converted into numeric vectors before feeding it to the model. (b) Training the model with raw inputs significantly takes a considerably longer duration. (c) The choice of the neural network model for feature extraction plays a vital role in the successful classification of the URL. (d) Huge numbers of hyper-parameter values are involved in D- based approaches, and the process of fine tuning the parameters is tedious.

Following the success of DL approaches in malicious URL detection and also considering the complexities involved, our research aims at proposing a novel DL-based malicious URL detection model that adopts the optimal techniques for preprocessing feature selection and classification. For experimentation, we have collected raw URL samples from various resources that consist of both malicious and benign URL data. Around half a million malicious URLs have been crawled from the Phish tank and ISCX-URL-2016 dataset and another half a million benign URLs are obtained from the Kaggle website.

Our study mainly focuses on alleviating the following issues that were prevailing in the existing phishing detection-based approaches. (i) Reliance of manually extracted URL features that requires a third-party service. (ii) Huge dimensions of the automatically extracted features that lead to training time overhead. (iii) Deprived preprocessing of URL data leading to inconsistency in identifying the important aspects of the URL. (iv) The poor response time of the model which is the consequence of the lack of optimality in the aforementioned issues.

Hence, we have proposed a detection model that incorporates optimal preprocessing and feature selection mechanism, and at the same time adapt necessary strategies to reduce the training time overhead and improvise the response time of the model. The steps involved in the proposed model are depicted in Figure 1.

3.1 | URL preprocessing

Since a DL model particularly accepts input in the form of numerical vectors, initially the data should be converted into either a single or multidimensional matrix with numerical values. Our model receives input in the form of URL strings in which each character has significance in determining the nature of the URL. Our approach employs OHE [27]-based preprocessing mechanism that converts every URL string into a numerical vector with N x M dimension. Each character in the URL is represented by a fixed length vector with M dimensions that consists of a sequence of 0's and 1's representing the position of the character. Depending on the length of the URL, a URL vector of N x M dimension is obtained.

In particular, M denotes the length of the maximum number of probable characters that might appear in a URL. As suggested in Ref. [28], there are 84 possible numbers of characters (a–z, A–Z, 0–9, - . !*'();:&= +$,/?#[]) through which an URL shall be formed. Hence we set the threshold value of M to be 84 such that our model captures all kinds of characters for evaluating the URL.

Here, N denotes the length of the URL, that is, the number of characters in the URL. Since the length of each URL is different from each other, it is not possible to allocate the value of N in accordance with the length of individual URLs. Hence, a fixed length is determined for all URLs by calculating the average length of all the URLs. Hence we set the threshold value of N to be 116, which is the mean length of all the URLS in the dataset. If a particular URL exceeds the fixed length value, then those extra values are trimmed. In contrast, if a URL possesses a shorter length than the fixed URL length value, then a zero padding mechanism is adopted in which 0's are appended to the vector.

Finally, the raw URL string will be transformed into 116 × 84 dimensional matrices with 116 rows and 84 columns, representing each character in a one hot encoded form after the preprocessing stage.

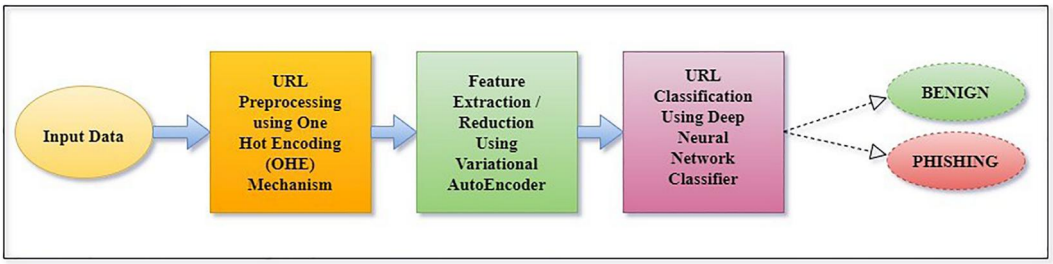


FIGURE 1 Steps involved in VAE-DNN Framework.

3.2 | Variational autoencoder (VAE)

After preprocessing the URL inputs, instead of directly feeding them to a neural network-based model for classification, our approach adopts a feature reduction/extraction technique to select certain inherent features of a URL to optimise the performance of the classifier. Reducing the dimensionality of the input URL significantly decreases the training time overhead associated with the classifier. The process of reducing the dimensionality of the input and recognising specific input features have certain pitfalls: (a) While reducing the dimensionality of the input, if suitable reduction mechanisms are not adopted, then there are chances for compromising important features that may lead to optimism. (b) Recognising valiant features from a URL input involves the consideration of inner relationship among the characters which is quite complex. Hence the neural network model should possess the aptitude to properly reduce the dimensions of the input and at the same time retain the important aspects of the input features.

In order to automatically extract salient features from the input URL vector, we developed an AutoEncoder (AE)-based feature extraction approach. The Auto encoders is a special form of feed forward neural network which is mainly designed to encode the input into a compressed and meaningful representation and then decode it back such that the reconstructed input is similar as possible to the original one [29].

There are two primary components of an Autoencoder model (a) Encoder and (b) Decoder. The Encoder part is dedicated to compress the original input vector into a reduced form which will be present in the latent space or sometimes referred to as the bottleneck layer of the model. From this intermediate layer, the decoder decodes the compressed data to reconstruct the original input data. The core principle behind this model is to minimise the loss incurred by calculating the difference among the original and the reconstructed data.

A traditional AE model can be constructed in one of the following ways. (a) Fixing the number of units in the hidden layer to be considerably lower than the number of units in the input layer. (b) Fixing the number of units in the hidden layer higher than the input layer units.

The former case is referred to as an undercomplete autoencoder in which the dimensions of the hidden layer are significantly lesser than the dimension of the input layer. The latter case is called an overcomplete autoencoder where the hidden layer has more units than the input layer. Both the undercomplete and overcomplete autoencoder suffer from the problem of generalisation since they get confined into the process of copying the original inputs directly without learning meaningful representation from the inputs. This is mainly due to the problem of non-regularisation in latent space of an autoencoder model, that is, data will be distributed in an uneven manner, and certain parts of the latent space do not represent the observed pattern leading to severe overfitting.

To avoid the problem of overfitting and non-regularisation in the latent space of a traditional AE model, we have adopted a Variational AutoEncoder (VAE), a special form of AE model that is capable of learning smooth latent space representation of the input data. The Helmholtz Machine [30], which was perhaps the first model to use a recognition model, inspired VAE. Its wake-sleep algorithm, on the other hand, was inefficient and did not optimise a single goal. Instead, the VAE learning rules are based on a single approximation to the maximum likelihood goal.

The basic principle behind the working of a VAE model is as follows: (I) Input Data is mapped to a latent space using a neural network. (i) The posterior and prior distribution of the latent space are modelled as Gaussian distribution. (ii) The output of the corresponding neural network is two parameters: (a) mean and (b) covariance, which are parameters of the posterior distribution. (II) A random sample from the latent space distribution is assumed to generate data that is similar to the input data. (III) The latent space vector is mapped to the generated data by using another neural network which in turns produces a reconstructed output corresponding to the mean of the Gaussian distribution.

The overview of VAE is shown in Figure 2. Variational Autoencoders is composed of three main components namely, Encoder, Decoder and Regularised loss function.

The encoder takes input training data, say X, and produces a latent representation, say Z, which in turn is generated using the parameters of the Gaussian distribution namely, mean(μ) and covariance(**Σ**). Here, the latent space is stochastic in nature that is, they are the parameters of a probability distribution.

The encoding function is given as

$$Q_{\varphi}(Z|X) \rightarrow Z \tag{1}$$

where **Z** in equation (1) can be expressed as

$$Z \rightarrow N(\mu(X), \Sigma(X)) \tag{2}$$

Here, the latent space follows a standard multivariate Gaussian distribution. The probability density function of a multivariate Gaussian is given by

$$P(Z: \mu, \Sigma) = \frac{1}{(2\pi)^{n}|\Sigma|} \exp\left(-\frac{1}{2}(Z-\mu)^{T}\Sigma^{-1}(Z-\mu)\right) \tag{3}$$

With sample vector $\mathbf{Z} = \begin{pmatrix} z1 \\ . \\ . \\ Zn \end{pmatrix}$, mean vector $\boldsymbol{\mu} = \begin{pmatrix} \mu 1 \\ . \\ . \\ \mu n \end{pmatrix}$ and Covariance matrix $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_n \\ \rho\sigma_1\sigma_n & \sigma_n^2 \end{pmatrix}$

Where $\rho\sigma_1\sigma_n$ are covariance and are considered to be diagonal and in turn expressed as an identity matrix with zero covariance and unit variance that is, $Z \sim \boldsymbol{N}(\mathbf{0}, \mathbf{I})$ for each

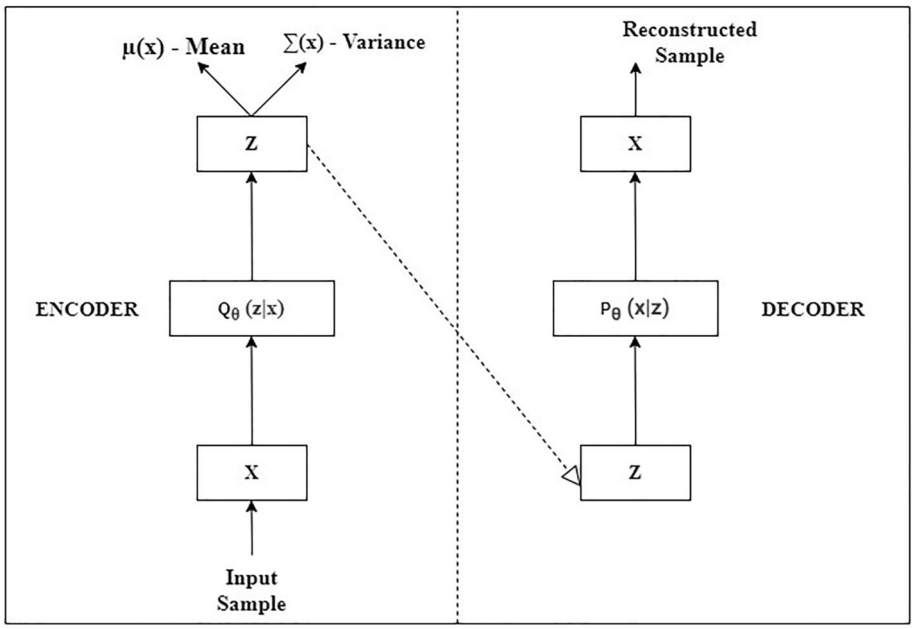


FIGURE 2 Traditional form of a Variational Autoencoder.

dimension and hence the variance for each dimension is one. Hence, to generate samples Z, only mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ vectors shall be considered, and the covariance matrix is not needed since they are diagonal, and there will be no interaction between the features. The sample Z obtained from the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are non-differentiable. Hence it is not possible to back propagate through the VAE model since the sampling is done based on non-differentiable parameters. To alleviate this issue, a standard technique called reparameterisation is adopted by sampling Z based on an additional parameter say $\boldsymbol{\epsilon}$, which is obtained from a unit standard normal distribution N (0, I) and is then multiplied by the variance vector and added with the mean vector. Since this sampling process is independent as it does not depend upon the parameters of both the encoder and decoder, the process of back propagation is feasible.

After applying the reparameterisation technique for sampling the input data samples, the final sample Z will be expressed as

$$Z = \mu + \sigma * \epsilon \tag{4}$$

In order to optimise the learning mechanism of the model, we used a log variance vector approach instead of a normal variance vector. The log variance vector is represented as

$$\text{Log variance} \rightarrow \log(\sigma 2) \tag{5}$$

By differentiating Equation (5), σ becomes

$$\log(\sigma 2) = 2.\log(\sigma) \rightarrow \frac{\log(\sigma^2)}{2} = \log(\sigma) \rightarrow \sigma = e^{\log(\sigma^2)/2} \tag{6}$$

Substituting σ in Equation (4) results in

$$Z = \mu + e^{\log(\sigma^2)/2} * \epsilon \tag{7}$$

Z is sampled from the output of the encoder and given as input to the decoder. The decoding function is represented as

$$P_\theta\ (X|Z) \rightarrow \widehat{X} \tag{8}$$

The decoder takes the input obtained from the latent space Z and outputs from the estimates of the input training data X. In particular, it enhances the dimension of the input data from lower to higher and reconstructs the input data. The reconstructed data $\widehat{X}$ is compared against the original input sample X, and the core expectation is that the obtained reconstructed data is as close as possible to the original input data.

The working principle of the VAE architecture is depicted in Figure 3 In order to train the model to generate the desired output, a loss function is calculated based on two terms namely, (a) Data Reconstruction loss and (b) Regulariser.

In general, the VAE loss function L is expressed as follows:

$$L = -D_{KL}\left[Q_\varphi(Z|X)\middle\|P_\theta(Z)\right] + E_{Q_\varphi}(Z|X)[\log(P_\theta(X|Z))] \tag{9}$$

The first part of the loss function - $\mathbf{D_{KL}}$ [$\mathbf{Q\varphi(Z|X)}\|$ $\mathbf{P_\theta(Z)}$] in Equation (9) is referred to as the Kullback-Lieber (KL) divergence loss term which is a regulariser that ensures that the parameters obtained from the encoder are as close to the unit normal distribution N (0, I).

In the KL divergence term, $Q_\varphi(Z|X)$ denotes the output of the encoder that provides mean μ(X) and the covariance

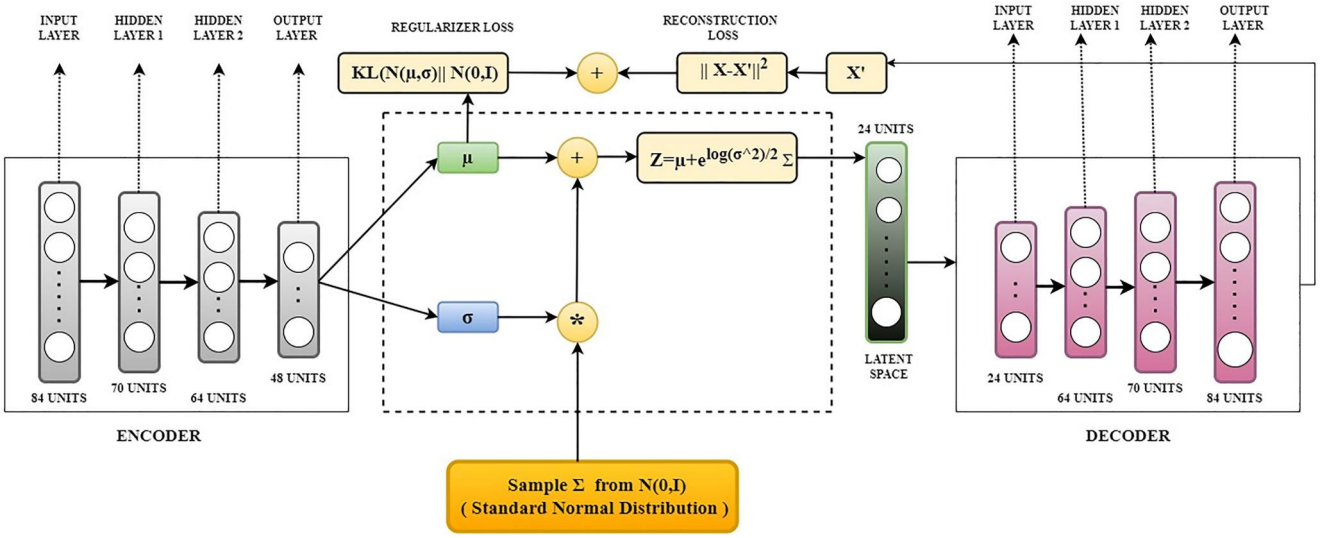


FIGURE 3 Working Principle of Variational Autoencoders (VAE) architecture

vector $\boldsymbol{\Sigma}(X)$ (constrained to be diagonal) corresponding to the Gaussian. And $P_\theta(Z)$ is another Gaussian with zero mean and unit standard deviation. Hence the KL divergence loss term $L1$ can be denoted as,

$$L1 = -D_{KL}\left[N(\mu, \sigma) + N(0, I)\right] \tag{10}$$

The second part of the loss function $\boldsymbol{E}_{Q_\varphi}(\boldsymbol{Z}|\boldsymbol{X})[\log(\mathbf{P}_\theta(\boldsymbol{X}|\boldsymbol{Z}))]$ in Equation (9) is referred to as the reconstruction loss, which is the output of the decoder that consists of the reconstructed input with respect to the sample Z. The difference between the original training samples and the reconstructed samples is calculated based on the mean squared error function $L2$ which is the squared difference between the input and the output and can be expressed as

$$L2 = \sqrt{\sum_{i=1}^{d}\left(X - \widehat{X}\right)^2} \tag{11}$$

Hence the overall VAE loss function L can be given as

$$L = -D_{KL}\left[N(\mu, \sigma) + N(0, I)\right] + \sqrt{\sum_{i=1}^{d}\left(X - \widehat{X}\right)^2} \tag{12}$$

3.3 | Proposed VAE-DNN architecture

To automatically extract higher-level abstract features from the input data and effectively identify malicious URLs, we have proposed a hybrid approach combining Variational Autoencoder (VAE) and Deep Neural network. The steps involved in the proposed framework are as follows:

1. Initially, the raw URL inputs are preprocessed by converting them into numerical vectors of a fixed dimension using OHE mechanism. 2. Then, the preprocessed URL vectors are passed as input to the VAE model for an unsupervised learning process. The model undergoes training in which it tries to reconstruct the original input through a series of forward and backward propagation mechanisms. Once an optimal loss is achieved, we extract the dimensionally reduced URL vector samples from the latent/bottleneck space of the VAE model. 3. The extracted low dimensional feature vectors are combined together to construct a new dataset which was fed as input to the DNN model for malicious URL detection. The dataset obtained from the VAE model is divided into training and testing dataset and passed into the DNN model for supervised training and classification. The detailed workflow of the proposed VAE-DNN framework is depicted in Figure 4.

3.3.1 | Feature extraction using unsupervised learning

In this section, we describe the design and steps involved in reducing the dimension of the input vectors using VAE architecture. An unsupervised learning methodology is adopted in which the raw URL inputs that were being converted into an OHE-based vector after preprocessing were given as input to the encoder part of the model.

The encoder part of the model is itself a fully connected feed forward neural network with one input layer, two hidden layers and an output layer comprising two vectors namely, mean and variance. The number of the units in the input layer of the encoder part is fixed to be 84, since it should match with the dimensions of the input. The number of hidden layer units is fixed to be 70 and 64 respectively. The numbers of units in the output layer of the encoder consisting of the mean and

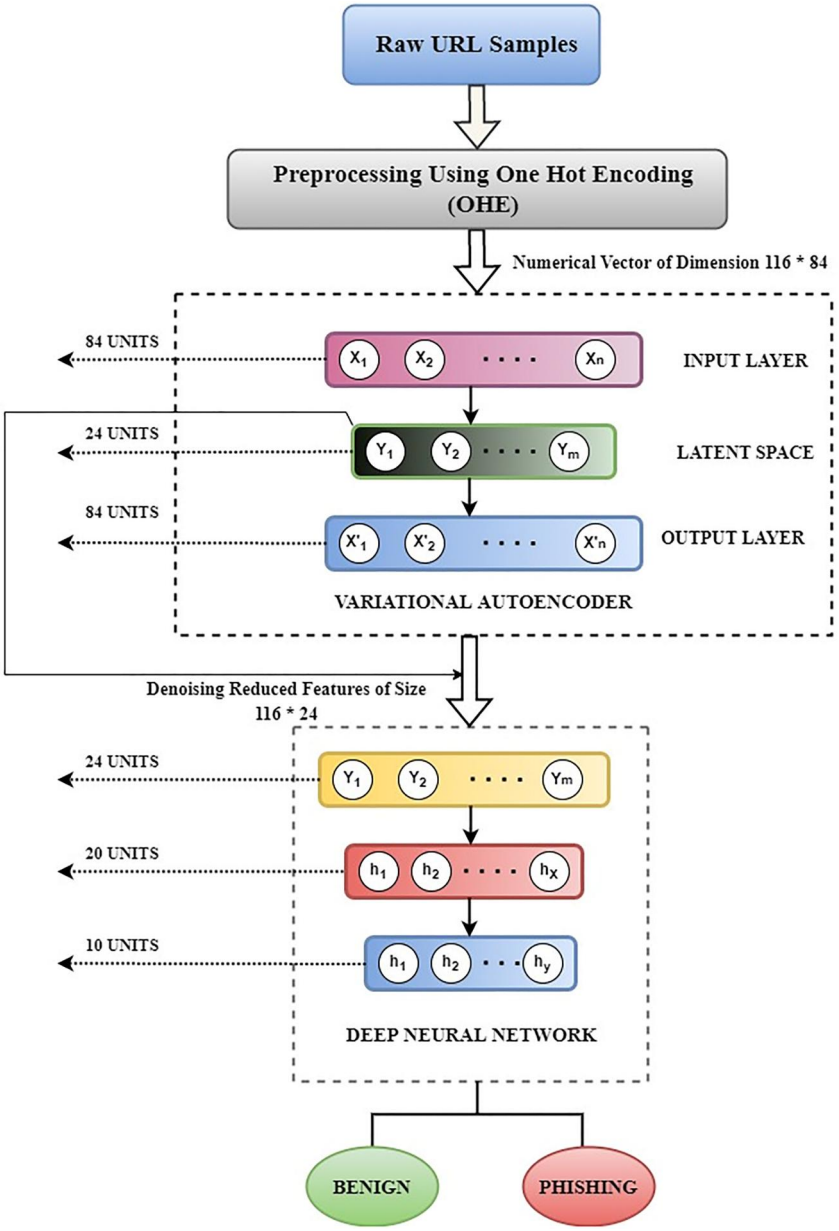


FIGURE 4 Workflow of the proposed VAE-DNN architecture.

variance vectors that are fixed to be 48, which is double the size of the latent space features. Latent space units L are fixed to be 24 based on several empirical experiments with respect to the loss function.

Similarly, the decoder part of the VAE model is also a fully connected feed forward neural network with one input layer, two hidden layers and an output layer. The number of units in the input layer is fixed to be 24, since the latent space inputs are fed into the decoder for training. Hidden layers one and two comprise 64 and 70 units, respectively. And the output layer consists of 84 units, which should be similar to the number of units in the input layer of the encoder part.

The training process of the VAE model comprises a sequence of forward and backward propagation mechanisms. In the forward pass, the URL vectors are propagated through the encoder in which lower dimensional URL samples are produced as output, which are then passed through the decoder where the lower dimensional vectors are again reconstructed back to the original input dimension. After the forward propagation, a loss function is calculated based on the difference among the original and the reconstructed output. In order to optimise the loss error, backward propagation is performed where a suitable optimisation technique is adapted in which the parameters of the decoder and encoder are adjusted and propagated backwards from the decoder to the encoder, and the process repeats.

Hence for training the VAE model, around one lakh URL input samples of dimension 116 × 84 were fed as input to the model. We set the number of epochs to be 25. The learning rate is fixed to be 0.001 and Adam's optimiser is used for the optimisation technique. Mean squared error and KL divergence loss term were used for calculating the loss function.

After the training process, the feature vectors from the bottleneck layer of the VAE model of dimension $116 \times L$ (L—Latent space size) were extracted to construct a new dataset with dimensionally reduced data.

3.3.2 | Supervised classification using deep neural network (DNN)

Once the features were extracted and a new dataset was constructed from the VAE architecture, finally those data were fed into a neural network model for malicious URL detection. In our work, the deep neural network architecture was proposed for effectively classifying malicious URLs. Around one lakh URL samples of $116 \times L$ (L—Latent space size) were collected from the VAE model and divided into two halves that is, 80% of the dataset were allocated for training the DNN model, and the remaining 20% of the data were retained for testing purposes.

The DNN architecture is composed of an input layer, two hidden layers and a probabilistic output layer. The number of units in the input layer was set according to the latent space size L. The hidden layers H1 and H2 were fixed to have units of length 20 and 10, respectively. The output layer is comprised two units. The input data were divided into equal batches of size 100. The number of training epochs was set to be 25, and the learning rate was fixed to be 0.001.

For minimising the loss function, the negative log-likelihood loss function was used. Batch normalisation was performed using a drop-out mechanism. SGD optimiser was used for the optimisation of the loss function. The Rectified Linear Unit was used as an activation function at every successive layer. For calculating the probabilistic outcome, the log softmax function was used as an activation function.

Finally, after the completion of training, the DNN model is tested against the testing data samples for malicious URL detection.

4 | RESULTS

4.1 | Dataset description

In our work, the URL samples were collected from different sources for experimentation. Exactly 99,658 URL samples were crawled from different resources. To balance the number of benign and malicious URL samples, almost equal numbers of benign and malicious URL samples were taken for a fair classification. The datasets used for experimentation are the Kaggle dataset and ISCX-URL-2016 dataset.

ISCX-URL-2016 dataset comprises around 1.5 lakh URLs categorised into five major categories namely Benign, phishing, malware, spam and defacement.

For our experimentation, we have collected around 50,000 malicious URL samples from this dataset (https://www.unb.ca/cic/datasets/url-2016.html) which is collected across the four different malicious categories (phishing, malware, spam and defacement). Kaggle is an open source dataset framework that comprises recently updated phishing and benign data across the Internet. We have collected around 50,000 benign URL samples from this forum (https://www.kaggle.com/xwolf12/malicious-and-benign-websites) for experimentation.

4.2 | Metrics

The proposed VAE-DNN model was evaluated based on the following metrics, namely the Confusion matrix, Classification accuracy, precision, recall, F1 score, True Positive (TP) Rate (TPR), True Negative (TN) Rate (TNR), FP Rate (FPR) and False Negative (FN) Rate (FNR).

A confusion matrix is a two-dimensional matrix used to represent the detection accuracy of the model based on the following terms, namely TP, TN, FP and FN. This matrix outputs the total number of samples being correctly and incorrectly identified by the classification model. Table 1 shows the general form of a confusion matrix.

True Positive refers to the number of benign samples being correctly identified as benign samples. True Negative value denotes the total number of malicious samples being correctly detected as malicious samples. False Positive is calculated based on the number of malicious samples being incorrectly identified as benign samples. False Negative is obtained by counting the number of benign samples that are incorrectly identified as malicious samples.

Classification accuracy of the model is calculated based on the ratio of the number of correctly identified URL samples to the number of samples. The accuracy of the model is given by

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{13}$$

Precision is one of the important metrics which can be calculated based on the ratio of the total number of correctly identified benign samples to the total number of samples being identified as benign.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{14}$$

TABLE 1 Confusion matrix

		Actual class: Benign	Actual class: Malicious
Predicted class	Benign	TP	FP
	Malicious	FN	TN

The term Recall or TP rate (TPR) or sensitivity is referred to as the total number of correctly identified benign samples to the actual number of benign samples in the given dataset.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{15}$$

False Positive Rate denotes the proportion of malicious samples incorrectly identified and is calculated based on the following formula,

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{16}$$

The term TNR or specificity is a proportion of the malicious samples correctly being identified and can be denoted as

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{17}$$

False Negative Rate denotes the proportion of benign samples incorrectly identified and can be expressed as

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{18}$$

The F1 score is measured based on the harmonic mean of precision and recall and is given by,

$$\text{F1 score} = 2 * \left(\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}\right) \tag{19}$$

4.3 | Experimentation design

The entire experiment was carried out by using the Google Colab Pro, a setup-free, cloud-based Jupyter notebook environment. For our work, we have used Graphics Processing Unit (GPU) of Google Colab Pro (High RAM). The hardware specifications offered by Google Colab Pro are a Tesla v100 PCIe GPU accelerator with a single precision performance of 14TFLOPS offering a memory bandwidth of 900 GB/sec along with HDD of size 125 GB and a memory size of 25 GB.

We have implemented our VAE-DNN model using Pytorch framework, an open source machine learning library for python which performs tensor computations on graphics processing units for building the different neural network models.

The proposed model comprises two distinct neural network architecture combined together for the purpose of feature extraction and classification of input data. The Variational Autoencoder (VAE) plays the vital role of reducing the dimension of the input data along with extracting higher-level abstract representation of the original data. The Deep Neural network (DNN), on the other hand, acts as a classifier by receiving the input features from the VAE model, which is actually the reduced version of the original input features.

To evaluate the performance of the proposed VAE-DNN architecture, various experiments have been conducted based on several metrics to assess the accuracy of the model in effectively detecting the malicious URLs. Following are the various steps taken to ensure the efficacy of the proposed model. (a) Initially, to evaluate the efficacy of the proposed model, the model was compared against a traditional deep neural network model that does not comprise a feature extraction mechanism. This traditional model was trained and tested against the original input data without feature reduction. The final classification results obtained from this model were recorded and set for comparison against the proposed VAE-DNN model. (b) For fixing the latent space size 'L' of the VAE model, various experiments were conducted to finalise a fixed length for the bottleneck space of the VAE architecture. Hence the VAE model was separately trained with the entire input data with different range of L values for a fixed number of iterations. (c) After finalising an optimal L value for the VAE model based on the experimental results, the dimensionally reduced feature vectors from the latent space of the model were given as input to the DNN for malicious URL detection. The performance of the model was assessed based on various metrics, namely confusion matrix, precision, recall, F1 score and accuracy. (d) Also, to highlight the advantages of incorporating the VAE model for feature selection, various AE-based neural network models were taken into consideration for the feature selection mechanism. A hybrid architecture combining different AE models and DNN was experimented, and the results were compared against the proposed VAE-DNN model. Following are the metrics used for evaluating the different AE-based models, namely TP rate (TPR), FP rate (FPR), TN rate (TNR) and FN rate (FNR). (e) Finally, to evaluate the responsiveness of the model in immediately identifying the malicious URLs, a random set of sample URLs were taken from the real-world environment and tested against the model for measurement. The response time of the different AE-based DNN models were taken for evaluation.

4.4 | Results

In this section we have described the process involved in analysing the performance of the proposed model. Various metrics have been involved in assessing the efficiency of the model with respect to malicious URL detection. Initially, the performance of the model is measured based on confusion matrix and classification accuracy. Then, the model's performance outcome is compared against a traditional neural network model. In addition, different autoencoders models were taken into consideration for experimentation, and their detection accuracy is assessed based on various metrics, namely precision, recall and F1 score. Finally, an ROC curve is plotted to observe the superiority of the proposed model in comparison to all the other experimented models.

4.4.1 | Hyper-parameters of VAE architecture

Our work focuses on deploying a VAE for extracting dimensionally reduced higher-level representation of the raw URL inputs. As described in section 3.2, the VAE model is composed of an encoder part, a latent space and a decoder part. During the training process, the model tries to reconstruct the original input through a series of forward and backward propagation. Once the training is completed, the features from the latent space of the model are fed as input to the deep neural network for further classification. The main purpose of training a VAE model with the original input features is to reduce the dimensionality of the input along with extracting significant features.

Hence, the number of units to be fixed in the latent space heavily determines the performance of the classifier, since those are the number of units to be fixed as an input layer for DNN. To identify an optimal latent space size L, experiments were conducted with various ranges of values, and a suitable L is finalised based on the final loss obtained after a fixed number of iterations during training.

The value of L was set as 5, 10, 24, 48 and 64. Since the original input URL is of dimension 116 × 84 after preprocessing, the values taken for L were fixed below 84. The L values were set accordingly, and the model was trained for a fixed 25 epochs. The final loss was assessed for different L values. Figure 5 shows the loss curves obtained for VAE models for different latent space sizes. For $L = 5$ and $L = 64$, the final loss achieved after 25 epochs does not reach an optimal value when compared with other L values.

For latent space $L = 10$ and $L = 48$; although the initial loss value obtained for the first epoch was considerably low when compared to other L values, the final loss value achieved was not as expected. For latent space size $L = 24$, the optimal loss value was achieved, reaching a final loss of 0.05.

Hence based on the experimental results, the final latent space size L was fixed to 24 since it reaches an optimal loss value after 25 epochs.

Apart from latent space size, the number of units in the input and output layer of VAE was fixed as 84. For calculating the loss value, MSE loss function and the KL divergence term were used. The learning rate was fixed as 0.001. For optimising the loss function, Adam's optimiser function is used.

4.4.2 | Performance analysis of the VAE-DNN model

After training the VAE model, a dataset is constructed based on the dimensionally reduced features extracted from the bottleneck layer of the VAE and fed as input to the DNN for training and testing. Around one lakh URLs of dimension 116 × 24 were split into training and testing data as 80:20 ratios. A total of 79,745 URL samples were taken for training, and the remaining 19,913 samples were allotted for testing the model. The number of training epochs was fixed to 25, and the learning rate was fixed to 0.001. The structure of the DNN classifier is already described in Section 3.3.2.

Figures 6 and 7 depicts the loss curve and accuracy curve of the VAE-DNN model. The final loss achieved after 25 epochs during training and testing was below 0.1. The classification accuracy of the model during training reached a maximum of 98.52% and testing accuracy reached the highest of 97.45%. The total time taken for training the VAE-DNN model took around 268 s.

In order to assess the detection accuracy of the model with respect to both benign and malicious samples, a confusion matrix was constructed to identify the total number of TP, TN, FP and FN samples. Among the 19,913 samples, our model was able to correctly identify 19,423 URL samples leading to a detection accuracy of 97.45%. In the testing dataset, the total numbers of benign samples were 10,002, and the numbers of malicious samples were 9911.

Table 2 shows the confusion matrix of the VAE-DNN model. As can be inferred from the table, out of the 19,913 samples, exactly 490 URL samples were wrongly identified. Among which 280 benign samples were wrongly identified as phishing samples, and 210 Phishing samples were misinterpreted as benign samples.

In order to further assess the performance of the proposed model, a comparative analysis of VAE-DNN is performed by constructing a model that is composed of only a deep neural network, and the autoencoder part that was integrated in our work for feature extraction was not included in this architecture. This newly constructed DNN model is composed of one input layer with 84 units, and two hidden layers with unit lengths 20 and 10, respectively, and an output layer with two units. This DNN model is structured in almost the same way as we organised DNN in our proposed work except for the number of units in the input layer which was set to 24 in our case.

The main intention of this experimentation is to check whether the influence of feature extraction based on VAE plays a vital role in effectively classifying the input URLs. Hence, the feature extraction part is taken out of the picture, and the DNN part alone is considered for classification with all the preprocessed original input URL of dimension 116 × 84. The model is trained and tested under the same circumstance with a similar hyper-parameter configuration under which our proposed model was implemented.

Figure 8 shows the detection accuracy of the DNN model without feature extraction mechanism.

As can be inferred from Figure 8, the maximum accuracy achieved by the model during the training phase was 92.5% and while testing the model it acquired the highest accuracy value of 89.95%. Hence from the results, it can be clearly seen that our proposed model provides better results in terms of accuracy when compared with the newly constructed model that does not comprise a feature extraction mechanism. Hence it can be summarised that the role of the VAE model in extracting higher-level abstract features from the inputs leads to an effective malicious URL detection.

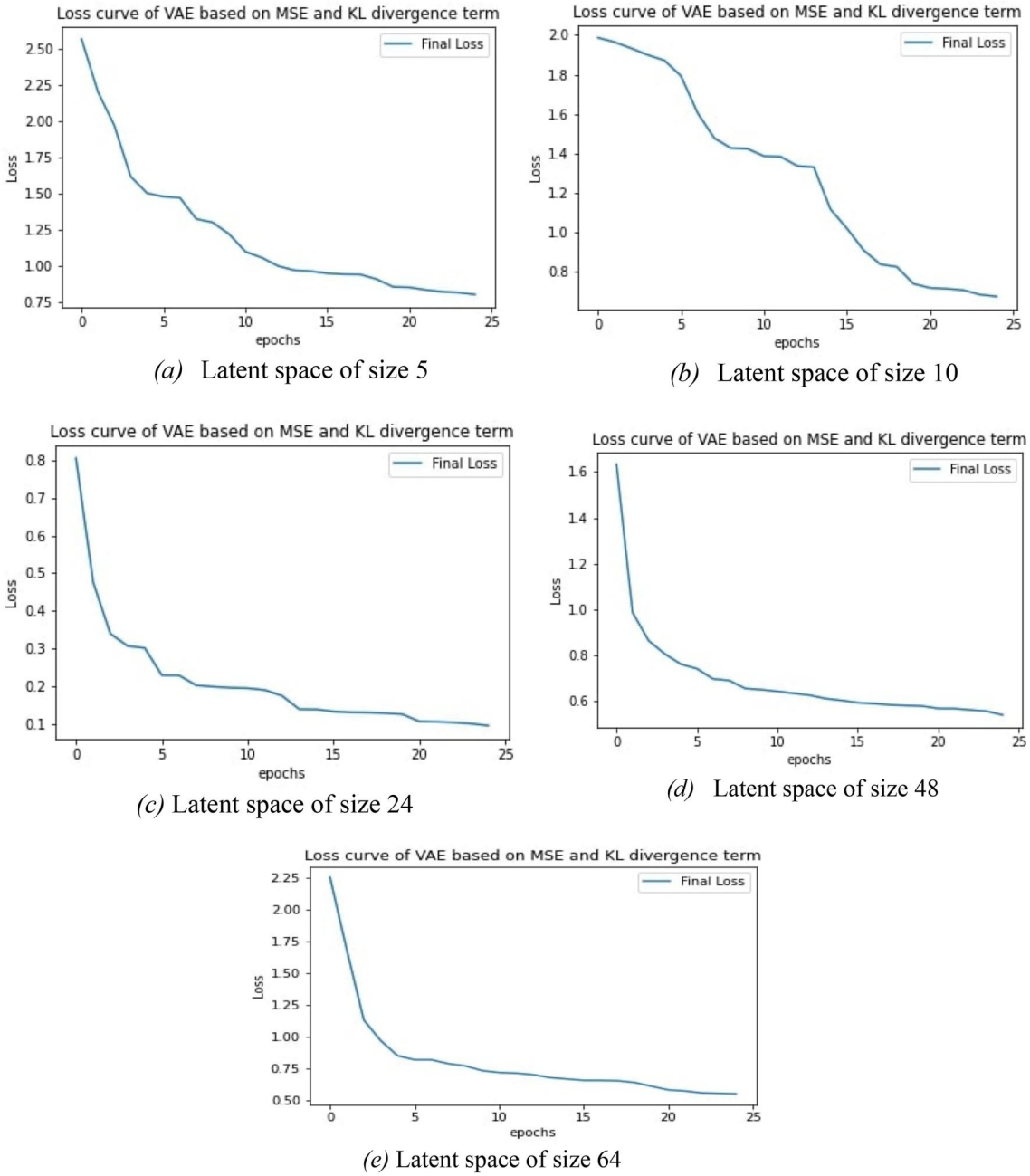


FIGURE 5 Experimentation with different latent space size for optimisation of loss value.

4.4.3 | Comparison analysis

In this section, our work focuses on experimenting with different techniques for dimensionality reduction in order to evaluate the proposed architecture. Although we have proven that incorporating the VAE model for dimensionality reduction performs exceptionally well through various experimentations, still we have not explored different techniques for dimensionality reduction in order to come to a conclusion.

Hence we have considered various autoencoder models for the task of dimensionality reduction and constructed a combined architecture, which is almost similar to that of the proposed architecture, by combining the autoencoder model with the DNN classifier for the process of feature selection and classification.

Following are the different architectures constructed based on autoencoders for comparative analysis.

- Traditional Vanilla Autoencoder-based DNN (TVAE-DNN)
- Deep Autoencoder-based DNN (Deep AE-DNN)
- Denoising Autoencoder-based DNN (Denoising AE-DNN)
- Sparse Autoencoder-based DNN (Sparse AE-DNN)
- Convolutional Autoencoder-based DNN (Convolutional AE-DNN)

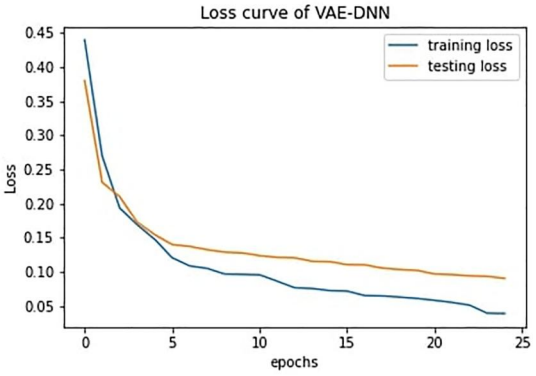


FIGURE 6 Loss curve of the VAE-DNN model.

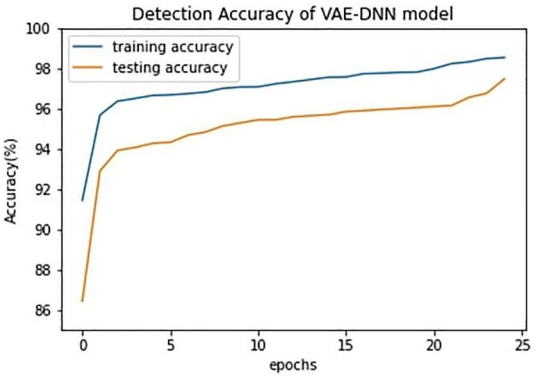


FIGURE 7 Detection Accuracy curve of the VAE-DNN model.

- Contractive Autoencoder-based DNN (Contractive AE-DNN)

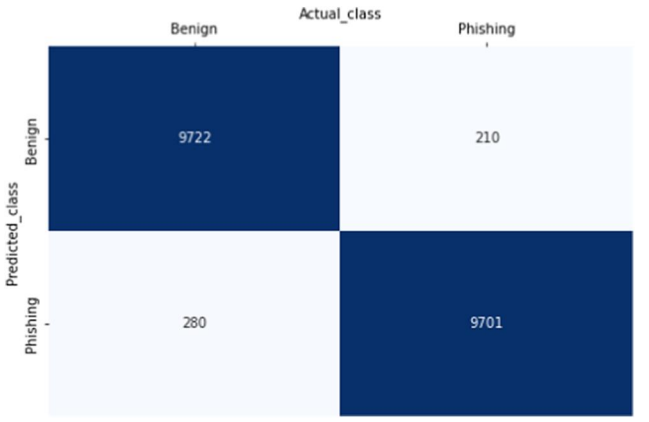
These models were trained and tested, and their performance was assessed based on various metrics namely precision, recall, F1 score and training time per epoch. The structure of these models is as follows:

For all the experimented models, the number of units in the input and the output layer of the AE architecture was fixed to be 84 units and the number of units in the hidden bottleneck layer was fixed to be 24 units in accordance with the proposed model for a fair comparison.

The TVAE-DNN model is composed of a traditional autoencoder-based architecture in which there will be one input layer, one hidden layer and one output layer. The Deep AE-DNN model is similar to the TVAE-DNN except for the fact that it consists of multiple hidden layers as opposed to a single hidden layer structure. For our experimentation, three hidden layers are included which comprise 64, 40 and 24 units, respectively.

The denoising AE-DNN model is a special form of an autoencoder model in which a certain form of noise or perturbation is added to the input features and fed for reconstruction. The reconstructed data is compared with the original data instead of the noisy input. This will overcome the problem of overfitting since it does not allow the model to simply copy the original input, instead it allows the model to decode a meaningful pattern from the original data.

TABLE 2 Confusion matrix of the VAE-DNN model



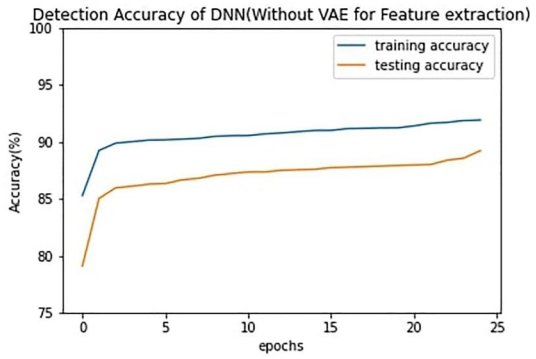


FIGURE 8 Accuracy curve of DNN without Variational Autoencoders (VAE) for feature extraction.

The sparse AE-DNN model is structured in a similar manner with respect to TVAE-DNN with an exception that a sparsity constraint is added while calculating the loss function, which penalises the activations of the hidden layer in order to ensure that only a few units are activated when an input is being fed to the model, leading to better regularisation.

The convolutional AE-DNN model is a special variant of CNN that is composed of convolutional and pooling layers stacked together to form the encoder as well as the decoder part. Hence both the encoder and decoder are individually constructed by CNN models and used for dimensionality reduction.

The structure of the Convolutional AE-DNN model is Input Layer - > Conv layer1 - > max pooling - > ReLu - > Conv layer2 - > max pooling - > ReLu - > flatten layer. Both the encoder and decoder are configured in accordance with the above structure with a fixed number of units in the input and output layer.

The contractive AE-DNN model is almost similar to the Sparse AE-DNN in which a penalty term is added to the loss function in order to make the model adapt to small changes in

TABLE 3 Precision, recall, F1 score and Accuracy of all the experimented models

Model	Precision (%)	Recall (%)	F1 Score (%)	Accuracy (%)
AE-DNN	92.77	90	91.36	91.45
Deep AE-DNN	94.39	92.02	93.10	93.25
Denoising AE-DNN	96.04	94.22	95.12	95.15
Sparse AE-DNN	95.83	93.82	94.81	94.85
Convolutional AE-DNN	96.91	94.88	95.88	95.91
Contractive AE-DNN	97.02	96.08	96.54	96.55
VAE - DNN	**97.89**	**97.20**	**97.54**	**97.45**

TABLE 4 True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR) and False Negative Rate (FNR) values for all the models

Algorithms	True positive rate (%)	False positive rate (%)	True negative rate (%)	False negative rate (%)
AE-DNN	90	7.08	92.92	10.01
Deep AE-DNN	92.02	5.52	94.48	7.98
Denoising AE-DNN	94.22	3.91	96.08	5.78
Sparse AE-DNN	93.82	4.17	95.88	6.18
Convolutional AE-DNN	94.88	3.06	96.94	5.19
Contractive AE-DNN	96.08	2.98	97.02	3.92
VAE-DNN	**97.20**	**2.19**	**97.88**	**2.80**

the input data resulting in a localised space contraction, yielding a robust feature on the activation layer.

Tables 3 and 4 clearly depict the performance of all the experimented models with respect to various metrics, namely precision, recall, F1 score, accuracy, TP rate, TN rate, FP rate and FN rate.

The results from Tables 3 and 4 suggest the following inference. In terms of detection accuracy, the VAE-DNN model outperforms all the other experimented models with the highest accuracy of 97.45%. The traditional VAE-DNN model was the least to perform in terms of detection accuracy. In terms of precision and recall both VAE-DNN and contractive AE-DNN produced close enough results which were the highest among all the others, reaching above 96%.

Both the Denoising AE-DNN and Convolutional AE-DNN have acquired higher precision values and almost reaching 96%, but in terms of recall measurement, both the models tend to produce average results. Regarding the F1 score value, apart from traditional and deep AE-based neural network models, all the other models achieved a maximum 95% with the highest value achieved by our proposed model, reaching 97.54%.

Although all the experimented models were producing better results in terms of accuracy and F1 score, it is mandatory to assess the model's ability in effectively identifying the malicious URL, which can be assessed based on two metrics, namely a FP rate and FN rate. As can be observed from Table 3, VAE-DNN has the lowest false alarm rate of 2.19 and FN rate of 2.80 when compared to all the other models in the experimentation. This result has proven that the VAE-DNN model is considerably better in terms of classification accuracy as well as accurately detecting the input URLs based on its categorisation.

TABLE 5 Area under curve (AUC) score for all the experimented models

Algorithms	AUC score
Vanilla AE-DNN	0.9351
Deep AE-DNN	0.9415
Sparse AE-DNN	0.9555
Denoising AE-DNN	0.9607
Convolutional AE-DNN	0.9699
Contractive AE-DNN	0.9789
Variational AE-DNN	**0.9858**

The AUC score was computed for all the experimented models in order to further validate the strength of the proposed model and is depicted in Table 5.

The Area under Curve is used to measure the capability of a model in discriminating classes. Maximum the value of AUC, better the outcome of the prediction. VAE-DNN achieves the maximum AUC score of 0.9858, which is far better than all the other models.

Figure 9 represents the AUC-ROC curve. This curve plots the tradeoff between a TP rate and FP rate at different classification thresholds. The Area under Curve score is calculated by measuring the entire 2-dimensional area beneath the ROC curve. It provides an aggregate measure of a model's performance across different threshold values.

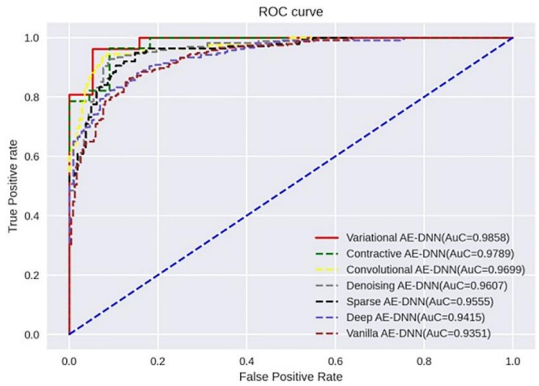


FIGURE 9 AUC-ROC curve for the experimented models

From Figure 9, it can be understood that the traditional AE-based models are not consistent in classifying the data across different threshold values as we can witness a clear fluctuation in the curve for both TVAE and Deep AE models. Among the other AE-based approaches, the contractive AE-based model performed better at varying thresholds; however, our VAE-DNN proved to be much more consistent than all the other models as it remained closer to value one for the maximum number of threshold values.

The response time of a model is an important measure to consider when using it in a real-time context. Because the model that will be deployed must be able to reply to URLs in real-time in a very short length of time. The time difference between the time the URL is fed and the projected results can be used to calculate a model's response time. Once the URL is fed into a model, it would go through various steps, including URL preprocessing, feature selection and prediction of whether the URL is benign or phishing.

As a result, a study was carried out to evaluate the tested models in terms of response time. Table 6 depicts the findings of the investigation. A random set of URLs from both the training and testing sets were fed into all of the experimented models for testing purposes, and response time was calculated for each model.

Table 6 suggests that the traditional VAE model possesses the least response time of 1.3 s due to the simplicity in the structuring of the model. Our proposed VAE-DNN model acquired a response time of 1.9 s which is the second fastest model to respond among all the models. Convolutional AE-DNN took the longest time to respond due to the complexity of its structure. The runtime achieved through VAE-DNN is optimal and is considered suitable to be deployed in a real-world environment.

TABLE 6 Response time of all the models

Algorithms	Response time (s)
Vanilla AE-DNN	1.3
Deep AE-DNN	2.4
Sparse AE-DNN	2.7
Denoising AE-DNN	3.5
Convolutional AE-DNN	4.1
Contractive AE-DNN	3.1
Variational AE-DNN	**1.9**

4.5 | Discussion

Based on the experimental analysis, few insights have been inferred regarding the impact of the proposed VAE-DNN model in malicious URL detection. The adoption of OHE mechanism in converting URL data to numerical vectors significantly improve the performance of the model since all the possible characters required to form the URL were taken into consideration, and a two dimensional vector is formed based on the average length of the URLs and fixed number of possible characters. As we fixed 84 as the fixed size length of each row in the constructed matrix, each URL will be represented by L X 84 dimensional vector representation which is quite big to process for a neural network model. However, this technique creates a vector that allows space for all kinds of possible characters to be represented in the matrix which will be helpful for the neural network models to effectively extract inherent features and classify URLs optimally.

To analyse the impact of the VAE model in the final classification of URLs, we have experimented with different auto encoder models and observed the final results obtained by the classifier by adopting each and every AE model as a feature extractor. The results suggest that among all the AE models selected for the feature extraction process, the VAE model produced the best possible results in terms of various metrics, namely precision, recall, F1 score etc. Also, the VAE-DNN approach was compared against the traditional DNN classifier that does not employ feature extraction mechanism, and the results suggested that our approach delivers 5% more accuracy than the standalone classifier. To finalise the length of the latent space size of VAE, various experiments were conducted in accordance with the final loss obtained, and the optimal latent space size of 24 was fixed to reduce the dimensionality of the input features.

Our approach eradicates the complexity involved in manual feature engineering and reliance on third party services to extract certain features. Our model simply accepts raw URL data which can be preprocessed and dimensionally reduced abstract higher-level features of the URL that were easily extracted using VAE. Although our model performs better in terms of classification accuracy, when exposed to a random set of URLs after being trained, the response time of the model is not quite optimal since it roughly took around 2 s to respond to a URL, which should definitely be improved.

5 | CONCLUSION AND FUTURE WORK

To overcome the complexities involved in identifying whether a particular website is legitimate or not, our work explores a DL-based phishing detection mechanism that combines the

strength of two different neural network models to effectively detect the malicious nature of a particular URL. Inspired by the reconstruction ability of the autoencoder model that has the aptitude to learn better representation of the input data and at the same time could reduce the dimensionality of the input, we have adopted the variational autoencoder (VAE) model for the process of feature extraction. Since DL models only accommodate numerical vectors, the URLs which are actually in the form of string have been converted into numerical matrices by using OHE mechanism. Finally, to classify the URL data as either benign or malicious, a deep neural network (DNN) classifier has been used. Our work mainly employs the VAE model for extracting abstract higher-level features from the raw URL and DNN model for classifying the URL. Experimental results demonstrate that the proposed model achieves a maximum accuracy of 97.85%, which is significantly higher than all the other experimental DL models. The novelty involved in the proposed model is the automatic extraction of inherent features from the URL that reveals the character level inner relationship existing within the individual URL, which heavily impacts the performance of the classifier. Although the detection accuracy of the proposed model is quite higher, the model exhibits a quite higher FP rate of 2.19%, which needs to be reduced further. To overcome this issue, in our future work we have planned to incorporate a generative modelling technique, which will allow us to generate fake URLs that resembles the original URL and train the model with a dataset comprising of both original and generated URLs such that our model will further get exposed to different variants of the URL, which might help in reducing the false alarm rate of the model.

AUTHOR CONTRIBUTIONS

Manoj Kumar Prabakaran: Conceptualization; Formal analysis; Investigation; Methodology; Software; Writing – original draft; Writing – review & editing. **Parvathy Meenakshi Sundaram**: Project administration; Supervision. **Abinaya Devi Chandrasekar**: Formal analysis; Validation; Visualization.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

Data openly available in a public repository that issues datasets with DOIs.

ORCID

Manoj Kumar Prabakaran https://orcid.org/0000-0001-8814-0793
Parvathy Meenakshi Sundaram https://orcid.org/0000-0002-1600-9136
Abinaya Devi Chandrasekar https://orcid.org/0000-0002-3736-6386

REFERENCES

1. Moore, T., Clayton, R.: Examining the Impact of Website Take-Down on Phishing. [Online]. http://www.bankname.freehostsite.com/login
2. Xiao, X., et al.: CNN–MHSA: a Convolutional Neural Network and multi-head self-attention combined approach for detecting phishing websites. Neural Network. 125, 303–312 (2020). https://doi.org/10.1016/j.neunet.2020.02.013
3. Wardman, B.: Computer Law Commons, Defense and Security Studies Commons, Forensic Science and Technology Commons, Information Security Commons, National Security Law Commons, OS and Networks Commons, Other Computer Sciences Commons, and the Social Control, Law, Crime, and Deviance Commons Scholarly Commons Citation Scholarly Commons Citation Wardman, Brad (2016). [Online]. https://commons.erau.edu/adfsl/2016/thursday/2
4. Phishing Email Reports and Phishing Site Trends 4 Brand-Domain Pairs Measurement 5 Brands & Legitimate Entities Hijacked by Email Phishing Attacks 6 Use of Domain Names for Phishing 7-9 Phishing and Identity Theft in Brazil 10-11 Most Targeted Industry Sectors 12 APWG Phishing Trends Report Contributors 13 Phishing Activity Trends Report Unifying the Global Response to Cyber Crime. [Online]. http://www.apwg.org
5. Al-Qahtani, A.F., Stefano, C.: The COVID-19 scamdemic: a survey of phishing attacks and their countermeasures during COVID-19. IET Inf. Secur. 16(5), 324–345 (2022). https://doi.org/10.1049/ise2.12073
6. Whittaker, C., Google Inc, B. Ryner Google Inc, and M. Nazif Google Inc: Large-Scale Automatic Classification of Phishing Pages
7. Liang, B., et al.: Cracking classifiers for evasion: a case study on the google's phishing pages filter. In: 25th International World Wide Web Conference, WWW 2016, pp. 345–356 (2016). https://doi.org/10.1145/2872427.2883060
8. IEEE Staff IEEE Staff: IEEE International Conference on Intelligence and Security Informatics (2012)
9. Cui, Q., et al.: Tracking phishing attacks over time. In: 26th International World Wide Web Conference, WWW 2017, pp. 667–676 (2017). https://doi.org/10.1145/3038912.3052654
10. Tang, L., Mahmoud, Q.H.: A survey of machine learning-based solutions for phishing website detection. Mach. Learn. Knowl. Extr. 3(3), 672–694 (2021). https://doi.org/10.3390/make3030034
11. Alkawaz, M.H., et al.: A comprehensive survey on identification and analysis of phishing websites based on machine learning methods. In: ISCAIE 2021 - IEEE 11th Symposium on Computer Applications and Industrial Electronics, pp. 82–87 (2021). https://doi.org/10.1109/ISCAIE51753.2021.9431794
12. da Silva, C.M.R., Feitosa, E.L., Garcia, V.C.: Heuristic-based strategy for Phishing prediction: a survey of URL-based approach. Comput. Secur. 88, 101613 (2020). https://doi.org/10.1016/j.cose.2019.101613
13. Yang, P., Zhao, G., Zeng, P.: Phishing website detection based on multidimensional features driven by deep learning. IEEE Access 7, 15196–15209 (2019). https://doi.org/10.1109/ACCESS.2019.2892066
14. Bu, S.J., Cho, S.B.: Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing url detection. Electronics 10(12), 1492 (2021). https://doi.org/10.3390/electronics-tnqh_9;10121492
15. He, Z., Zhou, J.: Inference attacks on genomic data based on probabilistic graphical models. Big Data Mining and Analytics 3(3), 225–233 (2020). https://doi.org/10.26599/BDMA.2020.9020008
16. Zhong, W., Yu, N., Ai, C.: Applying big data based deep learning system to intrusion detection. Big Data Mining and Analytics 3(3), 181–195 (2020). https://doi.org/10.26599/BDMA.2020.9020003
17. Wang, L., et al.: Effective algorithms to detect stepping-stone intrusion by removing outliers of packet RTTs. Tsinghua Sci. Technol. 27(2), 432–442 (2022). https://doi.org/10.26599/TST.2021.9010041
18. Haghighat, M.H., Li, J.: Intrusion detection system using voting-based neural network. Tsinghua Sci. Technol. 26(4), 484–495 (2021). https://doi.org/10.26599/TST.2020.9010022
19. Harinahalli Lokesh, G., BoreGowda, G.: Phishing website detection based on effective machine learning approach. J. Cyber Secur. Tech. 5(1), 1–14 (2021). https://doi.org/10.1080/23742917.2020.1813396

20. Saleem Raja, A., Vinodini, R., Kavitha, A.: Lexical features based malicious URL detection using machine learning techniques. Mater. Today Proc. 47, 163–166 (2021). https://doi.org/10.1016/j.matpr.2021.04.041
21. Gupta, B.B., et al.: A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment. Comput. Commun. 175, 47–57 (2021). https://doi.org/10.1016/j.comcom.2021.04.023
22. Gandotra, E., Gupta, D.: Improving spoofed website detection using machine learning. Cybern. Syst. 52(2), 169–190 (2021). https://doi.org/10.1080/01969722.2020.1826659
23. Wang, W., et al.: PDRCNN: precise phishing detection with recurrent convolutional neural networks. Secur. Commun. Network. 2019, 1–15 (2019). https://doi.org/10.1155/2019/2595794
24. Ali, W., Ahmed, A.A.: Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting. IET Inf. Secur. 13(6), 659–669 (2019). https://doi.org/10.1049/iet-ifs.2019.0006
25. Yang, L., et al.: An improved ELM-based and data preprocessing integrated approach for phishing detection considering comprehensive features. Expert Syst. Appl. 165, 113863 (2021). https://doi.org/10.1016/j.eswa.2020.113863
26. Bu, S.J., Cho, S.B.: Deep character-level anomaly detection based on a convolutional autoencoder for zero-day phishing url detection. Electronics 10(12), 1492 (2021). https://doi.org/10.3390/electronics10121492
27. Choong, A.C.H., Lee, N.K.: Evaluation of convolutionary neural networks modeling of DNA sequences using ordinal versus one-hot encoding method. In: 2017 International Conference on Computer and Drone Applications (IConDA), pp. 60–65 (2017). https://doi.org/10.1109/ICONDA.2017.8270400
28. Berners-Lee, T., Masinter, L., McCahill, M.: Uniform Resource Locators (URL) (1994).20
29. Bank, D., Koenigstein, N., and Giryes, R.: Autoencoders. arXiv preprint arXiv:2003.05991 (2020)
30. Dayan, P., et al.: The Helmholtz machine. Neural Comput. 7(5), 889–904 (1995). https://doi.org/10.1162/neco.1995.7.5.889

How to cite this article: Prabakaran, M.K., Meenakshi Sundaram, P., Chandrasekar, A.D.: An enhanced deep learning-based phishing detection mechanism to effectively identify malicious URLs using variational autoencoders. IET Inf. Secur. 17(3), 423–440 (2023). https://doi.org/10.1049/ise2.12106