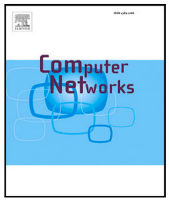
Computer Networks 206 (2022) 108826

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Efficient spam and phishing emails filtering based on deep learning

Safaa Magdy [a,*], Yasmine Abouelseoud [b], Mervat Mikhail [b]

[a] Computer Science, Software Engineer, Faculty of Engineering, Alexandria, Egypt
[b] Engineering Mathematics and Physics, Faculty of Engineering, Alexandria University, Alexandria 21544, Egypt

ARTICLE INFO

Keywords:
Ham
Spam
Phishing emails
Feature selection
SpamBase
Phishing corpus
Deep learning

ABSTRACT

Nowadays, spam emails represent a severe threat to security and cause a big waste in transmission time and users' time spent in browsing unsolicited bulk emails (UBE). This is in addition to a lot of bandwidth and large severs storage consumed by these spam emails, resulting in financial losses for organizations and annoying individual users. Another type of malicious emails is phishing emails, which aim to get sensitive information from users leading to credential theft. This forms a challenging threat in the cyberspace. Many machine learning (ML) approaches are used to classify emails as ham or spam emails. In this paper, a deep learning model is introduced that showed improvement in performance compared to state-of-the-art related studies. Three benchmark datasets are employed in our experiments, which include content-based features rather than text analysis techniques that may consume more time. Our classifier is used to discriminate between three classes for more general spam filtering. Different performance measures are used for model validation and testing. Moreover, the time consumed in both offline training and online detection stages is reported. The proposed classifier is designed with an eye on the validation accuracy achieving fast and competitive performance promoting its use in practical applications. A comparative study is presented to show that our work outperforms recent related studies.

1. Introduction

The volume of emails is growing rapidly as emails represent a primary, fast, and cheap communication tool in all fields. Accordingly, the need for more accurate spam filters has been raised. It is imperative to detect spam emails in near real time to have an effective and secure email filter. Phishing is a special form of social engineering attacks [1]. Phishing is designed to trick victims into entering their sensitive information, such as identity and financial-related data. Emails that have malicious attachments or redirecting URLs are common forms of phishing attacks. Thus, it becomes a security challenge to detect that threat. There are numerous types of phishing like bulk-phishing; where the attack is not especially targeted or tailored towards the recipient, spear-phishing; where the message is targeted at specific individuals or companies, clone-phishing; where the attackers send a legitimate email containing an attachment or link and replace it with a malicious version, and whaling; which focuses on senior individuals [2].

Spam filters can be implemented at all levels: firewalls of email servers, at mail transfer servers, and at email servers; where anti-spam tools support email protection at the network level [3]. Spam filters can be installed at the mail delivery agent. At the client level, the user can have personalized spam filters that detect such emails and automatically discard them. Famous internet service providers such as Yahoo and Google implemented machine learning approaches for email filtering, and continuously update their blacklists to avoid UBE threats efficiently [4].

Spreading malware through spam emails is a target of spammers in phishing attacks. Fake links are embedded in emails body that cause the users to be redirected to fake Web sites. In this attack, the fake links simulate well-known Web sites and thus seem less suspicious. Various methods have been implemented for spam detection such as blacklist-based methods, heuristic methods, metaheuristic methods and knowledge discovery methods such as data mining and machine learning methods [5].

In addition to application of machine learning techniques such as random forests (RF) and extra trees in spam filtering, artificial neural networks (ANNs) have been used to enhance the accuracy of these methods. The large number of features is a big challenge that slows down the learning process in ANNs, while decreasing the learning accuracy. To reduce the errors and long training time, feature selection techniques can be used such as information gain or Chi-square methods.

In this paper, a simple but efficient deep learning model is introduced for email classification. Different performance measures are considered and applied to construct a 3-classes email filter capable

* Corresponding author.
E-mail address: safaa_mgdy@yahoo.com (S. Magdy).

https://doi.org/10.1016/j.comnet.2022.108826
Received 31 March 2021; Received in revised form 12 August 2021; Accepted 29 January 2022
Available online 15 February 2022

of discriminating the emails classes, for further actions that can be done by the application that runs this classifier. We trained the neural network classifier using well-describing features extracted from benchmark datasets. The results show how that the performance of the neural network is sensitive to the feature selection approach used, focusing on both training accuracy and validation accuracy performance measures. Model selection approach is employed to decide on an effective neural network architecture.

Our contributions can be summarized as follows:

- Applying a well-tuned deep learning model to create a 3-fold classifier with performance outperforming that of other state-of-the-art studies.
- Extending the work in [4], where the authors experiments were based on machine learning approaches only and did not consider neural networks. Additionally, Chi-squared feature selection method has been considered.
- Showing the effect of several feature selection methods and reporting the effectiveness of such selectors on the classification performance.
- Using diverse benchmark datasets and testing the effect of training the proposed neural network based on each of them in terms of classification accuracy.
- Our experiments include a detailed description of the time consumed in each stage: feature extraction, model building and testing. All consumed test times have a maximum of 0.07856 s, which ensures that our classifier is real-time.
- A comparative study is conducted that gathers and analyzes numerous related recent studies, showing the competitiveness of the proposed approach.

The rest of the paper is organized as follows. A review of the related literature is provided in Section 2. The used datasets description and structure are discussed in Section 3. In Section 4, the proposed system architecture is introduced, showing the different stages in the model pipeline, starting from data preprocessing till reaching model validation. Moreover, feature extraction, feature selection, and deep neural network structure are included in this section. The experiments environment and performance evaluation measures are described in Section 5. Additionally, the obtained results are summarized, powered by a comparative study with recent papers in literature. Finally, Section 6 concludes our study and states the future work.

2. Related work

Spam is a serious issue that is not just annoying to the end-users, but also causes financial problems and raises security risks. Many researchers proposed machine learning algorithms to classify emails and to detect phishing attacks.

Several benchmark datasets are used in training spam email classifiers. SpamBase dataset [6] was used by [5,7–11]. In [8], authors analyzed SpamBase dataset using a data mining tool, searching for the best classifier and relevant features. Moreover, a model involving 24 machine learning classifiers was introduced in [10]. Authors in [7] examined Support Vector Machines (SVM) and local mixture SVM machine learning algorithms with variant kernel parameters, as well as Decision Trees and an ANN, showing the computational time and the limitations of each approach.

Using WEKA, authors in [9] employed 10 ML classifiers concluding that random forests is the best choice. More sophisticated approaches are used by the authors in [11], as they proposed a deep recurrent NN spam detector with accuracy 98.7%. Another artificial neural network approach is introduced by the authors in [5], where they used sine–cosine algorithm (SCA) as a feature selection method.

More datasets such as a combination of Ling-spam and Enron corpus have also been widely used as in [12,13]. Both studies employed a bag of words (BOW) to represent the email features, where the authors of [13] enriched their work by supporting artificial intelligence (AI) in a hybrid model of AI and BOW, achieving an accuracy of 98.27%. Furthermore, Spam Assissan and Nazario Phishing corpus are used in [14], where the authors employed an RF-based technique with information gain as a feature selector. They were able to reach a 99.1% accuracy.

Natural language processing (NLP) techniques are widely used in text analysis of email bodies, which can be applied to detect words existence and the corresponding weights of these words as features representing the emails in classifiers. In [15], the authors proposed an approach that employs NLP methods in text analysis for extracting email features to be fed into machine learning methods that detect malicious intents. In their approach, a topic blacklist is generated and a multinomial Naive Bayes (NB) classifier is used in phishing detection, achieving a precision of 95% using only semantic information. They trained their model using only 1000 ham emails and 1000 phishing ones, which may be insufficient for training. Moreover, their model depends only on semantic analysis of an email and does not consider neither sender/subject information nor phishing scripts, which are examined in our work.

As feature selection has a great impact on classification accuracy, Melvin et al. [16] introduced a compact feature representation using distributed memory(DM) and distributed bag of words (DBOW). They used the cosine similarity measure in computing distances between feature vectors. Their experiments were based on Enron and Trec07 datasets. They achieved an improvement in performance. Jungho et al. introduced in [17] a maleware classification method using an extended word2vec approach based on a long–short-term memory (LSTM) network [18]. Using word2vec is advantageous because it vectorizes data using fewer dimensions, thus reducing learning time compared to a one-shot encoding scheme.

Moreover, in the context of intrusion detection systems (IDS), a comparative study on feature selection methods for IDS was introduced by D. Selvamani et al. [19]. They explored feature selection methods based on Information Gain (IG), Gain Ratio, Symmetrical uncertainty, and Chi-square measures. They applied these techniques on datasets related to IDS, together with different classification methods, monitoring the performance. They concluded that IG achieved better results compared to other feature selection methods.

The concept of using optimization techniques in feature selection has been considered in literature, as the feature selection process aims to find the most relevant features according to some criteria. In [20], the authors introduced a hybrid approach that combines Radial Basis Function Neural Network (RBFNN) with Particle Swarm Optimization (PSO) algorithm for spam email filtering. They employed an RBFNN, as one of the most important types of artificial neural networks, for its simplicity and fast learning. PSO was used to find optimal centers of hidden neurons in the RBFNN. They improved the convergence and accuracy.

Additionally, in the context of metaheuristic optimization, the authors of [21] proposed a metaheuristic technique based on Binary Brain Storm Optimization for feature selection. They were able to achieve competitive results to the state-of-the-art methods. In [22], the authors proposed a novel hybrid supervised learning approach with a modified ML technique inspired from the human immunity system. Their algorithm used a probable spam word footprint scale to evaluate emails. A blocked-IP addresses database and frequently used spam confirmed keywords database were utilized to check emails. Their work increased the correct detection rate by about 5%.

However, using such metaheuristic optimization techniques in feature selection consumes a relatively long model-building time, where much time is spent in the process of exploring the features space to select the best features for a specific machine learning model.

Phishing is a cyber-crime, where phishers send bulk emails containing URLs and embedding links to convince the target user to visit their fake sites, misleading him to give his sensitive information. An

Enhanced Malicious URL Detection (EMUD) model was developed by S. Sankhwar et al. [2]. Their model focused on 14 relevant heuristics, which indicate whether the email under test is either a ham or phishing email. It is composed of two phases. In the first phase, EMUD model is applied to the URLs set to detect whether it is a phishing email by using the 14-URL heuristics. Next, in phase 2, a support vector machine (SVM) is used for classification. Their model accuracy is 93.01% that outperforms another competitive model for malicious URLs detection. They focused on malicious URLs and the discriminative features depending on URLs only without considering email body properties.

A. El Assal et al. [1] introduced a benchmark study for evaluating features in phishing detection applied to several datasets for URL, Web and email phishing, merged with legitimate emails from Enron dataset [23] and other datasets. Their objective was to investigate the benchmark framework with different robust and scalable detection methods. By changing the percentage of imbalance in the datasets, the authors concluded that imbalance affects the performance of the phishing classifier. They also studied how the training time changes according to the dataset size and machine learning technique applied.

It is noteworthy that, in most of the reviewed research in the field, researchers neither give enough attention to reporting the model building and testing times, nor they report the performance of their model for the testing dataset. However, in this paper, detailed investigation of the model building and testing times is provided. Additionally, the performance of our constructed model and its variants is reported for both the training and testing datasets.

3. Datasets

We used three benchmark datasets from UCI machine learning repository. The first is SpamBase [6], the second dataset is CSDMC2010 [24], and the third one is a combination of [4], which is constructed from ham-spam collection in SpamAssassin project, and phishing emails by Nazario [25]. One difference between these datasets is that SpamBase and CSDMC2010 have only two output classes distinguishing between ham and spam emails, while the Phishing_corpus is divided into three classes with phishing being added to the ham/spam classes. Another difference is that the first dataset is composed of features ready to be fed into the classification phase in the pipeline, while the second and the third datasets consist of a bulk of emails from which features can be extracted and ranked before proceeding with the classification step. Thus, CSDMC2010 and Phishing_corpus provide more freedom to the researcher in choosing a representative set of features to be extracted. Moreover, Phishing_corpus is further divided into three subsets for investigation purposes to enrich the research. The effect of data distribution among the different classes on the performance of the classifier is investigated and we demonstrate how the skewed data may reduce the classifier accuracy.

SpamBase is a benchmark dataset that contains 4601 mail messages, 1813 rows are classified as spam(39%) and 2788 as ham(61%). The dataset includes a set of 57 features extracted from each email. These features represent the frequencies of some discriminating words in the message body. It contains 48 features that are continuous (real numbers) representing the frequencies of specific words such as (address, remove, internet, order, mail, receive, free, business, credit, money, data, meeting, conference), other 6 continuous real features that indicate the frequencies of characters like (;, (, [, !,$,#), and features 55–57 measure the average, longest, and total length of sequences of consecutive capital letters. Finally, the class 0 or 1 for ham or spam emails; respectively.

The second dataset is CSDMC2010 spam corpus. It is one of the datasets employed in the data mining competition associated with the International Conference on Neural Information Processing(ICONIP) in 2010. This dataset contains 4327 email messages with 2949 ham emails (68.15%) and 1378 (31.85%) spam ones.

Table 1
Distribution of different classes in studied datasets.

Dataset	Ham	Spam	Phishing	Number of classes
SpamBase	2788	1813	–	2
CSDMC2010	2949	1378	–	2
Phishing_corpus DS1	2758	660	–	2
Phishing_corpus DS2	2758	–	2432	2
Phishing_corpus DS3	2758	660	2432	3

The third utilized dataset is the Phishing_corpus, which is constructed by merging emails from two benchmark datasets, SpamAssassin [26] to get ham/spam emails, and Nazario [25] to provide phishing emails. SpamAssassin dataset contains old phishing emails, which are close to spam. It has been used several times in literature [1]. The gathered phishing emails contain malicious links, malicious attachments, and phishing attempts. The raw email data resulting from this merge consists of ham, spam, and phishing emails as in [27].

The total number of rows in the Phishing_corpus is 5850, which are categorized as 2758 (47%) ham, 660 spam (11%), and 2432 phishing (42%). Three data subsets (DS1, DS2, and DS3) are constructed from the Phishing_corpus for investigation reasons. The distribution of each class in these subsets is shown in Table 1. The number of features selected to represent emails from Phishing_corpus and CSDMC2010 is 40, which can be divided into 5 groups as follows: body-based, subject-line-based, sender-address-based, url-based, and script-based features [27].

4. Proposed system architecture

Recently, artificial neural networks have been shown to succeed in detecting malware and in spam classification as in [28]. The strength of ANNs is in using one or a few number of hidden layers that produce a nonlinear model that captures difficult interactions between the input features and the target output class. However, ANNs require a relatively long training time to overcome the problem of being trapped in a local minimum. Deep learning increases the flexibility of the model design and accuracy over classical learning algorithms. It outperforms other solutions in many aspects, especially in extrapolation capability for new examples from a limited training dataset [29].

Our work investigates the power of applying neural networks with tuned hyperparameters for the purpose of accurate email classification as ham/spam or phishing. The training time and testing time is monitored as our target is to construct a real-time classifier. Moreover, we keep an eye on the validation accuracy to ensure the extrapolation capability of our developed classifier. Fig. 1 shows the steps and data flow in our proposed system. Preprocessing, features extraction and feature selection phases are essential to successful training of the neural network. In the learning phase, features of input data are fed to the network through the input layer, then passed through the hidden layers. Finally, the output layer predicts the corresponding email class. In the following subsections, each phase is described in detail.

4.1. Data preprocessing

As for the Phishing_corpus dataset, in the data preparation phase, the emails files – that are available in mbox and emb file formats – are read and each is transformed into an email object to be parsed and split into parts (header, address, body,..etc.) to be easily handled by the feature extraction functions. Mbox files may contain multiple messages that are captured through our extraction functions by traversing the email file. Next, all emails are put in a unified format, where each email is represented as a row in our features matrix. The target column (class) of the data row is filled with a value (1 or 2 or 3) according to whether the email is a ham or spam or phishing email; respectively. This is in accordance with the source of data; i.e. class 3 (phishing) is used as a

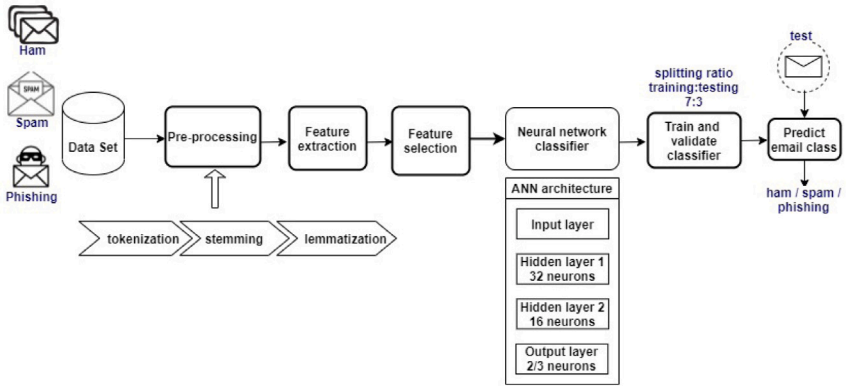


Fig. 1. Proposed system architecture for email classification.

label for files that come from Nazario dataset [25]. It is noteworthy that Nazario dataset is updated regularly to include new tricks of spammers and phishers.

Data preprocessing is done just after parsing the body part of the email. The purpose of this step is to get a standard canonical form from the raw text in the email. This can be done through tokenization, stemming and lemmatization. Tokenization is the process of breaking a stream of strings into pieces called tokens. Parsing is the process of text analysis for ease of interpretation and further processing. Stemming strips suffix and lemmitization converts a word to its base form. Putting all characters on small or capital case is called case folding. Spelling and typographical errors can be handled using similarity scoring to compare between intended words and actual spelling [4], and deciding if these two words are similar or not depends on a pre-defined threshold.

Regular expressions are used to extract different features from email body and subject. A regular expression (or regex) is a pattern that describes a certain portion of text mostly for the purpose of string matching [30]. Regular expressions are used in extracting URLs, links, domain names, and other features from the email body. A test email produced by [4] is used to examine the feature extraction functions. This test email has most of UBE characteristics.

4.2. Feature extraction

Feature engineering is the process of formulating the most discriminating features of an object [31]. It is the act of extracting features from raw data and transforming them into suitable formats for a machine learning model. Features engineering includes features extraction and features selection phases. It is a crucial step in the machine learning pipeline, as the relevant features can ease the difficulty of modeling, hence enable the system to output results of better quality.

The emails are represented by content-based features. Phishing_corpus and CSDMC2010 are represented based on the email features introduced and investigated by Toolan and Carthy [27]. These features as described in the previous section can be divided into five categories: body-based, subject line-based, sender address-based, URL-based, and script-based. A total of forty features have been considered in our experiments. Body-based features that we consider consist of 4 boolean variables as indicators of the presence or absence of (html-tags, forms, suspension word, verify_your_account word) in the body separately, and 5 numerical features related to the number of words/characters/other statistics of the body words besides the number of function words. Similarly, subject-line-based group of features include 2 boolean features indicating whether the email is a reply or forwarded message, 3 numerical features which are the number of words/characters and richness, and 3 features for checking the existence of verify/debit/bank words in the subject-line. Sender-address-based group of features involves 4 features (number of words/characters, two checks regarding sender's domain). URL-based group includes 13 features (indicator of the existence of the @ symbol, 'here' link, check if port $\neq$ 80, number of internal/external/image links, maximum number of periods, and others). Finally, script-based group contains 6 features (check if script/JavaScript/status-change script/popups/non-modal external JavaScript forms exist, number of on-click events). For more information about the functions that are used for extraction refer to [4].

Quantization and scaling (a.k.a. normalization) are some numeric feature engineering techniques that must be done to enhance the performance of the classification model.

4.3. Feature selection

Too many features leads to the overfitting problem, results in a more complex model, and longer processing times. Feature selection techniques remove non-useful features for model complexity reduction. However, feature selection is not about reducing the training time as some techniques increase the overall training time, but it aims to reduce the model scoring time.

Feature selection can be categorized into filter-based and wrapper-based methods. Wrapper-based category includes backward elimination method. Wrapper-based selection methods depend on trying to train the specified model by several combinations of features, and eliminate the worst features. Some classifiers can be used for features selection tasks like SVM and Radial Basis Function Network (RBFN) classifiers [32].

On the other hand, the univariant method and feature importance method belong to the first category. Tree-based classifiers like random forests can be used to determine the importance of the features; Feature Importance (FI) measure, through ranking the features with respect to their relevance in decision-making [33]. The univariant statistical filter is a best-k features selector based on a Chi-squared statistic. The effect of such filter is a significant decrease in training time and an improvement in accuracy. Feature selection methods that are employed in this study are Low variance, Principle Component Analysis (PCA), and Chi-squared (CHI) techniques, which are elaborated on in what follows.

4.3.1. Low variance

The variance is a measure for dispersion of the feature values about the corresponding means for the different email classes. It is computed as in Eq. (1)

$$Var(X) = \frac{1}{N-1}\sum(x_i - \bar{x})^2 \tag{1}$$

where X is the vector of the feature values for the dataset entries, $\bar{x}$ is the arithmetic mean of X. To use this measure for feature selection, the feature variance is compared to a certain threshold to determine which features have low variance and exclude them. A discriminating feature should show high variability across the different classes. The feature vector must be normalized before computing the variance. Thus, features values are reduced and redundancy is eliminated [34].

4.3.2. Principle component analysis

Principal component analysis is a dimensionality-reduction method that is widely used as a feature selection method. It transforms a large set of variables into a smaller one that preserves most of the information in the original set. There is a trade-off between dimensionality reduction and accuracy. Principal components are new variables that are linear combinations of the original variables. These combinations are formed in such a way that the new variables are uncorrelated and have the important information compressed into the first few components in the newly generated set [35]. The first step in PCA is to standardize the data to have the same scale. Then, a covariance matrix is computed in the second step. Thirdly, the eigenvectors and the eigenvalues of the covariance matrix are used to identify the principal components. Next, the eigenvectors are ordered according to the magnitude of the corresponding eigenvalues in descending order discarding those of low eigenvalues. Finally, a matrix of the eigenvectors with highest respective eigenvalues is used to compute the new features [36]. The number of principal components should be chosen such that a specified percentage of variance is retained.

4.3.3. Chi-squared test

Feature selection via Chi-squared test is another effective commonly used method [37]. It tests whether the feature under study and the class to which an email belongs are independent or not, according to a specific level of significance. If the test indicates that they are independent, then this feature should be discarded. Two hypotheses exist in the Chi-squared test: H_0 is the assumption that the feature and email class are independent, while the other hypothesis, H_1, is that the two are dependent. The Chi-squared test statistic is given by:

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{2}$$

where O_{ij} is the observed frequency of mutual occurrence and E_{ij} is the expected frequency that is asserted by the independence assumption H_0, r is the number of categories(bins) in the tested feature and c is the number of classes. r and c are the dimensions of the contingency table that contains the joint frequency distribution of the two variables (feature and class) which is used in probabilities calculations. The greater the value of the χ^2 statistic gives more evidence against H_0, hence, the more is the dependence between the tested feature and the corresponding class. Measuring this test statistic for each input feature and the target class feature, one can conclude the most relevant features and only select them in training the model.

4.4. Deep neural network construction

Algorithm 1 illustrates the steps of building the deep learning (DL) classifier, showing the data flow starting from extracting features from raw emails, till training and validating the neural network model.

Multilayer neural networks are more powerful than single layer ones. The number of neurons in both the input layer and the output layer are determined by the problem under study. In our classification model, the number of input layer neurons is equal to the number of extracted features, and the number of neurons in the output layer equals the number of target classes. Determining the number of hidden layers and the number of neurons in each hidden layer to properly fit the training data with adequate performance is the core of an ANN architecture.

According to Occam's Razor [38], the more complex is the model, the higher the possibility of errors and overfitting may occur. Thus, we start with one or two hidden layers. Most practical neural networks have 2 or 3 hidden layers and rarely have 4 ones. To construct a network, it is recommended to use a small number of parameters, hence, a small number of neurons must be included. Training should be stopped before overfitting occurs. In order to use early stopping

Result: Trained DL model
Input : emails dataset
Output: DL email classifier

Steps:

- Apply preprocessing steps.
- Extract emails features.
- Split data set into training and split data sets by ratio 7:3
- Initialize sequential deep learning model.
- Compile the classifier model.
- Fit the model by training set.
- Predict class of each email in test set and compare by actual target.
- Apply 10-fold cross-validation and retrain the model accordingly.
- Store the trained DL classifier.

Return: email classifier

Algorithm 1: DL model construction for email filtering

effectively, a clear stopping criterion must be defined. This crierion should be to stop when the error for the validation set goes up if more iterations are performed. Then, the training is stopped and the weights producing minimal errors on the validation set are finally stored. In a neural network, the activation function is responsible for transforming the summed weighted input to a node into the output by squashing the values into a suitable range. The rectified linear activation function (ReLU) is one of the widely used activation functions. Mathematically, it is defined as in Eq. (3). ReLU advantages are being computationally cheap and its use leads to fast convergence. When a hidden layer is fed by a range of input values, the ReLU function outputs more zeros, resulting in less neurons being activated and consequently this simplifies the model.

$$ReLU(x) = max(0, x) \tag{3}$$

SoftMax is a mathematical function that converts a vector of numbers into a vector of probabilities [39]. It is a generalization of logistic regression. It is often used as the activation function for the last layer of a multi-way classification network because the result could be interpreted as a probability distribution. To convert the probabilities back into an integer encoded class label, argmax function is used.

In our proposed model, the network architecture is composed of one input layer and two hidden layers. The optimization of the hyperparameters was done using a grid search as described in Algorithm 2 with 10-fold cross validation. Algorithm 2 shows in detail how we applied the grid search method to guide in selecting the best number of layers, the number of nodes in each layer and the learning rate. This procedure can take a long time, but it is done once during the offline training phase to get the simplest and most efficient network structure suitable for our training dataset by investigating combinations of selected hyperparameters ranges. The ANN architecture used for the Phishing_corpus and CSDMC2010 datasets is as follows: The input layer includes 40 neurons and the activation function is 'ReLU' for the hidden layers. The first and second hidden layers contain 32 and 16 nodes; respectively. The output layer in case of using datasets DS1, DS2 and CSDMC2010 contains 2 nodes (referring to two classes; ham and spam), while for the case of using dataset DS3 it contains 3 nodes (3 classes). The activation function of the last layer is 'softmax'. The neural network architecture employed in case of SpamBase dataset is quite similar. The input layer contains 57 neurons and the activation function is again 'ReLU' for the hidden neurons. The first and second hidden layers contain 32 and 16 nodes; respectively. The output layer contains 2 nodes. The activation function of the last layer is 'softmax'. It is noteworthy that the number of input neurons varies when feature selection methods are applied to our datasets.

4.5. Neural network training

The proposed deep-learning model uses supervised training for identifying malicious activities. Training is done through continuously updating the weights of the neural network nodes in each cycle till the resulting error is minimized. We used 'Adam' optimizer in the training phase to optimize the weights, and a learning rate equal to 0.001 is employed. Batch training is used and the batch size is set equal to 8 and the number of epochs is taken to be 200. The choice of the number of epochs is elaborated on in the results section.

4.5.1. Loss function

Neural networks are trained using an optimization process that requires a loss function to calculate the model error. Cross-entropy and mean squared error are the two main types of loss functions to use when training neural network models. The choice of the loss function is critical in defining the outputs and is application-dependent. Moreover, the loss function is directly related to the activation function used in the output layer of the neural network.

Categorical cross-entropy It is a loss function that is used in multi-way classification tasks, and designed to quantify the difference between two probability distributions. The categorical cross-entropy loss function is calculated as in Eq. (4)

$$Loss = -\sum_{i=1}^{c} y_i . \log \hat{y}_i \tag{4}$$

where $\hat{y}_i$ is the ith scalar value in the output, y_i is the corresponding target value in the supervised learning mode and c is the number of output classes. The minus sign ensures that the loss gets smaller when the distributions get closer to each other. The use of cross-entropy loss function improves the performance of models with sigmoid and softmax outputs, while mean squared error loss function may suffer from saturation and slow learning [40].

4.5.2. Optimization

We adopted "Adam" (Adaptive Moment Estimation) method in optimizing the weights while training our model. Adam optimization algorithm is an extension to stochastic gradient descent (SGD) that is widely used in deep learning applications. Stochastic gradient descent (SGD) maintains a single learning rate (α) for all weight updates and the learning rate does not change during training. Adam achieves the benefits of both SGD extensions; Adaptive Gradient (AdaGrad) and Root Mean Square Propagation (RMSProp) algorithms [41]. Instead of adapting the learning rates based on the average of the first moment of the gradients as in RMSProp, Adam uses the average of the second moments of the gradients which makes it converge faster. It is simple and computationally efficient showing rapid convergence. Mathematically, in gradient descent, the correction of a weight depends on the slope as in Eq. (5) [33]

$$Correction = \alpha \frac{\partial J}{\partial \theta_j} \tag{5}$$

where J is the cost function and θ is the weight parameter. Considering momentum, the correction with momentum is computed as Eq. (6)

$$Correction = \gamma * previous\ correction + \alpha \frac{\partial J}{\partial \theta_j} \tag{6}$$

where γ represents the ratio of the previous correction that should influence the current correction [33].

$$\theta_j = \theta_j - correction \tag{7}$$

Then, the value of θ_j is updated as in Eq. (7). The concept of momentum helps reaching the solution faster. The recommended value of γ starts with 0.5 and should be increased to 0.9 gradually [33].

4.5.3. Overfitting problem

There is always a gap between the performance of a model for the training dataset and that for the testing dataset, which is particularly large when the models are complex and the dataset is small. Fitting a model to a particular training dataset does not guarantee that it will provide good prediction performance on unseen test data. Thus, the model may not generalize well for new data. Early stopping is a common form of regularization to the cost function to avoid overfitting, in which the optimization technique is stopped after only a few iterations. One way to decide the stopping point is by holding out a part of the training data, and then testing the error of the model on the held-out set. The optimization algorithm is terminated when the error on the held-out set begins to rise. So, early stopping acts as a regularizer because it effectively restricts the parameter space.

4.6. Model validation

Model validation is the process of testing and choosing a model from candidate ones. One basic approach is using resampling techniques such as K-fold cross-validation. In K-fold cross validation, K refers to the number of groups that a given dataset is split into for sampling. In this method, the data is divided into K portions (or folds), where the data in the first fold is used as cross validation data and the data in the remaining K-1 folds is used as training data. Next, the data in the second fold is used as the cross validation set and the remaining data is used as the training set and so on, till the data in each fold has been considered for cross validation once [42]. Cross validation minimizes the effect of the random choice of the training and testing data and repeats K times the evaluation process to provide more accurate results.

Procedure: Tune DL Model()

Return: Model best hyperparameters

Begin:

- Initialize the model as a sequence of layers
- Loop through a specified range of layers (2:20)
  - Add a dense layer
  - Specify the number of nodes in that layer, try values from the set [16, 32, 64, 128, 256]
  - Set activation layer = 'ReLU'

  End loop
- Add output layer with nodes= number of classes, and activation function='softmax'
- Train the model using 'Adam' optimizer.
- Try learning rates [0.01, 0.001, 0.0001], Specify metric='accuracy'
- Get the best model from all trials.

End

Algorithm 2: DL model parameters tuning

5. Results

The simulation of our email classifier is implemented using Python open source language and WEKA tool for data analysis. WEKA open source framework is used for analyzing data and feature selection [43]. WEKA is widely used in data mining, data preprocessing, visualization and machine learning modeling. The execution time of our classifier is calculated using a personal computer running Ubuntu 20.04.1 LTS with CPU AMD A6, 2.6 GHz and with a 16 GB RAM. For training our neural network model, Google colab GPU is used instead of the local machine. We used TensorFlow(an end-to-end open source platform) as

Table 2
Some terminology.

Term	Description
n_i	Number of rows in dataset i, i = 1,2,3
b_t	Model building time
c_t	Email classification time
b_{tt}	Total building time
c_{tt}	Total classification time of an email
e_t	Features extraction time of an email

the backend of KERAS framework in deep learning network construction. Several performance metrics are measured in our experiments. Training and classification time is monitored. For accurate computation of the total time elapsed in each stage; offline and online, Eqs. (8) and (9) are used. Table 2 describes the meaning of each symbol used in Eqs. (8), (9).

$$b_{tt} = n_i * e_t + b_t \tag{8}$$

$$c_{tt} = e_t + c_t \tag{9}$$

Summarizing the previous Eqs. (8) and (9); 1- To obtain the building time, the total time for feature extraction of the whole dataset must be added to the corresponding model building time.

2- To obtain the total classification (testing) time, feature extraction time is added to the testing time of the corresponding model.

5.1. Performance evaluation measures

Measuring the correct number of classified emails is not sufficient because of the cost of UBE misclassification. So, other metrics that are derived from information retrieval and decision theory can aid to obtain better results [4]. When a ham email is misclassified as spam or phishing, this may lead to loss of this vital email, especially if the spam filter deletes suspicious emails automatically. While misclassifying phishing emails results in breaking of privacy.

Accuracy: This metric evaluates the average number of correctly classified emails over the whole email dataset as in Eq. (10).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{10}$$

where TP, TN, FP and FN are true positives, true negatives, false positives, and false negatives; respectively.

Precision: This metric indicates the reliability of the filter by measuring the proportion of true positive results to the total of positive results in the dataset as a whole. Its equation is (11)

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

Recall: Recall evaluates the classifier sensitivity. The formula of recall is given in Eq. (12)

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

F1-score: This metric reflects the balance between the precision and recall, as shown in Eq. (13).

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{13}$$

Matthew's correlation coefficient (MCC): It is a measure of the quality of multi-way (as well as binary) classification. It takes into account true and false positives and negatives and is generally regarded as a balanced measure that can be used even if the classes are of very different sizes [44]. The MCC value lies between −1 and +1. A coefficient of +1 represents perfect prediction, 0 an average random prediction and −1 an inverse prediction. It is computed as in .

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FN).(TP + FP).(TN + FN).(TN + FP)}} \tag{14}$$

MCC is preferred over accuracy and F1-score in case of imbalanced data [45], especially in binary classification.

Balanced detection ratio (BDR): This new metric, proposed by [46], is used to measure how many emails of minor class are detected correctly. Assuming that the positive class is the minor one, BDR and BDR% measures are calculated as in Eqs. (15) and (16); respectively.

$$BDR = \frac{TP}{1 + FP} \tag{15}$$

$$BDR\% = 100 * \frac{BDR}{TP + FN} \tag{16}$$

5.2. Proposed system results

Table 3 shows the different performance metrics of applying our ANN classifier to the SpamBase dataset with varying the number of epochs. It is clear that increasing the number of epochs causes a similar increment in the training time, while the accuracy value is fixed at a value of 0.9957(maximum value) and does not increase when the number of epochs is increased from 200 to 300. Fig. 2 depicts the effect of changing epochs number on the accuracy and loss values, indicating that 200 epochs is the best choice.

Similarly, Tables 4 and 5 show the effect of varying the number of epochs on the performance of our ANN model when applied to the Phishing_corpus dataset. In Table 5, accuracy/loss refers to training accuracy/loss, while Val. Acc./Loss refers to validation accuracy/loss. Validation accuracy measure is an important measure to be considered for model evaluation in addition to training accuracy. The big difference between them reflects learning errors like overfitting. It is clear that 200 epochs achieves the best compromise for the Phishing_corpus dataset as well. Thus, in what follows, we focus on 200-epochs in the remaining experiments results.

It is clear from Tables 3 and 5 that the testing time is sufficiently small to meet real-time requirement necessary in our application.

Table 6 illustrates the performance of applying different feature selection methods to the SpamBase dataset in training our ANN classifier. Compared to all-features results in Table 3, these features selection methods do not improve the performance significantly in case of predetermined number of epochs (200). The best accuracy is achieved when using LV-40 features selection method at the cost of increased building time.

To further extend our results, the performance of applying different feature selection methods to another two-class more recent dataset – the CSDMC2010 dataset – is shown in Table 7. Here, the use of PCA feature selection method with only 20 features yields an improvement in the training accuracy as well as the validation accuracy compared to all approaches shown in the table. The building time is slightly reduced compared to using all features. Using 20 features selected by PCA, the obtained results are summarized in Table 8. The accuracy increased by a slight value 0.008% approximately, while the building time increased by an average of 20 s (about 8%) when PCA features reduction method is used.

Table 9 shows the average performance measures for the case of using 30 features selected by the PCA method for the three Phishing_corpus data subsets. It is apparent that there is a slight increment in accuracy (0.011%) when 30 features are used, while the training time is approximately doubled compared to using all 40 features.

From Tables 9–11, it is clear that both PCA and LV feature selection methods consume approximately the same training time, which is about 1.8% of the original training time using all features. While Chi-squared (CHI) test method consumes less training(building) time (about 1.3% of the original building time). CHI method additionally succeeded in highly decreasing the validation loss to be a very small and negligible value. Also, a similar enhancement is achieved in the validation accuracy by the CHI method reaching a value of 0.9999.

For further elaboration, Figs. 3–4 visually summarize the results for the Phishing_corpus data subsets. Fig. 3-(a) shows that the MCC value of DS2 (the most balanced data subset) is the largest compared to DS1 and DS3 for all feature selection methods. CHI selection method

Table 3
Average performance metrics of applying our ANN classifier to the SpamBase dataset.

Epochs	SpamBase dataset (all features)							
	Accuracy	Precision	Recall	F1-score	MCC	Loss	Building time (s)	Testing time (s)
100	0.9905	0.9905	0.9905	0.9905	0.8537	0.0238	114.76452	0.06039
200	0.9957	0.9965	0.9965	0.9965	0.8427	0.0153	229.32218	0.05756
300	0.9950	0.9968	0.9968	0.9968	0.8519	0.0183	270.23723	0.06639

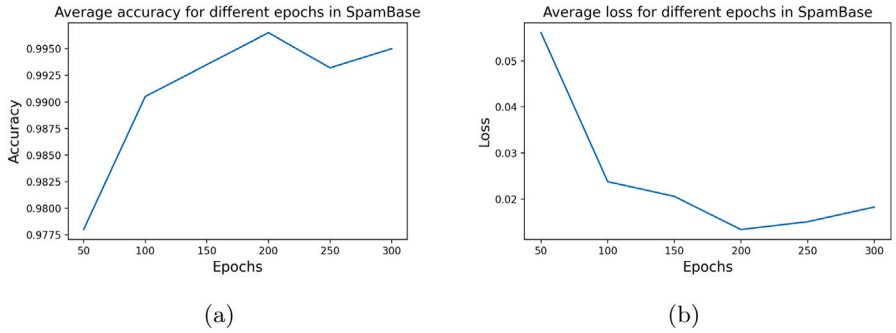


Fig. 2. The performance of our ANN classifier with changing the number of epochs for SpamBase dataset (a) accuracy (b) loss.

Table 4
Effect of changing the number of epochs on accuracy and building time when applying our ANN classifier to Phishing_corpus data subsets using all 40-features.

Dataset	100 epochs				300 epochs			
	Acc.	Loss	Building time (s)	Testing time (s)	Acc.	Loss	Building time (s)	Testing time (s)
DS1	0.9928	0.0193	58.73167	0.06509	0.9952	0.0128	311.15110	0.05084
DS2	0.9920	0.0195	105.44665	0.06681	0.9961	0.0111	505.08170	0.05158
DS3	0.9533	0.1213	114.13985	0.06077	0.9865	0.0487	577.21748	0.05291

Table 5
Average performance metrics of applying our ANN classifiers to the Phishing_corpus data subsets for 200 epochs and using all features.

DS	All features — 200 epochs									
	Acc.	Loss	Val. acc.	Val. loss	Prec.	Recall	F1-score	MCC	Building time (s)	Testing time (s)
DS1	0.9963	0.0065	0.960	0.3895	0.9965	0.9965	0.9965	0.8819	117.0894	0.0501
DS2	0.9982	0.0059	0.9859	0.2883	0.9965	0.9965	0.9965	0.9613	213.5986	0.0513
DS3	0.9891	0.0377	0.9561	0.6283	0.9860	0.9831	0.9844	0.8708	230.6510	0.0527

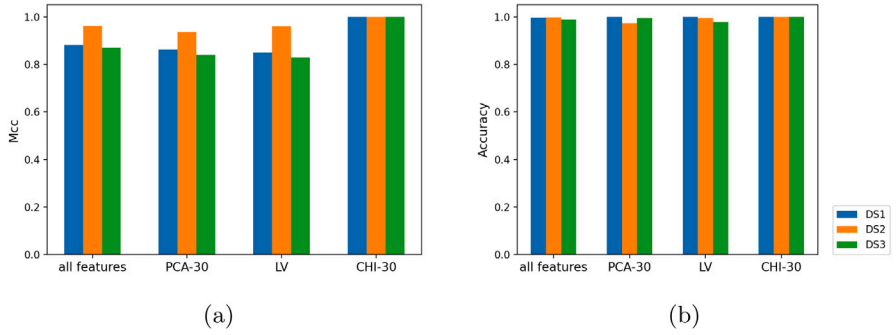


Fig. 3. The performance of different feature selection methods in Phishing_corpus data subsets using our ANN classifier with 200-epochs (a) MCC (b) Accuracy.

outperforms other feature selection methods in all measures, in addition to consuming the least training time. Figs. 5 and 6 show a slight increment in the MCC value and validation accuracy by applying the three investigated feature selection methods, whereas CHI method and LV method increased the training time by about 25% for SpamBase dataset. In Fig. 7, the BDR measure is investigated using all features for all datasets under study (SpamBase, Phishing_corpus three data subsets, and CSDMC2010 dataset). The BDR measure achieves its least values for DS3 data subset and CSDMC2010 dataset, while it achieves its highest value for DS1 data subset which is the most imbalanced dataset.

Early-stopping option in training the neural network means that the ANN training model monitors some specified measures like accuracy, loss, validation accuracy, and validation loss during learning and stops when the specified measure reaches an extreme value. In this situation,

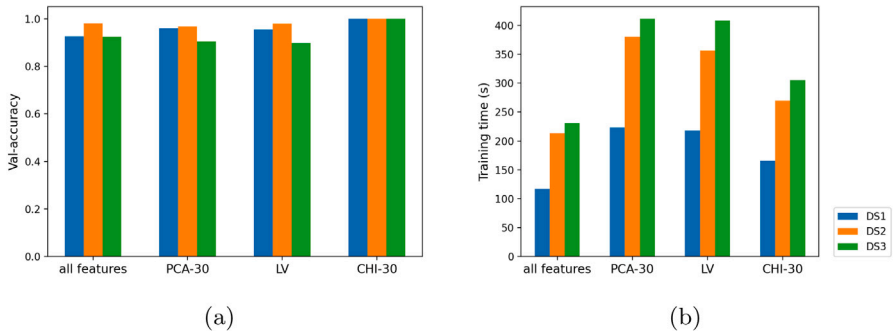


Fig. 4. The performance of different feature selection methods in Phishing_corpus data subsets using our ANN classifier with 200-epochs (a) Validation accuracy (b) Training time.

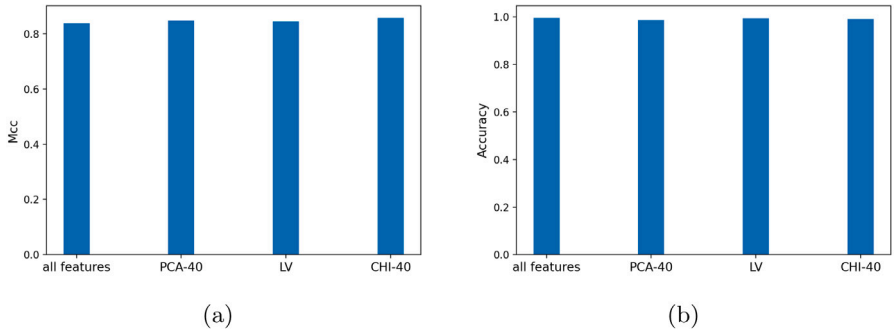


Fig. 5. The performance of different feature selection methods for SpamBase dataset using our ANN classifier with 200-epochs (a) MCC (b) Accuracy.

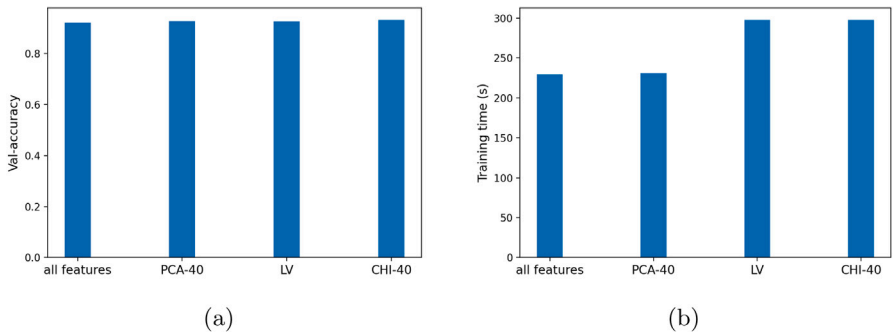


Fig. 6. The performance of different feature selection methods for SpamBase dataset using our ANN classifier with 200-epochs (a) Validation accuracy (b) Training time.

Table 6
Average performance metrics of applying our ANN classifier to the SpamBase dataset with CHI-30, CHI-40, PCA-30, PCA-40 , and LV-40 features selection methods.

Features	SpamBase — 200 epochs				
	Accuracy	Loss	Validation accuracy	Val-loss	Training time (s)
30-CHI features	0.9530	0.1123	0.8972	0.6053	300.7198
40-CHI features	0.9725	0.0625	0.9153	0.5148	295.3875
30-PCA features	0.9819	0.0446	0.9189	0.5436	215.2714
40-PCA features	0.9886	0.0348	0.9196	0.4977	216.5176
40-LV features	0.9936	0.0218	0.9276	0.6973	316.6261

Table 7
Average performance metrics of applying our ANN classifier to the CSDMC2010 dataset with all features, CHI-20, CHI-30, PCA-20, PCA-30 , and LV-20/30 features selection methods.

Features	CSDMC2010—200 epochs				
	Accuracy	Loss	Validation accuracy	Val-loss	Training time (s)
All features	0.9860	0.0392	0.8451	2.0721	382.6637
20-CHI features	0.9744	0.0614	0.8567	0.7957	341.41829
30-CHI features	0.9841	0.0402	0.8675	1.9581	382.69209
20-PCA features	0.9911	0.0282	0.8814	1.1021	361.89678
30-PCA features	0.9941	0.0159	0.8436	1.4374	342.30119
20-LV features	0.9720	0.0719	0.8575	0.6980	382.63426
30-LV features	0.9767	0.0547	0.8521	1.8308	382.62003

the epoch number is not pre-determined before starting the training process, instead, the optimum epoch number is automatically detected when the termination criterion is met. In the following tables, we show the effect of early stopping on enhancing the performance, with regard to the training time and the validation accuracy of the ANN model.

Table 12 shows the effect of applying early stopping criterion on average performance measures for the Phishing_corpus dataset, where the validation accuracy is monitored and used as a stopping criterion. It is apparent that the building time is significantly reduced but the accuracy decreased. In Fig. 8, several performance metrics of applying our ANN model to DS3 data subset of the Phishing_corpus are plotted against the epoch number. It is clear that the testing or validation loss, contrary to the training loss, increases after a small number of epochs.

Table 8
Average performance metrics of applying our ANN classifier to the Phishing_corpus data subsets using 20 features selected by the PCA method.

Dataset	PCA 20-features — 200 epochs					
	Accuracy	Loss	Val-Acc	Val-Loss	Building time (s)	Testing time (s)
DS1	0.9993	0.0019	0.9610	0.4224	153.5962	0.0586
DS2	0.9970	0.0058	0.9756	0.5841	254.3837	0.05326
DS3	0.9920	0.0274	0.9123	0.7627	274.6161	0.05932

Table 9
Average performance metrics of applying our ANN classifier on the Phishing_corpus data subsets using 30 features selected by the PCA method.

Dataset	PCA 30-features — 200 epochs					
	Accuracy	Loss	Val-Acc	Val-Loss	Building time (s)	Testing time (s)
DS1	0.9960	0.011	0.9630	0.4305	223.311	0.05237
DS2	0.9999	0.0034	0.9769	0.4183	380.179	0.05198
DS3	0.9941	0.0201	0.9162	1.0041	411.357	0.05563

Table 10
Average performance metrics of applying our ANN classifier to the Phishing_corpus data subsets for 30 features selected by the Chi-Squared test method.

Dataset	CHI 30-features — 200 epochs					
	Accuracy	Loss	Val-Acc	Val-Loss	Building time (s)	Testing time (s)
DS1	0.9999	0.00000001	0.9999	0.0000000267	165.7677	0.06729
DS2	0.9999	0.0000012	0.9999	0.0000000136	270.0492	0.05839
DS3	0.9999	0.000008	0.9994	0.00072385	304.8681	0.06816

Table 11
Average performance metrics of applying our ANN classifier to the Phishing_corpus data subsets for the Low Variance (LV) feature selection method with a threshold of 0.01.

Dataset	Low Variance 0.01 threshold — 200 epochs					
	Accuracy	Loss	Val-Acc	Val-Loss	Building time (s)	Testing time (s)
DS1(22 features)	0.9952	0.0175	0.9522	0.5780	217.7766	0.05275
DS2(21 features)	0.9973	0.0103	0.9788	0.3103	356.0573	0.05366
DS3(23 features)	0.9814	0.0481	0.9026	0.6574	408.1177	0.05802

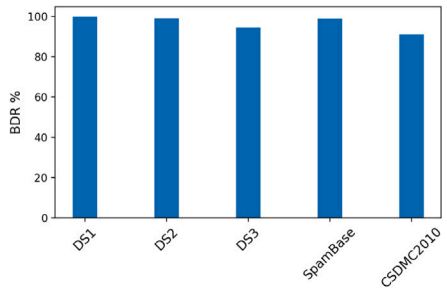


Fig. 7. Balanced detected rate (BDR) metric for all datasets.

Table 12
Average performance metrics of applying our ANN classifier to the Phishing_corpus dataset using all features with early-stopping applied.

Dataset	All features — early stopping					
	Accuracy	Loss	Val-Acc	Val-Loss	Building time (s)	Testing time (s)
DS1	0.9498	0.2325	0.9259	0.2325	7.00728	0.05328
DS2	0.9550	0.1081	0.9531	0.1227	9.71945	0.05439
DS3	0.9368	0.1771	0.9293	0.2038	10.16566	0.05576

In Table 13, the Chi-squared test feature selection method is applied with 30 features to the Phishing_corpus data subsets, and the

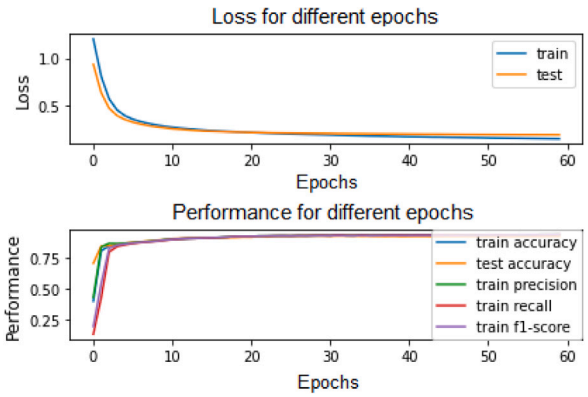


Fig. 8. Average performance metrics of training and testing applying suggested ANN model to DS3 dataset for different epochs.

Table 13
Average performance metrics of applying our ANN classifier on the Phishing_corpus dataset using 30 features selected by CHI method with early-stopping condition.

Dataset	CHI 30-features — early stopping					
	Accuracy	Loss	Val-Acc	Val-Loss	Building time (s)	Testing time (s)
DS1	0.9998	0.0012	0.9990	0.0023	3.66198	0.05704
DS2	0.9996	0.0021	0.9987	0.0060	6.26265	0.05461
DS3	0.9999	0.002	0.9989	0.0040	8.77136	0.05685

Table 14
Average performance metrics of applying our ANN classifier to the Phishing_corpus dataset with 30 features selected using PCA and applying early-stopping condition.

Dataset	PCA 30-features — early stopping					
	Accuracy	Loss	Val-Acc	Val-Loss	Building time (s)	Testing time (s)
DS1	0.9558	0.1199	0.9503	0.1693	6.939373	0.05427
DS2	0.9702	0.0772	0.9621	0.1048	12.28673	0.05587
DS3	0.9241	0.2325	0.9060	0.2951	16.28686	0.05741

early-stopping option is employed. Comparing Tables 12 and 13, it is noticed that the effect of applying the CHI feature selection method is decreasing the training time by about 50% and increasing the accuracy and validation accuracy by about 0.05%. The reason of such a significant drop in the training time is the rapid convergence to the optimum validation accuracy, in addition to an improvement in the accuracy reaching values in the range from 0.9996 to 0.9999. Figures 9–11 show that the training stopped after approximately one epoch, which explains the small training time(3.66–8.771 s). The 30-features selected by the CHI method ignored the 10 most irrelevant features, which in turn makes the ANN more simple, reaching the optimum model faster with the best accuracy for both the training set and the validation/testing set.

Table 14 shows the results in case of selecting 30-features by the PCA feature reduction method. Compared to the results in Table 12, an average increment of 0.01% in accuracy occurred after applying PCA feature selection method, but the training time increased by an average of 3 s in the three datasets. Table 15 presents the performance metrics of our ANN model when applied in conjunction with low variance feature selection method to the Phishing_corpus data subsets and using early stopping. The average accuracy is decreased by 0.02% after LV feature reduction with an average increment of 2.5 s in the building time.

On the other hand, when applying the early stopping condition and CHI selector to SpamBase dataset, the training and validation accuracy approximately become stable when the number of epochs exceeds 3. The same behavior occurs in the loss function for both the training and

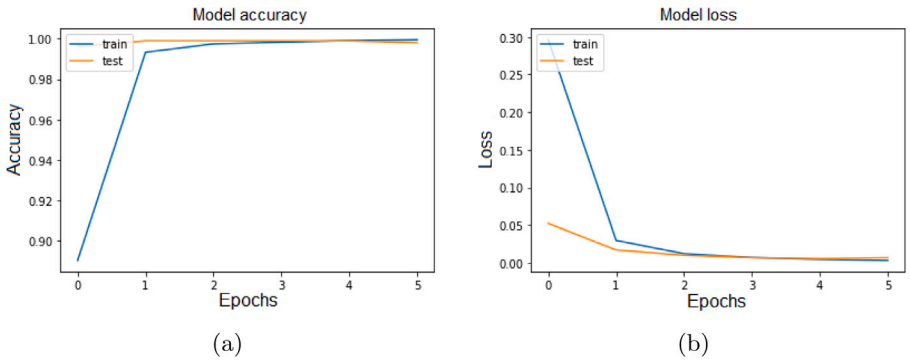


Fig. 9. The performance for the different number of epochs in DS1 dataset of Phishing_corpus with early stopping condition (a) accuracy (b) loss.

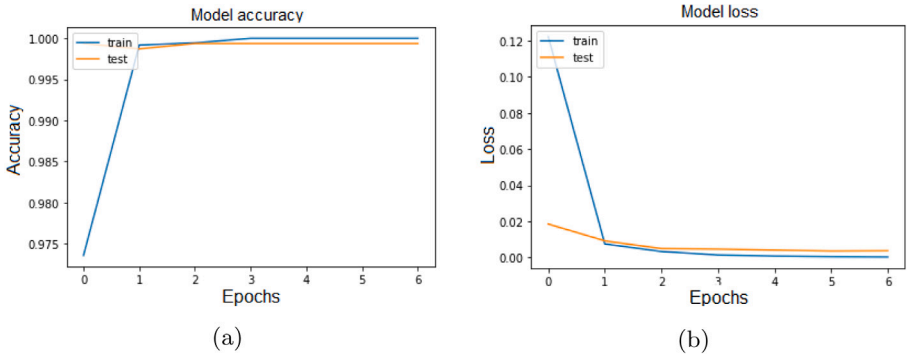


Fig. 10. The performance for the different number of epochs in DS2 dataset of Phishing corpus early stopping condition (a) accuracy (b) loss.

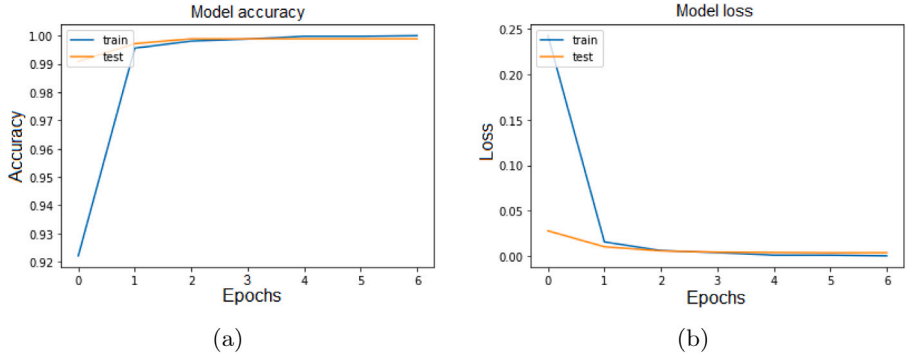


Fig. 11. The performance for the different number of epochs in DS3 dataset of Phishing corpus early stopping condition (a) accuracy (b) loss.

Table 15
Average performance metrics of applying our ANN classifier to the Phishing_corpus dataset with Low Variance feature selection method and applying early-stopping condition.

Dataset	LV - 0.01 threshold — early stopping					
	Accuracy	Loss	Val-Acc	Val-Loss	Building time (s)	Testing time (s)
DS1(22 features)	0.9332	0.1799	0.9201	0.2464	8.81448	0.05002
DS2(21 features)	0.9380	0.1695	0.9409	0.1630	7.11418	0.05615
DS3(23 features)	0.8750	0.3269	0.8758	0.3766	13.14045	0.05703

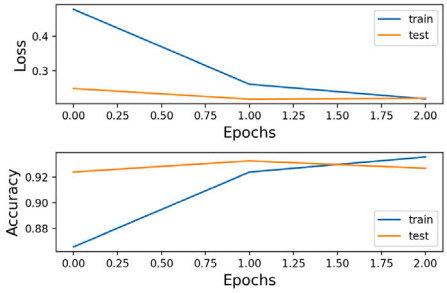


Fig. 12. The performance for different number of epoch in SpamBase with early stopping condition and 30 features selection using the CHI method.

the validation sets as depicted in Fig. 12. The top 10 features selected from the SpamBase dataset are displayed in Fig. 13.

For SpamBase, as clear from Table 16, in case of using only 30 features (about 50% of the original features), the PCA method achieves an accuracy exceeding that of the CHI method by 0.05%, but the training time increases by 2 s. For 40-features (about 75% of the original set), the accuracy of the PCA method exceeds that of the CHI method by 0.01%, and the training time consumed by the PCA method is less than that consumed by the CHI method by 2.6 s. As for the LV feature selection method, when we tried a threshold of 0.01, all

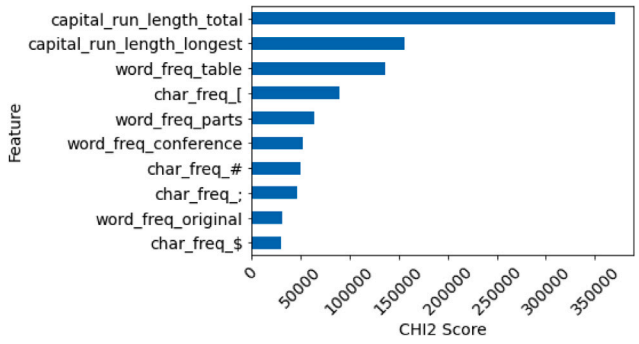


Fig. 13. Top 10 features ranked by the CHI feature selection method for SpamBase.

Table 16
Average performance metrics of applying our ANN on the SpamBase dataset with all features, CHI-30, CHI-40, PCA-30, PCA-40, and 40-LV feature selection and applying early-stopping condition.

Features	SpamBase — early stopping				
	Accuracy	Loss	Validation accuracy	Val-loss	Training time (s)
All features	0.9432	0.1578	0.9269	0.1953	5.5309
30-CHI features	0.8786	0.3055	0.8646	0.3230	3.2312
40-CHI features	0.9103	0.2457	0.9182	0.2535	6.6091
30-PCA features	0.9272	0.1934	0.9240	0.2119	5.2529
40-PCA features	0.9265	0.2161	0.9203	0.2078	4.0559
40-LV features	0.9245	0.2095	0.9138	0.2178	7.1335

57 features are selected and no features are eliminated. Thus, we tried another threshold 0.345 to get 40 features only; which have the highest variance. The performance of 40-LV features is close to that of 40-PCA features except that it consumes double the training time of the PCA method. In SpamBase, selecting 40 features is better than selecting 30 ones, as this means more information is captured. The selection percentage 75% achieves better results compared to 50%. Using all features in SpamBase for training gives the best accuracy and validation accuracy. Meanwhile, the training time just decreased by 1.5 s in case of selecting 40-PCA features.

The superiority of the results obtained for the Phising_corpus dataset over those for the SpamBase dataset can be attributed to the more careful selection of the set of features. For the Phising_corpus, the emails raw data is available and there is more freedom in the choice of the set of features to be extracted. As in Section 3, most SpamBase dataset features are binaries indicating the existence/absence of some spam keywords in the email, which are similar to blacklists that can be encapsulated in one feature indicating the number of spam keywords in the email body. Moreover, the features selected to represent the Phishing_corpus include distinct categories of features that are highly discriminative in spam detection, like the number of links as a URL-based feature. Phishing_corpus features properly analyze the email and capture the details of each email part (address, domain, body,.. etc.) that cover spam and phishing emails characteristics carefully.

5.3. A comparative study

To demonstrate the competitiveness of our proposed ANN-based email classifier, a comparison is provided in Tables 17–19 showing related recent studies results, whose experiments were conducted over the same or similar datasets.

In [7], the authors used Matlab ANN toolbox to implement a two-layer feedforward neural network to detect spam emails applied to the SpamBase dataset. They used the hyperbolic tangent function as an activation function for the hidden layer and a sigmoidal function for the output layer. They reported the results of varying the number of hidden layer neurons and the corresponding accuracy and time. They examined the effect of feature reduction to 7 features using information gain (IG). The stopping criterion was chosen to be the level at which accuracy dropped below 90%. This caused a notable decrease in accuracy to be 90.56%. Their best NN architecture included 5 nodes in the hidden layer achieving an accuracy of 94.02% within a training time of 70.81 s.

In [20], the authors achieved an accuracy of 93.1% when they employed a Radial Basis Function Neural Network (RBFNN) combined with Particle Swarm Optimization (PSO) as a hybrid classification approach. Furthermore, the authors in [9] used Weka to apply ten different machine learning classifiers to the SpamBase dataset. They applied the Infinite Latent Feature Selection (ILFS) method achieving a 95.4% accuracy using a Random Forest (RF) classifier.

Table 17
A comparative study on SpamBase dataset.

Paper	ML methods	Best results	Tools
[7]	SVM, LM-SVM, ANN and DT classifiers	ANN accuracy 0.94	MATLAB 2009b
[9]	10 classifiers	Random Forest accuracy 0.9545	Weka
[8]	Fishing filter for feature selection	Random forest accuracy 0.99	TANAGRA tool
[20]	Radial Basis Function NN and PSO	RBFPSO accuracy 0.93	Matlab
[10]	24 classifiers tested	Random committee accuracy 0.94.	Weka
[11]	5 ML classifiers and introduced recurrent NN	Their proposed approach accuracy 0.987	Python and Keras
[5]	6 ML classifiers and proposed feature selection for ANN	Their proposed approach accuracy 0.9792	Matlab
Our work	Including all features	ANN accuracy 0.9983	Python and Keras

Table 18
A comparative study on CSDMC2010 dataset.

Paper	ML methods	Best results	Tools
[47]	Linear SVM and AdaBoost	AdaBoost accuracy 0.9888	Python
[48]	Gaussian NB, MultiNomial NB, Linear SVM, RBF-SVM, LR by gradient descent, artificial bee colony with LR	Their proposed model (1000 features) accuracy 0.987	Python
[49]	Individual classifiers, bagging, and boosting ensemble approaches	The boosting approach accuracy 0.9891	Java
[50]	RF, Boosted RF, Bagged RF, SVM and NB	Bagged RF accuracy 0.9519	Java
[51]	NN, Firefly algorithm for feature selection and NB	Their proposed approach accuracy 0.97	Map reduce in hadoop
Our work	ANN	PCA-30 feature selector accuracy 0.9941	Python and Keras

Table 19
A comparative study on Phishing-related datasets.

Paper	Dataset	ML methods	Best results	Tools
[14]	Spam Assisan, Nazario phishing	information gain in feature selection	RF accuracy 0.991	Weka, Java
[12]	Ling-spam, Enron Corpus	SVM, Naïve Bayes using features from BOW and TF–IDF	SVM accuracy 0.93, BOW with TF–IDF accuracy 0.99	Python
[13]	Combination of Ling-spam and Enron Corpus	Hybrid model using AI and BOW	ANN accuracy 0.9827	Python
[15]	Enron+Nazario phishing	NLP and ML MultinomialNB	Their approach precision 0.95, recall 0.91	Python
Our work	SpamAssisan, Phishing_corpus	ANN	Avg. Accuracy 0.99946 , Avg. Precision 0.99366, Avg. Recall 0.9935	Python

R. K. Kumar et al. [8] applied 15 machine learning classifiers using TANAGRA [52] data mining tool and employed four feature selection methods. RF classification algorithm is applied to the reduced features using fisher filtering feature selection, achieving more than 99% accuracy in spam detection. Further researches on SpamBase dataset include the work in [10], where the authors used Weka to apply different machine learning algorithms. They concluded that using Random committee classifier achieved the best accuracy reaching 94.28%.

More recent papers that use SpamBase include [11], where the authors developed a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) with a Support Vector Machine (SVM) for spam email detection. CART algorithm is used in that study for feature selection, reducing to 15 features, they reported an accuracy of 98.7%. Sin-Cos Algorithm (SCA); a more sophisticated feature selection method, is applied in [5]. Using ANN, the authors reached an accuracy of 97.92%. They used Root-Mean-Squared-Error (RMSE) to compare between several machine learning classifiers. Random Forest (RF) was ranked second to their proposed SCA-based model.

From Table 17, it is clear that our proposed ANN classifier using all features achieves the best accuracy with the RF-based model in [8] being the closest competitor. Table 18 includes recent papers that used the CSDMC2010 dataset in their experiments in comparison with our proposed ANN model. In [49], AdaBoost achieved better accuracy compared to bagging of classifiers. Kause et al. in [47] also reported boosting approaches that achieved an accuracy close to that of [49] using meta-data only. So, their classifier is fast as it discards body-based features and consequently avoids the time complexity of text processing. Artificial Bee Colony (ABC) algorithm is used in tuning the ML model hyperparameters in [48], where the authors used this approach with linear logistic regression (LR) to search for the best hyperparameters. The authors experimented with a variant number of tokens (500 and 1000) for model training. The best accuracy is achieved by using 1000 tokens in their LR classifier. Shams et al. in [50] employed readability-features, that measure the errors in the text, along with traditional text-based features. Authors tested their model on four datasets using bagged RF classifier achieving an accuracy of 0.9519 for the CSDMS2010 dataset. In [51], spam filters are investigated in a distributed environment to benefit from parallel processing in increasing the efficiency of classification. The authors used the Firefly algorithm as a feature selector to explore the feature space to obtain the best discriminative ones according to the performance of their Naïve Bayesian (NB) classifier. They trained their model on the SpamBase and CSDMC2010 datasets, achieving an accuracy of 0.97 for the CSDMC2010 dataset. It is apparent from Table 18 that our proposed model when applied in conjunction with the PCA feature selection method achieves the best accuracy.

The comparison with papers that employed the Phishing_corpus dataset is summarized in Table 19. In [14], A. Yasin et al. introduced a classifier to determine the class of an email whether it is a ham or phishing email. Their proposed model utilized linguistic processing techniques to enhance the ability to detect similarity between emails using similar semantic term meaning and they used term-frequency–inverse-document-frequency (TF–IDF) concept to weight each phishing term. They used IG in feature selection and examined 5 machine learning classifiers. The best achieved accuracy was 99.1% by a RF in their proposed model. M. A. Hassan in [12] highlighted the importance of the correct coupling of the choice of the feature extraction method and the corresponding classifier. They used text-based features and bag-of-words (BOW) model with TF–IDF in building six machine learning classifiers. The resulting class is either a ham or a spam and the best results obtained were 93% and 99% for BOW with TF–IDF and SVM, respectively. S. Douzi et al. [13] introduced a hybrid approach for spam filtering based on a neural network model with Paragraph Vector Distributed Memory (PV-DM), where PV-DM is used to represent emails context compactly besides pertinent features. They proved that the proposed model is more resistant to differences in the language system and message cohesion. Their proposed model achieved a 98.27% accuracy. In [15], T. Peng et al. suggested a phishing classifier that employs natural language processing and a machine learning classifier, performing semantic analysis of the email text. Their approach determines whether the nature of the sentence is a question or a command using topic blacklist to detect malicious emails. They used a multinomial NB machine learning approach to generate the topic blacklist as commonly used in text classification. The output class is phishing or non-phishing with a precision of 95%. To sum up, using our ANN model with only 4 layers, we achieved competitive performance compared to the state-of-the-art models. Further improvement is achieved by using the Chi-squared feature selection method on the Phishing_corpus, with only 30 features and applying the early stopping option during training. The best accuracy is 0.9999 and the building time is 0.877 s for DS3 with 0.9989 validation accuracy.

6. Conclusions

Discriminating between spam and non-spam emails is a great challenge for email service providers and users as well. Several state-of-the-art research papers study the performance of email classifiers and how to optimize it. As our comparative study showed, the results of our work are superior to those obtained in the investigated studies in terms of accuracy and consumed time. Most of email classifiers focus on ham/spam or ham/phishing classes, whereas our classifier supports 3-fold classes ham or spam or phishing. We implemented a simple

and accurate predictive ANN model that consists of only two hidden layers and the training time is acceptable, as it is done once offline. While the testing time is fast enough to be applicable in email server providers and spam detector tools. The time consumed is reported for the different phases in our system (extraction, training, and testing). Our approach is applied to three standard datasets that represent real world samples of UBE emails to actually avoid misclassification that can cause severe problems like losing important emails or exposure to phishing attacks that aim to steal users' information. Careful selection of hand-crafted features to be extracted from the raw email data available in the Phishing_corpus dataset and the CSDMC2010 dataset assisted in obtaining our competitive results.

For future work, the proposed model can be further improved by adopting a mechanism for enhanced linguistic processing that combines such text-based features with existing content-based ones. Moreover, more sophisticated features selection methods, like auto-encoder in ANNs, can be used such that better classification results can be obtained.

CRediT authorship contribution statement

Safaa Magdy: Conceptualization, Investigation, Methodology, Software, Writing – original draft, Writing – review & editing, Visualization. **Yasmine Abouelseoud:** Conceptualization, Investigation, Methodology, Validation, Supervision, Writing – review & editing. **Mervat Mikhail:** Conceptualization, Supervision, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. El Aassal, S. Baki, A. Das, R.M. Verma, An in-depth benchmarking and evaluation of phishing detection research for security needs, IEEE Access 8 (2020) 22170–22192.

[2] S. Sankhwar, D. Pandey, R. Khan, Email phishing: an enhanced classification model to detect malicious URLs, EAI Endorsed Trans. Scalable Inf. Syst. 6 (21) (2019).

[3] V. Christina, S. Karpagavalli, G. Suganya, Email spam filtering using supervised machine learning techniques, Int. J. Comput. Sci. Eng. (IJCSE) 2 (09) (2010) 3126–3129.

[4] T. Gangavarapu, C. Jaidhar, B. Chanduka, Applicability of machine learning in spam and phishing email filtering: review and approaches, Artif. Intell. Rev. (2020) 1–63.

[5] R.T. Pashiri, Y. Rostami, M. Mahrami, Spam detection through feature selection using artificial neural network and sine–cosine algorithm, Math. Sci. (2020).

[6] M. Hopkins, E. Reeber, G. Forman, J. Suermondt, Spambase dataset, Hewlett-Packard Labs, 1999, https://archive.ics.uci.edu/ml/datasets/spambase.

[7] S.A. Saab, N. Mitri, M. Awad, Ham or spam? A comparative study for some content-based classification algorithms for email filtering, in: MELECON 2014-2014 17th IEEE Mediterranean Electrotechnical Conference, IEEE, 2014, pp. 339–343.

[8] R.K. Kumar, G. Poonkuzhali, P. Sudhakar, Comparative study on email spam classifier using data mining techniques, in: Proceedings Of The International MultiConference Of Engineers And Computer Scientists, Vol. 1, 2012, pp. 14–16.

[9] M. Bassiouni, M. Ali, E. El-Dahshan, Ham and spam e-mails classification using machine learning techniques, J. Appl. Secur. Res. 13 (3) (2018) 315–331.

[10] S. Sharma, A. Arora, Adaptive approach for spam detection, Int. J. Comput. Sci. Iss. (IJCSI) 10 (4) (2013) 23.

[11] M. ALAUTHMAN, Botnet spam E-mail detection using deep recurrent neural network, Int. J. 8 (5) (2020).

[12] M.A. Hassan, N. Mtetwa, Feature extraction and classification of spam emails, in: 2018 5th International Conference On Soft Computing & Machine Intelligence, ISCMI, IEEE, 2018, pp. 93–98.

[13] S. Douzi, F.A. AlShahwan, M. Lemoudden, B. El Ouahidi, Hybrid email spam detection model using artificial intelligence, Int. J. Mach. Learn. Comput. 10 (2) (2020).

[14] A. Yasin, A. Abuhasan, An intelligent classification model for phishing email detection, 2016, arXiv preprint arXiv:1608.02196.

[15] T. Peng, I. Harris, Y. Sawa, Detecting phishing attacks using natural language processing and machine learning, in: 2018 Ieee 12th International Conference On Semantic Computing, Icsc, IEEE, 2018, pp. 300–301.

[16] M. Diale, T. Celik, C. Van Der Walt, Unsupervised feature learning for spam email filtering, Comput. Electr. Eng. 74 (2019) 89–104.

[17] J. Kang, S. Jang, S. Li, Y.-S. Jeong, Y. Sung, Long short-term memory-based malware classification method for information security, Comput. Electr. Eng. 77 (2019) 366–375.

[18] Y. Goldberg, O. Levy, Word2vec explained: deriving Mikolov others, negative-sampling word-embedding method, 2014, arXiv preprint arXiv:1402.3722.

[19] D. Selvamani, V. Selvi, A comparative study on the feature selection techniques for intrusion detection system, Asian J. Comput. Sci. Technol. 8 (1) (2019) 42–47.

[20] M.A.M. Foqaha, Email spam classification using hybrid approach of RBF neural network and particle swarm optimization, Int. J. Netw. Secur. Appl. 8 (4) (2016) 17–28.

[21] J.P. Papa, G.H. Rosa, A.N. de Souza, L.C. Afonso, Feature selection through binary brain storm optimization, Comput. Electr. Eng. 72 (2018) 468–481.

[22] A.J. Saleh, A. Karim, B. Shanmugam, S. Azam, K. Kannoorpatti, M. Jonkman, F.D. Boer, An intelligent spam detection model based on artificial immune system, Information 10 (6) (2019) 209.

[23] C. Project, Enron spam datasets.

[24] CSDMC2010 spam corpus, in: International Conference on Neural Information Processing, ICONIP, 2010, https://gitlab.koehn.com/bkoehn/deep-spam/tree/69b93eff3885ba19474a115d43e621f421754cdd/CSDMC2010_SPAM.

[25] J. Nazario, The online phishing corpus, 2018, https://monkey.org/~jose/phishing/.

[26] SpamAssassin dataset, 2003, https://www.kaggle.com/beatoa/spamassassin-public-corpus.

[27] F. Toolan, J. Carthy, Feature selection for spam and phishing detection, in: 2010 ECrime Researchers Summit, IEEE, 2010, pp. 1–12.

[28] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, Survey of intrusion detection systems: techniques, datasets and challenges, Cybersecurity 2 (1) (2019) 20.

[29] G. Thamilarasu, S. Chawla, Towards deep-learning-driven intrusion detection for the internet of things, Sensors 19 (9) (2019) 1977.

[30] A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, M. Alazab, A comprehensive survey for intelligent spam email detection, IEEE Access 7 (2019) 168261–168295.

[31] A. Zheng, A. Casari, Feature Engineering For Machine Learning: Principles And Techniques For Data Scientists, " O'Reilly Media, Inc.", 2018.

[32] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Comput. Electr. Eng. 40 (1) (2014) 16–28.

[33] G. Rebala, A. Ravi, S. Churiwala, An Introduction to Machine Learning, Springer, 2019.

[34] S. Das, U.M. Cakmak, Hands-On Automated Machine Learning: A Beginner's Guide to Building Automated Machine Learning Systems Using AutoML And Python, Packt Publishing Ltd, 2018.

[35] D.J. Hand, Principles of data mining, Drug Saf. 30 (7) (2007) 621–622.

[36] Z. Jaadi, A step-by-step explanation of principal component analysis, 2019, https://builtin.com/data-science/step-step-explanation-principal-component-analysis.

[37] I.S. Thaseen, C.A. Kumar, Intrusion detection model using fusion of chi-square feature selection and multi class SVM, J. King Saud Univ.-Comput. Inf. Sci. 29 (4) (2017) 462–472.

[38] P. Domingos, The role of Occam's razor in knowledge discovery, Data Min. Knowl. Discov. 3 (4) (1999) 409–425.

[39] C.C. Aggarwal, et al., Neural networks and deep learning, Springer (2018) 1–52.

[40] J. Brownlee, Better Deep Learning: Train Faster, Reduce Overfitting, And Make Better Predictions, Machine Learning Mastery, 2018.

[41] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[42] G. Rebala, A. Ravi, S. Churiwala, Machine learning definition and basics, in: An Introduction to Machine Learning, Springer, 2019, pp. 1–17.

[43] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, ACM SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.

[44] M. Bekkar, H.K. Djemaa, T.A. Alitouche, Evaluation measures for models assessment over imbalanced data sets, J. Inf. Eng. Appl. 3 (10) (2013).

[45] D. Chicco, G. Jurman, The advantages of the matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, BMC Genomics 21 (1) (2020) 1–13.

[46] A. Aassal, L. Moraes, S. Baki, A. Das, R. Verma, Anti-phishing pilot at ACM IWSPA 2018: Evaluating performance with new metrics for unbalanced datasets, in: Proc. IWSPA-AP Anti Phishing Shared Task Pilot 4th ACM IWSPA, 2018, pp. 2–10.

[47] T. Krause, R. Uetz, T. Kretschmann, Recognizing email spam from meta data only, in: 2019 IEEE Conference On Communications And Network Security, CNS, IEEE, 2019, pp. 178–186.

[48] B.K. Dedeturk, B. Akay, Spam filtering using a logistic regression model trained by an artificial bee colony algorithm, Appl. Soft Comput. 91 (2020) 106229.
[49] L. Rokach, Ensemble Learning: Pattern Classification Using Ensemble Methods, World Scientific, 2019.
[50] R. Shams, R.E. Mercer, Classifying spam emails using text and readability features, in: 2013 IEEE 13th International Conference On Data Mining, IEEE, 2013, pp. 657–666.
[51] K.R. Dhanaraj, V. Palaniswami, Firefly and Bayes classifier for email spam classification in a distributed environment, Aust. J. Basic Appl. Sci. 8 (17) (2014) 118–130.
[52] Tanagra - a free data mining software for teaching and research, http://eric.univ-lyon2.fr/~ricco/tanagra/en/tanagra.html.

Safaa Magdy: received her B.Sc. degree in Computer Science and Automatic Control from Faculty of Engineering, Alexandria University in 1993. She obtained her M.Sc. degree in Engineering Mathematics from Alexandria University in 2019. Her research interests include database management, security, big data, data mining, neural networks, and machine learning. Currently, she aims to receive her Ph.D. degree in the field of intrusion detection, artificial neural networks and deep learning.

Yasmine Abouelseoud: received her B.Sc. in Computer Science, M.Sc. and Ph.D. degrees in Engineering Mathematics from Alexandria University in Egypt in 2001, 2005, 2008, respectively. She is currently a Professor at the Engineering Mathematics department, Faculty of Engineering, Alexandria University. Her research interests include information security, signal processing and engineering optimization with over forty publications in these fields.

Mervat Mikhail: received her B.Sc. in Computer Science, M.Sc. and Ph.D. degrees in Engineering Mathematics from Alexandria University in Egypt in 2002, 2008, 2016, respectively. She is currently a lecturer at the Engineering Mathematics department, Faculty of Engineering, Alexandria University. Her research interests include computer graphics, information security and image processing.