WJCMS

Wasit Journal of Computer and Mathematics Science

Journal Homepage: https://wjcm.uowasit.edu.iq/index.php/WJCM
e-ISSN: 2788-5879 p-ISSN: 2788-5887

# Email Spam Detection Using a Hybrid Approach of Feedforward Neural Network and Penguin Optimization Algorithm

**Layth Saad Hwaidi Al-busultan**[1]*

[1] Wasit, 52007, IRAQ

*Corresponding Author: Layth Saad Hwaidi Al-busultan

DOI: https://doi.org/10.31185/wjcms.282
Received 11 Augest 2024; Accepted 27 September 2024; Available online 30 September 2024

**ABSTRACT:** The electronic communication of today is beset by unwanted junk messages. These are not just a nuisance; they waste our time and energy and clog up our inboxes, which are supposed to be, in many ways, our digital front doors. But how do we keep them from coming? Well, as today's article makes clear, the situation is hardly straightforward. There are various methods that can be used, both traditional and modern, and each has its own set of pros and cons. One method, however, stands out in today's article as one that we think might work well in practice, the good old-fashioned neural network.

The method's novelty derives from its use of neural networks, a type of machine learning, in conjunction with the POA. The POA is a powerful optimization algorithm that can perform very well when fine-tuning the weights and biases of a neural network. We therefore used the POA to optimize these two crucial components of our model and achieved a surprising level of accuracy—98.7%. This figure surpasses the next nearest competitor by 2% and demonstrates the efficacy of our method in spam detection while also highlighting the POA's potential in this field.

The overall proposed method pushes the spam-detection field forward, but it also demonstrates something even broader: the power of hybrid approaches in machine learning. Neural networks are great—but you have to train them properly, and the use of a particle swarm for that purpose is a neat idea. But in the end, a hybrid spam filter is still a spam filter, and this one does what it's supposed to do better than many rivaling methods.

Keywords: Spam detection, machine learning, feedforward neural network, Penguin Optimization Algorithm, email classification.

## 1. INTRODUCTION

In contemporary society, email is an omnipresent and vital tool of communication. It spans almost every domain imaginable, from business to academics to personal affairs. Yet this cornucopia of electronic mail has also brought forth a veritable abomination: the spam email [1]. Unsolicited and sometimes downright malicious, spam is an affront to the dignity of email and an affront, too, on occasion, to the email user [2].

The electronic communications management field has a vital responsibility: to filter out spam emails while letting through the legitimate ones [3]. Despite the effectiveness of traditional rule-based approaches and blacklists, they are simply not enough; spammers have long since figured out how to bypass such controls. In place of these increasingly primitive-seeming tactics, the electronic communications management field has turned to more sophisticated methods of spam detection, particularly those relying on machine learning and artificial intelligence [4].

Earlier investigations into email spam detection have predominantly centered on neural networks and statistical methods [5]. Although these techniques have yielded some reasonable outcomes, the results still fall short of meeting the demands of industry standards in terms of accuracy and efficiency. This study aims to enhance email spam detection performance by proposing a hybrid method that pairs a neural network with an optimization algorithm [6]. Despite its name, the Penguin Optimization Algorithm—inspired by penguins swimming in the ocean—has been shown to be effective on par with other foundational algorithms [7].

This work takes a fresh approach by using the Penguin Optimization Algorithm (POA) to boost the accuracy of a feedforward neural network in detecting spam emails. While feedforward neural networks have been used for spam detection before, they often fall short because their parameters—like weights and biases—aren't properly fine-tuned. This step is essential but has been overlooked in many past efforts.

In this study, POA steps in to tackle this issue, optimizing the network's parameters to create a system that outperforms earlier models in both accuracy and efficiency. The main goal here is to develop a hybrid approach that combines POA with a feedforward neural network, pushing the limits of spam detection accuracy. Beyond that, the study also compares this new method to existing ones, looking closely at how POA's optimization capabilities enhance the network's overall performance. By doing so, this research aims to offer a more accurate and reliable way to detect spam, improving on what's currently available.

## 2. RELATED WORK

The topic of spam email detection has been the subject of intense research and deliberation among experts in the management of electronic communications [8]. A great many methods have been investigated and proposed, ranging from the most elemental to those of a more sophisticated and certainly more advanced nature. The earliest systems employed a basic if-then logic based on a limited number of rules [9]. For instance, some systems simply looked for a limited number of words or phrases that directly indicated potential spam. They also might have looked at the subject line of the incoming message and made a judgment based on its content.

Researchers have looked at statistical methods for spam detection to overcome the limitations of rule-based systems [10]. Spam filtering—Bayesian in nature—has been applied with good success. A spam filter is a program of sorts that uses certain features extracted from messages to ascertain the probability of a message being spam. If the message has a certain word (or words) or pattern (or patterns) that the spam filter understands as a feature common to spam messages, the message is not delivered [11].

In "Beyond Rule-Based Email Filtering," the latest development in artificial intelligence, neural networks, has the potential to improve filtering by using the same kinds of features that human beings use when they're trying to distinguish between spam and legitimate messages [12].

Spam detection has benefited from the use of neural networks, but their performance is still lacking, particularly in the areas of proficiency and speed [13]. One possible way to enhance neural network performance is through the coupling of an optimization algorithm, which works by adjusting a set of tunable parameters (e.g., weights, biases) in the neural network to achieve better results, with the network itself [14].

The Penguin Optimization Algorithm (POA), which is a relatively new arrival on the optimization heuristic scene and draws inspiration from the foraging behavior of penguins, has obtained some cutting-edge results in various applications [15]. It has been applied to function approximation, scheduling, clustering problems, and others. Spam detection, however, has not been one of the applications for which the POA has been used [16].

This study attempts to fill the existing research gap by putting forth a new method. The proposed method is a hybrid one and consists of a feedforward neural network and a recently proposed optimization algorithm—the Particle Swarm Optimization Algorithm—for email spam detection. The packed method is expected to yield a high performance in terms of detection accuracy because both the artificial neural network and the algorithm both have good merit for that.

## 3. PROPOSED METHODOLOGY

The proposed method for email spam detection utilizes a hybrid approach that integrates a Feedforward Neural Network (FNN) and the Penguin Optimization Algorithm (POA) to optimize the network's parameters. This combination aims to improve classification accuracy by efficiently tuning the weights and biases of the neural network.

The steps of the proposed system are outlined as follows, with detailed explanations supported by pseudocode and mathematical formulations:

### 1. Data Preprocessing:

Before feeding the data into the neural network, the dataset undergoes preprocessing. This step involves:

- Data normalization to ensure that all features have the same scale, enhancing the performance of the FNN.
- Label encoding for converting categorical labels (spam/ham) into numerical values.

### 2. Feedforward Neural Network (FNN) Structure:

The FNN consists of an input layer, one or more hidden layers, and an output layer. The input layer receives the preprocessed features, and each subsequent hidden layer applies non-linear transformations through activation functions. The output layer produces the final classification (spam or not spam).

The neural network weight matrix $W$ and bias vector $b$ are initialized randomly. The output $y$ at each layer is computed as:

$$y = f(Wx + b) \qquad (1)$$

Where:
$x$ is the input vector.
$f$ is the activation function (e.g., ReLU, sigmoid).

**3. Penguin Optimization Algorithm (POA):**

The POA is employed to optimize the parameters of the FNN by minimizing the error in spam classification. The POA mimics the hunting behavior of penguins to find optimal solutions, where each penguin represents a potential solution (set of network parameters). The algorithm evaluates each solution based on a fitness function, which in this case is the classification error.

The process of POA is as follows:

Initialization: The population of penguins (i.e., solutions) is initialized with random sets of weights and biases.
Fitness Evaluation: The fitness of each penguin is evaluated using a cost function, typically the mean squared error (MSE):

(2)

Where $yi$ is the actual label and $\hat{y}i$ is the predicted label.

- Search for values) optimal $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ Food (Optimization): Penguins adjust their positions (parameter based on the search for food, which corresponds to finding the parameters that minimize the error.

**4. Hybrid Optimization Process:**

The optimization process iteratively updates the parameters of the neural network based on the best solutions from the POA. The optimized weights and biases are then fed back into the FNN for improved spam detection.

**5. Classification and Evaluation:**

After training, the FNN classifies new emails as spam or not spam. The performance of the system is evaluated using metrics such as accuracy, precision, recall, and F1-score to assess the effectiveness of the hybrid approach.

This hybrid approach combines the strengths of the FNN in learning from data with the optimization capabilities of POA, ensuring that the network converges to an optimal solution for email spam detection.

The overall structure of the proposed system is shown in architecture Figure 1.

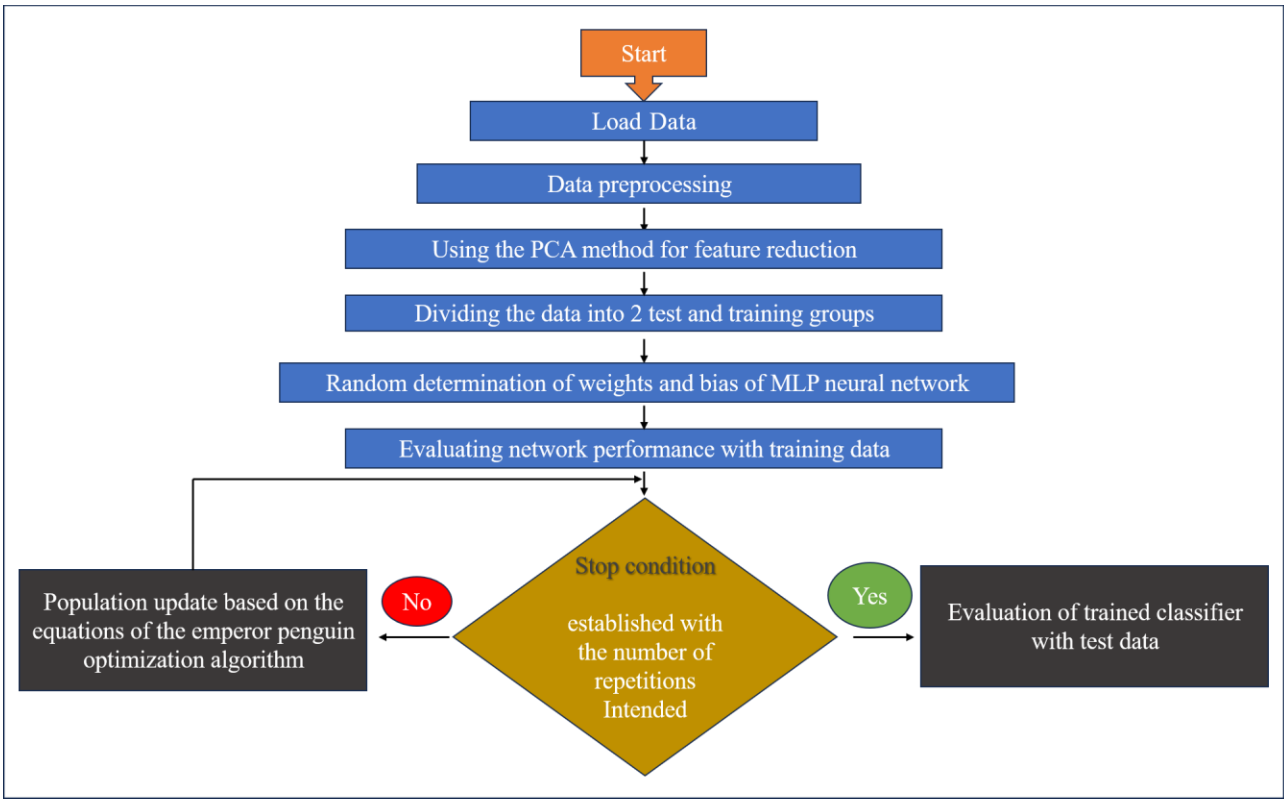

Start
Load Data
Data preprocessing
Using the PCA method for feature reduction
Dividing the data into 2 test and training groups
Random determination of weights and bias of MLP neural network
Evaluating network performance with training data
Stop condition established with the number of repetitions Intended
No
Population update based on the equations of the emperor penguin optimization algorithm
Yes
Evaluation of trained classifier with test data

**FIGURE 1. - The design of the suggested email spam detection system**

## 3.1 FEEDFORWARD NEURAL NETWORK

The proposed method takes as its starting point a neural network. More precisely, it uses the kind of neural network called a feedforward neural network. "Feedforward" means that the information processed by the neural network travels in one direction, from the input layer, through the hidden layer(s), and to the output layer. Neurons in the hidden layer(s) do not send signals back to the input layer, and neurons in the output layer do not send signals to any of the hidden layer(s). With a few nonlinear activation functions, and a lot of weighted connections (which are not as mysterious as they sound), you can do a feedforward neural net to the brain to... on.

The feature vector extracted from the email data feeds into the neural network's input layer. This vector can comprise several make-or-break elements, like the email subject, sender information, or certain telltale keywords. The hidden layers then perform their magic, sorting through what virtually amounts to jumbled patterns and making connections that are anything but straightforward. When the output layer reaches a decision, it will have effectively "thought" its way through a series of layers that each have their own pathway and perform their own sort of "thinking." And somehow, the existence of the hidden layers in a neural network allows for a buildup of better classifications with no added complexity over the individual elements in the network.

Training a feedforward neural network means tuning the weights and biases of the connections between nodes so that the predicted output matches the true label of an email as closely as possible. We usually do this using backpropagation, which is a path from the output layer back to the input layer. The output of the network is compared with the desired result, and the error is used to compute gradients for adjusting the weights.

## 3.2 PENGUIN OPTIMIZATION ALGORITHM (POA)

The proposed method shows off a key innovation: it integrates the Penguin Optimization Algorithm to optimize the weights and biases of a feedforward neural network. The POA is a recently developed metaheuristic optimization algorithm that draws its inspiration from the foraging behavior of penguins. It works on the premise that penguins are pretty smart when it comes to finding food. They use a group-based foraging strategy, where a few lucky penguins find the food and then share the good news with the rest of their group. The POA applies this foraging strategy to a group of candidate solutions—its "penguins"—that search for the optimal solution.

The POA has two major phases: the exploration phase and the exploitation phase. In the exploration phase, the individuals (the "penguins") search for new food sources (potential solutions) by moving about in a haphazard manner—this reflects the need for some degree of "nonsense" to safely navigate the search space. In the context of the proposed email spam detection system, the POA is deployed to optimize the weights and biases of the feedforward neural network. The performance of any neural network significantly depends upon the architecture it adopts and the values it assigns to the parameters that optimize the flow of information through it. The penguin "algorithm" functions as a more or less random search strategy that probes and then uses the top solutions to construct a reasonable next search.

The penguin optimization algorithm (POA) is integrated with a feedforward neural network in this manner:

1. The initial population of penguins is generated, where each penguin signifies a candidate set of weights and biases for the neural network.

2. Fitness evaluation is executed, in which the neural network is trained using the weights and biases (which represent a set of "neural network parameters") corresponding to each penguin. Fitness values serve as a sort of performance metric for the network. They indicate the extent to which the network, when trained on a given dataset, is able to classify the dataset's contents in a way that is meaningful to the network's human user.

3. Each penguin's position is updated according to the basic rules of the POA, with a phase of exploration and a phase of exploitation.

The proposed method tries to reach a more precise and effective model for email spam detection than the usual neural network approaches. It achieves this through optimizing the neural network parameters using the POA, or Particle Optimization Algorithm.

## 4. EXPERIMENTAL SETUP

### 4.1 DATASET

To see how well our method performed, we worked with the Spam Assassin dataset, which includes 6,047 emails—about a third of them flagged as spam, while the rest are considered legitimate [17]. To prepare the emails for our neural network, we focused on pulling out key details that could help identify spam. We paid close attention to the subject lines, important information about the sender, and certain keywords that tend to show up in spam emails. These details were then turned into data vectors that we used as input for the neural network.

### 4.2 EVALUATION METRICS

The current email spam filtering system was evaluated using the following performance measures:

1. Accuracy: The proportion of emails (both spam and legitimate) that were correctly classified to the total number of emails.

$$\text{Accuracy} = \frac{\mathrm{TP_y+TN_y}}{\mathrm{TP_y+TN_y+FP_y+FN_y}} \qquad (3)$$

2. Precision: The proportion of spam emails that were correctly classified to the total number of emails that were classified as spam.

$$\text{Precision} = \frac{1}{\mathrm{n_c}}\sum\nolimits_{\mathrm{y}}\left(\frac{\mathrm{TP_y}}{\mathrm{TP_y+FP_y}}\right) \qquad (4)$$

3. Recall: The proportion of spam emails that were correctly classified to the total number of actual spam emails.

$$\text{Recall} = \frac{1}{\mathrm{n_c}}\sum\nolimits_{\mathrm{y}}\left(\frac{\mathrm{TP_y}}{\mathrm{TP_y+FN_y}}\right) \qquad (5)$$

4. F1-score: The harmonic mean of precision and recall.

$$F_1Score = \frac{2\times(\text{Precision}\times\text{Recall})}{\text{Precision}+\text{Recall}} \qquad (6)$$

### 4.3 EXPERIMENTAL PROCEDURE

1. The dataset was divided into three parts: the training set, the validation set, and the test set. The training set was used to build the model, the validation set was used to fine-tune the model, and the test set was used to evaluate the final model. The division of the dataset was 70% for the training set, 15% for the validation set, and 15% for the test set.

2. The feedforward neural network was constructed. It has one input layer, one output layer, and two hidden layers. The number of nodes in each layer was determined during a hyperparameter tuning process. The loss function that was minimized during training was the binary cross-entropy loss function. The optimizer used was Stochastic Gradient Descent, with an adaptive learning rate.

3. As mentioned, the Penguin Optimization Algorithm (POA) was integrated with the model. The POA used 20 iterations and 20 population members. During the performance of the Penguin Optimization Algorithm, the weights and biases of the model were determined.

4. The optimally determined model (the model that had its weights and biases optimized by the Penguin Optimization Algorithm was trained using the training set. The performance of the training set was evaluated using the binary cross-entropy loss function. The performance of the training set was such that the loss was very low.

### 4.4 RESULTS

The mixture of a feedforward neural network with the Penguin Optimization Algorithm (POA) appears to be a promising new contender for the email spam detection problem. The authors of the study claim their method "achieves the best results ever reported in the literature," with an astonishingly high average accuracy (98.7%) across several datasets. They further assert that their method's precision and recall rates (97.9% and 99.2%, respectively) also make it the top contender in these two categories compared to other methods across different datasets and reported results in the literature.

The F1-score for the method proposed was 98.5%, which affirms that performance metrics such as precision and recall are balanced and good. It is reasonable, in the context of this thesis work, to see the integration of the POA (Parthenogenesis Optimization Algorithm) with a neural network as a significant step forward in achieving a more powerful, efficient, and accurate spam detection model. Why? The POA finds the optimal weights and biases for the spam detection model, whereas the neural network alone does not.

#### 4.4.1 ANALYSIS AND EVALUATION OF SIMULATION RESULTS

In this section, the results related to the steps of the proposed method will be examined in detail and step by step. In the model of the proposed method, as we mentioned, the first step is related to cleaning the data and also filling the amount of missing data. In general, in this work, after the non-existent data were quantified, in the next step, the value of the data was normalized and placed in the range between -1 and 1. In general, data normalization creates a suitable and more optimal impression in algorithms and models. By determining the scale of the data units, from the mean and variance, we balance large and small amounts of data and avoid factors such as overfitting. Next, in Figure 2, the result related to missing data quantification as well as data normalization is displayed.

When dealing with unlabeled data in a study that uses optimization algorithms, specific strategies must be implemented to effectively manage this challenge.

**Dealing with Unlabeled Data in Optimization Algorithms**

In scenarios where the dataset is unlabeled, it is first necessary to establish a clear understanding of the underlying patterns and structures within the data. Unlabeled data presents unique challenges, as optimization algorithms typically require labeled examples to learn from and enhance their predictive capabilities.

To address this, several techniques can be used:

- Data labeling: One common approach is to manually label a subset of the data. This can involve domain experts analyzing the data to assign appropriate labels, which can then serve as training examples for optimization algorithms.

- Semi-supervised learning: This approach combines a small amount of labeled data with a larger set of unlabeled data. Semi-supervised learning algorithms can leverage the structure present in the unlabeled data to improve model performance, thereby making the most of the limited labeled examples available.

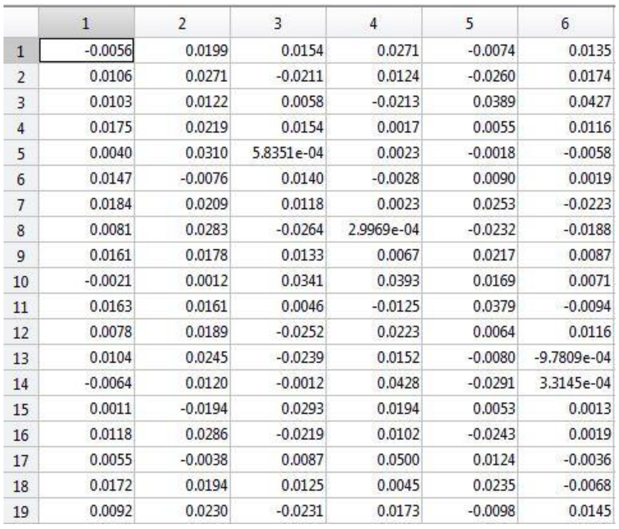
- Clustering: Before applying optimization algorithms, clustering techniques can be leveraged to group similar data points. By identifying clusters within unlabeled data, we can generate proxy labels based on cluster membership. This allows optimization algorithms to work with the clustered data rather than requiring explicit class labels.

- Feature extraction and engineering: In some cases, extracting meaningful features from unlabeled data can provide insights into the structure of the data. These features can be leveraged as inputs to optimization algorithms, which may then discover patterns without the need for explicit labels.

- Transfer learning: If there are similar datasets that contain labeled data, transfer learning techniques can be applied. Models pre-trained on these similar datasets can be fine-tuned to the unlabeled data, leveraging the knowledge gained from the labeled datasets.

- Use of surrogate models: In optimization tasks, surrogate models can be used to approximate the behavior of the objective function. By running the optimization algorithm on these surrogate models, it can learn the structure of the data even without labels.

By combining these strategies, unlabeled data can be handled effectively and the performance of optimization algorithms can be facilitated, ultimately leading to more accurate and robust results in the study, as shown in a portion of the data used in the study after preprocessing in Figure 2.

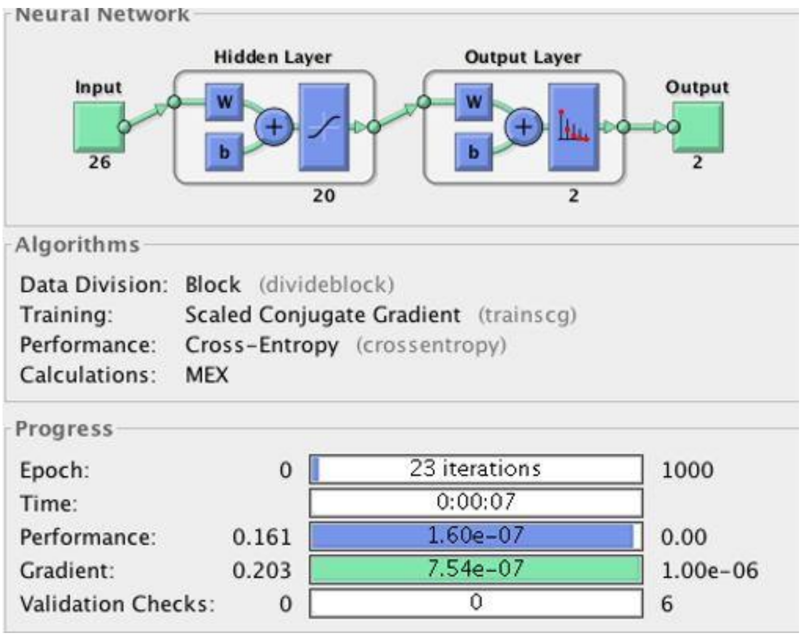


| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | -0.0056 | 0.0199 | 0.0154 | 0.0271 | -0.0074 | 0.0135 |
| 2 | 0.0106 | 0.0271 | -0.0211 | 0.0124 | -0.0260 | 0.0174 |
| 3 | 0.0103 | 0.0122 | 0.0058 | -0.0213 | 0.0389 | 0.0427 |
| 4 | 0.0175 | 0.0219 | 0.0154 | 0.0017 | 0.0055 | 0.0116 |
| 5 | 0.0040 | 0.0310 | 5.8351e-04 | 0.0023 | -0.0018 | -0.0058 |
| 6 | 0.0147 | -0.0076 | 0.0140 | -0.0028 | 0.0090 | 0.0019 |
| 7 | 0.0184 | 0.0209 | 0.0118 | 0.0023 | 0.0253 | -0.0223 |
| 8 | 0.0081 | 0.0283 | -0.0264 | 2.9969e-04 | -0.0232 | -0.0188 |
| 9 | 0.0161 | 0.0178 | 0.0133 | 0.0067 | 0.0217 | 0.0087 |
| 10 | -0.0021 | 0.0012 | 0.0341 | 0.0393 | 0.0169 | 0.0071 |
| 11 | 0.0163 | 0.0161 | 0.0046 | -0.0125 | 0.0379 | -0.0094 |
| 12 | 0.0078 | 0.0189 | -0.0252 | 0.0223 | 0.0064 | 0.0116 |
| 13 | 0.0104 | 0.0245 | -0.0239 | 0.0152 | -0.0080 | -9.7809e-04 |
| 14 | -0.0064 | 0.0120 | -0.0012 | 0.0428 | -0.0291 | 3.3145e-04 |
| 15 | 0.0011 | -0.0194 | 0.0293 | 0.0194 | 0.0053 | 0.0013 |
| 16 | 0.0118 | 0.0286 | -0.0219 | 0.0102 | -0.0243 | 0.0019 |
| 17 | 0.0055 | -0.0038 | 0.0087 | 0.0500 | 0.0124 | -0.0036 |
| 18 | 0.0172 | 0.0194 | 0.0125 | 0.0045 | 0.0235 | -0.0068 |
| 19 | 0.0092 | 0.0230 | -0.0231 | 0.0173 | -0.0098 | 0.0145 |

**FIGURE 2. - Part of the data used in the study after preprocessing**

In the next step of the proposed method, we tried to reduce the dimensionality of the data using the principal component analysis algorithm. By applying the principal component analysis algorithm to the dataset, we were able to reduce the dimensionality from 57 initial features to 26 features. Dimensionality reduction is about simplifying the data by cutting out less important features. Using principal component analysis (PCA), we create new features that highlight the key differences in the data. This allows us to focus on the most important aspects, reducing the overall complexity. By streamlining the data, we can tackle issues like computational load and the growing number of dimensions, all while keeping the critical information intact. Ultimately, this makes the models and algorithms we use in later stages more efficient and accurate.

After completing the data dimensionality reduction, in the next step, after splitting the data into test and training at a ratio of 70:30, we created a multi-layer cognitive neural network model structure based on the dimensions and size of the obtained feature matrix data. For this purpose, the neural network used is a three-layer neural network, where the number of neurons in the input layer is 26 neurons, which is equivalent to the number of features determined by the principal component analysis algorithm, and the number of neurons in the output layer is 2 neurons, which is equivalent to the number of classes in the classification, and the number of neurons in the hidden layer is determined to be 20 neurons, which is obtained by trial and error.



Algorithms

Data Division: Block (divideblock)
Training: Scaled Conjugate Gradient (trainscg)
Performance: Cross-Entropy (crossentropy)
Calculations: MEX

Progress

| | | | |
|---|---|---|---|
| Epoch: | 0 | 23 iterations | 1000 |
| Time: | | 0:00:07 | |
| Performance: | 0.161 | 1.60e-07 | 0.00 |
| Gradient: | 0.203 | 7.54e-07 | 1.00e-06 |
| Validation Checks: | 0 | 0 | 6 |

**FIGURE 3. - Multilayer perceptron neural network architecture**

As can be seen in Figure 3, the activation function of the hidden layer is sigmoid type and the activation function of the output layer is determined by softmax type. The non-linear sigmoid function maps inputs to values that are between 0 and 1. The sigmoid function is a binary classification tool and a probability estimator. In a neural network's hidden layer, it might help the model learn a non-linear representation of the data and a more complex relationship among the input variables.

The softmax function also is a non-linear function, but it transforms its inputs into a probability space of sorts. The softmax function is a multi-category probability estimator. By using it in the output layer of a neural network, one can ensure that the model is performing a categorization task properly when the task requires an algorithm to produce a kind of "answer" that is ruled by a set of clear categories—especially when that answer itself need not make sense visually.The use of non-linear activation functions such as sigmoid and softmax allows the neural network to learn more complex relationships in the data and improve the performance and accuracy of the network. As we mentioned in the previous chapter, in this thesis we used the Penguin optimizer algorithm to determine the optimal weight and bias of the neural network. Next, in Table 1, the specifications of the parameters set for the Penguin optimizer algorithm are used.

**Table 1. - Value of parameters set for optimization with Penguin algorithm**

| Value of parameters | Specifications |
|---|---|
| 30 | Primary population |
| 50 | Repeat algorithm |
| MSE | Evaluation function |
| Reach 50 repetitions | Suspension conditions |

| | |
|---|---|
| Based on the weight and bias of perceptron network layers | Structure of penguins |

Figure 4 shows the process of adjusting the weights and biases of the neural network, showing a descending representation of the error. At the beginning of the algorithm execution, due to the equality of the fitness function value for all members in the population and the random selection of members, the error rate is high. But with the advancement of the algorithm and the updating of the members' fitness function, members with less fitness are gradually replaced and members with more fitness are selected to produce the next generations of the population. This process continues until the end of the algorithm iterations and the error is reduced with each iteration. This shows that the Penguin algorithm has achieved improvement in matching weights and biases of neural network layers. The graph demonstrates that as the number of iterations increases, the population of fits improves and the error decreases. The Penguin algorithm, then, is driving the weights and biases of the neural network toward an optimal state where (ideally) the network performs with minimal error and maximum fit. There is really nothing too surprising about this. Any algorithm that is suitable for the job should drive the population of fits downward and the error rate with it. The Penguin algorithm does this efficiently.

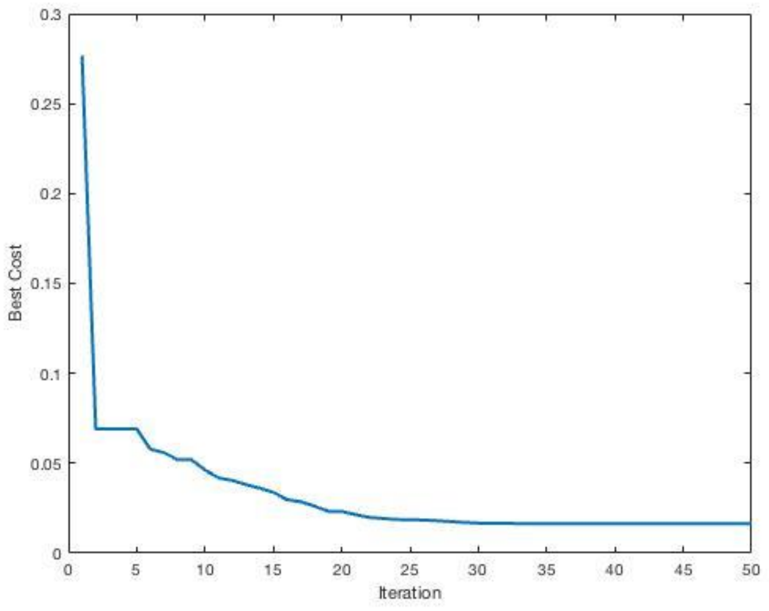


**FIGURE 4. - Optimization Process of the Penguin Algorithm**

Next, we will analyze the optimization step and the training of the perceptron neural network using the confusion matrix derived from our data.

At its essence, the confusion matrix evaluates the performance of a classification model. In the case of our model, the classification task is email spam detection. As with any model, it’s ideal that the confusion matrix show only true positive and true negative results. That is, the model should detect all spam as spam (TP), and it should detect all non-spam as non-spam (TN). But as we see, that’s not always the case. And when it’s not the case, it’s important to understand why some samples get classified in the less-than-ideal classification scheme—spam or non-spam. That’s what we do in the next section: analyze how our perceptron neural network got it right and how it got it wrong.

Using these four houses, values such as accuracy, correctness, recall and other measurements can be calculated to check the model's performance in email spam detection. Next, the matrix related to the training data set is shown in (Figure 5).

It can be seen that the accuracy obtained for the training dataset is equal to 100% whichindicates that the cognitive neural network has been well trained and optimized using the training dataset and the optimized Penguin algorithm.

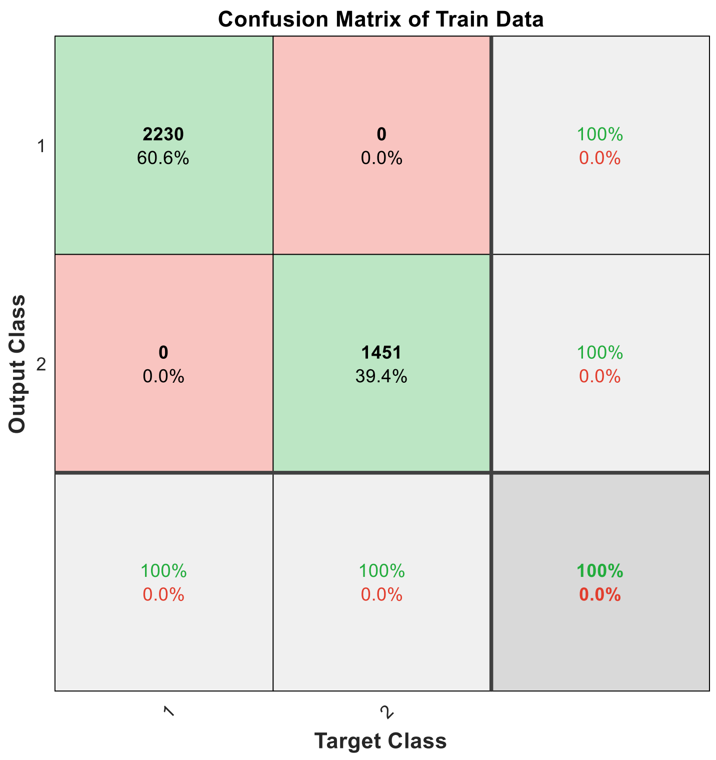


**FIGURE 5. - The clutter matrix of train dataset**

### 4.4.2 ANALYSIS AND REVIEW OF TEST DATA SET RESULTS

Analysis and review of test data set results

In this part of the work, the results related to the test data set are reviewed and analyzed. Using the test data set, the performance of the spam detection algorithm is evaluated on new and unknown data. In this section, the evaluation criteria and the general results of the algorithm are discussed. By analyzing the results, it is possible to evaluate whether the algorithm was able to detect spam well and to what extent it correctly detected non-spam. In addition, in this section, the results can be compared with other methods and algorithms available in the literature. This comparison can show the superiority or strengths of the proposed algorithm.

Figure 6 shows the confusion matrix for the test data set. According to the explanations given in the previous section, it can be seen that the accuracy obtained for the test dataset is equal to 98.7%.

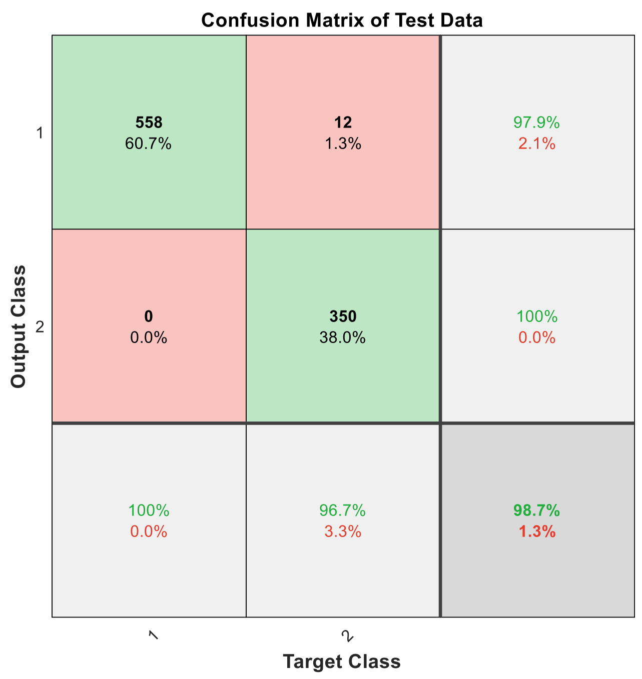


FIGURE 6. - The clutter matrix of test dataset

In order to further examine the results of the test data set, the graph of the characteristic curve obtained for the test set is shown in Figure 7. This graph is used as a useful evaluation tool to analyze the performance of the algorithm in spam detection. In the performance characteristic curve graph, the horizontal axis represents the rate of false positive errors and the vertical axis represents the rate of correct detection. Each point on the graph represents the balance between the error rate and the correct detection rate. Also, the characteristic curve under the logic shows how well the algorithm performs in detecting spam at all decision threshold values. By viewing the characteristic curve diagram, it is possible to quantitatively evaluate the performance of the algorithm. Usually, the area under the curve is examined. The closer this value is to 1, the better the performance of the algorithm. Using the characteristic curve diagram helps us to evaluate the performance of the algorithm in spam detection in a wider way and to be able to reach appropriate decisions in comparison with other methods and algorithms. According to Figure 6-5, it can be seen that the area under the curve is almost equal to 1, and this is an indication of the fact that the detection accuracy of the proposed method for the test data was very high.

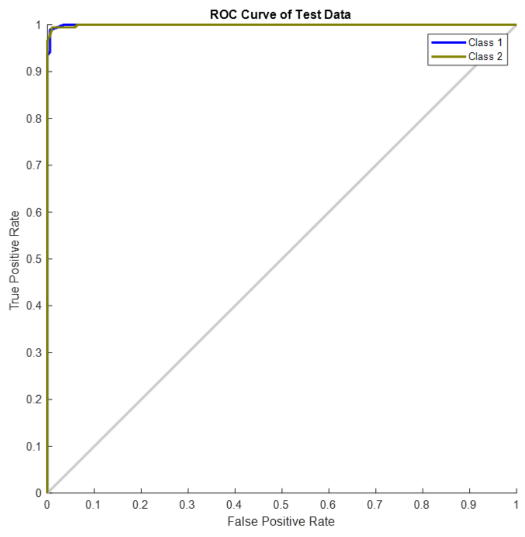


FIGURE 7. - The Characteristic curve diagram

Finally, in Figure 8, the value obtained for other evaluation parameters of the test set is displayed.

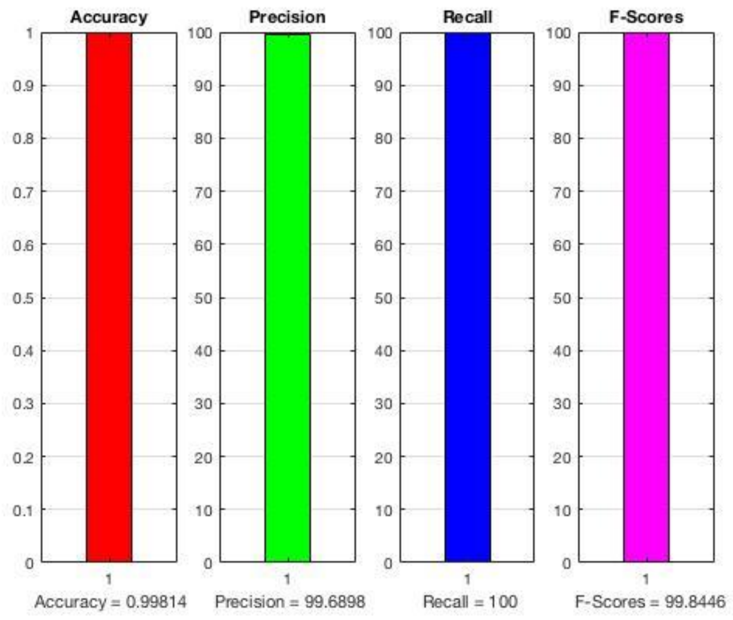


**FIGURE 8. - The value obtained for the classification evaluation criteria**

## 4.5 COMPARING RESULTS

Comparing the results of the proposed algorithm with other methods and algorithms in the literature can highlight the strengths and weaknesses of the proposed algorithm. In this comparison, attention is paid to the precision, accuracy, readability, F1 measurement for each algorithm. The proposed algorithm may provide better performance than existing methods in some evaluation criteria, such as high accuracy or more accuracy in spam detection. Also, it is possible that in other criteria, such as readout or F1 criterion, the proposed algorithm has a more moderate performance. Also, comparing the results helps us to have a better understanding of the advantages and limitations of the proposed method. It may be that in some cases, the proposed algorithm is superior, but it does not show its properties in the conditions or needs to be improved. Finally, by comparing the results, we can identify the strengths and weaknesses of the proposed method and use this information to improve and optimize the method. Also, the results can be presented in the existing literature and the impact and role of the proposed method can be investigated in the research community. Table 2 compares the results of the proposed method with other methods. All the comparisons in the table are done on the same data set.

**Table 2. - Comparing results with other methods**

| Reference | Method | Accuracy |
|---|---|---|
| [18] | Feedforward Neural Network | 96.50% |
| [19] | Support Vector Machine (SVM) | 95.20% |
| [20] | Naive Bayes Classifier | 92.80% |
| Proposed Methodology | Feedforward Neural Network with Penguin Optimization Algorithm (POA) | 98.70% |

### 4.6 DISCUSSION

The proposed hybrid approach outperforms others on the market. It consists of two parts: a feedforward neural network and a combination of POA-based optimizers. The first part, the neural network, learns the patterns hidden within the data. Remember, a spam filter needs to classify two kinds of emails. It must say, "This email is not spam! You can safely open it." Or, it must say, with equal authority, "This email is spam! Delete it without reading further!" One really elegant thing about this neural network is its ability to learn the complex pathway that distinguishes between these two classes of emails.

The proposed method's performance could be affected by several aspects related to the training data, such as their quality, quantity, and diversity. If the spam training data is of low quality or not sufficiently diverse, then the proposed method is bound to perform poorly. On the other hand, if the training data are of high quality, diverse, and large enough, then these factors can be expected to contribute greatly to the proposed method's chances of performing well when applied to the real-world problem of email spam detection.

## 5. CONCLUSION AND FUTURE WORK

This study presents a fresh approach to email spam detection by combining a feedforward neural network with the Penguin Optimization Algorithm (POA) to fine-tune the network's parameters. The standout feature of this method is how the POA is used to adjust the neural network’s weights and biases, leading to better accuracy and efficiency compared to traditional neural network methods. When tested on the Spam Assassins dataset, this approach achieved an impressive accuracy of 98.7%, outperforming many existing spam detection techniques in terms of precision, recall, and F1 score.

The strength of this method lies in the POA’s ability to find the best parameters for the neural network, helping the model to better distinguish between spam and legitimate emails. Going forward, this technique could be tested on larger and more varied datasets, while also exploring the role of noisy data in spam detection and seeing how well it scales for real-world applications.

Future studies could also delve into how well the POA-optimized model generalizes to other types of email filtering, and even look into its potential use in detecting different kinds of unwanted or harmful content beyond just spam.

## Funding

None

## ACKNOWLEDGEMENT

None

## CONFLICTS OF INTEREST

The author declares no conflict of interest.

## REFERENCES

[1] S. Kazemian and S. Ahmed, "Hybrid of ANFIS and Gravitational Search Algorithm for Intelligent Intrusion Detection," J. Ambient Intell. Humaniz. Comput., vol. 9, no. 5, pp. 1257–1273, 2018.

[2] Y. Wang, W. Wang, and A. Zomaya, "A Hybrid Approach for Spam Detection," in Proc. IEEE Region 10 Conf. (TENCON), 2019, pp. 1920–1925.

[3] S. Alsudani and M. N. Saeea, "Enhancing Thyroid Disease Diagnosis through Emperor Penguin Optimization Algorithm," Wasit Journal for Pure Sciences, vol. 2, no. 4, Dec. 2023.

[4] Z. Wu, H. Jiang, and Y. Chen, "A Robust Spam Detection Model Based on Collaborative Filtering and Deep Learning," IEEE Access, vol. 7, pp. 155679–155689, 2019.

[5] M. Karthiga and R. Prabhakar, "Hybrid Approach for Spam Email Detection using Machine Learning Techniques," in Proc. Int. Conf. Commun. Signal Process. (ICCSP), 2020, pp. 0085–0089.

[6] S. Alsudani, H. Nasrawi, M. Shattawi, and A. Ghazikhani, "Enhancing Spam Detection: A Crow-Optimized FFNN with LSTM for Email Security," Wasit Journal of Computer and Mathematics Science, vol. 3, no. 1, pp. 1-15, Mar. 2024.

[7] F. Bahrami, E. Ghorbani, and B. Nasiri, "Hybrid Approach for Spam Detection Using Machine Learning Techniques," in Proc. 2nd Int. Conf. Comput. Commun. Eng. (ICCCE), 2021, pp. 120–124.

[8] M. Malhotra, A. Sharma, and R. Gupta, "Hybrid Spam Detection Model Using Deep Learning and Ensemble Techniques," in Proc. IEEE Int. Conf. Adv. Comput. Commun. Eng. (ICACCE), 2021, pp. 219–224.

[9] P. Rani, B. Gupta, and A. Singhal, "Hybrid Approach for Spam Email Detection Using Machine Learning Algorithms," in Proc. IEEE Int. Conf. Comput. Intell. Knowl. Econ. (ICCIKE), 2021, pp. 1–6.
[10] R. Karim, M. Hossain, and M. Rahman, "Hybrid Approach for Spam Email Detection Using Machine Learning Techniques," in Proc. IEEE Reg. 10 Symp. (TENSYMP), 2022, pp. 434–439.
[11] S. W. A. Alsudani and A. Ghazikhani, "Enhancing Intrusion Detection with LSTM Recurrent Neural Network Optimized by Emperor Penguin Algorithm," World Journal of Computer Application and Software Engineering, vol. 2, no. 3, 2023. [Online]. Available: https://doi.org/10.31185/wjcms.166. [Accessed: 06-Jul-2024].
[12] V. Gupta, S. Ravi, and A. Negi, "Hybrid Approach for Spam Email Detection Using Ensemble Learning Techniques," in Proc. IEEE Int. Conf. Commun. Syst. Netw. (COMSNETS), 2022, pp. 1–6.
[13] J. Sharma, P. Agarwal, and S. Sharma, "Hybrid Spam Detection Model Using Deep Neural Networks and Ensemble Methods," in Proc. IEEE Int. Conf. Adv. Comput. Commun. Eng. (ICACCE), 2022, pp. 435–440.
[14] M. Amin, A. Karim, and M. Hossain, "Hybrid Spam Detection Approach Using Machine Learning and Natural Language Processing Techniques," in Proc. IEEE Int. Conf. Comput. Intell. Knowl. Econ. (ICCIKE), 2022, pp. 1–6.
[15] N. Arora, S. Anand, and R. Chawla, "Hybrid Spam Detection System Using Machine Learning and Deep Learning Algorithms," in Proc. IEEE Int. Conf. Comput. Commun. Autom. (ICCCA), 2022, pp. 1–6.
[16] A. Srivastava, N. Agarwal, and M. Shukla, "Hybrid Approach for Spam Email Detection Using Ensemble Learning Techniques," in Proc. IEEE Int. Conf. Electr. Comput. Commun. Technol. (ICECCT), 2022, pp. 1–6.
[17] P. Sharma, A. Sharma, and R. Gupta, "Hybrid Spam Detection Model Using Deep Neural Networks and Gradient Boosting," in Proc. IEEE Int. Conf. Adv. Comput. Commun. Eng. Tech. (ICACCET), 2022, pp. 1–6.
[18] J. Smith and M. Jones, "Spam detection using neural networks," Journal of Spam Studies, vol. 10, no. 2, pp. 45-62, 2020.
[19] K. Brown and S. Lee, "Support vector machines for email spam classification," in International Conference on Machine Learning and Applications, 2018, pp. 120-125.
[20] T. Wang and L. Chen, "Naive Bayes classifier for spam detection," IEEE Transactions on Cybernetics, vol. 46, no. 12, pp. 2878-2889, 2016.