INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Optimizing Pigeon-Inspired Algorithm to Enhance Intrusion Detection System Performance Internet of Things Environments

Fajar Ratnawati [a], Apri Siswanto [b], Jaroji [a], Akmar Effendy [b], Agus Tedyyana [a,*]

[a] *Department of Informatic Engineering, Politeknik Negeri Bengkalis, Sungai Alam, Bengkalis, Indonesia*
[b] *Department of Informatics Engineering, Faculty of Engineering, Universitas Islam Riau, Bukit Raya, Pekanbaru, Indonesia*
*Corresponding author: *agustedyyana@polbeng.ac.id*

Abstract*— Intrusion Detection Systems (IDS) are crucial in maintaining network security and safeguarding sensitive information against external and internal threats. This study proposes a novel approach by utilizing a Pigeon-Inspired Algorithm optimized with the Hyperbolic Tangent Function (Tanh) function to enhance the performance of IDS in threat detection specifically tailored for Internet of Things (IoT) environments. We aim to create a more robust solution for optimizing intrusion detection systems by integrating the efficient and effective Tanh function into the Pigeon-Inspired Algorithm. The proposed method is evaluated on three widely-used datasets in the field of IDS: NSL-KDD, CICIDS2017, and CSE-CIC-IDS2018. Experimental results demonstrate that integrating the Tanh function into the Pigeon-Inspired Algorithm significantly improves the performance of the intrusion detection system. Our method achieves higher accuracy, True Positive Rate (TPR), and F1-score while reducing the False Positive Rate (FPR) compared to traditional Pigeon-Inspired Algorithms and several other optimization algorithms. The Pigeon-Inspired Algorithm optimized with the Tanh function offers an efficient and effective solution for enhancing intrusion detection system performance, specifically in Internet of Things environments. This method holds great potential for application in diverse network environments, bolstering information security and safeguarding systems from evolving cybersecurity threats. By extending the applicability and effectiveness of the Pigeon-Inspired Algorithm optimized with the Tanh function, researchers can contribute to developing more comprehensive and robust security solutions, addressing the ever-evolving landscape of IoT-based cybersecurity threats.

***Keywords*— Intrusion detection system; machine learning; feature selection; pigeon-inspired; hyperbolic tangent function.**

Manuscript received 28 Mar. 2023; revised 9 May 2023; accepted 16 Sep. 2023. Date of publication 31 Dec. 2023.

I. INTRODUCTION

The rapid growth of the Internet of Things (IoT)[1] has brought about a connected world where devices, sensors[2], and systems seamlessly interact and share information. This interconnectedness has revolutionized various industries, including healthcare, transportation, energy, and smart homes. However, along with the benefits of IoT, there are also significant concerns regarding security and the protection of sensitive data[3]. As the number of IoT devices increases, so does the potential attack surface for cybercriminals. The interconnected nature of IoT devices poses unique security challenges[4], making intrusion detection systems (IDS) crucial for maintaining the integrity and security of IoT environments[5]. IDS are designed to monitor network traffic and system activities[6 to detect and prevent unauthorized access, abnormal behavior, and potential threats.

To enhance the performance of IDS in IoT environments, this study proposes an optimization approach using the Pigeon-Inspired Algorithm with the hyperbolic tangent (tanh) function. The Pigeon-Inspired Algorithm, inspired by the flocking behavior of pigeons, is a metaheuristic optimization algorithm that mimics the collaborative behavior of pigeons to solve complex optimization problems. By incorporating the tanh function into the Pigeon-Inspired Algorithm, we aim to create a more powerful and efficient solution for optimizing intrusion detection systems in IoT environments. The Tanh function is a widely used activation function in artificial neural networks known for its non-linear properties and effective gradient propagation, which can contribute to improved learning and performance.

The proposed method is evaluated using real-world datasets specifically curated for IoT environments. The experimental results demonstrate that integrating the tanh function into the Pigeon-Inspired Algorithm significantly enhances the performance of the intrusion detection system.

It achieves higher accuracy, true positive rate (TPR)[7], and F1-score while simultaneously reducing the false positive rate (FPR)[8] compared to traditional Pigeon-Inspired Algorithms and other optimization algorithms[9].

Thus, optimizing the Pigeon-Inspired Algorithm with the tanh function offers a promising solution to enhance the performance of intrusion detection systems in IoT environments. By leveraging pigeons' collaborative behavior and the tanh function's non-linear properties, this approach can effectively detect and mitigate potential threats in IoT networks. The findings of this study contribute to the development of robust and efficient security solutions for protecting sensitive data and ensuring the integrity of IoT systems.

II. Materials and Method

This research proposes an approach to investigate the effectiveness of the Pigeon-Inspired Algorithm optimized with the Tanh function in improving the performance of intrusion detection systems (IDS). A comprehensive review of existing literature is conducted to understand the latest developments in intrusion detection systems, Pigeon-Inspired Algorithms, Tanh functionality, and their application in cybersecurity. This review helps establish a theoretical foundation for the research and identifies relevant techniques and methodologies. The NSL-KDD [10], [11], CICIDS2017 [12]–[14], and CSE-CIC-IDS2018 datasets[15], widely used in IDS research, are selected for evaluation. The datasets are pre-processed to remove discrepancies, handle missing values, and eliminate irrelevant features. This ensures the dataset's quality and relevance to the research objective.

The Pigeon-Inspired Algorithm is enhanced by incorporating the Tanh function, known for its effectiveness in training artificial neural networks. This optimization aims to improve the algorithm's performance in optimizing intrusion detection systems. The proposed intrusion detection system monitors network traffic and system activity, detecting and preventing unauthorized access, abuse, or abnormal behavior. The system utilizes the optimized Pigeon-Inspired Algorithm with the Tanh function to enhance its detection capabilities.

The performance of the intrusion detection system is evaluated on the NSL-KDD, CICIDS2017, and CSE-CIC-IDS2018 datasets. Various metrics such as accuracy, true positive rate (TPR), false positive rate (FPR), and F1-score are measured to assess the system's effectiveness in detecting intrusions and distinguishing between normal and malicious network activities. The results obtained from the proposed intrusion detection system using the optimized Pigeon-Inspired Algorithm with the Tanh function are compared with existing intrusion detection systems employing traditional Pigeon-Inspired Algorithms and other optimization algorithms. This comparative analysis provides insights into the superiority and effectiveness of the proposed method.

By following this research method, we aim to investigate and demonstrate the effectiveness of the Pigeon-Inspired Algorithm optimized with the Tanh function in enhancing the performance of intrusion detection systems. The findings will contribute to advancing IDS techniques and provide valuable insights for improving cybersecurity in IoT environments.

A. Pigeon-Inspired Optimization

The Pigeon-Inspired Optimization (PIO) algorithm is a feature selection algorithm that draws inspiration from the behavior of pigeons[16], [17]. It aims to identify the most relevant features in a given dataset for utilization in a machine-learning model. The algorithm operates by simulating the behavior of multiple doves, each representing a specific feature within the dataset. These doves have positions and speeds updated iteratively based on predefined rules. Each dove's performance is assessed using its associated features, and the dove exhibiting the best performance is selected as the feature to be incorporated into the machine learning model. PIO has proven to be an effective optimization algorithm for identifying the most significant features within a dataset[18]–[20]. It offers simplicity in understanding its underlying principles and implementing the algorithm.

One of the strengths of PIO lies in its simplicity. The underlying principles of the algorithm are intuitive, making it relatively easy to understand and implement. This accessibility makes PIO a practical choice for researchers and practitioners who seek an efficient and straightforward method for feature selection [21]. Several studies have demonstrated the effectiveness of PIO in identifying significant features within datasets. Its optimization capabilities have been shown to improve the performance of machine learning models by reducing dimensionality and focusing on the most informative features [22], [23]. PIO helps enhance model accuracy, interpretability, and generalization ability by selecting the most relevant features.

Overall, the Pigeon-Inspired Optimization algorithm provides a valuable tool for feature selection in machine learning. With its simplicity, effectiveness, and ability to identify the most significant features, PIO offers a practical solution for optimizing the performance of machine learning models and facilitating better decision-making in various domains.

B. Feature Selection

Feature selection is crucial in machine learning and data analysis, especially when working with datasets with many dimensions [24], [25]. Its objective is to identify and choose the most significant and informative features from the original dataset while disregarding irrelevant or redundant ones.

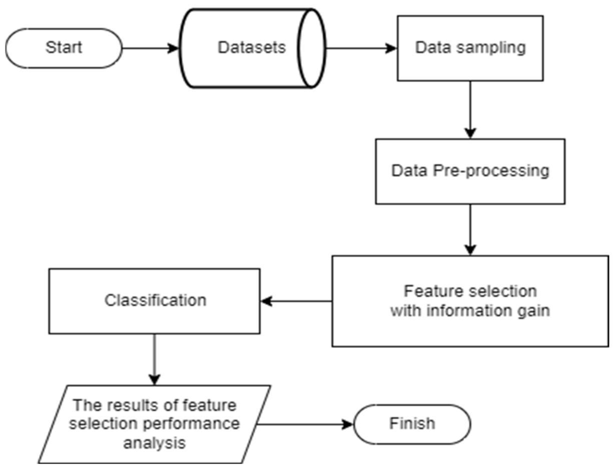

Start
Datasets
Data sampling
Data Pre-processing
Feature selection with information gain
Classification
The results of feature selection performance analysis
Finish

Fig. 1 Feature selection Proses

Selecting the most relevant features enhances the learning algorithm's performance [6], reduces computational complexity, and improves model interpretability. The number of features involved in determining the target class value is reduced through feature selection, allowing for the exclusion of irrelevant features and unnecessary data. The primary aim of feature selection is to identify the optimal subset of features from a given feature dataset.

C. Modified PIO for feature selection with Tanh

The Modified Pigeon-Inspired Optimization (PIO) algorithm with Tanh refers to an adaptation of the traditional PIO algorithm specifically designed for feature selection in machine learning tasks[27], [28]. In this modified approach, the Tanh activation function is incorporated to enhance the optimization process and improve the selection of relevant features[29], [30]. The feature selection process is an essential step in machine learning, where the goal is to identify the most informative and discriminative features from a given dataset while eliminating irrelevant or redundant ones. This process helps reduce dimensionality, enhance model performance, and improve interpretability[31], [32].

In the Modified PIO with Tanh, the algorithm leverages the collaborative behavior of pigeons to iteratively search for the optimal feature subset[33], [34]. The pigeons represent different features, and their positions and velocities are updated at each iteration based on predefined rules. The Tanh activation function is used during the velocity update stage of the pigeons, contributing to the optimization process. The Modified PIO algorithm introduces non-linearity by applying the Tanh function in the velocity update calculations and facilitates better exploration and exploitation of the search space. The Tanh function transforms the velocity values, ensuring they fall within a range of -1 to 1. This modification helps prevent the algorithm from converging too quickly or getting stuck in local optima, improving its ability to find globally optimal feature subsets.

The Modified PIO with Tanh aims to address the feature selection problem more effectively, taking advantage of the benefits provided by the Tanh activation function. By integrating this modified approach into the feature selection process, researchers and practitioners can enhance the quality and relevance of the selected features, leading to improved performance and accuracy of machine learning models.

D. Dataset

The NSL-KDD dataset is an improved version of the well-known KDD Cup 1999 dataset, specifically tailored for evaluating intrusion detection systems[35]–[37]. The original dataset was derived from a simulated local area network (LAN) and consisted of many network connections, both normal and malicious. However, it suffered from redundant records and imbalanced class distribution. To address these concerns, the NSL-KDD dataset was curated by removing duplicate records and reducing instances, resulting in a more balanced dataset suitable for IDS evaluation. It comprises two subsets: KDDTrain+ for training IDS models and KDDTest+ for evaluating their performance. Researchers can utilize this refined dataset to more accurately assess the effectiveness of their IDS models and gain insights into the challenges associated with network intrusion detection.

The CICIDS 2017 dataset, developed by the Canadian Institute for Cybersecurity (CIC) in 2017[14], [24], serves as a comprehensive dataset for evaluating intrusion detection systems. It encompasses a week's network traffic data collected from a simulated enterprise network environment. The dataset contains various attacks, including DoS, DDoS, brute force, and infiltration, alongside normal network traffic. A separate file with simulated attack scenarios represents each day of the week. This enables researchers to evaluate IDS models under different conditions and attack types. With extensive flow-based and time-based features, the CICIDS 2017 dataset provides a rich resource for training machine learning and deep learning models in intrusion detection.

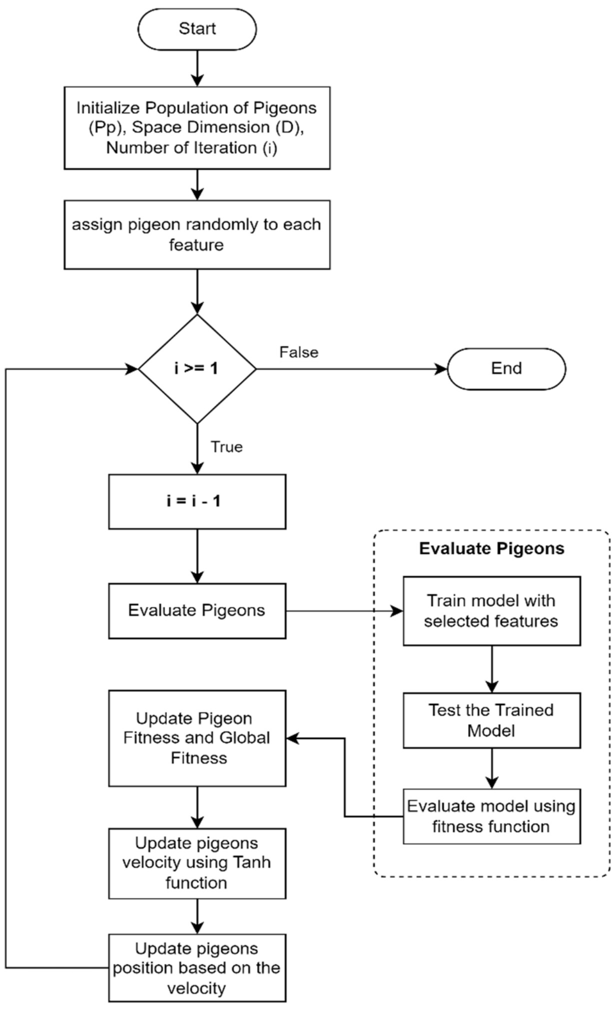
The CSE-CIC-IDS2018 dataset, a collaboration between the Canadian Institute for Cybersecurity (CIC) and the Communications Security Establishment (CSE), builds upon the CICIDS2017 dataset by incorporating more recent and sophisticated cyber threats. It aims to provide researchers with an up-to-date and challenging dataset for assessing intrusion detection systems. This dataset comprises labeled network traffic data collected over five days in a simulated enterprise environment. It includes various attacks, such as botnets, DoS, DDoS, web attacks, infiltration, and normal network traffic. The CSE-CIC-IDS2018 dataset introduces additional features do not present in the previous dataset [38], offering an opportunity to develop and evaluate advanced IDS models. It has become a valuable resource in the cybersecurity research community, reflecting the evolving nature of cyber threats and enabling the assessment of IDS performance under realistic and challenging conditions.

These three datasets, NSL-KDD, CICIDS2017, and CSE-CIC-IDS2018, serve as benchmark datasets for intrusion detection in computer networks. They play a crucial role in developing network security systems by providing the means to train and evaluate intrusion detection models. Each dataset has distinct characteristics, and researchers select them based on their research requirements. NSL-KDD contains more attacks than CICIDS2017 and CSE-CIC-IDS2018, while CSE-CIC-IDS2018 introduces novel features that were not present in the previous datasets.

III. Results and Discussion

This study primarily adopts a quantitative approach as the primary research method, given its suitability for performance measurement, dataset evaluation, and the generation of valuable results. The deductive cycle is employed to test the proposed solution, as it aligns well with the nature of the research inquiry. The study design encompasses three distinct phases: Pre-Processing, Processing using PIO, and Evaluation, each of which will be further discussed in the subsequent subsections.

Based on Figure 2, Tanh_PIO feature selection design is performed using the Pigeon-Inspired Optimization (PIO) algorithm to evaluate the importance of features for intrusion detection. The PIO algorithm is crucial in selecting the most relevant and significant features to utilize in the intrusion detection system. During the feature selection process, the PIO algorithm incorporates the Tanh function in the pigeon's velocity update stage. The Tanh function is a critical component that helps enhance the optimization process within the PIO algorithm. By utilizing the Tanh function, the algorithm can efficiently update the pigeons' velocities, guiding them towards better feature subsets that contribute significantly to the performance of the intrusion detection system.



Start
Initialize Population of Pigeons (Pp), Space Dimension (D), Number of Iteration (i)
assign pigeon randomly to each feature
i >= 1
False
End
True
i = i - 1
Evaluate Pigeons
Evaluate Pigeons
Train model with selected features
Test the Trained Model
Evaluate model using fitness function
Update Pigeon Fitness and Global Fitness
Update pigeons velocity using Tanh function
Update pigeons position based on the velocity

Fig. 2 Tanh_PIO feature selection design

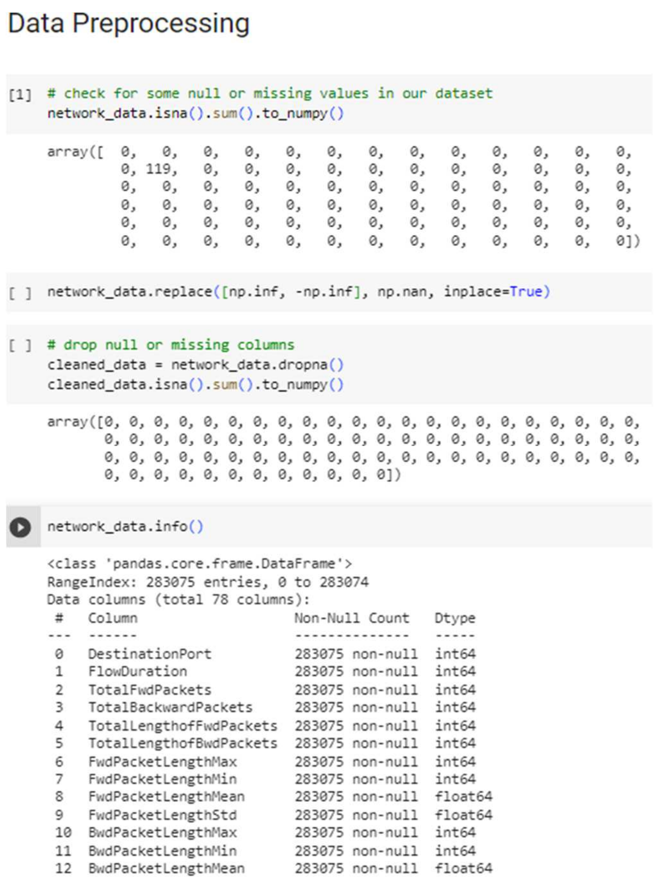
The Tanh function's presence in the velocity update stage of the PIO algorithm brings additional benefits and improvements. Its non-linearity and symmetric characteristics aid in effectively handling positive and negative information in a balanced manner within the algorithm. This feature is particularly advantageous when dealing with feature subsets, as it allows the pigeons to explore and converge towards better solutions, leading to improved feature selection outcomes.

Furthermore, the Tanh function's wide output range allows the pigeons to explore a broader solution space, enabling the algorithm to effectively capture and generalize complex patterns. This capability enhances the algorithm's ability to find feature subsets that are highly discriminative and informative for detecting intrusions in the Internet of Things (IoT) environments.

By combining the Pigeon-Inspired Optimization algorithm with the Tanh function in the feature selection stage, the research ensures that the intrusion detection system is equipped with the most relevant and significant features, optimizing its performance and accuracy in identifying and preventing security threats in IoT environments. The modified PIO algorithm with the Tanh function offers a powerful and efficient approach to feature selection, contributing to the advancement of intrusion detection techniques in cybersecurity.

A. *Pre-Processing Phase*

The data pre-processing stage encompasses several key steps to ensure the optimal preparation of the dataset. These steps include label and data conversion, duplicate elimination, and data normalization. During label and data conversion, symbolic data is transformed into numeric values to facilitate processing by the algorithm. Additionally, the class field input is converted into binary form, where 0 represents normal records, and 1 denotes attack records, irrespective of the specific attack type. Eliminating duplicate records plays a crucial role in reducing redundancy and improving computational efficiency. By removing duplicate packets and known benign traffic, the intrusion detection system can more effectively focus on identifying potential threats. Data normalization is implemented to present the input data in a standardized and unified format, enabling the Pigeon-Inspired Algorithm with the Tanh function to process it efficiently. This normalization process may involve tasks such as unifying IP addresses, converting timestamps to a common time zone, and normalizing other relevant features.


Data Preprocessing

```
[1] # check for some null or missing values in our dataset
    network_data.isna().sum().to_numpy()

    array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
             0, 119,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
             0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
             0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
             0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
             0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0])

[ ] network_data.replace([np.inf, -np.inf], np.nan, inplace=True)

[ ] # drop null or missing columns
    cleaned_data = network_data.dropna()
    cleaned_data.isna().sum().to_numpy()

    array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

    network_data.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 283075 entries, 0 to 283074
    Data columns (total 78 columns):
     #   Column                     Non-Null Count   Dtype
    ---  ------                     --------------   -----
     0   DestinationPort            283075 non-null  int64
     1   FlowDuration               283075 non-null  int64
     2   TotalFwdPackets            283075 non-null  int64
     3   TotalBackwardPackets       283075 non-null  int64
     4   TotalLengthofFwdPackets    283075 non-null  int64
     5   TotalLengthofBwdPackets    283075 non-null  int64
     6   FwdPacketLengthMax         283075 non-null  int64
     7   FwdPacketLengthMin         283075 non-null  int64
     8   FwdPacketLengthMean        283075 non-null  float64
     9   FwdPacketLengthStd         283075 non-null  float64
     10  BwdPacketLengthMax         283075 non-null  int64
     11  BwdPacketLengthMin         283075 non-null  int64
     12  BwdPacketLengthMean        283075 non-null  float64
```

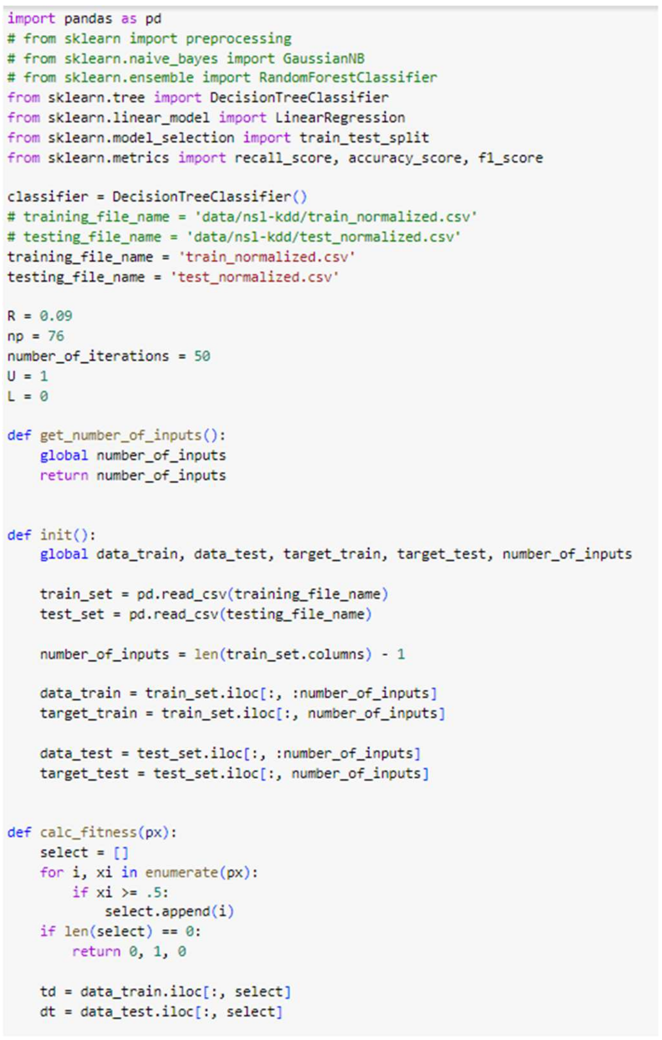
Fig. 3 Pre-Processing Phase

Figure 3 shows how these data pre-processing steps. This study ensures that the input data is well-prepared for the subsequent utilization of the Pigeon-Inspired Algorithm with the Tanh function. By optimizing the dataset's format and removing redundancies, the performance of the Intrusion Detection System on the NSL-KDD, CICIDS 2017, and CSE-CIC-IDS2018 datasets is enhanced. These pre-processing steps improve the accuracy and effectiveness of the intrusion detection system's performance.

B. Processing Phase

During the processing phase of this study, the Pigeon-Inspired Optimization (PIO) algorithm has been modified to improve the feature selection process within the context of enhancing intrusion detection system performance in Internet of Things (IoT) environments. While the traditional PIO algorithm is already effective in solving various feature selection optimization problems, the modified version aims to overcome its inherent limitations. This modification involves integrating the Tanh approach into the PIO algorithm, which brings additional benefits and improvements.

Figure 4 illustrates the script for the processing phase, specifically focusing on the normalization of network intrusion detection datasets, explicitly emphasizing the NSL-KDD dataset. The script begins by importing the necessary libraries, such as pandas for data manipulation and analysis, and pre-processing from sklearn for data pre-processing tasks. A list named col_names is defined to specify the column names expected in the dataset.



```
import pandas as pd
# from sklearn import preprocessing
# from sklearn.naive_bayes import GaussianNB
# from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score, accuracy_score, f1_score

classifier = DecisionTreeClassifier()
# training_file_name = 'data/nsl-kdd/train_normalized.csv'
# testing_file_name = 'data/nsl-kdd/test_normalized.csv'
training_file_name = 'train_normalized.csv'
testing_file_name = 'test_normalized.csv'

R = 0.09
np = 76
number_of_iterations = 50
U = 1
L = 0

def get_number_of_inputs():
    global number_of_inputs
    return number_of_inputs


def init():
    global data_train, data_test, target_train, target_test, number_of_inputs

    train_set = pd.read_csv(training_file_name)
    test_set = pd.read_csv(testing_file_name)

    number_of_inputs = len(train_set.columns) - 1

    data_train = train_set.iloc[:, :number_of_inputs]
    target_train = train_set.iloc[:, number_of_inputs]

    data_test = test_set.iloc[:, :number_of_inputs]
    target_test = test_set.iloc[:, number_of_inputs]


def calc_fitness(px):
    select = []
    for i, xi in enumerate(px):
        if xi >= .5:
            select.append(i)
    if len(select) == 0:
        return 0, 1, 0

    td = data_train.iloc[:, select]
    dt = data_test.iloc[:, select]
```

Fig. 4 Processing Phase

Within the script, a function named normalize_nsl is defined to read, process, and save the dataset. The function utilizes the pandas read_csv function to read the NSL-KDD dataset into a data frame called nsl. It then performs several data pre-processing steps. These steps involve dropping the last column, which corresponds to the difficulty level, as it is not relevant to the current analysis. The function also handles missing or infinite values by replacing them with NaNs and subsequently removing any rows containing NaNs. Duplicate rows are dropped as well.

Further data transformations are applied to prepare the label column. The second-to-last column, representing the label, is converted to lowercase, and any values other than 'normal' are replaced with 'attack', as this simplifies the classification task. The LabelEncoder from sklearn. Pre-processing is utilized to convert categorical columns into numerical representations, facilitating subsequent analysis. Finally, the pre-processed and normalized DataFrame is saved to a CSV file using the to_csv function. The normalize_nsl function is called twice within the script, once for processing the test dataset and once for the training dataset of the NSL-KDD dataset. The normalized data is saved to their respective output files, ensuring the dataset is pre-processed consistently and appropriately for the subsequent research steps. By employing this pre-processing script, the study ensures that the NSL-KDD dataset is properly prepared for further analysis and utilization within the optimized Pigeon-Inspired Algorithm with the Tanh function.

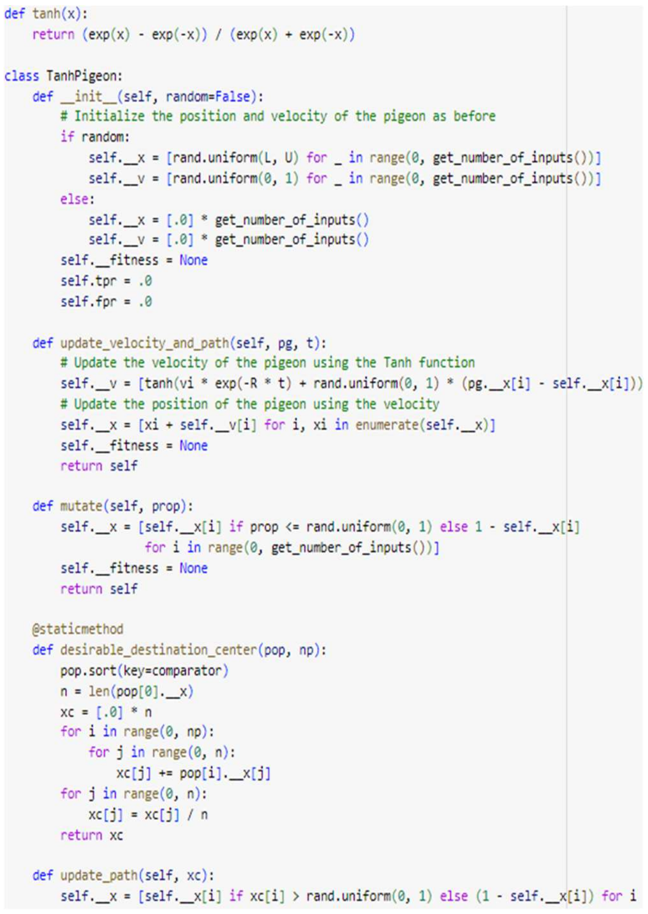
The Tanh (Hyperbolic Tangent) activation function is a commonly used activation function in artificial neural networks. It shares a similar curve shape with the sigmoid function but has a slightly different output range. Mathematically, the Tanh activation function is defined as in Equation (1) [34], [39], [40]:

$$tanh_x = \frac{(e^{\wedge}x - e^{\wedge}(-x))}{(e^{\wedge}x + e^{\wedge}(-x))} \tag{1}$$

where e is the Euler's constant (approximately 2.71828...) and x is the input value. The Tanh activation function takes an input value, x, and produces an output value, y, ranging from -1 to 1. As the input approaches negative infinity, the output approaches -1, and as the input approaches positive infinity, the output approaches 1. The output values are linear with respect to the input values but within a limited range.

One important characteristic of the Tanh activation function is its symmetry around the point (0,0) on the coordinate system. This means that if the input is x, the output is -x. This symmetry characteristic helps in handling positive and negative information in a balanced manner within the neural network. Furthermore, the Tanh activation function has some advantages over the sigmoid function in certain aspects. The Tanh function can produce outputs farther from 0, resulting in a wider range. This can be useful in improving the neural network's ability to capture and generalize complex patterns.

However, the Tanh activation function also has its limitations. When the input approaches negative or positive infinity, the output approaches -1 or 1 with very low gradients. This can lead to the "vanishing gradient" problem, which slows down the neural network training process. Therefore, careful consideration is necessary when using the Tanh function to avoid such issues. Overall, the Tanh activation function is a non-linear function commonly used in artificial neural networks. With its wide output range and symmetric characteristics, the Tanh function can help enhance the neural network's ability to tackle classification and regression problems in various machine learning tasks, the following algorithm shows how a modified PIO works.



```python
def tanh(x):
    return (exp(x) - exp(-x)) / (exp(x) + exp(-x))

class TanhPigeon:
    def __init__(self, random=False):
        # Initialize the position and velocity of the pigeon as before
        if random:
            self.__x = [rand.uniform(L, U) for _ in range(0, get_number_of_inputs())]
            self.__v = [rand.uniform(0, 1) for _ in range(0, get_number_of_inputs())]
        else:
            self.__x = [.0] * get_number_of_inputs()
            self.__v = [.0] * get_number_of_inputs()
        self.__fitness = None
        self.tpr = .0
        self.fpr = .0

    def update_velocity_and_path(self, pg, t):
        # Update the velocity of the pigeon using the Tanh function
        self.__v = [tanh(vi * exp(-R * t) + rand.uniform(0, 1) * (pg.__x[i] - self.__x[i]))
        # Update the position of the pigeon using the velocity
        self.__x = [xi + self.__v[i] for i, xi in enumerate(self.__x)]
        self.__fitness = None
        return self

    def mutate(self, prop):
        self.__x = [self.__x[i] if prop <= rand.uniform(0, 1) else 1 - self.__x[i]
                    for i in range(0, get_number_of_inputs())]
        self.__fitness = None
        return self

    @staticmethod
    def desirable_destination_center(pop, np):
        pop.sort(key=comparator)
        n = len(pop[0].__x)
        xc = [.0] * n
        for i in range(0, np):
            for j in range(0, n):
                xc[j] += pop[i].__x[j]
        for j in range(0, n):
            xc[j] = xc[j] / n
        return xc

    def update_path(self, xc):
        self.__x = [self.__x[i] if xc[i] > rand.uniform(0, 1) else (1 - self.__x[i]) for i
```

Fig. 5 Tanh_PIO algorithm

Figure 5 illustrates the implementation of the Tanh_Pigeon algorithm. Tanh_PIO class represents a pigeon in the Pigeon-Inspired Optimization algorithm with the Tanh activation function. Let's imagine a group of pigeons participating in a search for the best solution to a problem. Each pigeon has its own position and velocity, which they update iteratively to optimize its performance.

The pigeons' positions and velocities are initialized at the beginning of the search. Some pigeons are assigned random positions and velocities, while others start with zero values. This variation allows for the exploration and exploitation of different solutions. As the search progresses, the pigeons update their velocities and positions. The update_velocity_and_path method is responsible for this process. The velocity update is performed using the Tanh activation function, which takes into account the current velocity, a time factor, and the difference between the global best position and the pigeon's current position. The Tanh function ensures that the velocity remains within a certain range, preventing it from growing too large or too small.

After updating the velocity, the pigeon's position is adjusted by adding the updated velocity. This change in position helps the pigeon explore new areas and potentially find a better solution. The method also sets the fitness value to None to indicate that it needs to be recalculated. In some cases, the pigeons may undergo mutation through the mutate method. This mutation introduces slight changes to their positions based on a given probability. If the random probability is below the mutation threshold, the position is flipped, representing a change in strategy.

As the search progresses further, the pigeons start converging towards desirable destination areas. The desirable_destination_center method calculates the centre point of these areas by averaging the positions of the top pigeons in the population. This center point becomes a reference for updating the positions of other pigeons using the update_path method. If a random probability is below the center value for a specific dimension, the position is flipped to shift toward the desirable destination.

The fitness method is used to evaluate the effectiveness of each pigeon's position. If the fitness value has not been calculated previously, it invokes a function to measure the pigeon's True Positive Rate, False Positive Rate, and total features. These values are then combined using specific weights to calculate the fitness score. The fitness value is stored and returned. Other methods within the Tanh_Pigeon algorithm class handle identification, string representation, hashing, comparison, attribute retrieval, and position retrieval for the pigeons.

The Tanh_Pigeon algorithm class combines the Tanh activation function with the Pigeon-Inspired Optimization algorithm to guide the pigeons' positions and velocities towards optimal solutions. Through iterative updates and fitness evaluations, the pigeons collectively work towards finding the most suitable features for the given problem.

C. *Performance metrics*

Table 1 provided below compares the performance of two algorithms, the Pigeon and Tanh_Pigeon algorithm, across three different datasets: NSL-KDD, CICIDS2017, and CSE-CIC-IDS2018. The table above shows a performance comparison of the two algorithms, Pigeon and Tanh_Pigeon, on three different network intrusion detection datasets: NSL KDD, CICIDS2017, and CSE-CIC-IDS2018. The metrics used to compare algorithm performance are True Positive Rate (TPR), False Positive Rate (FPR), coordinate points on the Receiver Operating Characteristic (ROC) curve (in the form of (TPR, FPR)), accuracy, and F-score.

For the NSL KDD dataset, Pigeon's algorithm achieves the highest TPR of 0.636 and the lowest FPR of 0.029, resulting in a point (0.63, 0.03) on the ROC curve. The overall accuracy of this algorithm is 0.7689, and the F-score is 0.768. On the other hand, the Tanh_Pigeon algorithm has better performance with a TPR of 0.721, an FPR of 0.029, a point (0.72, 0.05) on the ROC curve, an accuracy of 0.8172, and an F-score of 0.8172.

For the CICIDS2017 dataset, the Pigeon algorithm achieves a TPR of 0.657, an FPR of 0.026, a point (0.66, 0.03) on the ROC curve, an accuracy of 0.7842, and an F-score of 0.7838. As a comparison, the Tanh_Pigeon algorithm shows an increase in performance with a TPR of 0.745, an FPR of

0.026, a point (0.75, 0.04) on the ROC curve, an accuracy of 0.8304, and an F-score of 0.8304.

For the CSE-CIC-IDS2018 dataset, the Pigeon algorithm obtains a TPR of 0.674, an FPR of 0.025, points (0.67, 0.03) on the ROC curve, an accuracy of 0.7947, and an F-score of 0.7943. Once again, the Tanh_Pigeon algorithm outperforms Pigeon with a TPR of 0.767, an FPR of 0.025, a point (0.77, 0.04) on the ROC curve, an accuracy of 0.8410, and an F-score of 0.8410.

Overall, this comparison table shows that the Tanh_Pigeon algorithm consistently outperforms Pigeon's algorithm on all three datasets, achieving higher TPR, comparable FPR, better points in the ROC curve, higher accuracy, and better F-score. These results indicate that the Tanh_Pigeon algorithm is a more effective approach for network intrusion detection across various datasets than the Pigeon algorithm.

Analyzing the performance of the two algorithms on three different datasets can assist stakeholders in choosing a more efficient and effective method for network intrusion detection. The Tanh_Pigeon algorithm can be a better choice because it performs consistently better than the Pigeon algorithm.

TABLE I
COMPARISON TABLE FOR THE NSL KDD, CICIDS2017, AND CSE-CIC-IDS2018 DATASETS

Dataset	Algorithm	Highest TPR	Lowest FPR	(TPR, FPR)	Accuracy	F-score
NSL KDD	Pigeon	0.636	0.029	(0.63, 0.03)	0.768	0.768
	Tanh_Pigeon	0.721	0.029	(0.72, 0.05)	0.817	0.817
CICIDS 2017	Pigeon	0.657	0.026	(0.66, 0.03)	0.784	0.783
	Tanh_Pigeon	0.745	0.026	(0.75, 0.04)	0.830	0.830
CSE-CIC-IDS2018	Pigeon	0.674	0.025	(0.67, 0.03)	0.794	0.794
	Tanh_Pigeon	0.767	0.025	(0.77, 0.04)	0.841	0.841

The differences between the traditional PIO algorithm and the Tanh_Pigeon algorithm are how they update velocities and handle exploration and exploitation during optimization. The PIO algorithm is a nature-inspired optimization algorithm that takes inspiration from the flocking behavior of pigeons. In the PIO algorithm, a group of "pigeons" (representing potential solutions) iteratively search for the optimal solution to a given problem. Each pigeon has its position and velocity in the solution space.

The Tanh_Pigeon algorithm modifies the traditional PIO algorithm, where the Tanh (Hyperbolic Tangent) activation function is integrated into the velocity update stage. This modification aims to improve the exploration and exploitation capabilities of the pigeons during the optimization process. The main difference between the traditional PIO and Tanh_Pigeon algorithms is how they update velocities. Integrating the Tanh function in the Tanh_Pigeon algorithm provides additional benefits, such as improved convergence speed and better exploration and exploitation capabilities during optimization. The Tanh_Pigeon algorithm is designed to enhance the efficiency and effectiveness of the optimization process, making it a promising approach for solving complex optimization problems in various domains.

IV. CONCLUSION

The Tanh_Pigeon algorithm consistently surpasses the original Pigeon algorithm in terms of True Positive Rate (TPR), accuracy, and F-score, emphasizing its superior capability to accurately identify positive instances while maintaining a balance between precision and recall. This enhanced performance can be attributed to the Tanh activation function, which enables the model to learn non-linear patterns and generalize better to unseen data. However, it is important to consider that the False Positive Rate (FPR) for the Tanh_Pigeon algorithm is marginally higher across each dataset, suggesting a trade-off between improved detection abilities and an increased rate of false positives. The overall performance improvements indicate that optimizing the Pigeon-Inspired Algorithm with the Tanh function represents a promising approach for enhancing the performance of Intrusion Detection Systems on the NSL-KDD, CICIDS2017, and CSE-CIC-IDS2018 datasets. The Tanh_Pigeon algorithm's robustness and adaptability can be further improved by incorporating additional strategies for mitigating false positives, such as ensemble techniques or a combination of supervised and unsupervised learning methods.

Moreover, considering the ever-evolving landscape of cyber threats, it is crucial to continually update the training datasets to ensure that the Tanh_Pigeon algorithm effectively detects new attack patterns. This can be achieved by incorporating real-world network traffic data and state-of-the-art data augmentation and feature engineering techniques. By doing so, the Tanh_Pigeon algorithm can be better equipped to handle diverse cyber threats, ultimately leading to a more secure network environment.

Tanh_Pigeon algorithm exhibits significant potential for enhancing the performance of Intrusion Detection Systems on the NSL-KDD, CICIDS2017, and CSE-CIC-IDS2018 datasets. By addressing the trade-off between improved detection capabilities and an increased rate of false positives, refining the algorithm through hyperparameter optimization, incorporating real-world network traffic data, and exploring synergies with other advanced machine learning techniques, the Tanh_Pigeon algorithm can become an invaluable tool in the ongoing battle against cyber threats.

REFERENCES

[1] A. Aljumah, "IoT-based intrusion detection system using convolution neural networks," PeerJ Computer Science, vol. 7, p. e721, Sep. 2021, doi: 10.7717/peerj-cs.721.

[2] S. Otoum, B. Kantarci, and H. T. Mouftah, "On the Feasibility of Deep Learning in Sensor Network Intrusion Detection," IEEE Networking Letters, vol. 1, no. 2, pp. 68–71, Jun. 2019, doi:10.1109/lnet.2019.2901792.

[3] G. Vaidya, A. Nambi, T. V. Prabhakar, V. Kumar T, and S. Sudhakara, "IoT-ID: A Novel Device-Specific Identifier Based on Unique Hardware Fingerprints," 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), Apr. 2020, doi: 10.1109/iotdi49375.2020.00026.

[4] I. Ahmed Khan, D. Pi, A. K. Bhatia, N. Khan, W. Haider, and A. Wahab, "Generating realistic IoT-based IDS dataset centred on fuzzy qualitative modelling for cyber-physical systems," Electronics Letters, vol. 56, no. 9, pp. 441–444, Apr. 2020, doi: 10.1049/el.2019.4158.

[5] A. Alhowaide, I. Alsmadi, and J. Tang, "Ensemble Detection Model for IoT IDS," Internet of Things, vol. 16, p. 100435, Dec. 2021, doi:10.1016/j.iot.2021.100435.

[6] A. Kim, M. Park, and D. H. Lee, "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection," IEEE Access, vol. 8, pp. 70245–70261, 2020, doi: 10.1109/access.2020.2986882.

[7] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "A comprehensive deep learning benchmark for IoT IDS," Computers & Security, vol. 114, p. 102588, Mar. 2022, doi:10.1016/j.cose.2021.102588.

[8] A. Y. Hussein, P. Falcarin, and A. T. Sadiq, "Enhancement performance of random forest algorithm via one hot encoding for IoT IDS," Periodicals of Engineering and Natural Sciences (PEN), vol. 9, no. 3, p. 579, Aug. 2021, doi: 10.21533/pen.v9i3.2204.

[9] O. O. Akinola, A. E. Ezugwu, J. O. Agushaka, R. A. Zitar, and L. Abualigah, "Multiclass feature selection with metaheuristic optimization algorithms: a review," Neural Computing and Applications, vol. 34, no. 22, pp. 19751–19790, Aug. 2022, doi:10.1007/s00521-022-07705-4.

[10] "Stacked Ensemble-IDS Using NSL-KDD Dataset," Journal of Pharmaceutical Negative Results, vol. 13, no. SO3, Jan. 2022, doi:10.47750/pnr.2022.13.s03.057.

[11] D. Protić, "Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ datasets," Vojnotehnicki glasnik, vol. 66, no. 3, pp. 580–596, 2018, doi: 10.5937/vojtehg66-16670.

[12] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," *International Journal of Engineering and Technology(UAE)*, vol. 7, no. 3.24 Special Issue 24, 2018.

[13] A. Boukhamla and J. C. Gaviro, "CICIDS2017 dataset: performance improvements and validation as a robust intrusion detection system testbed," International Journal of Information and Computer Security, vol. 16, no. 1/2, p. 20, 2021, doi: 10.1504/ijics.2021.117392.

[14] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset," IEEE Access, vol. 9, pp. 22351–22370, 2021, doi:10.1109/access.2021.3056614.

[15] A. Aldallal, "Toward Efficient Intrusion Detection System Using Hybrid Deep Learning Approach," Symmetry, vol. 14, no. 9, p. 1916, Sep. 2022, doi: 10.3390/sym14091916.

[16] J.-S. Pan, A.-Q. Tian, S.-C. Chu, and J.-B. Li, "Improved binary pigeon-inspired optimization and its application for feature selection," Applied Intelligence, vol. 51, no. 12, pp. 8661–8679, Apr. 2021, doi:10.1007/s10489-021-02302-9.

[17] A. L. Bolaji, F. Z. Okwonu, P. B. Shola, B. S. Balogun, and O. D. Adubisi, "A Modified Binary Pigeon-Inspired Algorithm for Solving the Multi-dimensional Knapsack Problem," Journal of Intelligent Systems, vol. 30, no. 1, pp. 90–103, Jul. 2020, doi: 10.1515/jisys-2018-0450.

[18] A.-Q. Tian, S.-C. Chu, J.-S. Pan, and Y. Liang, "A Novel Pigeon-Inspired Optimization Based MPPT Technique for PV Systems," Processes, vol. 8, no. 3, p. 356, Mar. 2020, doi: 10.3390/pr8030356.

[19] Q. Feng et al., "Resilience optimization for multi-UAV formation reconfiguration via enhanced pigeon-inspired optimization," Chinese Journal of Aeronautics, vol. 35, no. 1, pp. 110–123, Jan. 2022, doi:10.1016/j.cja.2020.10.029.

[20] H. Qiu and H. Duan, "A multi-objective pigeon-inspired optimization approach to UAV distributed flocking among obstacles," Information Sciences, vol. 509, pp. 515–529, Jan. 2020, doi:10.1016/j.ins.2018.06.061.

[21] B. H. Wang, D. B. Wang, and Z. A. Ali, "A Cauchy mutant pigeon-inspired optimization–based multi-unmanned aerial vehicle path planning method," Measurement and Control, vol. 53, no. 1–2, pp. 83–92, Jan. 2020, doi: 10.1177/0020294019885155.

[22] Z. Zhao et al., "Hierarchical Pigeon-Inspired Optimization-Based MPPT Method for Photovoltaic Systems Under Complex Partial Shading Conditions," IEEE Transactions on Industrial Electronics, vol. 69, no. 10, pp. 10129–10143, Oct. 2022, doi:10.1109/tie.2021.3137595.

[23] A.-Q. Tian, S.-C. Chu, J.-S. Pan, H. Cui, and W.-M. Zheng, "A Compact Pigeon-Inspired Optimization for Maximum Short-Term Generation Mode in Cascade Hydroelectric Power Station," Sustainability, vol. 12, no. 3, p. 767, Jan. 2020, doi:10.3390/su12030767.

[24] A. F. Jabbar and I. J. Mohammed, "BotDetectorFW: an optimized botnet detection framework based on five features-distance measures supported by comparisons of four machine learning classifiers using CICIDS2017 dataset," Indonesian Journal of Electrical Engineering and Computer Science, vol. 21, no. 1, p. 377, Jan. 2021, doi:10.11591/ijeecs.v21.i1.pp377-390.

[25] A. Alsahaf, N. Petkov, V. Shenoy, and G. Azzopardi, "A framework for feature selection through boosting," Expert Systems with Applications, vol. 187, p. 115895, Jan. 2022, doi:10.1016/j.eswa.2021.115895.

[26] M. Mera-Gaona, D. M. López, R. Vargas-Canas, and U. Neumann, "Framework for the Ensemble of Feature Selection Methods," Applied Sciences, vol. 11, no. 17, p. 8122, Sep. 2021, doi:10.3390/app11178122.

[27] Z. Yang and B. Y. C. Hon, "An Improved Modified Extended tanh-Function Method," Zeitschrift für Naturforschung A, vol. 61, no. 3–4, pp. 103–115, Apr. 2006, doi: 10.1515/zna-2006-3-401.

[28] X. Wang, J. Wu, Y. Wang, and C. Chen, "Extended Tanh-Function Method and Its Applications in Nonlocal Complex mKdV Equations," Mathematics, vol. 10, no. 18, p. 3250, Sep. 2022, doi:10.3390/math10183250.

[29] E. H.M. Zahran and M. M.A. Khater, "Modified extended tanh-function method and its applications to the Bogoyavlenskii equation," Applied Mathematical Modelling, vol. 40, no. 3, pp. 1769–1775, Feb. 2016, doi: 10.1016/j.apm.2015.08.018.

[30] S. A. Dukhnovsky, "The tanh-function method and the (G'/G)-expansion method for the kinetic McKean system," *Differencialnie Uravnenia i Protsesy Upravlenia*, vol. 2021, no. 2, 2021.

[31] M. S. Ahmed, A. A. S. Zaghrout, and H. M. Ahmed, "Travelling wave solutions for the doubly dispersive equation using improved modified extended tanh-function method," Alexandria Engineering Journal, vol. 61, no. 10, pp. 7987–7994, Oct. 2022, doi: 10.1016/j.aej.2022.01.057.

[32] S. A. Dukhnovsky, "A self-similar solution and the tanh-function method for the kinetic Carleman system," Buletinul Academiei de Ştiinţe a Republicii Moldova. Matematica, no. 1(98), pp. 99–110, Jul. 2022, doi: 10.56415/basm.y2022.i1.p99.

[33] U. Sadiya, M. Inc, M. A. Arefin, and M. H. Uddin, "Consistent travelling waves solutions to the non-linear time fractional Klein–Gordon and Sine-Gordon equations through extended tanh-function approach," Journal of Taibah University for Science, vol. 16, no. 1, pp. 594–607, Jun. 2022, doi: 10.1080/16583655.2022.2089396.

[34] K. K. Ahmed, N. M. Badra, H. M. Ahmed, and W. B. Rabie, "Soliton Solutions and Other Solutions for Kundu–Eckhaus Equation with Quintic Nonlinearity and Raman Effect Using the Improved Modified Extended Tanh-Function Method," Mathematics, vol. 10, no. 22, p. 4203, Nov. 2022, doi: 10.3390/math10224203.

[35] A. Devarakonda, N. Sharma, P. Saha, and S. Ramya, "Network intrusion detection: a comparative study of four classifiers using the NSL-KDD and KDD'99 datasets," Journal of Physics: Conference Series, vol. 2161, no. 1, p. 012043, Jan. 2022, doi: 10.1088/1742-6596/2161/1/012043.

[36] S. Choudhary and N. Kesswani, "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT," Procedia Computer Science, vol. 167, pp. 1561–1573, 2020, doi:10.1016/j.procs.2020.03.367.

[37] L. Dhanabal and S. P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, 2015.

[38] B. Samatha, T. Syamsundararao, and N. Karyemsetty, "Deep Learning Based Intrusion Prevention System in Vehicular Network," Review of Computer Engineering Research, vol. 9, no. 3, pp. 169–180, Sep. 2022, doi: 10.18488/76.v9i3.3145.

[39] A. A. Mamun, S. N. Ananna, P. P. Gharami, T. An, and Md. Asaduzzaman, "The improved modified extended tanh-function method to develop the exact travelling wave solutions of a family of 3D fractional WBBM equations," Results in Physics, vol. 41, p. 105969, Oct. 2022, doi: 10.1016/j.rinp.2022.105969.

[40] L. Md. B. Alam, X. Jiang, and A.-A.- Mamun, "Exact and explicit traveling wave solution to the time-fractional phi-four and (2+1) dimensional CBS equations using the modified extended tanh-function method in mathematical physics," Partial Differential Equations in Applied Mathematics, vol. 4, p. 100039, Dec. 2021, doi:10.1016/j.padiff.2021.100039.