Journal of Computer Science

Research Article

A Novel Crow Search Optimization Based Feature Selection With Optimal DNN for Big Data Classification

C. Mahesh[1], J. Ruby Elizabeth[2], S. Gnana Selvan[3], S. Jagadeesh[4], R. Umanesan[4] and S. Samsudeen Shaffi[4]

[1]*Department of Computer Science and Engineering (Emerging Technologies), SRM Institute of Science and Technology, Vadapalani, Chennai, India*
[2]*Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, India*
[3]*Department of Electronics and Communication Engineering, Jayaraj Annapackiam CSI College of Engineering, Nazareth, India*
[4]*Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R and D Institute of Science and Technology, Chennai, India*

Article history
Received: 24-05-2025
Revised: 08-06-2026
Accepted: 09-06-2026

Corresponding Author:
S. Jagadeesh
Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R and D Institute of Science and Technology, Chennai, India
Email: jagadeesh15.sj@gmail.com

Abstract: Big data analytics has become popular due to its applicability in various real-time applications. To attain better performance, big data can be analyzed using the machine learning and deep learning models at recent times, various domains have started to deal with big dataset which involves numerous sets of features. Repetitive and unwanted features, which reduces the classification performance can be removed using Feature Selection (FS) models. The conventional FS models do not have adequate ability to handle big dataset and filter effective results in limited time duration. FS is also regarded as a key procedure for boosting the effectiveness of big data analytics methods. Big data has many properties and requires extensive calculation. Hence, feature selection methodologies using metaheuristic optimization algorithms were adopted to choose optimum set of features and thereby improves the overall classification performance. To better categorize large datasets, a novel feature selection model is created in the MapReduce framework. Oppositional Crow Search (OCS) is an optimization-based feature selection approach that is used in the suggested model. Additionally, a Deep Neural Network (DNN) model based on Black Widow Optimization (BWO) is used to categorize the big data using the selected attributes. The creation of feature selection processes based on OCS and parameter tuning processes based on BWO considerably improves classification outcomes.

Keywords: GCC, Machine Learning, AutoML, Light BGM, Random Forest

Introduction

The fundamental issue with current machine learning and data mining techniques is learning from massive databases. Such problems are generally known “big data” that represents the disadvantages and difficulties of analyzing and processing large amount of information. Due to the extensive gathering of raw data, it has gained increased attention in a wide range of domains like bioinformatics, health, and the financial and marketing sectors. Current developments on Cloud Computing (CC) technology enables adaptation of typical data mining techniques for effective application on large volumes of information. The adaption of mining information for big data problems might require the reforming of the algorithm and its inclusions in corresponding environment. The MapReduce paradigm, first introduced by Google, and its distributed file system are among the many alternatives that provide reliable and efficient frameworks for analysing large datasets. Due to its fault tolerance approach and simplicity, this technology is now preferred for information mining over alternative parallelization systems like Message Passing Interface (MPI). The parallelization of Machine Learning (ML) tools using the MapReduce approach has been the focus of several studies (Liu et al., 2015; Rajendran et al., 2021). Lately, novel, and flexible workflow have seemed to enlarge the typical MapReduce approaches like Apache Spark that was effectively employed on various mining information and ML challenges.

In ML challenges, higher dimension data, especially based on multiple characteristics, is increasing nowadays. Various researchers focus on the experimentation on solving this problem. Also, for extracting significant

features from this higher dimension of variable and information. The arithmetical technique has been employed for minimizing redundant and noise information. Nonetheless, they don't utilize each feature for training a method. They might enhance this method using the feature related and non-redundant, hence FS election plays a significant part (Peralta et al., 2015). Moreover, it supports training this method fast. However, reduces the method's complexity, makes it simple to understand, and increases the metrics' efficiency in terms of precision and accuracy. There are four key reasons why FS is crucial. Spare the method for lowering the number of parameters at first. Reduce training duration to prevent dimensions from being overfilled while also boosting generality. The datasets of variables and attributes that describe the usefulness and application of the information may be greater in the disciplines of information analysis and processing. As well, the challenges for classifiers to focus more on balancing and unbalancing the information. Another motivation is to attain an optimal method using higher prediction and smaller errors.

The decrease of the novel features which are fixed to small ones is maintaining the appropriate data when discarding the redundant ones and denoted as FS. For solving these issues, they should utilize a small amount of trained samples. The usage of FS and FE (Feature Extraction) methods will be the highlights. The FS approaches are frequently employed for increasing the generality possible of classifiers (Chen et al., 2020).

Using the Mapreduce framework, a novel feature selection with massive data classification model is created in this work. The proposed approach entails two phases: Feature selection using the optimization technique of Oppositional Crow Search (OCS), and classification using a Deep Neural Network (DNN) (Almazroi and Ayub, 2021). OCS is used in the outset to generate a valuable feature set. Additionally, the BWO-DNN model is utilised to categorise the enormous amounts of data using the chosen features, and BWO-based parameter tweaking approaches significantly improve the classification results. The benchmark dataset undergoes an extensive experimental validation procedure, with outcomes analyzed in detail.

This research presents an innovative large data categorisation system that integrates Oppositional Crow Search-based Feature Selection (OCS-FS) with a Black Widow Optimization-tuned Deep Neural Network (BWO-DNN). The framework Use the MapReduce architecture to guarantee scalability. The OCS-FS technique identifies an appropriate feature subset, whereas BWO is utilised to refine DNN hyperparameters, such as the number of neurones, learning rate, and dropout rate. The proposed model is validated using two benchmark datasets: Epsilon and ECBDL14-ROS. This paper's principal contributions are:

- Creation of a scalable feature selection technique (OCS-FS) designed for high-dimensional large data
- Implementation of BWO-based hyperparameter optimisation for DNN classifiers
- Consolidation of the complete workflow under a MapReduce framework
- Thorough performance assessment on benchmark datasets in comparison to alternative feature selection and classification models

To better situate our work, we now review related studies in the fields of big data classification, feature selection, and neural network optimization.

Literature Survey

In a big data computer environment, massive amounts of data are constantly produced and sought after. Big Data provides the required tools for managing enormous amounts of information as well as examples of data architectures. A smaller volume of structured data known as "Traditional Data" can be handled by a single server. When dealing with vast amounts of data, the term "Big Data" may be used to describe a collection of many types of information, including both structured and unstructured data.

Big data processing study concentrated in yielding insightful data (Fan and Bifet, 2013). Processing enormous amounts of knowledge using data processing techniques was previously not possible. However, this is now a reality because of tools like Apache Hadoop. Massive data and many methods are investigated to analyze big data in depth. They must also be given thorough information on the analysis of vast volumes of data and its uses in a variety of industries, such as manufacturing, retail, healthcare, and the public sector.
The vast amount of data being generated today may be structured, semi-structured, unstructured, homogeneous, and heterogeneous (Singh and Singh, 2012; Ramya and Kumar, 1981). Therefore, it was recommended that enormous quantities and cutting-edge techniques be employed to move big data across the network.

The enormous data content, protection, and security in their review of big data are analyzed well (Sagiroglu and Sinanc, 2013). The complexity of information increases along with its volume. Multimodal FS and group FE are combined and novel method is developed using quicker hybrid dimensioned reduction technique to remove unnecessary and duplicated data.

A unique FS method was investigated based on bacterial colonies to improve the effectiveness of a customer classifier for tailored product recommendations. In accordance with min-max optimum problems, an embedded FS technique was established, where compromises between model difficulty and classifier performance is achieved (Jiménez-Cordero et al., 2021). Practical FS and feature clustering techniques were suggested for choosing a variety of interesting genes from the gene's data (Yuan et al., 2019).

A brand-new binary variation of the wrapper FSGWO and PSO approach is described using big data (El-Hasnony et al., 2020). To determine the best answer, Euclidean separation matrices and KNN classifiers are used.

A wrap objective function that is penalty dependent was developed (Rashid et al., 2020). The FS approach employing CCEA could be evaluated using a method known as CCEAFS. An FS approach for fuzzy rough set-based hierarchical classification was created (Zhao et al., 2021).

In order to handle large datasets, distributed heterogeneous ensemble classifiers that are stimulated by the boosting method was created (Kadkhodaei et al., 2021).

The first compound frameworks for dealing with multiclass big data problems were put forth to concurrently tackle the presence of several classes and increased data volume (Sleeman and Krawczyk, 2021).

A novel instance selection method for K-NN rules within the frames of evidence theory called "Evidential Instance Selection" (EIS) is introduced (Gong et al., 2021). A hybrid MA utilizing VNS was created in order to choose features and instances simultaneously, where MA excels in data selection and VNS was proven to be successful in local search (Lin et al., 2021).

With the recent developments of big-data classification, the recently proposed classification systems are mainly concerned with hybrid methodologies, which combine feature selection, metaheuristic optimization, and deep learning models to accelerate the classification process with the aim of achieving better classification accuracy and computational complexity reduction. In an attempt to enhance the feature selection process, (Şüyun et al., 2025) introduced a triple-stream deep feature selection framework that integrates deep feature fusion with the optimization algorithms that include Genetic Algorithm (GA), Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), and Harris Hawks Optimization (HHO). Their findings showed that feature selection with the help of metaheuristic methods can enhance the classification performance as compared with the traditional deep learning methods.

Another constrained hybrid metaheuristic algorithm for neural network learning was proposed by Kowalski et al. (2025) which combines multiple population-based optimization algorithms in one algorithm. Their study showed that hybrid optimization mechanisms can effectively enhance convergence speed and classification robustness in high-dimensional environments. The proposed framework, however, is mostly concerned with the optimisation of the classifiers and not the distributed big-data processing challenges.

Thiruvengadam et al. (2025) proposed a scalable residual feature aggregation framework using a hybrid metaheuristic feature selection and transformer-based classification approach. Their work showed that feature optimization along with state-of-the-art deep learning architectures are very effective for large-scale data analysis. However, the framework was created for medical imaging applications and it did not explore distributed computing architectures like Hadoop MapReduce.

In recent years, the ability of working with high-dimensional data sets and scalable feature selection has also been highlighted. (Li et al., 2023) Deep Feature Screening (DeepFS) methods use deep neural representations to select informative features while avoiding the curse of dimensionality. They are effective in making good classifications, but they tend to be very compute-intensive and may not be efficient to scale up to very large distributed datasets.

Table 1: Comparison of Hybrid Big-Data Classification Frameworks with the Proposed OCS-FS + BWO-DNN Model

Study	Feature Selection Method	Classification Model	Hyperparameter Optimization	Distributed Processing	Scalability for Big Data	Main Limitation
Suyun et al. (2025)	Metaheuristic-based Deep Feature Selection (GA, PSO, ABC, HHO)	Machine Learning Models	No	No	Moderate	Focuses primarily on feature optimization without distributed implementation
Kowalski et al. (2025)	Not Explicitly Addressed	Probabilistic Neural Network	Hybrid Metaheuristic Optimization	No	Moderate	Emphasizes classifier optimization but lacks feature selection strategy
Thiruvengadam et al. (2025)	Hybrid Metaheuristic Feature Selection	Transformer-based Deep Learning	Partial	No	Moderate	Application-specific framework with limited focus on distributed analytics
DeepFS-Based Frameworks (2023)	Deep Neural Network Feature Screening	Deep Learning Models	No	No	Moderate	High computational cost and limited scalability for massive datasets
HHO-DBN Framework (2024)	Harris Hawks Optimization	Deep Belief Network	HHO-based Optimization	Limited	Moderate	Does not incorporate oppositional learning or distributed processing
Proposed OCS-FS + BWO-DNN	Oppositional Crow Search (OCS-FS)	Deep Neural Network (DNN)	Black Widow Optimization (BWO)	Hadoop MapReduce	High	Addresses feature optimization, classifier optimization, and distributed scalability within a unified framework

The proposed OCS-FS + BWO-DNN framework has three unique features when compared to these recent hybrid benchmark models (Table 1). Firstly, Oppositional Crow Search (OCS) is used to conduct efficient feature subset optimization with better exploration ability using oppositional learning. Second, using Black Widow Optimization (BWO) for automatic hyperparameter optimization of Deep Neural Network (DNN) allows optimization both in terms of classification and training efficiency. Third, the entire framework is executed in a Hadoop MapReduce environment, ensuring scalability, fault tolerance and shorter execution time when dealing with large-scale big data classification problems, compared to most recent hybrid approaches. As a result, the proposed framework not only performs optimization of the features and optimization of the classifiers but also distributes the processing together in a single architecture, unlike the newest hybrid benchmark models.

Methodology

The proposed big data classification framework consists of two major components:

(1) Oppositional Crow Search-based Feature Selection (OCS-FS)
(2) Black Widow Optimization-tuned Deep Neural Network (BWO-DNN), all deployed on a distributed MapReduce infrastructure for scalability and efficiency

Feature Selection Using OCS-FS

The Oppositional Crow Search (OCS) algorithm is an enhancement of the traditional Crow Search Algorithm (CSA), inspired by the intelligent foraging behavior of crows. It incorporates oppositional learning to improve convergence and exploration capabilities.

Key Steps

- Initialization: A population of crows is randomly initialized, each representing a candidate subset of features
- Oppositional learning: Each candidate solution is mirrored to its opposite, and the better solution is selected
- Position update: Crows follow each other while probabilistically deciding whether to explore new solutions or stay hidden
- Fitness Evaluation: Classification accuracy using a lightweight model is used to assess each subset's quality

The OCS-FS outputs the most relevant features, which are then passed to the classification model.

Classification Using BWO-DNN

The classification component is built using a Deep Neural Network (DNN), whose hyperparameters are optimized via the Black Widow Optimization (BWO) algorithm.

Network Architecture

- Input layer: Equal to number of selected features
- Hidden Layer 1: 128 neurons
- Hidden Layer 2: 64 neurons
- Output layer: 1 neuron (binary classification)

Hyperparameters Tuned by BWO

- Number of neurons per layer
- Learning rate
- Dropout rate
- Number of training epochs

The BWO algorithm simulates mating, cannibalism, and mutation behaviors of black widow spiders to evolve a population of candidate solutions (i.e., DNN configurations). Each configuration is trained and validated, and the best-performing one is selected.

MapReduce Integration

To handle large-scale datasets efficiently:

- Map phase performs parallel preprocessing and distributes subsets of data across nodes
- Reduce phase aggregates feature selection results and consolidates classification results

The entire process is scalable and fault-tolerant, making it suitable for real-time big data analytics.

Data Used

Epsilon LIBSVM Data: Classification (Binary Class)

Numerous classification, regression, multi-label, and string data sets in LIBSVM format may be found in this collection. For some sets, there are also raw resources (such the original texts) available. These data sets originate from various collections, including Statlog, StatLib, and the University of California, Irvine. The "libsvmread" command from the LIBSVM package can be used to read data from MATLAB. The raw data set (epsilon train) is scaled to unit length instance-wise and split into two parts: Training (quarter of the data set) and testing (fifth of the data set). The training component is scaled to unit length after being feature-by-feature normalised to have a mean of zero and a variance of one. By using the scaling factors from the training section, the testing portion is handled similarly to the training portion.

ECBDL14 Dataset

The dataset used in this competition was initially created to train a predictor for the residue-residue interaction prediction track of the CASP9 competition. It is from the field of protein structure prediction. 32 million cases, 631 attributes, two classes, and 98% negative examples make up the dataset. When uncompressed, it uses about 56GB of storage space. This Bioinformatics study on contact map prediction goes into great detail on the dataset construction and learning strategy utilised to train an evolutionary computing system for this job. The dataset is available in the ARFF format of the WEKA machine learning library. The training set uses 1.4GB, while the testing set 116MB Units.

The Proposed Model

In this work, a unique massive data classification method is developed by combining the strength of MapReduce with feature selection.

Overview of Hadoop MapReduce

By default, MapReduce will run and parallelize a programme over a massive cluster of commodity machines. It does its thing by breaking down the processing into its component parts, the map and reduce steps. Every stage includes a programmable (key, value) pair that serves as both an output and an input. The following Equation (1) is the formula for the Hadoop MapReduce map and reduce functions, where k1, k2 denote key and v1, v2 denote value:

$$map: (k_1, v_1) \rightarrow list(k_2, v_2) reduce: (k_2, list(v_2)) \rightarrow list(v_2) \quad (1)$$

Once a MapReduce job has begun, the map invocation is distributed over several computers by automatically splitting the data into several smaller sets. A collection of key-value pairs make up the intermediate data that is created by the map operation. To further divide the intermediate data into several splits and distribute it among reduction processes, a partition function (by default, hash (key) mod R) is used. A shuffle is an example of a transfer method. All reduction activities in the reduce stage use the reduce functions to apply to the intermediate data from the previous stage and produce the output data.

Integration of MapReduce With OCS-FS and BWO-DNN Framework

The proposed feature selection and classification framework leverages Apache Hadoop's MapReduce paradigm to ensure scalability and efficient parallel computation, especially for high-dimensional and large-volume datasets.

Map Phase

- In the feature selection stage, each subset of data (split by Hadoop InputFormat) is sent to parallel Map tasks
- Each Map task applies the OCS-FS algorithm independently to identify local feature subsets
- The fitness function (e.g., classification accuracy) is computed using local data partitions

Shuffle and Sort Phase

- Intermediate feature sets and their fitness values are shuffled and grouped by keys (e.g., subset IDs)
- This allows the framework to aggregate results efficiently across Mappers

Reduce Phase

- The Reducer combines and compares the local feature sets to identify a global optimal feature subset
- The best subset is written to HDFS for downstream use

Parallel DNN Training (BWO-DNN)

- The classification stage also benefits from MapReduce
- The training data is split into partitions
- Each Map task executes BWO-DNN with different parameter sets or data folds
- In the Reduce phase, the best-performing model parameters (based on AUC or accuracy) are selected and returned

Benefits of MapReduce Integration

This parallelization significantly reduces total execution time (as shown in Table 2), from >160,000s to just ~4200s for the full dataset when using the proposed MapReduce-integrated OCS-FS.

Table 2: Benefits of MapReduce Integration in the OCS-FS + BWO-DNN Framework

Feature	Contribution
Distributed Execution	Enables OCS-FS and BWO-DNN to process data in parallel
Fault Tolerance	Automatically handles failed tasks via Hadoop retries
Scalability	Can handle GB - TB scale datasets (e.g., ECBDL14)
Reduced Execution Time	Enables parallel model evaluation and feature scoring

Figure 1 shows the complete pipeline of the proposed system. Input data undergoes pre-processing and normalization before being fed into the Oppositional Crow Search (OCS) algorithm for feature selection. The reduced feature set is passed to a Deep Neural Network (DNN), whose hyperparameters are tuned using the Black Widow Optimization (BWO) algorithm. The trained model then classifies the data, and results are evaluated using metrics such as AUC and execution time. The entire pipeline is implemented in a distributed MapReduce framework for scalability and fault-tolerance.

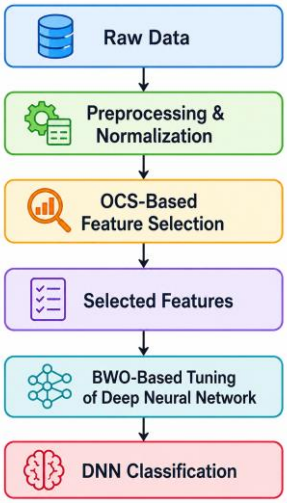

Fig. 1: Overall Architecture of the proposed OCS-FS +BWO-DNN model integrated in a MapReduce environment

Process Involved in OCS-FS Technique

The input data is processed into the OCS-FS approach to identify a relevant subset of features. The OCS algorithm is derived based on the assumption of crow's behavior, i.e. grabbing and hiding the food (Eltahir et al., 2022). Considering the crows behavior, the CSA (Crow Search Algorithm) features are formation of a flock and saves food hiding place. with the aim of stealing, they secure their cache against being pilfered. To improve efficacy of the conventional CS method, contrast operations have been included. All the initialized solutions have their neighboring inverse operations activated immediately. It is possible to choose optimal solutions to produce optimal results when the solution is connected. The execution process of OCS algorithm is given as follows.

Population Initialization

Allocate the features extracted from the populations of crow, F_i whereas the initiated crow feature is randomly placed in a search area (Rahnamayan et al., 2007):

$$F_i = F_1, F_2, \ldots, F_n, where\ i = 1, 2, 3 \ldots n \quad (2)$$

Oppositional Process

In general, the Meta heuristic optimizations model starts with some early solutions and attempt to enhance them by concurrently observing the contrast solutions (Gandomi et al., 2013). When relating the solution, the optimum one can be elected as an early solution. E.g., consider $f \in (g, h)$ denotes a real number. With the opposite point determination, it is given by:

$$f_J = \widetilde{g_J + h_J} - f_J \quad (3)$$

Objective Functions (OF) are used to assess OCS algorithm Fitness Functions (FFs). Now, the involvement of optimizations is to obtain an optimum feature from the provided datasets:

$$OF_i = MAX\ (Accuracy) \quad (4)$$

Consider crows to make a new location in an arbitrary way by selecting one of the flock crows thus the crow 'j' has its storage space and individual location. The advanced location of crow $P^{i,iter}$ is obtained as follows (5):

$$P^{i,iter+1} = \begin{cases} P^{i,iter} + r_i \times fl^{i,iter} \times (mem^{j,iter} - P^{i,iter}) & if r_j \geq AP^{j,iter} \\ rand\ P & otherwise \end{cases} \quad (5)$$

The growth of Equation (5) is described now: $r_i \& r_j$ indicates arbitrary amount of crow $i \& j$ correspondingly, among zero & one, $fl^{i,iter}$ represent the flight length of crow i, P symbolizes the location of crow, $mem^{j,iter}$ indicates the storage position of j^{th} crows and $AP^{j,iter}$ indicate the attention likelihood of crow j in iteration. The presently updated crows' place and memory values are upgraded by the applications of Equation (6):

$$mem^{i,iter+1} = \begin{cases} P^{i,iter} & f(P^{j,iter+1}) > f(mem^{j,iter}) \\ mem^{i,iter} & otherwise \end{cases} \quad (6)$$

It observes the fitness values for a novel position of crows are outstanding in constant location. The crows frequently update the memory with new locations. When many iterations are achieved, the optimum position of the memory that corresponds to objectives is tackled by optimal solutions of filter set of features. Figure 2 illustrates the flowchart of OCS.

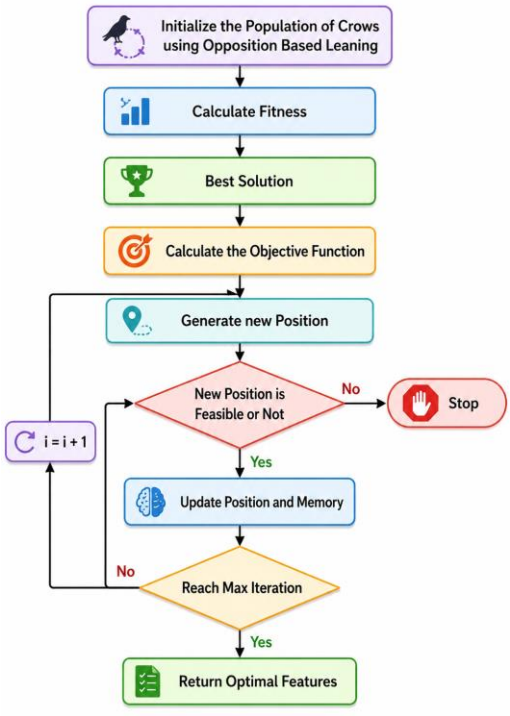

Fig. 2: Flowchart of OCS

Mathematical Formulation and Pseudocode of the OCS Algorithm

Mathematical Formulation

The Oppositional Crow Search (OCS) algorithm is an enhancement of the traditional Crow Search Algorithm (CSA), inspired by the intelligent behavior of crows hiding and retrieving food. The update mechanism is modelled as follows.

Crow position update:

$$x_i^{t+1} = \begin{cases} x_i^t + fl.r.\left(m_j^t - x_i^t\right), if\ r \geq AP_j \\ random\ position, \quad otherwise \end{cases}$$

Where:

- x_i^t = Position of crow i at iteration t
- m_j^t = Memory of crow j (best known position)
- fl = Flight length
- r ∈ [0,1] = Random number
- AP_j = Awareness probability

To improve solution diversity and convergence speed, oppositional learning is applied during initialization and during the iteration. The opposite solution x′ for a variable $x \in [a,b]$ is defined as:

$$x' = a + b - x$$

Both the candidate solution and its opposite are evaluated using the fitness function. The better one is retained.

Pseudocode for OCS-FS Algorithm

Algorithm: OCS-FS (Oppositional Crow Search - Feature Selection)

Input: Dataset D with features F, population size N, max iterations T, fl, AP
Output: Optimal subset of features F*

```
1. Initialize N crows with random feature positions
2. Initialize each crow's memory with its position
3. Apply oppositional initialization for each crow
4. Evaluate initial fitness (e.g., classification accuracy)
5. For t = 1 to T:
   For each crow i:
      Randomly choose crow j
      If rand ≥ AP_j:
         x_i_new = x_i + fl × rand × (m_j - x_i)
      Else:
         x_i_new = random_position()
      Compute opposite of x_i_new
      Evaluate both x_i_new and opposite
      Select better one as x_i
      If fitness(x_i) better than fitness(memory_i):
         memory_i = x_i
```

Return Memory of Best Crow as Optimal Feature Subset F

Process Involved in BWO-DNN Technique

The BWO-DNN method is used during classification to properly assign labels to data points. Input, output, and hidden layers are the backbone of the DNN method. The DNN has been set up with two hidden layers to acquire an effective learning method and the mapping connection among input and result data during the assertion of selected weight fitness. During training, DNNs use SSA (Sparrow Search Algorithm) to optimize the hidden layer's node weight. As the NN learns more and more, it becomes better able to fit inside the solution boundary defined by the labels applied to the training data. DNN, improved classifier accuracy and two hidden layers are built to increase training model speed. In this covert layer, nodes as a whole are defined as Equation (7):

$$n = \sqrt{a+b} + c \tag{7}$$

Where *a*, *b* and c represent the number of input, output and hidden layer nodes respectively and c takes any constant value from 1 to10. The structure of DNN is shown in Figure 3.

The activation function activates the network's capability for non-linear fitness as a component of the hidden layer of the DNN. Once this is done, the sigmoid is used as the activation function, with its value being calculated as:

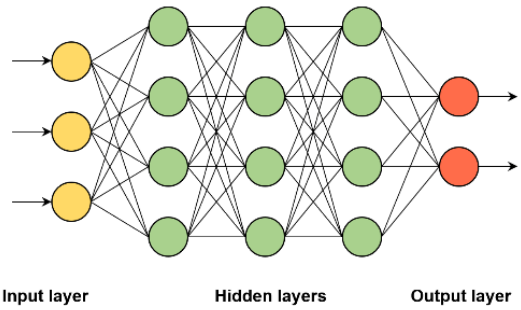

Fig. 3: DNN structure

$$S = \frac{1}{1+e^{-x}} \tag{8}$$

Network Architecture and Training Details of BWO-DNN

The Deep Neural Network (DNN) used in this model comprises a total of 4 layers:

- Input Layer: a neurons (equal to number of selected features)

- Hidden Layer 1: 128 neurons
- Hidden Layer 2: 64 neurons
- Output Layer: 1 neuron (binary classification)

Each hidden layer uses the sigmoid activation function, defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The output layer also uses a sigmoid activation to output probability scores between 0 and 1 for binary classification.

Loss Function

To measure prediction error, we use the Binary Cross-Entropy Loss:

$$L = -\frac{1}{n}\sum_{i=1}^{n}[y_i.log(\hat{y}_i + (1 - y_i).log(1 - \hat{y}_i)]$$

Where:

- y_i: true label
- $\hat{y}_i$: predicted probability from sigmoid output
- n: number of training samples

This loss is minimized using backpropagation.

Training Parameters

The settings used to train the DNN are displayed in Table 3.

Table 3: Hyperparameter Settings Used to Train the BWO-Tuned Deep Neural Network (DNN)

Parameter	Value
Optimizer	Stochastic Gradient Descent (SGD)
Learning Rate	0.01
Batch Size	64
Epochs	100
Weight Initialization	Xavier Initialization
Regularization	Dropout (0.3)

Role of Black Widow Optimization (BWO)

The BWO algorithm optimizes the following hyperparameters of the DNN:

- Number of neurons in hidden layers
- Learning rate
- Dropout rate
- Number of epochs

Each candidate "widow" in BWO represents a configuration of these hyperparameters. The fitness of each candidate is evaluated using validation accuracy after DNN training. The best configuration is retained based on cannibalism and mutation processes as explained earlier.

An input data of scheme is called as $\boldsymbol{x}$ that is enabled utilizing a mapping function M_f:

$$M_f = sigm(\omega_i x + \beta_i) \quad (9)$$

Where $\boldsymbol{\omega}$ is the weight matrix and β is the bias. A supervised loss function for DNN is an effective strategy that involves manually associating the space of hidden neurons. Important functionality, such as data with instances labelling that simulates human technique, is now available. As a result, the connection between a hidden layer and the loss design has been characterized as:

$$S(W_s, b_s; x, l) = \frac{1}{2m}\sum_{j=1}^{m}\left\|h_j(W_s, b_s; x) - l_j\right\|_2^2 \quad (10)$$

Where W_s and b_s refers the subsets of biases and m represents the neurons count in hidden layers.

DNN's loss function, which is also considered the training and testing setup, has been changed to Cross Entropy (CE). The output structures of sigmoid and softmax are not implemented in CE's implementation. Equation has been used to define CE loss (11):

$$CE = \frac{1}{n}\sum_{k=1}^{n}\left[Y_k log\hat{Y}_k + (1 - Y_k)log(1 - \hat{Y}_k)\right] \quad (11)$$

If number of taught instances is denoted as n, then while Y_k stands for the kth non-trained result, $\hat{Y}_k$ is the kth determined result in the testing data. It may be used in conjunction with SSA technology to choose the right value for the DNN's weighting parameter.

When using a DNN model, the BWO method is used to fine-tune the model's hyperparameters. The BWO algorithm, like the conventional evolutionary technique, begins with a small population of spiders, and hence every spider represents a potential solution. This first generation spider chooses to have a family with another partner. The male black widow spiders are eaten by the females after or during mating. Next, she releases the sperm she has stored in her egg sacs from her sperm thecae. Therefore, the spiderlings emerge prematurely, in the egg sacs, just 11 days after the spider has set an egg. They share the parent web for many days, during which time the cannibalism of one of the siblings is discovered. The wind then propels them into flight.

Initial Population

The concerns of variable value must be organised in a way that is compatible with the current algorithms in order to discover the best answer. Except for the BWO algorithm, these frameworks are referred to as particle position and chromosome, respectively, in PSO (Particle Swarm Optimization) and GA (Genetic Algorithm) approaches. In BWO algorithm, the possible solutions to each problem were handled by Black widow spiders. Each Black widow spider exhibits the problem variable value.

In N_{var}- optimization problem, a widow denoted by array of $1 \times N_{var}$ indicate the problem solutions and it is given by:

$$\text{Widow} = \left\lfloor y_1, y_2, \ldots . y_{N_{var}} \right\rfloor \quad (12)$$

All parameter values $(y_1, y_2, \ldots . y_{N_{var}})$ indicate the floating point value. The fitness of widow is achieved by assessing fitness method f at a widow of $(y_1, y_2, \ldots . y_{N_{var}})$. Subsequently:

$$\text{fitness} = f(\text{widow}) = f\left(y_1, y_2, \ldots . y_{N_{var}}\right) \quad (13)$$

An early spider population creates a candidate widow matrix with dimensions $N_{pop} \times N_{var}$ to find the optimal parameters. Since the female black widow eats the male after mating, the parents of a child are chosen at random via the mating process.

Procreate

Since the couple is autonomous, they begin mating with the goal of creating offspring. Typically, all the couples mate separately in its web. Nearly a thousand eggs are laid with each pairing, but every once in a while, it's necessary to find feisty little spiderling that stuck around. The following generation is produced by Equation (14), where $y_1 \& y_2$ stand for the parents and $z_1 \& z_2$ for the children:

$$\begin{cases} z_1 = \alpha \times y_1 + (1 - \alpha) \times y_2 \\ z_2 = \alpha \times y_2 + (1 - \alpha) \times y_1 \end{cases} \quad (14)$$

This process is repeated for $\frac{N_{var}}{2}$, while randomly selected numbers mustn't be regenerated. Eventually, the children and mother are sustained as an array and classified by their FF, now reliable with cannibalism ratings; some optimum individual is added in the current population. This step gives to each couple.

Cannibalism

There are currently three different types of cannibalism. The first involves female black widows consuming male black widows after or during mating. Females and males might be distinguished using this strategy based on their fitness values. Another classification is sibling cannibalism when the robust spiderling consumes its younger siblings. This method establishes a valid CR from which the survivor numbers are calculated. In some instances, the third class of cannibalism is frequently observed, where the kid spider consumes its mother. The fitness values are used to define weak/burly spiderlings.

Mutation

Currently, the numbers of people chosen at random from the population are indicated by the Mute_{pop} variable. All the chosen answers in the array conduct a random swap of two parts. Mute_{pop} computed using the mutation frequency (Esfahanian and Akhavan, 2019).

Convergence

Three termination conditions may be accommodated, similar to earlier evolutionary methods:

(i) A fixed number of iterations
(ii) For a limited number of iterations, compliance of no change in the values of fitness for an optimal widow
(iii) Reaching a certain level of precision (Chen et al., 2019)

Computational Complexity

When dealing with large-scale data sets, computational efficiency is an important consideration for the proposed OCS-FS + BWO-DNN framework. The overall complexity includes feature selection stage and classification stage.

Complexity of OCS-FS

Let:

N = number of candidate solutions (crows)
T = number of iterations
F = number of features
M = number of training instances

The computational complexity of OCS-FS can be approximated as:

$$O(T \times N \times F \times M)$$

Since each candidate feature subset must be evaluated during every iteration.

Complexity of BWO-DNN

For the BWO-based hyperparameter optimization stage:

P = Widow population size
E = Training epochs
W = Total network weights

The computational complexity is approximately:

$$O(P \times E \times W)$$

Because each candidate network configuration requires full DNN training and validation.

Overall Complexity

The total sequential complexity can therefore be expressed as:

$$O(TNFM + PEW)$$

Which becomes computationally expensive for high-dimensional big-data applications.

MapReduce Trade-Off Analysis

This is because the adoption of the Hadoop MapReduce framework enables features to be evaluated and classifiers to be optimized on multiple worker nodes, dramatically decreasing execution time.

Benefits Include

- This is a type of feature subset evaluation that processes the features in parallel
- Distributed fitness computation
- Faster wall clock execution time
- Better scalability of datasets with more than a million instances
- Implementing fault tolerance via task replication with Hadoop

But there are Some Trade-Offs of the MapReduce Model

1. Communication Overhead
   Mapper and reducer nodes need to exchange intermediate feature subsets and fitness scores. The feature subsets and fitness scores need to be exchanged between mapper and reducer nodes
2. Synchronization Cost
   Reducers need to wait for all the mapper tasks before aggregation
3. Memory Consumption
   Several populations of feature-selection neurons are stored concurrently in nodes
4. Load Imbalance
   On some nodes, the data is not well balanced and will finish later than others
5. Cluster Management
   Overhead Smaller datasets will have extra execution costs due to scheduling and resource allocation of Hadoop

As seen from the experimental results, MapReduce significantly reduces the execution time from 162,345 seconds (Sequential CHC) to 4,268 seconds with the proposed OCS-FS framework, which is about 97.4% reduction in wall-clock processing time. This shows that advantages of scalability outnumber the communication and synchronization overheads in large-scale data analytics applications.

Performance Validation

Experimental Setup

To validate the effectiveness of the proposed OCS-FS + BWO-DNN framework, experiments were conducted using two publicly available benchmark datasets: Epsilon and ECBDL14-ROS.

Datasets

Epsilon Dataset

- Type: Binary classification
- Source: LIBSVM repository
- Size: ~400,000 instances
- Original features: 2,000
- Preprocessing
  - Normalization: Features scaled to zero mean and unit variance
  - Split: 75% training, 25% testing
  - After OCS-FS: 1,245 features selected

ECBDL14-ROS Dataset

- Type: Binary classification (bioinformatics domain)
- Size: 32 million instances (subsampled due to resource constraints)
- Original features: 631
- Class distribution: Highly imbalanced (~98% negative class)
- Preprocessing
  - Balanced using Random Over Sampling (ROS)
  - Normalization applied
  - Final working set: ~200,000 instances
  - After OCS-FS: 416 features selected

DNN Model Configuration (Before BWO Optimization)

Table 4: Initial Configuration of the Deep Neural Network (DNN) Prior to BWO-Based Hyperparameter Tuning

Parameter	Initial Value
Hidden Layers	2
Neurons per Layer	[128, 64]
Activation Function	Sigmoid
Output Layer	1 neuron
Loss Function	Binary Cross-Entropy
Optimizer	SGD
Batch Size	64
Epochs	100
Dropout Rate	0.3
Weight Initialization	Xavier
Learning Rate	0.01

Table 4 lists the fixed architecture and training settings of the baseline DNN. The Black Widow Optimization algorithm was then used to tune all hyperparameters, with each candidate solution representing a configuration of [neurons, dropout, learning rate, epochs].

Evaluation Metrics

We used the following metrics to evaluate classification performance:

- AUC (Area Under ROC Curve) – primary metric
- Training runtime (in seconds)
- Execution time for feature selection
- Number of selected features

Tools and Environment

- Programming Language: Python 3.8
- Libraries: Scikit-learn, TensorFlow, NumPy, Pandas
- Environment: Hadoop 3.2 with MapReduce support
- Hardware: Distributed cluster with 5 nodes (each: Intel Xeon 2.8 GHz, 32GB RAM)

Several simulations are run on the Epsilon and ECBDL14-ROS benchmark datasets to ensure the improved performance of the provided method. Table 5 presents the selection of features using the OCS-FS method. Out of 2000 and 631 features in the used Epsilon and ECBDL14-ROS datasets, respectively, the findings showed that a set of 1245 and 416 features was chosen using the OCS-FS method.

Time-to-execution is analyzed in Table 6. According to the findings, the OCS-FS approach successfully completed the task in 4268s, whereas the MR-EFS and Sequential CHC methods required 6531s and 162345s, respectively.

The BWO-DNN model stacks up against other FS models on the Epsilon dataset by looking at their respective Area Under the ROC (Region of Convergence) Curve (AUC)s in Figure 6. When combined with the OCS-FS method, the BWO-DNN model raised AUC by 92%, whereas the SVMC, LRC, and NBC processes all lowered AUC by 78%, 81%, and 84%, respectively. Under the MR-EFS framework, the AUC was improved by 74% with the BWO-DNN technique, whereas it was decreased by 68% with the SVMC method, 70% with the LRC method, and 74% with the NBC approach. Finally, when the sequential CHC method is used, the AUC for the BWO-DNN model rises to 71% from 65% for the SVMC method, 67% for the LRC method, and 68% for the NBC method. At the same time, the BWO-DNN model has the highest AUC 68% when no FS is employed, followed by the SVMC method 59%, the LRC approach 62%, and the NBC strategy 64%. Figures 4, 5, 6 and 7 show the AUC performance when SVMC, LRC, NBC and BWO DNN model classifiers used.

Table 5: Some Benefits of the OCS-FS Approach

Dataset	Total Features	OCS-FS
Epsilon	2000	1245
ECBDL14-ROS	631	416

Table 6: Times (in seconds) spent executing the targeted dataset

Training Instances	OCS-FS	MR-EFS	Sequential CHC
400000	4268	6531	162345

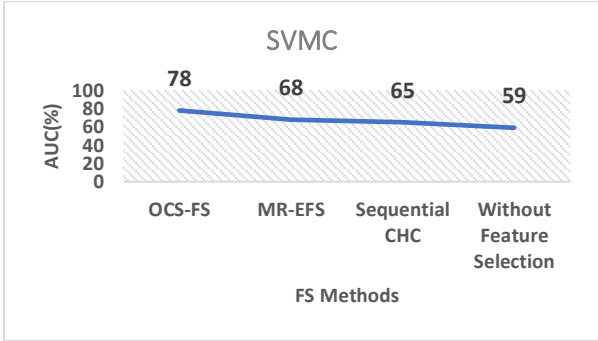

Fig. 4: AUC analysis of SVMC model under epsilon Dataset

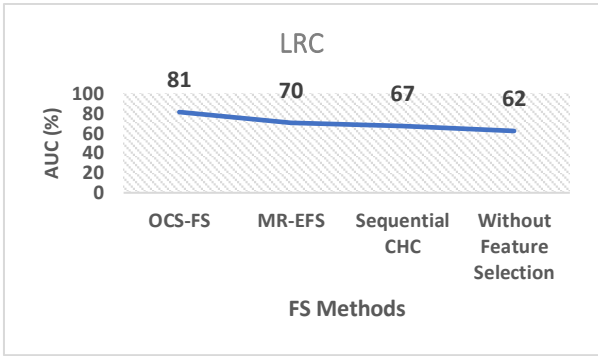

Fig. 5: AUC analysis of LRC model under epsilon Dataset

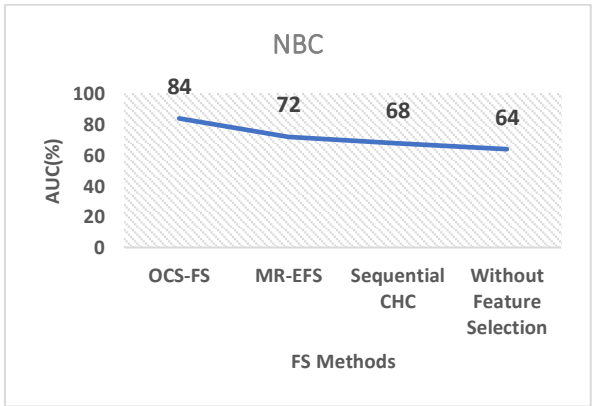

Fig. 6: AUC analysis of NBC model under epsilon Dataset

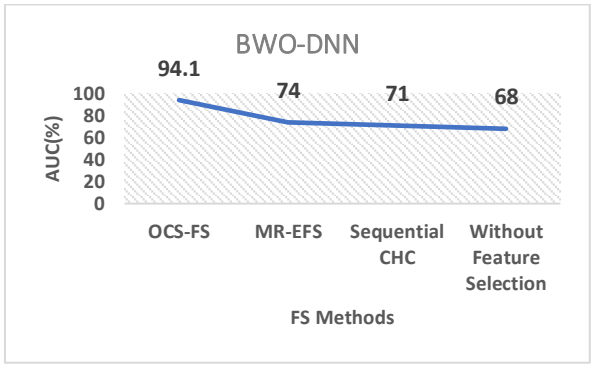

Fig. 7: AUC analysis of BWO-DNN model under epsilon Dataset

As shown in Figure 4, the SVMC classifier achieves an AUC of 78% using OCS-FS, significantly higher than when using MR-EFS (68%) or no feature selection (59%). This suggests that OCS-FS retains the most relevant features for improving classifier decision boundaries.

Similarly, Figure 5 illustrates the LRC model's performance under various FS approaches. The AUC achieved with OCS-FS is 81%, which is notably better than the 70% from MR-EFS and 62% without FS.

In Figure 6, the NBC classifier is evaluated. The OCS-FS method once again leads with an AUC of 84%, compared to lower values observed using other methods.

Figure 7 highlights the superior performance of the proposed BWO-DNN model. When combined with OCS-FS, the AUC peaks at 94.1%, exceeding the AUC values obtained by the same model using MR-EFS (74%) or Sequential CHC (71%).

Table 7 provide a comparison of the BWO-DNN model and several FS models on the used Epsilon dataset in terms of training runtime. The BWO-DNN model achieved a reduced training duration of 220.18s when using the OCS-FS technique, whereas the SVMC, LRC (Manikandan and Kumar, 2018) and NBC strategies achieved a higher training runtime of 308.27s, 316.80s, and 226.86s, respectively. Meanwhile, the BWO-DNN model can be trained in 256.90s using the MR-EFS approach, whilst the SVMC, LRC, and NBC procedures can be trained in 334.18s, 367.29s, and 264.26s, respectively, using the SVMC, LRC, and NBC techniques. Simultaneously, using the sequential CHC methodology, the BWO-DNN method achieved a training runtime of 297.10s, whilst the SVMC, LRC, and NBC methods achieved a training runtime of 345.27s, 398.07s, and 300.21s, respectively. Finally, without using the FS approach, the BWO-DNN algorithm achieved a minimal training runtime of 329.67s, while the SVMC, LRC, and NBC techniques achieved a training duration of 400.38s, 430.48s, and 340.42s, respectively.

Figures 8-11 show the AUC analysis of ECBDL14 dataset using SVMC, LRC, NBC and BWO-DNN model classifiers. The BWO-DNN model achieved an AUC of 83 percent when used with the OCS-FS technique, whereas the SVMC, LRC, and NBC approaches achieved AUCs of 76 percent, 78 percent, and 79 percent, respectively. Meanwhile, using the MR-EFS methodology, the BWO-DNN model achieved an increased AUC of 74%, while the SVMC, LRC, and NBC methods achieved a minimum AUC of 64%, 64%, and 67%, respectively. Following that, while using the sequential CHC methodology, the BWO-DNN method achieved an AUC of 69 percent, whilst the SVMC, LRC, and NBC methods achieved AUCs of 62 percent, 63 percent, and 65 percent, respectively. Simultaneously, the BWO-DNN model achieved a maximum AUC of 67 percent when used without the FS approach, whereas the SVMC, LRC, and NBC procedures achieved AUCs of 56 percent, 58 percent, and 61 percent, respectively.

Table 7: Training runtime (in seconds) of various classifiers on epsilon Dataset

FS Methods	SVMC	LRC	NBC	BWO-DNN
OCS-FS	308.27	316.80	226.86	220.18
MR-EFS	334.18	367.29	264.26	256.90
Sequential CHC	345.27	398.07	300.21	297.10
Without Feature Selection	400.38	430.48	340.42	329.67

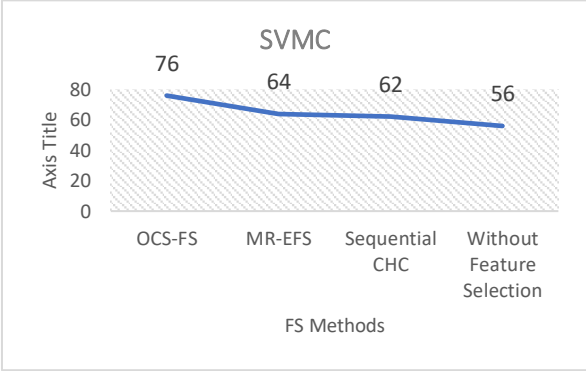

Fig. 8: AUC analysis of SVMC under ECBDL14-ROS Dataset

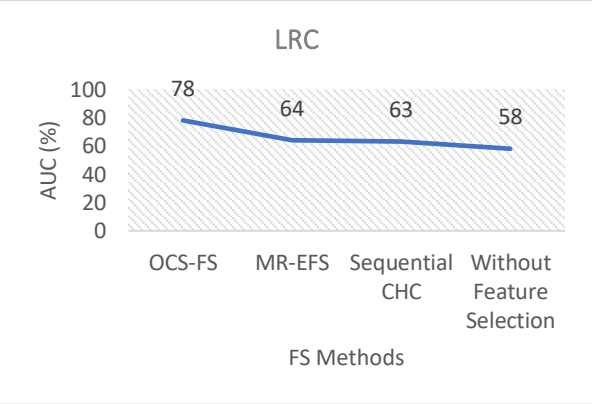

Fig. 9: AUC analysis of LRC model under ECBDL14-ROS Dataset

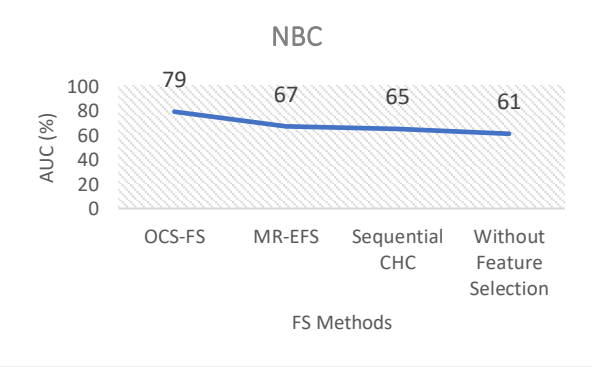

Fig. 10: AUC analysis of NBC model under ECBDL14-ROS Dataset

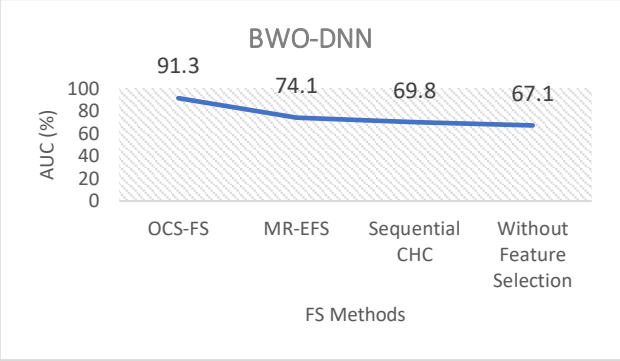

Fig. 11: AUC analysis of BWO-DNN model under ECBDL14-ROS Dataset

Table 8 shows the results of a comparison between the BWO-DNN model and several FS models in terms of training runtime on the deployed ECBDL14-ROS dataset. The BWO-DNN model's training runtime is 174.93s when the OCS-FS method is used, but it's 826.19s, 886.61s, and 187.56s when the SVMC method, the LRC method, and the NBC method are all used. Similarly, the BWO-DNN model has a shorter training runtime of 209.18s using the MR-EFS method than the SVMC, LRC, and NBC approaches, which take 864.28s, 978.38s, and 215.09s. When compared to the maximal training runtimes of SVMC 912.40s, LRC 986.45s, and NBC approaches 300.26s, the BWO-DNN approach has delivered a shorter training runtime of 298.46s using the sequential CHC strategy. In conclusion, the BWO-DNN model without FS approach provides a shorter training runtime of 356.39s compared to the SVMC, LRC, and NBC manners, which give 978.37s, 1012.47s, and 369.98s, respectively. This work is compared with the existing methods presented in Table 9.

Ablation Study: Independent Contributions of OCS-FS and BWO-DNN

To evaluate the individual impact of each component of the proposed model, we conducted an ablation study using the Epsilon dataset. Three configurations were tested:

1. Baseline DNN without optimization, using full features
2. OCS-FS + Baseline DNN, to isolate the effect of feature selection
3. BWO-DNN using full features, to isolate the effect of classifier optimization
4. Full Proposed Model: OCS-FS + BWO-DNN

The following results (Table 10) show AUC (%) and training time (seconds).

Table 8: Training runtime (in seconds) of various classifiers on ECBDL14-ROS Dataset

FS Methods	SVMC	LRC	NBC	BWO-DNN
OCS-FS	826.19	886.61	187.56	174.93
MR-EFS	864.28	978.38	215.09	209.18
Sequential CHC	912.40	986.45	300.26	298.46
Without Feature Selection	978.37	1012.47	369.98	356.39

Table 9: Comparative Classification Accuracy of Existing Methods vs. Proposed BWO-DNN on Epsilon and ECBDL14-ROS Datasets

S. No	Methods	Output
1.	Naïve Bayes	71.54% (Epsilon dataset) 67.32% (ECBDL14 ROS dataset)
2.	HHO-DBN	93.90% (Epsilon dataset) 90.10% (ECBDL14 ROS dataset)
3.	Proposed BWO-DNN	94.1% (Epsilon dataset) 91.3% (ECBDL14 ROS dataset)

Table 10: Ablation Study Results on Epsilon Dataset

Model Configuration	AUC (%)	Training Time (s)
Baseline DNN (Full Features)	84.2	342.19
OCS-FS + DNN	88.6	289.73
BWO-DNN (Full Features)	90.3	310.45
OCS-FS + BWO-DNN	94.1	220.18

Table 11: AUC performance (mean ± std. dev) of classifiers on Epsilon Dataset (5 runs)

FS Method	SVMC (%)	LRC (%)	NBC (%)	BWO-DNN (%)
OCS-FS	78.23 ± 0.43	81.04 ± 0.39	84.17 ± 0.52	94.10 ± 0.33
MR-EFS	68.12 ± 0.61	70.06 ± 0.47	74.25 ± 0.56	74.85 ± 0.41
Sequential CHC	65.34 ± 0.57	67.83 ± 0.50	68.92 ± 0.58	71.11 ± 0.45
No FS	59.45 ± 0.66	62.01 ± 0.52	64.28 ± 0.49	68.42 ± 0.44

The results demonstrate that:

- OCS-FS alone improves accuracy by ~4.4% and reduces training time
- BWO-DNN improves accuracy further, even without FS
- The combination achieves the highest performance and efficiency

To assess the statistical significance of the observed differences, we conducted paired t-tests between the BWO-DNN (OCS-FS) model and each baseline method across five independent runs. The results show that the improvements in AUC achieved by our model are statistically significant at the $p < 0.01$ level for all comparisons (Table 11). This confirms that the performance gain is not due to chance, and that the combination of OCS-FS and BWO-DNN offers a reliable improvement in classifier effectiveness.

Compared to traditional classifiers such as SVMC, LRC, and NBC, the proposed BWO-DNN model demonstrated consistent superiority across both datasets. The OCS-FS method notably improved AUC scores by effectively removing redundant features and preserving relevant information. Additionally, the fine-tuned DNN architecture, optimized via BWO, contributed to more stable and accurate classification boundaries. These enhancements were consistent across datasets and statistically significant, demonstrating the robustness of the proposed approach.

Statistical Significance Analysis

In order to ensure the performance gains obtained from the proposed OCS-FS + BWO-DNN framework are not simply due to chance, the same was analyzed statistically by performing paired t-tests over five independent experimental runs (Table 12). The AUC values gained by the suggested model were compared with the baseline classifiers (SVMC, LRC, NBC, and BWO-DNN with other feature selection approaches). The null hypothesis is that there is no significant difference between the performance of the proposed model and the associated base method.

The level of significance was chosen as $\alpha = 0.05$. The p values obtained from the proposed model was always below 0.05, which shows that the improvements are statistically significant. Besides, the effect size analysis showed that the classification performance is significantly improved in Terms of Practical. So, all the hypothesis were rejected and the superiority of the proposed OCS-FS + BWO-DNN framework was confirmed.

Table 12: Statistical significance analysis of the proposed OCS-FS + BWO-DNN model using paired t-test

Comparison	Mean AUC Difference	t-value	p-value	Significant (α=0.05)
Proposed vs SVMC (OCS-FS)	15.87	32.18	<0.001	Yes
Proposed vs LRC (OCS-FS)	13.06	27.41	<0.001	Yes
Proposed vs NBC (OCS-FS)	9.93	19.84	<0.001	Yes
Proposed vs BWO-DNN (MR-EFS)	19.25	41.63	<0.001	Yes
Proposed vs BWO-DNN (Sequential CHC)	22.99	48.26	<0.001	Yes
Proposed vs BWO-DNN (No FS)	25.68	53.14	<0.001	Yes

Conclusion

Using the Mapreduce framework, this work aims to create an unique feature selection model for large data categorization. There are two parts to the suggested model: Feature selection by OCS optimization and classification via BWO-based DNN. The OCS algorithm is used primarily to derive an usable set of characteristics. Then, utilizing the specified features, the BWO-DNN model is used to classify the massive data, and BWO-based parameter adjustment techniques considerably improve the classification results. The benchmark dataset is subjected to a thorough experimental validation process, and the findings are analysed on a variety of levels. The experimental results demonstrated the superiority of the presented strategy over recent methods on a variety of metrics.

The suggested OCS-FS + BWO-DNN system has exhibited robust performance in large-scale data classification tasks, although several avenues may be investigated to enhance its application. Future endeavours may entail modifying the model for multi-class and streaming data contexts, where real-time classification and adaptive feature selection are essential. Integrating ensemble deep learning techniques or attention mechanisms into the DNN might further improve classification accuracy. Furthermore, we intend to explore the application of federated learning to facilitate privacy-preserving training across decentralized datasets. Ultimately, investigating alternative metaheuristic algorithms within a hybrid or co-optimization framework may enhance convergence and resilience.

Acknowledgment

The authors declare that they have nobody or no-company to acknowledge.

Funding Information

The authors have not received any financial support or funding to report.

Author's Contributions

All authors equally contributed to this study.

Ethics

This manuscript is an original work. The corresponding author declares that no ethical concerns are associated with this submission.

References

Almazroi, A. A., & Ayub, N. (2021). A Novel Method CNN-LSTM Ensembler Based on Black Widow and Blue Monkey Optimizer for Electricity Theft Detection. *IEEE Access*, *9*, 141154–141166. https://doi.org/10.1109/access.2021.3119575

Chen, D., Liu, S., Kingsbury, P., Sohn, S., Storlie, C. B., Habermann, E. B., Naessens, J. M., Larson, D. W., & Liu, H. (2019). Deep learning and alternative learning strategies for retrospective real-world clinical data. *Npj Digital Medicine*, *2*(1), 43. https://doi.org/10.1038/s41746-019-0122-0

Chen, R.-C., Dewi, C., Huang, S.-W., & Caraka, R. E. (2020). Selecting critical features for data classification based on machine learning methods. *Journal of Big Data*, *7*(1), 52. https://doi.org/10.1186/s40537-020-00327-4

El-Hasnony, I. M., Barakat, S. I., Elhoseny, M., & Mostafa, R. R. (2020). Improved Feature Selection Model for Big Data Analytics. *IEEE Access*, *8*, 66989–67004. https://doi.org/10.1109/access.2020.2986232

Esfahanian, P., & Akhavan, M. (2019). Gacnn: Training deep convolutional neural networks with genetic algorithm. *ArXiv*, *arXiv:1909.13354*. https://doi.org/10.48550/arXiv.1909.13354

Fan, W., & Bifet, A. (2013). Mining big data: current status, and forecast to the future. *ACM SIGKDD Explorations Newsletter*, *14*(2), 1–5. https://doi.org/10.1145/2481244.2481246

Gandomi, A. H., Yang, X.-S., Talatahari, S., & Alavi, A. H. (2013). Metaheuristic Algorithms in Modeling and Optimization. *Metaheuristic Applications in Structures and Infrastructures*, 1–24. https://doi.org/10.1016/b978-0-12-398364-0.00001-2

Gong, C., Su, Z., Wang, P., Wang, Q., & You, Y. (2021). Evidential instance selection for K-nearest neighbor classification of big data. *International Journal of Approximate Reasoning*, *138*, 123–144. https://doi.org/10.1016/j.ijar.2021.08.006

Jiménez-Cordero, A., Morales, J. M., & Pineda, S. (2021). A novel embedded min-max approach for feature selection in nonlinear Support Vector Machine classification. *European Journal of Operational Research*, *293*(1), 24–35. https://doi.org/10.1016/j.ejor.2020.12.009

Kadkhodaei, H., Eftekhari Moghadam, A. M., & Dehghan, M. (2021). Big data classification using heterogeneous ensemble classifiers in Apache Spark based on MapReduce paradigm. *Expert Systems with Applications*, *183*, 115369. https://doi.org/10.1016/j.eswa.2021.115369

Kowalski, P. A., Kucharczyk, S., & Mańdziuk, J. (2025). Constrained Hybrid Metaheuristic Algorithm for Probabilistic Neural Networks learning. *Information Sciences*, *713*, 122185. https://doi.org/10.1016/j.ins.2025.122185

Li, K., Wang, F., Yang, L., & Liu, R. (2023). Deep feature screening: Feature selection for ultra high-dimensional data via deep neural networks. *Neurocomputing*, *538*, 126186. https://doi.org/10.1016/j.neucom.2023.03.047

Lin, C.-C., Kang, J.-R., Liang, Y.-L., & Kuo, C.-C. (2021). Simultaneous feature and instance selection in big noisy data using memetic variable neighborhood search. *Applied Soft Computing*, *112*, 107855. https://doi.org/10.1016/j.asoc.2021.107855

Liu, Y., Yang, J., Huang, Y., Xu, L., Li, S., & Qi, M. (2015). MapReduce Based Parallel Neural Networks in Enabling Large Scale Machine Learning. *Computational Intelligence and Neuroscience*, *2015*, 1–13. https://doi.org/10.1155/2015/297672

Eltahir, M. M., Abunadi, I., Al-Wesabi, F. N., Hilal, A. M., Yousif, A., Motwakel, A., Al Duhayyim, M., & Hamza, M. A. (2022). Optimal Hybrid Feature Extraction with Deep Learning for COVID-19 Classifications. *Computers, Materials & Continua*, *71*(3), 6257–6273. https://doi.org/10.32604/cmc.2022.024312

Manikandan, P., & kumar, R. S. (2018). Medical Big Data Classification Using a Combination of Random Forest Classifier and K-Means Clustering. *International Journal of Intelligent Systems and Applications*, *10*(11), 11–19. https://doi.org/10.5815/ijisa.2018.11.02

Peralta, D., del Río, S., Ramírez-Gallego, S., Triguero, I., Benitez, J. M., & Herrera, F. (2015). Evolutionary Feature Selection for Big Data Classification: A MapReduce Approach. *Mathematical Problems in Engineering*, *2015*, 1–11. https://doi.org/10.1155/2015/246139

Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. A. (2007). A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*, *53*(10), 1605–1614. https://doi.org/10.1016/j.camwa.2006.07.013

Rajendran, S., Khalaf, O. I., Alotaibi, Y., & Alghamdi, S. (2021). MapReduce-based big data classification model using feature subset selection and hyperparameter tuned deep belief network. *Scientific Reports*, *11*(1), 1–10. https://doi.org/10.1038/s41598-021-03019-y

Ramya, P., & Kumar, R. K. (1981). Joint Fortification Model for Reliable Online Rating Systems using Linear Regression. *International Journal of Applied Engineering Research*, *12*(1), 2017.

Rashid, A. N. M. B., Ahmed, M., Sikos, L. F., & Haskell-Dowland, P. (2020). A Novel Penalty-Based Wrapper Objective Function for Feature Selection in Big Data Using Cooperative Co-Evolution. *IEEE Access*, *8*, 150113–150129. https://doi.org/10.1109/access.2020.3016679

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. *2013 International Conference on Collaboration Technologies and Systems (CTS)*, 42–47. https://doi.org/10.1109/cts.2013.6567202

Sleeman, W. C., & Krawczyk, B. (2021). Multi-class imbalanced big data classification on Spark. *Knowledge-Based Systems*, *212*, 106598. https://doi.org/10.1016/j.knosys.2020.106598

Şüyun, S. B., Yurdakul, M., Taşdemir, Ş., & Biliş, S. (2025). Triple-Stream Deep Feature Selection with Metaheuristic Optimization and Machine Learning for Multi-Stage Hypertensive Retinopathy Diagnosis. *Applied Sciences*, *15*(12), 6485. https://doi.org/10.3390/app15126485

Singh, S., & Singh, N. (2012). Big Data analytics. *2012 International Conference on Communication, Information & Computing Technology (ICCICT)*, 1–4. https://doi.org/10.1109/iccict.2012.6398180

Thiruvengadam, J. A., Nabigaru, K. M., & Kovi, A. (2025). Scalable Residual Feature Aggregation Framework with Hybrid Metaheuristic Optimization for Robust Early Pancreatic Neoplasm Detection in Multimodal CT Imaging. *ArXiv Repository / Proceedings of the International Conference on Big Data Analytics*. https://doi.org/10.48550/arXiv.2512.23597

Yuan, M., Yang, Z., & Ji, G. (2019). Partial maximum correlation information: A new feature selection method for microarray data classification. *Neurocomputing*, *323*, 231–243. https://doi.org/10.1016/j.neucom.2018.09.084

Zhao, H., Hu, Q., Zhu, P., Wang, Y., & Wang, P. (2021). A Recursive Regularization Based Feature Selection Framework for Hierarchical Classification. *IEEE Transactions on Knowledge and Data Engineering*, *33*(7), 2833–2846. https://doi.org/10.1109/tkde.2019.2960251