



School of Pure and Applied Sciences

Department of Computing and Informatics

**RHINO OPTIMISATION ALGORITHM FOR HYPERPARAMETER OPTIMISATION IN MACHINE LEARNING MODELS**

by

SIMISANI NDABA

Student ID number: 24020000

BIS (Business) (UoB), MSc (Computer Information Systems) (UoB)

A Concept Paper Submitted to the School of Pure and Applied Sciences in Partial Fulfilment of the Requirements for the Award of the Degree of Doctor of Philosophy in Information Systems of BIUST

**Supervisor(s): Prof, Rajalakshmi Selvaraj**

Department of Computing and Informatics

School of Pure and Applied Sciences, BIUST

E-mail Address: selvarajr@biust.ac.bw

**Dr. Hlomani Hlomani**Department of Computing and Informatics

School of Pure and Applied Sciences, BIUST

E-mail Address: hlomanihb@biust.ac.bw

December, 2024

# ABSTRACT

The Rhino Optimization Algorithm (ROA) is a novel bio-inspired metaheuristic technique designed for hyperparameter optimization in machine learning models, particularly Support Vector Machines (SVM) and Neural Networks. Hyperparameter tuning is crucial for model performance, yet it is challenging and time-consuming, especially in high-dimensional spaces. Metaheuristic algorithms like ROA, which mimic natural behaviours, efficiently balance exploration and exploitation to find optimal solutions. In ROA, the exploration phase mimics random foraging behaviours, while the exploitation phase refines the search in promising regions. This study applies ROA to optimize hyperparameters in SVM and Neural Networks, aiming to enhance accuracy, convergence speed, and solve high-dimensional space. The ROA will be tested on the CEC 2010,2013 and 2017 benchmark functions with expected improvements in accuracy, convergence rates, and robustness compared to traditional optimisation algorithms and other bio-inspired algorithms. The Friedman test will statistically assess and evaluate the ROA results with other bio-inspired metaheuristic algorithms to compare the performance of different optimization algorithms across multiple datasets to reduce the value to a 2.00 Friedman score.

# TABLE OF CONTENTS

[1. ABSTRACT 2](#_Toc184979150)

[2. TABLE OF CONTENTS 3](#_Toc184979151)

[3. LIST OF ABBREVIATIONS 5](#_Toc184979152)

[4. SECTION 1. INTRODUCTION 6](#_Toc184979153)

[1.1 Background 7](#_Toc184979154)

[1.2 Problem Statement 8](#_Toc184979155)

[1.3 General Objective . 8](#_Toc184979156)

[1.4 Specific Objectives 8](#_Toc184979157)

[1.5 Hypothesis 9](#_Toc184979158)

[1.6 Research Questions 9](#_Toc184979159)

[1.7 Expected Outcome 10](#_Toc184979160)

[1.8 Justification of the study 10](#_Toc184979161)

[5. SECTION 2. LITERATURE REVIEW 11](#_Toc184979162)

[2.1 An Overview of Scalable Methods 11](#_Toc184979163)

[2.1.1 Clouded Leopard Optimization algorithm 11](#_Toc184979164)

[2.1.2 Walrus Optimization Algorithm 11](#_Toc184979165)

[2.1.3 Osprey Optimization Algorithm 12](#_Toc184979166)

[2.1.4 Northern Goshawk Optimization 12](#_Toc184979167)

[2.1.5 Zebra Optimization Algorithm 12](#_Toc184979168)

[2.1.6 Duck Swarm Algorithm 12](#_Toc184979169)

[2.1.7 Harris Hawks Optimizer – no optimum speed cost 13](#_Toc184979170)

[2.1.8 Pufferfish Optimization Algorithm 13](#_Toc184979171)

[2.2 An Overview of Non-Scalable Algorithms 13](#_Toc184979172)

[2.2.1 Bat Algorithm‑Tuned CNN Hyperparameters 13](#_Toc184979173)

[2.2.2 Elk Herd Optimizer 14](#_Toc184979174)

[2.2.3 Pelican Optimisation Algorithm 14](#_Toc184979175)

[2.2.4 Wild Horse Optimizer 14](#_Toc184979176)

[2.2.5 Fennec Fox Optimization Algorithm 14](#_Toc184979177)

[2.2.6 Tasmanian Devil Optimization 15](#_Toc184979178)

[2.2.7 Hippopotamus Optimization Algorithm 15](#_Toc184979179)

[2.2.8 Salp Swarm Algorithm 15](#_Toc184979180)

[2.2.9 Particle Swarm Optimization Grid Search for Support Vector Regression 15](#_Toc184979181)

[2.2.10 Hybrid Bald Eagle-Crow Search Algorithm 16](#_Toc184979182)

[2.2.11 Beetle Antennae Search (BAS) algorithm 16](#_Toc184979183)

[2.2.12 Crow Search Algorithm 16](#_Toc184979184)

[2.3 Identification of the Existing Research Gaps 17](#_Toc184979185)

[6. SECTION 3. THEORETICAL FUNDAMENTALS / METHODOLOGY 18](#_Toc184979186)

[3.1 Research Design 18](#_Toc184979187)

[3.1.2 Phase 1: Rhino Optimiser Algorithm Modelling 19](#_Toc184979188)

[3.1.3 Phase 2: Utilize ROA for Hyperparameter Optimization 21](#_Toc184979189)

[3.1.4 Phase 3: Assess Scalability 22](#_Toc184979190)

[3.1.5 Phase 4: Examine Convergence 23](#_Toc184979191)

[3.1.5.1 Steps to Perform Convergence Metric 23](#_Toc184979192)

[3.1.6 Phase 5: Demonstrate ROA’s Benefits 24](#_Toc184979193)

[7. SELECTED REFERENCES 25](#_Toc184979194)

# 3. LIST OF ABBREVIATIONS

|  |  |  |
| --- | --- | --- |
| **S. No** | **Abbreviation** | **Meaning** |
| 1 | ROA | Rhino Optimisation Algorithm |
| 2 | HPO | Hyperparameter Optimisation |
| 3 | SVM | Support Vector Machine |
| 4 | CNN | Convolution Neural Networks |
| 5 | RNN | Recurrent Neural Networks |
| 6 | NN | Neural Networks |
| 7 | LSTM | Long Short Neural Network |
| 8 | BA | Bio-inspired algorithms |
| 9 | PSO | Particle Swarm Optimiser |
| 10 | ML | Machine Learning |
| 11 | CLO | Clouded Leopard Optimization |
| 12 | WOA | Walrus Optimization Algorithm |
| 13 | OOA | Osprey Optimization Algorithm |
| 14 | NGO | Northern Goshawk Optimization |
| 15 | ZOA | Zebra Optimization Algorithm |
| 16 | DSA | Duck Swarm Algorithm |
| 17 | HHO | Harris Hawks Optimizer |
| 18 | PFOA | Pufferfish Optimization Algorithm |
| 19 | EHO | Elk Herd Optimizer |
| 20 | POA | Pelican Optimization Algorithm |
| 21 | WHO | Wild Horse Optimizer |
| 22 | FFA | Fennec Fox Optimization Algorithm |
| 23 | TDO | Tasmanian Devil Optimization |
| 24 | HO | Hippopotamus Optimization Algorithm |
| 25 | SSA | Salp Swarm Algorithm |
| 26 | PSOGS-SVR | Particle Swarm Optimization Grid Search for Support Vector Regression |
| 27 | HBE-CSA | Hybrid Bald Eagle-Crow Search Algorithm |
| 28 | BAS | Beetle Antennae Search Algorithm |
| 29 | CSA | Crow Search Algorithm |

# SECTION 1. INTRODUCTION

According to Das et al. (2024), the tuning of hyperparameters in machine learning models can be framed as an optimization problem, as it entails navigating through a set of objective functions to maximize model accuracy. Hyperparameters, defined by Abdulsaed et al. (2023) are the configuration parameters that dictate the behaviour and performance of a machine learning model, are crucial in this process. The hyperparameters of machine learning models (ML), such as the learning rate and dropout rate, exert a direct influence on the learning dynamics, stability, and classification accuracy of the network. The primary objective of hyperparameter optimization (HPO) is to identify the most effective hyperparameter settings that yield optimal results in the shortest time (Li et al., 2021). This domain, a significant subfield of machine learning, emphasizes refining the hyperparameters of selected algorithms to enhance overall performance (Kadr et al., 2024). Algorithms designed to perform this task are referred to as hyperparameter optimisers (Claesen and De Moor, 2015; Feurer and Hutter, 2019).

Bio-inspired metaheuristic algorithms that use the natural behaviours of animals have been used in hyperparameter optimisation studies in recent years in finding optimal solutions. Various natural behaviours of animals have been investigated as inspiration to create algorithms such as the Spotted Hyena Optimizer (Dhiman and Kumar, 2017), Whale Optimization Algorithm (Mirjalili and Lewis, 2016), Chameleon Swarm Algorithm (Braik ,2021) and the Marine Predator Algorithm (Faramarzi et al, 2020).

The behaviour of Rhinoceroses has yet to be thoroughly investigated as a model for hyperparameter optimization.

The Rhino Optimization Algorithm (ROA) tackles the hyperparameter optimization task by emulating Rhinoceros exploratory foraging by performing an initial broad search across the hyperparameter space. Like a Rhinoceros would focus on an area with plenty of resources, it switches to a more focused exploitative search within this subset of hyperparameters once it finds regions with promising results. This allows the ROA to strike a balance between exploitation (focused tuning within a profitable range) and exploration (randomized searching of wide parameter spaces) which is crucial for hyperparameter tuning. The efficiency of ROA is increased by this bio-inspired mechanism especially in high-dimensional spaces where conventional techniques might not work as well. ROA can effectively find the best hyperparameters by adaptively switching between exploration and exploitation based on past results which improves model performance while using less computing power. This analogy illustrates how the algorithm mimics biological processes, categorizing it as a bio-inspired algorithm (Haque et al., 2021; Trojovská and Dehghani, 2022).

In comparison to one of the most contemporary Grey Wolf Optimiser (GWO) developed by Mirjalili et al (2014), the ROA and the GWO are by bio-inspired algorithms, but they differ significantly in their approaches to exploration and exploitation. ROA emulates the foraging behaviour of rhinoceroses, where the search is guided by adaptive foraging tactics that enable dynamic shifts between broad exploration and focused exploitation. When a promising region is located, the ROA intensifies its search around that area, mimicking how rhinoceroses may linger near a rich water source. This targeted search process allows the ROA to converge rapidly in high-dimensional spaces by focusing computational resources on fruitful areas. In contrast, the GWO models the hierarchical hunting strategy of grey wolves, where the alpha, beta, and delta wolves coordinate a group-based pursuit of prey. GWO relies on position updating guided by these leaders, balancing exploration and exploitation as the wolves close in on the optimal solution. However, GWO’s reliance on a hierarchical structure can lead to slower adaptation in dynamic environments, as it may converge prematurely if the leaders are trapped in local optima. Unlike GWO’s structured approach, ROA’s flexible foraging technique offers an adaptable framework that allows for refined exploration without being bound to fixed leader-follower dynamics, making it more robust in complex, high-dimensional search spaces.

## 1.1 Background

Abdulsaed et al. (2023) argue that traditional methods such as Grid search, Random search, and Bayesian optimization suffer from high-dimensional space complexity and may fail to effectively explore the entire hyperparameter space. As a result, there is a growing need for more advanced techniques that can efficiently optimize hyperparameters in machine learning models. According to Das et al. (2024), metaheuristic approaches have gained substantial traction in recent years due to their ability to converge both locally and globally, making them suitable for a wide array of optimization challenges. Metaheuristic algorithms are generally classified into single-solution-based and population-based methods, with the latter including evolution-inspired, swarm-inspired, physics-based, human-based, bio-based, and mathematically grounded algorithms. These stochastic methods, which draw inspiration from natural processes, aim to guide an initial population toward a global optimum, offering near-optimal solutions within a reasonable computational time (Agrawal et al., 2021).

Metaheuristics have emerged as one of the most effective stochastic methods for optimizing and solving various optimization challenges. Zhang et al. (2021) describe the solutions generated by metaheuristics as quasi-optimal, acknowledging that these solutions may not necessarily coincide with the global optima. Dehghani et al. (2021) and Trojovská and Dehghani (2022) highlight that the pursuit of better quasi-optimal solutions remains the driving force behind the continuous development of numerous metaheuristic algorithms. Consequently, a variety of metaheuristic techniques have been inspired by natural phenomena, such as biological processes, animal behaviour, and other evolutionary mechanisms.

Haque et al. (2021) note that bio-inspired algorithms (BA) represent unique heuristic approaches designed to mimic natural strategies for solving optimization problems. These bio-optimization algorithms (BoAs) typically follow a two-phase process: exploration and exploitation. Due to their rapid convergence, minimal parameter requirements, and straightforward implementation, BoAs have found widespread application in tasks such as parameter optimization, fault prediction, bioinformatics, and image processing (Trojovská et al., 2022). Numerous studies, including those by Al-Betar et al. (2024), Açıkkar and Altunkol (2023), Hussien et al. (2020), Ravikiran et al. (2024), Nirmal et al. (2024), and Sun et al. (2022), have integrated BoAs with machine learning models to improve hyperparameter tuning. Some of the most widely recognized legacy optimization techniques include Differential Evolution (Storn and Price, 1997), Harmony Search (Geem et al. (2001), Ant Colony Optimization (Dorigo and Di Caro, 1999), Firefly Algorithm (Yang, 2010), Cuckoo Search (Yang and Deb, 2009), Gravitational Search Algorithm (Rashedi et al.,2009), and Grey Wolf Optimizer (Mirjalili et al., 2014). These algorithms are often inspired by the social behaviour of animals (e.g., foraging, migration, courtship) as well as the evolution of natural systems and human social behaviour.

In addition to the ROA comparison to the GWO as discussed in the Introduction, similarly, unlike bird-based models like Particle Swarm Optimization, which focus on collective flock behaviour, the Rhino algorithm emphasizes individual momentum and exploratory adaptation.

In contrast to Ant Colony Optimization, which relies on pheromone-guided paths and collective intelligence, the Rhino algorithm models a more independent and aggressive exploration approach.

## 1.2 Problem Statement

Hyperparameter optimization algorithms often fail to efficiently identify optimal solutions within expansive and complex search spaces, particularly in high-dimensional contexts. The Rhino Optimization Algorithm (ROA) inspired by Rhino food search foraging and adaptability behaviours addresses enhancing search space navigation and improving convergence rates creates a flexible balance between exploration and exploitation phases.

## 1.3 General Objective Develop a new bio-inspired algorithm, the ROA based on the foraging food search and adaptive behaviours of Rhinoceroses to overcome the shortcomings of existing hyperparameter optimization technique.

## 1.4 Specific Objectives

Based on the general objectives and its future perspectives of achievement, some of the specific objectives are propounded. They are the following:

1. Create a model that mimics rhino behaviour emphasizing their social and foraging habits to serve as the basis for the Rhino Optimization Algorithm (ROA).
2. Utilize ROA to optimize hyperparameters and evaluate its efficacy in intricate architectures using deep learning frameworks such as SVM CNNs RNNs and LSTMs.
3. Assess ROA's scalability on datasets of varying sizes and complexities to understand its performance on larger optimization problems and its real-world applicability.
4. Examine the convergence of the ROA to determine how well it achieves optimal or nearly optimal outcomes for machine learning applications.
5. Demonstrate the ROAs benefits and contribution to hyperparameter tuning research compare its performance with that of the most popular hyperparameter optimization algorithms.

## 1.5 Hypothesis

Based on the research objectives proposed in the research, the hypothetical statements are recommended for testing during the course of research experiments.

**Hypothesis-1:**

*H*11​: The Rhino Optimization Algorithm achieves better hyperparameter optimization results than existing

metaheuristic algorithms.

**Hypothesis-2:**

*H*12: The convergence rates of the Rhino Optimization Algorithm are significantly faster than those of

traditional optimization algorithms in hyperparameter tuning.

**Hypothesis-3:**

*H*13: The Rhino Optimization Algorithm is generalizable across various domains and problem sets beyond

hyperparameter tuning.

**Hypothesis-4:**

*H*14: The Rhino Optimization Algorithm significantly improves model performance in deep learning architectures such as SVM, CNNs, RNNs, and LSTMs compared to conventional methods.

## 1.6 Research Questions

1. How might the Rhino Optimization Algorithm (ROA) be built using a computational model of rhino behaviour

that highlights their social and feeding behaviours?

2. To what extent can the Rhino Optimization Algorithm (ROA) optimize hyperparameters for intricate deep

learning frameworks such as SVM CNNs RNNs and LSTMs?

3. How does the Rhino Optimization Algorithm (ROA) perform on more significant optimization problems that

are pertinent to real-world applications and how scalable is it across datasets of different sizes and

complexities?

4. How effectively does the Rhino Optimizer Algorithm (ROA) converge to optimal or near-optimal solutions in

machine learning applications?

5. How does the Rhino Optimizer Algorithm (ROA) perform in comparison to the most well-known

hyperparameter optimization algorithms and what are its advantages and contributions to hyperparameter

tuning?

## 1.7 Expected Outcome

By producing high-quality solutions with quicker convergence resilience to local optima and adaptability to noisy datasets the Rhino Optimizer Algorithm (ROA) is anticipated to greatly advance hyperparameter optimization in machine learning while preserving scalability and requiring little parameter tuning. The field of optimization research will benefit from its comparative advantage over well-established metaheuristic algorithms which will facilitate interdisciplinary applications and encourage further advancements in optimization methodologies.

## 1.8 Justification of the study

This paper presents the Rhino Optimizer Algorithm (ROA) a state-of-the-art metaheuristic that draws inspiration from Rhinoceroses social dynamics adaptive foraging and territorial behaviours. The ROA which was created to fill important gaps in optimization methods improves search agent cooperation and dynamically strikes a balance between exploration and exploitation tactics while adjusting to changing environmental circumstances. Based on rhinos natural foraging tactics which entail teamwork to optimize resource use while averting dangers the ROA uses these behaviours to create a strong algorithmic framework. This structure offers a flexible and effective search process by mitigating common issues like premature convergence and stagnation in local optima. By addressing a broad spectrum of intricate issues in computer science engineering and other fields the ROA makes a substantial contribution to the development of metaheuristic optimization methods.

# 5. SECTION 2. LITERATURE REVIEW

The development of bio-inspired metaheuristic optimisers and their integration to machine learning models have created various studies from the behaviour of animals. This literature review examines various journal articles, providing a comprehensive overview of the status of research, identifying existing research gaps, and justifying the research methods for developing the novel Rhino Optimisation Algorithm (ROA). Most of the surveyed studies do not use traditional machine learning models. Instead, they focus on the metaheuristic optimization algorithms inspired by animal behaviour they have chosen to use to simulate.

## 2.1 An Overview of Scalable Methods

The following section surveys the developed bio-inspired optimisation algorithms that have worked on large and diverse datasets. Most of the developed bio-inspired optimisation algorithms have been developed by Eva Trojovská and Mohammad Dehghani as follows.

### 2.1.1 Clouded Leopard Optimization algorithm

Trojovská and Dehghani’s (2022) Clouded Leopard Optimization (CLO) algorithm divides its activities into phases of exploration and exploitation simulating the nocturnal hunting and daytime resting behaviours of clouded leopards. It was applied to 68 benchmark functions (unimodal, multimodal, CEC 2015, and CEC 2017) and four engineering tasks. The CLO performed with a speed optimum cost of 2996.4342, better than ten other metaheuristic algorithms. Due to this equilibrium, the algorithm can function effectively across several domains on larger datasets and on higher-dimensional problems. However, testing the algorithm on a wider range of real-world problems to validate its robustness and versatility.

### 2.1.2 Walrus Optimization Algorithm

The Walrus Optimization Algorithm (WOA) by Trojovský and Dehghani (2022) is organized into phases of exploration migration and exploitation which simulates walrus behaviours such as feeding migrating and evading predators. It obtained a speed optimum cost of 2998. 6108 after testing on four engineering design problems and 68 benchmark functions. Adaptive strategies are successfully simulated by the WOA demonstrating significant optimization potential. To improve robustness and versatility, improvements could concentrate on expanding testing to a variety of real-world problems and better balancing exploration and exploitation.

### 2.1.3 Osprey Optimization Algorithm

The Osprey Optimization Algorithm (OOA) by Dehghani and Trojovský (2023) was developed to address complex engineering optimization problems by replicating the hunting behaviour of ospreys which can scale across a variety of use cases. In the exploration phase, the algorithm simulates the osprey's technique of locating fish underwater, thus identifying potential solutions. The exploitation phase involves refining these solutions, analogous to the osprey catching fish and transporting them to a suitable location for further processing, thereby improving the optimization process.

The OOA is applied to solve various optimization problems, including 29 standard benchmark functions from the CEC 2017 test suite and 22 real-world constrained optimization problems from the CEC 2011 test suite. However, potential areas for improvement could include further enhancing the balance between exploration and exploitation and testing the algorithm on a wider range of real-world problems to validate its robustness and versatility.

### 2.1.4 Northern Goshawk Optimization

Dehghani et al. (2021) developed the Northern Goshawk Optimization (NGO) algorithm inspired by the hunting strategies of the northern goshawk. The algorithm mimics two key phases of the goshawk’s predatory behaviour: prey identification (exploration) and the chase-and-escape phase (exploitation). The NGO algorithm is applied to solve various optimization problems, including 68 different objective functions and four engineering design problems and obtained a speed optimum cost of 2994.2471. The algorithm can manage intricate search spaces with flexible tactics for expanded optimization assignments. In addition, the application of NGO in solving optimization problems in different sciences and comparing it with other existing algorithms are other suggestions for further studies.

### 2.1.5 Zebra Optimization Algorithm

The Zebra Optimization Algorithm (ZOA) developed by Trojovská et al. (2022) mimics the Zebra foraging (exploration) and predator defence (exploitation) behaviours are modelled by the Zebra Optimization Algorithm (ZOA). When it was used to solve four engineering problems and 68 benchmark functions the cost speed was 2998. 5189. ZOA is useful for complicated issues because it is scalable and strikes a balance between exploration and exploitation. This balance could be further refined and its resilience tested in a variety of real-world situations.

### 2.1.6 Duck Swarm Algorithm

The Duck Swarm Algorithm (DSA) by Zhang and Wen (2024) is inspired by the resource gathering (exploitation) and food searching (exploration) the Duck Swarm Algorithm (DSA) simulates the foraging habits of ducks. It has been used to solve six engineering optimization problems and CEC benchmark functions which are categorized into unimodal, multimodal, and fixed-dimension numerical optimization problems yielding a speed optimum cost of 2996.403. Furthermore, in comparison to current algorithms the DSA showed better performance in Wireless Sensor Networks (WSN) node deployment optimization. Possible improvements include balancing exploration and exploitation better and testing its robustness in more real-world scenarios to confirm its reliability and applicability.

### 2.1.7 Harris Hawks Optimizer – no optimum speed cost

Shehab et al (2022) developed the Harris Hawks Optimizer (HHO) modelled after the cooperative hunting tactics of Harris hawks which include coordinated group efforts and surprise pouncing. By adjusting tactics in response to the preys energy dynamics, the HHO effectively strikes a balance between exploration and exploitation in complex search spaces. Due to its ease of use and adaptability, it can be used in a variety of fields including networking power systems and image processing. However, premature convergence and other problems show that more work and extensive testing are required to improve its resilience and suitability for practical situations.

### 2.1.8 Pufferfish Optimization Algorithm

The Pufferfish Optimization Algorithm (PFOA), proposed by Al-Baik et al. (2024), models pufferfish behavior to solve complex optimization problems. It integrates exploration and exploitation strategies for high-dimensional challenges and demonstrates effectiveness on CEC 2017 test suites (dimensions 10–100), CEC 2011 constrained problems, and four engineering design tasks, achieving an optimum cost of 2996.3882. Despite its strengths, improvements are needed to enhance exploration-exploitation balance and validate performance across diverse real-world scenarios.

## 2.2 An Overview of Non-Scalable Algorithms

Optimization algorithms face difficulties when dealing with large datasets and complex models because of their high computational cost’s requirements for fine-tuning and limitations in high-dimensional spaces.

### 2.2.1 Bat Algorithm‑Tuned CNN Hyperparameters

Ravikiran et al (2024) achieved a high classification accuracy of 0.96 when they investigated the optimization of sheep breed classification using CNNs and the Bat algorithm which simulates bat echolocation to modify parameters like learning rate and dropout for better performance. The dataset used includes 40,000 images of sheep, divided into training, validation, and testing sets. Although the Bat algorithm works well for increasing model accuracy, its scalability and suitability for larger datasets or complex models may be limited by its dependence on parameters and computational intensity especially in multiple iterations.

### 2.2.2 Elk Herd Optimizer

Al-Betar et al. (2024) introduced the Elk Herd Optimizer (EHO) that simulates elk herd breeding behaviours across three phases: rutting season, calving season, and selection season. During the rutting season, dominant bulls lead groups, with the size of each group determined by the leaders' fitness. In the calving season, new solutions are generated based on interactions between leaders and followers. The selection season then merges all groups, with the fittest individuals selected for the next cycle. Although it is good at simulating natural behaviours, its phase-based structure and computational cost for large problems may limit its scalability.

### 2.2.3 Pelican Optimisation Algorithm

Trojovský and Dehghani (2022) introduced the Pelican Optimization Algorithm (POA) aimed at solving complex engineering challenges. The algorithm is modelled after the hunting behaviour of Pelicans. In the exploration phase, Pelicans (search agents) randomly detect and move towards potential prey (solutions). The exploitation phase involves refining the search by focusing on the area surrounding the prey, allowing the pelicans to converge on the optimal solution. The performance of EHO was assessed using 29 benchmark optimization problems from the CEC-2017 special sessions on real-parameter optimization and four traditional real-world engineering design problems. The Pelican Optimiser generated an optimum speed reducer cost of 2994.471. Despite its use in complicated engineering problems the POA may not be as scalable as it could be due to its dependence on specific behaviours.

### 2.2.4 Wild Horse Optimizer

The Wild Horse Optimizer (WHO) was developed by Naruei and Keynia (2022) inspired by the social behaviour of wild horses. The algorithm simulates group dynamics such as grazing, mating, leadership, and dominance.

The algorithm consists of five main steps: creating an initial population, grazing and mating of horses, leadership and leading the group by the leader (stallion), exchange and selection of leaders, and saving the best solution.

The performance of WHO was assessed using several sets of test functions such as CEC2017 and CEC2019 and compared with popular and new optimization methods. This behaviour promotes diversity and exploration within the search space. Like many optimization algorithms, WHO may suffer from premature convergence, where it gets stuck in local optima. The algorithm's performance can be sensitive to parameter settings, which may require fine-tuning for different problems.

### 2.2.5 Fennec Fox Optimization Algorithm

Trojovská et al. (2022) introduced the Fennec Fox Optimization (FFA) algorithm consisting of two main phases: exploration (digging) and exploitation (escaping), which simulate the foxes' natural behaviours. FFO was tested on 68 standard benchmark functions and four engineering design problems, showing its effectiveness in solving real-world optimization tasks. The optimiser speed reducer design problem optimum cost generated 2996.348. The computational demands and stochastic behaviour in complex search spaces may limit the algorithms scalability.

### 2.2.6 Tasmanian Devil Optimization

The Tasmanian Devil Optimization (TDO) algorithm (Dehghani et al. (2022) emulates the feeding behaviour of Tasmanian devils. The TDO algorithm incorporates two primary strategies: attacking live prey and scavenging carrion. TDO was tested on 23 standard objective functions, including unimodal, high-dimensional multimodal, and fixed-dimensional multimodal benchmark functions. It was also applied to four real-world engineering design problems, showing its effectiveness in solving complex optimization tasks. The optimum speed reducer cost generated 2997.016. Although it performs well on common objective functions scalability may be problematic when used for more complicated or large-scale problems.

### 2.2.7 Hippopotamus Optimization Algorithm

The Hippopotamus Optimization (HO) algorithm by Al-Baik et al. 2024 emulates the navigation and territorial defence strategies exhibited by Hippos, incorporating critical elements such as position updating, defensive tactics, and evasion methods. The performance of HO was assessed using 161 benchmark functions, categorized into unimodal, high-dimensional multimodal, fixed-dimensional multimodal functions, and the CEC 2019 and CEC 2014 test suites. In high-dimensional or large-scale optimization tasks its navigational and territorial strategies might be less flexible.

### 2.2.8 Salp Swarm Algorithm

Abdulsaed et al. (2023) used the coordinated movement of Salp swarms which makes it easier to explore the hyperparameter space by simulating interactions within the swarm to introduce the Salp Swarm Algorithm (SSA) hyperparameters in CNN. To achieve strong optimization performance, each Salp is impacted by the best solution neighbouring solutions and a random factor. When tested on the MNIST and Fashion-MNIST datasets SSA outperformed other techniques with high accuracies of 99. 46 percent and 94. 53 percent respectively. SSAs high computational costs with large datasets reliance on fine-tuning parameters and performance problems in high-dimensional spaces are some of its drawbacks which the study points out can limit its scalability and viability for real-time applications.

### 2.2.9 Particle Swarm Optimization Grid Search for Support Vector Regression

To improve hyperparameter optimization for Support Vector Regression (SVR), Açıkkar and Altunkol (2023) created the PSOGS algorithm a combination of Grid Search (GS) and Particle Swarm Optimization (PSO). When their PSOGS-SVR was tested on five benchmark datasets such as QSAR Fish Toxicity and Energy Efficiency, it performed noticeably faster than GS on SVR by 2.8% and 6.2% better prediction accuracy. Notwithstanding these encouraging outcomes, the study points out drawbacks including the algorithms reliance on parameter choice and its incapacity to manage continuous variables because of the discrete search space indicating that more work is required to improve scalability particularly in high-dimensional spaces.

### 2.2.10 Hybrid Bald Eagle-Crow Search Algorithm

Nirmal et al. (2024) introduced the Hybrid Bald Eagle-Crow Search Algorithm (HBE-CSA) for optimizing Gaussian Mixture Model (GMM) parameters in speaker verification tasks, merging the Bald Eagle Search Algorithm (BESA) for exploitation and the Crow Search Algorithm (CSA) for exploration. In comparison to individual algorithms and the Expectation Maximization (EM) algorithm, HBE-CSA demonstrated better performance including higher log-likelihood values improved convergence and lower equal error rates particularly in noisy conditions when tested on the Librispeech and AURORA datasets across a range of Signal-to-Noise Ratios (SNRs). In addition to highlighting HBE-CSAs potential for wider speech processing applications this balance between the exploration and exploitation phases allows for more reliable speaker verification. The algorithm's performance on larger and more complex datasets needs further validation to ensure its robustness and versatility.

### 2.2.11 Beetle Antennae Search (BAS) algorithm

Sun et al. (2022) applied a back-propagation neural network (BPNN) optimized by the Beetle Antennae Search (BAS) algorithm to predict the unconfined compressive strength of high-strength concrete. The model was trained on a dataset of 324 HSC samples and evaluated using root-mean-square error (RMSE) and the correlation coefficient (R). The BAS-tuned BPNN achieved high prediction accuracy, yielding a correlation coefficient (R = 0.9893) and an RMSE of 1.5158 MPa. Despite its success, the study identified limitations due to the relatively small dataset and the stochastic behaviour of the BAS algorithm, recommending further data collection and model refinement to improve the robustness of predictions. While the BAS works well for small datasets, its scalability to larger more complex scenarios is limited by its stochastic nature and dependence on small datasets.

### 2.2.12 Crow Search Algorithm

Hussien et al. (2020) aimed to apply the Crow Search Algorithm (CSA) in domains such as power engineering computer science and image processing. The CSA was inspired by the ways crows store and retrieve food.

Although the CSA has been applied to a wide range of fields, including power systems, computer science, machine learning, and civil engineering, demonstrating its flexibility and effectiveness, compared to other metaheuristic algorithms, it performed with the lowest Freidman test of 2.73. Even though CSA is useful, it frequently gets stuck in local optima so changes are required to make it more applicable to complex or high-dimensional problems.

## 2.3 Identification of the Existing Research Gaps

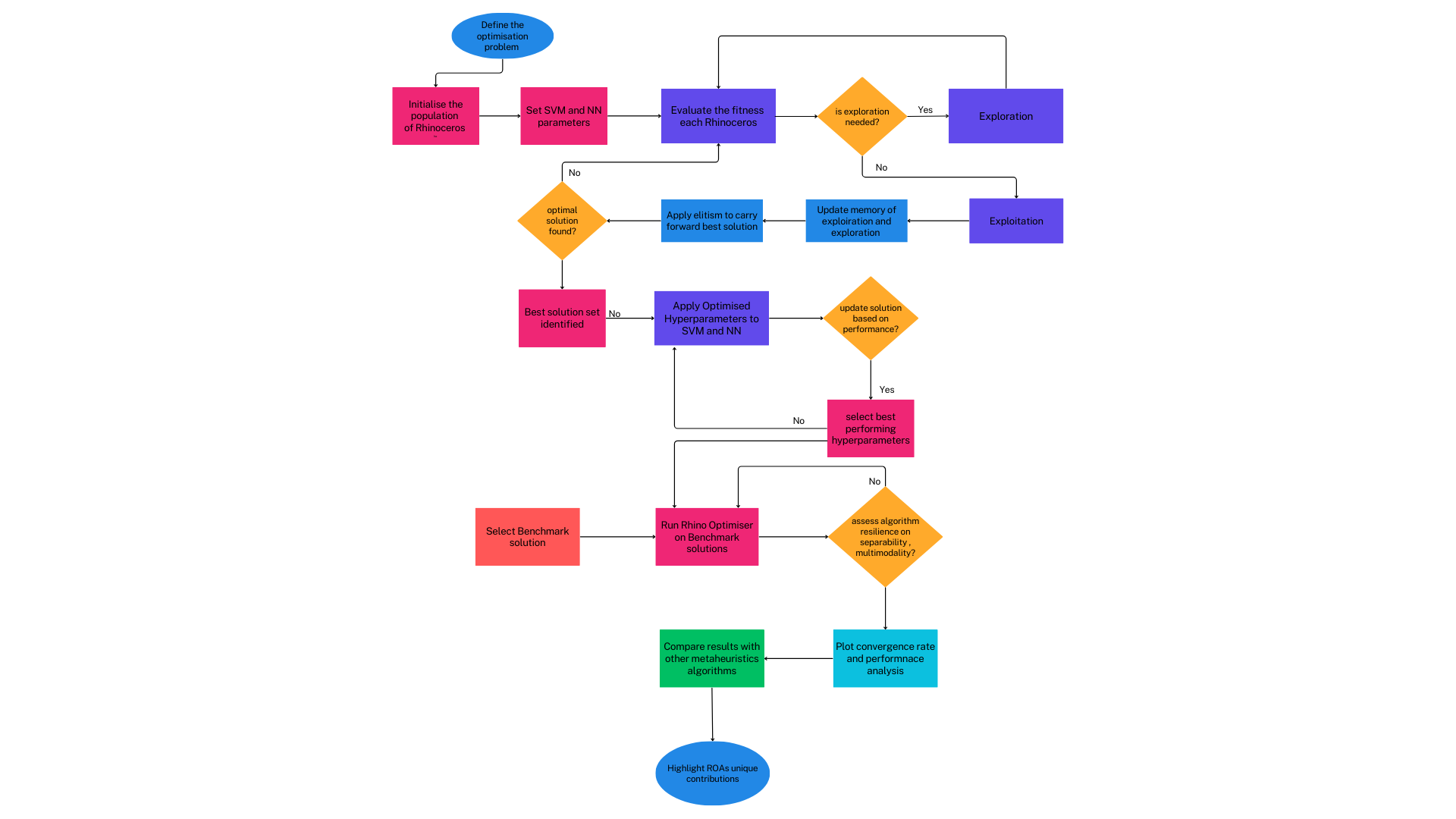
The goal of several bio-inspired algorithms such as Clouded Leopard Optimization (CLO) and Wild Horse Optimizer (WHO) is to strike a balance between the phases of exploitation (local search) and exploration (global search). Maintaining this equilibrium is still difficult though and frequently results in sluggish or early convergence in a variety of contexts. Although they perform well in controlled settings hybrid methods such as the Bald Eagle-Crow Search Algorithm (HBE-CSA) have trouble with noisy real-world datasets. There is a need for reliable techniques that generalize under a variety of circumstances because stochastic algorithms like the Beetle Antennae Search (BAS) and the Salp Swarm Algorithm (SSA) are sensitive to initial conditions and exhibit inconsistent performance on smaller datasets.

# 6. SECTION 3. THEORETICAL FUNDAMENTALS / METHODOLOGY

This research proposes a novel methodology for the ROA. The methodology is divided into three distinct phases, the Initiation phase, Fitness Evaluation, Exploration and Exploitation. Each phase incorporates innovative approaches to address current limitations and improve the accuracy and applicability of ROA.

## 3.1 Research Design

The research work designs a novel model for working on high-dimensional spaces, improving the balance between exploration and exploitation and finding the local optima. While exploitation concentrates on focusing the search around known good solutions, exploration entails searching widely across the hyperparameter space to find potentially better solutions. The novel ROA strikes a balance between these approaches to enhance performance by optimizing successful regions (exploitation) and prevent becoming mired in suboptimal areas (exploration). This equilibrium is essential for effectively determining the optimal hyperparameter set by initially examining the space and then taking advantage of promising regions as the algorithm approaches convergence. The following model of the ROA flowchart consists of the objectives the research is setting to achieve in Figure.1.



**Figure 1: ROA Research Design flowchart**

### 3.1.2 Phase 1: Rhino Optimiser Algorithm Modelling

A systematic quantitative approach is needed to construct an optimization algorithm which require a formal representation that enables the algorithm to methodically look for the best answer. In hyperparameter optimization the objective function (like model error or accuracy) that must be maximized or minimized is specified by the mathematical model. The algorithm iteratively investigates various combinations of the hyperparameters which are treated as variables in this model to optimize the objective function. To determine the ideal set of hyperparameters ROA can employ mathematical techniques (such as exploration exploitation and search heuristics) within a mathematical framework. A mathematical model also makes it possible to quantify performance and offers a foundation for comparing various hyperparameter configurations guaranteeing that ROA can determine the optimal configuration based on the model’s assessment. This methodical approach is described in the following steps.

#### 3.1.2.1 Initialization Phase

* Define the optimization problem as minimizing or maximizing an objective function , where d is a candidate solution in the *d*-dimensional search space.
* **Population Initialization**: Initialize a population of *N* rhinoceroses (candidate solutions), where each individual is a vector in the search space.

Here, *r* is a random vector uniformly distributed between 0 and 1, and ​ and ​ represent the lower and upper bounds of the solution space.

* Initialize parameters: Set the maximum number of iterations exploration-exploitation balance factor, and other algorithmic constants.

#### 3.1.2.2 Fitness Evaluation and Solution Update

* Fitness Function Evaluation: Evaluate the fitness of each solution using the objective function
* Selection: Select the best-performing rhinoceros, i.e., the one with the lowest (or highest for maximization problems) objective function value:

=min {, , ..., }

* Convergence Criteria: The algorithm terminates if the maximum number of iterations is reached, or if the improvement in fitness is less than a threshold ϵ.

#### 3.1.2.3. Exploration Phase

Random Movement: During the exploration phase, each rhinoceros moves in a randomized manner across the search space to explore potential regions.

= + .*.* *(* *- rand*

* is the position of rhinoceros *i* at iteration *t*.
* ​ is the best solution found so far.
* is the scaling factor for the influence of the best solution.
* is a random number in the range [0, 1] that controls the intensity of the exploration.
* *rand* is a random perturbation vector to enhance randomness in movement.
* controls the effect of the random perturbation on the solution's position.

#### 3.1.2.4. Exploitation Phase

* Focused Foraging (Local Search): In this phase, rhinoceroses move towards promising regions where high-quality solutions have been found.

= . ( *( - + . M .* *(*

* α is the momentum factor that determines how aggressively the rhinoceroses converge towards
* *M* is the inertia coefficient controlling the rate of convergence.
* and are randomly selected individuals from the population (to introduce diversity).
* ζ is the weight assigned to this diversification term.

### 3.1.3 Phase 2: Utilize ROA for Hyperparameter Optimization

#### 3.1.3.1 Define SVM and Neural Network Models:

In this phase, the aim is to evaluate the performance of the ROA by integrating it with SVM and Neural Networks. SVMs and neural networks have more intricate high-dimensional hyperparameter spaces that need precise adjustments for optimal performance they are better suited for testing using the Rhino Optimizer Algorithm (ROA). While neural networks necessitate careful tuning of learning rates number of layers and neurons per layer support vector machines (SVMs) rely on parameters such as the kernel type regularization and gamma. Decision trees and less complex classification algorithms such as k-Nearest Neighbours or Logistic Regression on the other hand have fewer and less sensitive hyperparameters which makes them simpler to optimize. For sophisticated optimization algorithms like ROA these more straightforward models are less difficult because they frequently do not need the same degree of complex hyperparameter tuning. In this phase, the steps are as follows,

* Define the structure of the SVM and the Neural Networks.
* For SVM, parameters to be optimized include the kernel type, regularization parameter (C), and gamma.
* For Neural Networks, parameters include the number of layers, number of neurons in each layer, activation function, learning rate, and dropout rate.

#### 3.1.3.2 Initialize Rhino Optimizer:

* Input: Predefined population size, number of iterations, and bounds for hyperparameters (for SVM and NN).
* Initialize the population of solutions (randomly generated sets of hyperparameters).
* Set up the Rhino Optimizer with predefined parameters for exploration and exploitation.

#### 3.1.3.4 Optimization Process:

* Input: Initialized population, SVM, and NN models.
  1. For each hyperparameter set (solution) in the population:
     + Apply the current hyperparameters to the SVM and Neural Network models.
     + Train the SVM and NN models using the training dataset.
     + Evaluate the performance (e.g., accuracy, F1-score) on the testing dataset.
  2. Use the fitness (model performance) to update the solutions during the exploration and exploitation phases.
  3. After a number of iterations, select the best-performing hyperparameter set.

#### 3.1.3.5 Training with Optimized Hyperparameters:

* Input: Best hyperparameter sets for SVM and NN (from the Rhino Optimizer).
* Retrain the SVM and Neural Network models using the optimized hyperparameters on the entire training dataset.
* Evaluate the models on the testing dataset.
* Output: Performance metrics (accuracy, precision, recall, F1-score) for both SVM and NN with optimized hyperparameters.

#### 3.1.3.6 Performance Comparison:

* Compare the performance of the models before and after hyperparameter optimization.
* Compare the performance of SVM and Neural Networks when optimized using the Rhino Optimizer.

### 3.1.4 Phase 3: Assess Scalability

The performance of optimization algorithms must be assessed using optimization benchmark functions. They offer a consistent method for evaluating various algorithms and comprehending their advantages and disadvantages. They facilitate comprehension of an algorithms ability to identify the best answer in various situations. Benchmark functions come in different varieties such as constrained separable non-separable unimodal and multimodal functions. The CEC benchmark functions are widely used to evaluate large-scale global optimization algorithm. This study has chosen to use the following benchmark functions.

* CEC 2010: This suite includes benchmark functions specifically designed for large-scale global optimization. These functions test the ability of algorithms to handle problems with a high number of decision variables, typically more than 100.
* CEC 2013: This suite extends the CEC 2010 benchmark functions and introduces new features such as imbalance between subcomponents and overlapping functions. It aims to better represent real-world large-scale optimization problems.
* CEC **2017**: This suite includes 30 benchmark functions, including composition, shifted, and rotated functions, designed to test the performance of optimization algorithms on challenging problems

The benchmark functions were chosen for their high dimensionality to test the ROA on problems with many dimensions using these functions which is crucial for assessing scalability. The benchmarks complexities evaluate the resilience and flexibility of algorithms the functions incorporate a variety of challenges including multimodality separability and non-separability. The benchmark functions are applicable to real-world optimization problems because they are made to resemble them. For instance, the Ackley Rastrigin Rosenbrock and Schwefel functions are frequently used benchmark functions. In optimization research these functions are frequently utilized.

### 3.1.5 Phase 4: Examine Convergence

### An optimization algorithms effectiveness and efficiency in moving closer to the ideal solution over time can be assessed using convergence testing. To do this, the algorithms speed accuracy and stability must be examined. 3.1.5.1 Steps to Perform Convergence Metric

1. **Define the Objective Function**: The CEC 2010, 2013 and CEC 2017 benchmark functions have been identified for the optimization algorithm will solve.
2. **Initialize the Algorithm**: Set up the Rhino Optimization Algorithm with initial parameters, including population size, number of iterations, and any specific parameters related to the algorithm.
3. **Run the Algorithm**: Execute the Rhino Optimization Algorithm on the chosen objective functions. Record the best solution found at each iteration or generation.
4. **Track Performance Metrics**: Measure the performance of the algorithm using the following metrics:
   * **Best Solution**: The best solution found by the algorithm at each iteration.
   * **Average Solution**: The average solution quality across multiple runs of the algorithm.
   * **Standard Deviation**: The variability in the solutions found by the algorithm across multiple runs.
5. **Plot Convergence Curve**: Create a convergence curve by plotting the best solution found at each iteration against the number of iterations. This curve shows how quickly the algorithm converges to the optimal solution.
6. **Compare with Other Algorithms**: Run the same objective function with different metaheuristic optimization algorithms and compare their convergence curves. This helps in understanding the relative performance of the algorithms.
7. **Statistical Analysis**: Perform statistical tests, such as the Wilcoxon signed-rank test or the Friedman test, to determine if the differences in performance are statistically significant.

### 3.1.6 Phase 5: Demonstrate ROA’s Benefits

1. **Competitor Algorithms**:

Select popular hyperparameter optimization algorithms (e.g., Bayesian Optimization, Grid Search).

2. P**erformance Comparison**:

Apply all algorithms to identical problems and datasets. Compare results based on key metrics (e.g., accuracy, precision, recall, speed).

3.**Statistical Analysis**: Perform statistical tests (e.g., t-tests) to validate performance differences.

4. **Documentation**: Highlight ROA’s unique features, advantages, and limitations in hyperparameter optimization.

## 7. SELECTED REFERENCES

1. Abdulsaed, E., Alabbas, M., and Khudeyer, R. (2023). Hyperparameter Optimization for Convolutional Neural

Networks using the Salp Swarm Algorithm. Informatica, 47(9). <https://doi.org/10.31449/inf.v47i9.5148>.

2. Açıkkar, M., and Altunkol, Y. (2023). A novel hybrid PSO-and GS-based hyperparameter optimization algorithm

for support vector regression. *Neural Computing and Applications*, *35*(27), 19961-19977.

<https://link.springer.com/article/10.1007/s00521-023-08805-5>.

3. Agrawal, P., Abutarboush, H. F., Ganesh, T., and Mohamed, A. W. (2021). Metaheuristic algorithms on feature

selection: A survey of one decade of research (2009-2019). IEEE Access, 9, 26766-26791.

<https://ieeexplore.ieee.org/abstract/document/9344597>.

4. Al-Baik, O., Alomari, S., Alssayed, O., Gochhait, S., Leonova, I., Dutta, U., ... and Dehghani, M. (2024). Pufferfish

Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems.

Biomimetics, 9(2), 65. <https://doi.org/10.3390/biomimetics9020065>.

5. Al-Betar, M. A., Awadallah, M. A., Braik, M. S., Makhadmeh, S., and Doush, I. A. (2024). Elk herd optimizer: a

novel nature-inspired metaheuristic algorithm. Artificial Intelligence Review, 57(3), 48.

<https://link.springer.com/article/10.1007/s10462-023-10680-4>.

6. Alibrahim, H., and Ludwig, S. A. (2021, June). Hyperparameter optimization: Comparing genetic algorithm

against grid search and bayesian optimization. In 2021 IEEE Congress on Evolutionary Computation (CEC) (pp. 1

1551 1559). IEEE.

7. Afape, J. O., Willoughby, A. A., Sanyaolu, M. E., Obiyemi, O. O., Moloi, K., Jooda, J. O., and Dairo, O. F. (2024).

Improving millimetre-wave path loss estimation using automated hyperparameter-tuned stacking ensemble

regression machine learning. Results in Engineering, 22, 102289.

8. Amiri, M. H., Mehrabi Hashjin, N., Montazeri, M., Mirjalili, S., and Khodadadi, N. (2024). Hippopotamus

optimization algorithm: a novel nature-inspired optimization algorithm. Scientific Reports, 14(1), 5032.

<https://doi.org/10.1038/s41598-024-54910-3>.

9. Braik, M. S. (2021). Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design

problems. Expert Systems with Applications, 174, 114685.

10. Cavalcanti, J. H., Kovács, T., and Kő, A. (2022). Production system efficiency optimization using sensor data,

machine learning-based simulation and genetic algorithms. Procedia CIRP, 107, 528-533.

11. Claesen, M., and De Moor, B. (2015). Hyperparameter search in machine learning. arXiv preprint

arXiv:1502.02127.

12. Das, M., Mohan, B. R., Guddeti, R. M. R., and Prasad, N. (2024). Hybrid Bio-Optimized Algorithms for

Hyperparameter Tuning in Machine Learning Models: A Software Defect Prediction Case Study. Mathematics,

12(16), 2521.<https://www.mdpi.com/2227-7390/12/16/2521>

13. Dehghani, M., Hubálovský, Š., and Trojovský, P. (2021). Northern goshawk optimization: a new swarm-based

algorithm for solving optimization problems. Ieee Access, 9, 162059-

162080.<https://ieeexplore.ieee.org/abstract/document/9638618>.

14. Dehghani, M., Hubálovský, Š., and Trojovský, P. (2022). Tasmanian devil optimization: a new bio-inspired

optimization algorithm for solving optimization algorithm. IEEE Access, 10, 19599-19620.

<https://ieeexplore.ieee.org/document/9714388>.

15. Dehghani, M., and Trojovský, P. (2023). Osprey optimization algorithm: A new bio-inspired metaheuristic

algorithm for solving engineering optimization problems. Frontiers in Mechanical Engineering, 8, 1126450.

<https://doi.org/10.3389/fmech.2022.1126450>.

16. Dhiman, G., and Kumar, V. (2017). Spotted hyena optimizer: a novel bio-inspired based metaheuristic

technique for engineering applications. Advances in Engineering Software, 114, 48-70.

17. Dorigo, M., and Di Caro, G. (1999, July). Ant colony optimization: a new meta-heuristic. In Proceedings of the

1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406) (Vol. 2, pp. 1470-1477).

IEEE.<https://ieeexplore.ieee.org/abstract/document/782657>.

18. Faramarzi, A., Heidarinejad, M., Mirjalili, S., and Gandomi, A. H. (2020). Marine Predators Algorithm: A nature-

inspired metaheuristic. Expert systems with applications, 152, 113377.

19. Feurer, M., and Hutter, F. (2019). Hyperparameter optimization. *Automated machine learning: Methods,*

*systems, challenges*, 3-33.

20. Geem, Z. W., Kim, J. H., and Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search.

simulation, 76(2), 60-68.<https://doi.org/10.1177/003754970107600201>.

21. Haque, N. I., Khalil, A. A., Rahman, M. A., Amini, M. H., and Ahamed, S. I. (2021, September). Biocad: Bio-

inspired optimization for classification and anomaly detection in digital healthcare systems. In 2021 IEEE

International Conference on Digital Health (ICDH) (pp. 48-58). IEEE.

<https://ieeexplore.ieee.org/document/9581267>.

22. Hussien, A. G., Amin, M., Wang, M., Liang, G., Alsanad, A., Gumaei, A., and Chen, H. (2020). Crow search

algorithm: theory, recent advances, and applications. IEEE Access, 8, 173548-173565.

<https://ieeexplore.ieee.org/document/9195808>.

23. Li, Y., Zhang, Y., and Cai, Y. (2021). A new hyper-parameter optimization method for power load forecast based

on recurrent neural networks. Algorithms, 14(6), 163. <https://doi.org/10.3390/a14060163>.

24. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp Swarm algorithm: a bio-inspired

optimizer for engineering design problems. Adv Eng Softw 114:163–191

25. Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69,

46-61.<https://doi.org/10.1016/j.advengsoft.2013.12.007>.

26. Mirjalili, S., and Lewis, A. (2016). The whale optimization algorithm. Advances in engineering software, 95, 51-

67.

27. Naruei, I., and Keynia, F. (2022). Wild horse optimizer: A new meta-heuristic algorithm for solving engineering

optimization problems. Engineering with computers, 38(Suppl 4), 3025-3056.

<https://link.springer.com/article/10.1007/s00366-021-01438-z>.

28. Nirmal, A., Jayaswal, D., and Kachare, P. H. (2024). A Hybrid Bald Eagle-Crow Search Algorithm for Gaussian

mixture model optimisation in the speaker verification framework. Decision Analytics Journal, 10, 100385.

<https://doi.org/10.1016/j.dajour.2023.100385>.

29. Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). GSA: a gravitational search algorithm. Information

sciences, 179(13), 2232-2248.<https://doi.org/10.1016/j.ins.2009.03.004>.

30. Ravikiran, H. K., Jayanth, J., Sathisha, M. S., and Bindu, K. (2024). Optimizing Sheep Breed Classification with

Bat Algorithm-Tuned CNN Hyperparameters. SN Computer Science, 5(2), 219.

<https://doi.org/10.1007/s42979-023-02544-z>.

31. Shehab, M., Mashal, I., Momani, Z., Shambour, M. K. Y., AL-Badareen, A., Al-Dabet, S., ... and Abualigah, L.

(2022). Harris hawks optimization algorithm: variants and applications. Archives of Computational Methods

in Engineering, 29(7), 5579-5603. <https://link.springer.com/article/10.1007/s11831-022-09780-1>.

32. Storn, R., and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization

over continuous spaces. Journal of global optimization, 11, 341-359.,

<https://link.springer.com/article/10.1023/a:1008202821328>.

33. Sun, J., Wang, J., Zhu, Z., He, R., Peng, C., Zhang, C., ... and Wang, X. (2022). Mechanical performance prediction

for sustainable high-strength concrete using bio-inspired neural network. Buildings, 12(1), 65.

<https://www.mdpi.com/2075-5309/12/1/65>.

34. Trojovská, E., Dehghani, M., and Trojovský, P. (2022). Fennec fox optimization: a new nature-inspired

optimization algorithm. *IEEE Access*, *10*, 84417-84443. <https://ieeexplore.ieee.org/document/9853509>.

35. Trojovská, E., Dehghani, M., and Trojovský, P. (2022). Zebra optimization algorithm: A new bio-inspired

optimization algorithm for solving optimization algorithm. Ieee Access, 10, 49445-

49473.<https://ieeexplore.ieee.org/abstract/document/9768820>

36. Trojovská, E., and Dehghani, M. (2022). Clouded leopard optimization: a new nature-inspired optimization

algorithm. IEEE Access, 10, 102876-102906. <https://ieeexplore.ieee.org/document/9899451>.

37. Trojovský, P., and Dehghani, M. (2022). Pelican optimization algorithm: A novel nature-inspired algorithm for

engineering applications. Sensors, 22(3), 855. <https://doi.org/10.3390/s22030855>.

38. Trojovský, P., and Dehghani, M. (2022). Walrus optimization algorithm: a new bio-inspired metaheuristic

algorithm. <https://doi.org/10.21203/rs.3.rs-2174098/v1>.

39. Wolpert, D. H., and Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on

evolutionary computation, 1(1), 67-82.

40. Yang, X. S., and Deb, S. (2009). Cuckoo search via Lévy flights. In 2009 World congress on nature and biologically

inspired computing (NaBIC) (pp. 210-214). 10.1109/NABIC.2009.5393690.

41. Yang, X. S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.

Zhang, F., Chen, J., Mao, T., and Wang, Z. (2021). Feedback interval optimization for MISO LiFi systems. IEEE

Access, 9, 136811-136818.<https://ieeexplore.ieee.org/abstract/document/9557310>.

42. Zhang, M., and Wen, G. (2024). Duck swarm algorithm: theory, numerical optimization, and applications.

Cluster Computing, 1-29. <https://link.springer.com/article/10.1007/s10586-024-04293-x>.