Faculty of Sciences

Department of Computer Science and Information Systems

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 Faculty of 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 Computer Science and Information Systems

Faculty of science, BIUST

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

Dr. Hlomani Hlomani

Department of Computer Science and Information Systems

Faculty of science, BIUST

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

September, 2024

1. ABSTRACT

The Rhino Optimization Algorithm (ROA) is a 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. Testing will be conducted using datasets like MNIST and those from the UCI repository, with expected improvements in accuracy, convergence rates, and robustness compared to traditional methods like grid search.

2. TABLE OF CONTENTS

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	NFL	No Free Lunch
10	PSO	Particle Swarm Optimiser
11	ML	Machine Learning

4. 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 Mirjalili et al (2014), the ROA and the GWO are both 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.

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.

Problem Statement

The problem in the hyperparameter optimisation task is that many existing algorithms, such as the Crow Search Algorithm (Hussien et al., 2020) and the Elk Herd Optimizer (Al-Betar et al., 2024) often struggle with the context of finding optimal solutions to complex problems with a large search space. These limitations hinder their applicability in high-dimensional and complex solution spaces which creates computational inefficiency, where a flexible and efficient optimization strategy is crucial (Zhang and Wen, 2024).

Despite the advancement of various nature-inspired metaheuristic algorithms in addressing these complex optimization problems, algorithms such as the Wild Horse Optimizer (Naruei and Keynia, 2022) and the Harris Hawks Optimizer (Shehab et al. 2022) have significant challenges in achieving a balanced and adaptive exploration-exploitation strategy resulting in suboptimal solution quality in certain scenarios (Naruei and Keynia, 2022; Trojovský and Dehghani,2022).

Therefore, the question of whether the development of new algorithms, such as the Zebra (Trojovská et al., 2022), Pufferfish (et al., 2024), Walrus (Trojovský and Dehghani, 2022), Osprey (Dehghani and Trojovský, 2023), Duck (Zhang and Wen, 2024), and Goshawk (Dehghani et al., 2021) algorithms, remains necessary despite the wide array of existing metaheuristics represents a central issue in the analysis of optimization algorithms. The

resolution to this inquiry is grounded in the No Free Lunch (NFL) theorem by Wolpert and Macready (1997) who state that there is no assurance that an algorithm exhibiting superior performance in optimizing a specific subset of problems will maintain equivalent efficacy across all other optimization challenges.

General Objective

Developing a new bio-inspired algorithm, the ROA based on the collective social behaviours of Rhinoceroses is the main goal of this study to overcome the shortcomings of existing hyperparameter optimization techniques. The ROA strives to balance the exploration and exploitation phases typical of bio-inspired algorithms providing a novel approach to optimization tasks by simulating important aspects of Rhinoceros foraging food-searching and adaptive behaviours.

The effectiveness of ROA will be compared to other metaheuristic algorithms with an emphasis on high-dimensional spaces computational cost and precision in hyperparameter tuning for machine learning models. To ascertain the ROAs adaptability to different optimization challenges its generalizability will also be evaluated across datasets with varying sample sizes. Lastly by investigating whether Rhinoceroses adaptive territorial behaviour and momentum-based movement can hasten the convergence to optimal or nearly optimal solutions in machine learning models and other complex systems the study seeks to increase convergence rates in optimization tasks. Especially in high-dimensional spaces where conventional techniques might not work as well this bio-inspired mechanism increases ROAs efficacy. ROA can effectively find the ideal hyperparameters by adaptively switching between exploration and exploitation based on past results improving model performance while using less computing power.

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. Formulate a model of Rhinoceros Behaviour to simulate Rhinoceros' behaviour from their foraging and social patterns. This model will underpin the proposed Rhino Optimization Algorithm (ROA) and serve as the basis for its optimization process.
2. Apply ROA to Deep Learning Algorithms on prominent deep learning frameworks, including Support Vector Machine (SVM), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory Networks (LSTMs) to evaluate the effectiveness of the Rhino Optimization Algorithm in hyperparameter tuning for complex deep learning architectures.
3. Evaluate the Scalability of ROA across various datasets of differing sizes and complexities to determine how well ROA performs as the scale of the optimization problem increases, thereby providing insights into its applicability in real-world scenarios.

4. Analyse the convergence properties of the Rhino Optimization Algorithm in relation to its optimization tasks. This objective will focus on understanding how quickly and effectively the algorithm reaches optimal or near-optimal solutions, which is essential for practical applications in machine learning.
5. Compare the performance of ROA with established state-of-the-art hyperparameter optimization algorithms. This objective will quantitatively assess the relative advantages of the proposed algorithm, contributing to the literature on algorithmic performance in hyperparameter tuning.

Hypothesis/Research Questions

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

Hypothesis-1:

μ01= {*The Rhino Optimization Algorithm does not achieve better hyperparameter optimization results than existing metaheuristic algorithms*}.

Hypothesis-2:

μ02= {*The convergence rates of the Rhino Optimization Algorithm are not significantly faster than those of traditional optimization algorithms in hyperparameter tuning*}.

Hypothesis-3:

μ03= {*The Rhino Optimization Algorithm is not generalizable across various domains and problem sets beyond hyperparameter tuning*}.

Hypothesis-4:

μ04 = {*The Rhino Optimization Algorithm does not significantly improve model performance in deep learning architectures such as SVM, CNNs, RNNs, and LSTMs compared to conventional methods*}.

Research Questions

1. What are the specific factors influencing the effectiveness of the ROA in hyperparameter tuning compared to existing metaheuristic algorithms?
2. How do the exploration and exploitation phases modelled after Rhinoceros' behaviour contribute to the overall performance of the ROA in various optimization tasks?
3. To what extent does the ROA improve convergence rates in complex machine learning models, such as CNN and RNN?
4. How does the ROA perform across different datasets and problem domains compared to established

optimization algorithms like Bayesian optimization and Particle Swarm Optimization (PSO)?

5. What is the impact of varying hyperparameter settings on the performance of the ROA, and how can this inform best practices for algorithm tuning?

Expected Outcomes

The expected outcomes from the ROA can be categorized into several key areas, reflecting its performance, applicability, and contributions to the field of optimization:

1. Enhanced Optimization Performance:
    - Improved Solution Quality: The ROA is expected to deliver high-quality solutions for hyperparameter optimization tasks in machine learning, potentially outperforming existing algorithms in terms of accuracy and error rate.
    - Faster Convergence: The algorithm should demonstrate quicker convergence to optimal solutions compared to traditional optimization methods, especially in high-dimensional search spaces.
2. Robustness and Adaptability:
    - Resistance to Local Optima: By leveraging its unique leader-follower dynamic and adaptive exploration-exploitation balance, the ROA aims to reduce the likelihood of getting stuck in local optima, a common issue in many optimization algorithms.
    - Applicability to Noisy Datasets: the ROA is designed to handle variable and noisy datasets effectively, making it suitable for real-world applications in diverse fields such as finance, engineering, and data science.
3. Scalability:
    - Parameter Efficiency: The algorithm should require minimal parameter tuning, addressing scalability issues often seen in other algorithms. This feature will facilitate its application in larger and more complex optimization problems without substantial overhead.
4. Comparative Advantage:
    - Validation Against Established Algorithms: ROA's effectiveness will be validated against well-known metaheuristic algorithms (e.g., Particle Swarm Optimization, Genetic Algorithms), providing a clear benchmark for its performance and utility.
    - Contribution to the Field of Metaheuristics: By offering a novel optimization framework based on rhino behaviour, RO is expected to contribute to the advancement of metaheuristic research, providing insights and methodologies for future algorithm development.

5. Interdisciplinary Applications:
    - Wider Application Spectrum: The adaptability and effectiveness of RO should enable its use across various domains, enhancing predictive performance in areas such as machine learning, engineering, and other scientific disciplines.
    - Influence on Future Research: The insights gained from the application and performance of ROA may inform future research directions, encouraging further innovations in optimization techniques and algorithms.

Justification of the study

This study proposes the development of the ROA, an innovative algorithm inspired by the social dynamics, adaptive foraging, and territorial behaviours of Rhinoceroses. The aim is to address existing gaps by creating an algorithm that not only enhances collaboration among search agents but also dynamically adjusts its exploration and exploitation strategies based on environmental conditions. The objective is to provide a more robust and efficient optimization tool capable of tackling a wider array of complex problems in engineering, computer science, and other fields, ultimately contributing to the body of knowledge in metaheuristic optimization techniques.

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. Separating the algorithms into two major categories—Scalable and Non-Scalable—to classify the literature review according to scalability. Scalability generally refers to an algorithms capacity to efficiently manage complicated problems or large datasets whether in terms of computational resources time complexity or the ability to scale up to higher-dimensional spaces.

Methods that are scalable.

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. Due to this equilibrium, CLO can function effectively across several domains on larger datasets and on higher-dimensional problems. CLO performed better than ten other metaheuristic algorithms in key performance metrics when tested using 68 benchmark functions and four engineering design problems. Even though CLO successfully strikes a balance between exploration and exploitation it might not always be the best option and in some circumstances newer algorithms might perform better.

Dehghani et al. (2021) introduced 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). In the exploration phase, the goshawk randomly selects and targets prey, enhancing the algorithm's global search capabilities. During the exploitation phase, the goshawk chases the prey, thereby improving the efficiency of local search to refine the optimization process. The NGO can manage intricate search spaces with flexible tactics for expanded optimization assignments.

According to Trojovská et al. (2022) the Zebra Optimization Algorithm (ZOA) mimics two essential behaviours incorporated into this algorithm: foraging and predator-defence tactics. The ZOA guides the search for the best answers during the foraging phase by simulating how zebras follow a pioneer zebra to find food. The algorithms' ability to efficiently explore and take advantage of the search space is improved during the defence phase by imitating zebras escape patterns and group defence strategies to avoid predators. This dual-phase method enhances the algorithms resilience in resolving challenging optimization problems while also mirroring the adaptive tactics of zebras. The ZOA dual-phase approach exhibits flexibility in both exploration and exploitation it is scalable and effective for complex problems.

The Duck Swarm Algorithm (DSA) by Zhang and Wen (2024) draws inspiration from the foraging behaviour of ducks. The algorithm models two main phases: exploration, which simulates the search for food, and exploitation, which mimics the foraging process. These phases are critical in balancing the algorithm's global and local search capabilities. According to Zhang and Wen (2024) the DSA is capable of modelling both local and global search phases and is flexible enough to operate in expansive and intricate search spaces.

The Harris Hawks Optimizer (HHO) by Shehab et al. (2022) is inspired by the cooperative hunting strategies of Harris hawks. The HHO algorithm simulates the hawks' hunting tactics, such as surprise pouncing and coordinated group efforts, to optimize solutions in complex search spaces. HHO has been successfully applied across various domains, including image processing, power systems, networking, and medical applications, owing to its adaptive switching between different hunting strategies based on the prey's energy and escape patterns. The successful use in a variety of fields image processing power systems demonstrates its adaptability to a variety of problem types.

The Pufferfish Optimization Algorithm (PFOA) (Al-Baik et al. ,2024) aimed at addressing complex optimization problems through the emulation of pufferfish behaviour. The algorithm employs both exploration and exploitation strategies to identify optimal solutions within a search space, demonstrating its effectiveness for tackling exploration and exploitation high-dimensional challenges. PFOA initializes a population of candidate solutions, which are then iteratively refined through random search processes, with the objective of approaching the global optimum as closely as possible. This method reflects the adaptive strategies observed in nature, making it a valuable tool in the field of optimization research.

The Walrus Optimization Algorithm (WaOA) (Trojovský and Dehghani,2022) emulates various behaviours exhibited by walruses in their natural habitat, including feeding, migrating, escaping threats, and engaging with predators. The algorithm is systematically structured into three distinct phases: exploration, migration, and exploitation. This modelling effectively captures the adaptive strategies of walruses, thereby enhancing the algorithm's potential to address complex optimization challenges problems and simulating adaptive strategies.

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.

Inability to scale 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.

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. 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. So even though it seems promising more study is required to improve the algorithms effectiveness and versatility in high-dimensional environments.

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.

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. 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.

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. It begins with an initial randomly generated population, which is divided into groups led by stallions. Horses graze in the vicinity of their leader, and foals leave their group upon reaching puberty to join other groups, preventing inbreeding. This behaviour promotes diversity and exploration within the search space. When working with larger

populations or extremely complex search spaces the dynamic group-based approach known as Wild Horse Optimizer (WHO) (Naruei & Keynia 2022) may encounter scalability issues.

Trojovská et al. (2022) introduced the Fennec Fox Optimization (FFA) algorithm which is modelled on the unique digging and escape behaviours of the fennec fox. During the exploitation phase, the algorithm mimics the fox's digging ability, performing a local search to find optimal solutions. In the exploration phase, the algorithm simulates the fox's escape strategy, enabling a global search across the solution space. The computational demands and stochastic behaviour in complex search spaces may limit the algorithms scalability.

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. The performance of this algorithm is rigorously evaluated against 23 standard objective functions, showcasing its robust capabilities in both exploration and exploitation. Although it performs well on common objective functions scalability may be problematic when used for more complicated or large-scale problems.

The Hippopotamus Optimization (HO) algorithm (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. These behaviours are mathematically formulated to enhance both search efficiency and convergence speed within the algorithm. In high-dimensional or large-scale optimization tasks its navigational and territorial strategies might be less flexible.

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.

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-SVR and had comparable prediction accuracy increasing hyperparameter

optimization efficiency. 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.

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.

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. algorithms that are hybrid and performance-based benchmarks.

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 it showed a slower rate of convergence than algorithms like Grey Wolf Particle Swarm and Genetic Algorithm evaluations using SVMs Neural Networks and metrics like fitness functions convergence speed and solution quality demonstrated CSAs efficacy in complex optimization. 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.

Identification of the Existing Research Gaps

Exploration vs. Exploitation Trade-off:

Many of the discussed algorithms, such as Wild Horse Optimizer (WHO) and Clouded Leopard Optimization (CLO), balance exploration and exploitation phases (Trojovský and Dehghani, 2022). However, despite promising results, there remains a trade-off between exploration (global search) and exploitation (local search), as these algorithms can struggle to maintain a balance in different optimization environments (Al-Betar et al., 2024; Shehab et al., 2022). This limitation can lead to either premature convergence (favouring exploitation) or slow convergence (favouring exploration).

Convergence Issues in Noisy or Real-World Conditions:

Algorithms such as Hybrid Bald Eagle-Crow Search Algorithm (HBE-CSA) demonstrate superior performance in controlled settings but face challenges when applied to real-world, noisy datasets (Nirmal et al., 2024). The need for more robust methods to maintain solution quality in noisy conditions or high variability environments is evident.

Stochastic Behaviour and Dataset Constraints:

Some algorithms, like Beetle Antennae Search (BAS) and Salp Swarm Algorithm (SSA), show limitations due to their stochastic nature and sensitivity to initial conditions (Sun et al., 2022; Abdulsaed et al., 2023). These issues, combined with smaller datasets, suggest a need for methods that can better generalize across diverse datasets without compromising prediction accuracy.

ROA compared to bio-inspired algorithms

The Rhino Optimization Algorithm (ROA) is notable for its strong method of resolving some of the issues that other bio-inspired algorithms encounter. The Clouded Leopard Optimization (CLO) algorithm for example may not always perform optimally in some situations particularly when more recent algorithms are able to adapt more rapidly despite striking a balance between exploration and exploitation. Similarly, the ROAs ability to adaptively balance exploration and exploitation in even higher-dimensional and more complex problems could be advantageous to the Northern Goshawk Optimization (NGO) algorithm which successfully manages complex search spaces by simulating the goshawks dynamic hunting phases. The efficiency of ROA in local and global searches is improved by its iterative learning from prior findings which may cut down on the computation time needed by CLO NGO and other algorithms for efficient optimization in big high-dimensional datasets.

Furthermore, compared to certain algorithms such as the Zebra Optimization Algorithm (ZOA) and Duck Swarm Algorithm (DSA) which perform well in dual-phase exploration and exploitation but may have scalability issues in large search spaces, ROA more successfully handles scalability and continuous variable optimization. Although algorithms such as the Harris Hawks Optimizer (HHO) and the Pufferfish Optimization Algorithm (PFOA) exhibit

high domain adaptability they are vulnerable to scalability issues when used in large or extremely complex search spaces. By dynamically altering exploration and exploitation tactics based on prior iterations, ROAs adaptive mechanisms provide an advantage over other methods making it more effective for handling larger datasets without sacrificing accuracy or convergence speed. Due to its versatility ROA is also positioned as a flexible substitute for phase-locked algorithms like the Fennec Fox Optimization (FFA) and the Pelican Optimization Algorithm (POA) which rely on animal behaviours that might restrict flexibility across various optimization scenarios. By constantly learning from previous search results and modifying its tactics accordingly ROA can get around problems like scalability and local optima trapping that plague algorithms like the Bat Algorithm and the Crow Search Algorithm (CSA).

Justification of the Study

The ROA addresses significant research gaps in optimization techniques by incorporating behaviours observed in Rhinoceroses. Central to its design is the robust balance between exploration and exploitation phases, which reflects the Rhino's natural foraging strategies. In the wild, Rhinos often work collectively to find food, demonstrating adaptive behaviour to maximize resource utilization while avoiding threats. This social dynamic inspires the RO's algorithmic structure, allowing for a more effective search process that avoids common pitfalls of falling into local optima or slow convergence in complex landscapes, flexible search dynamics of the Rhino Optimizer align with the rhino's adaptability in varying environmental conditions.

Summary

The Rhino Optimization Algorithm (ROA) introduces a novel bio-inspired method that strikes a balance between exploration and exploitation in intricate high-dimensional search spaces thereby filling important gaps in hyperparameter optimization. To overcome the drawbacks of current algorithms ROA dynamically focuses on promising regions while avoiding local optima. This approach is modelled after the foraging behaviours of Rhinoceroses which explore widely but concentrate on productive areas when found. This technique is especially useful for fine-tuning hyperparameters in intricate machine learning tasks like SVMs and neural networks. It provides scalable effective search capabilities that can be tailored to a variety of applications that demand accurate model tuning and increased computational efficiency.

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.

Theoretical Methodology

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. 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 comprised of four major phases of work as given in Figure.1.

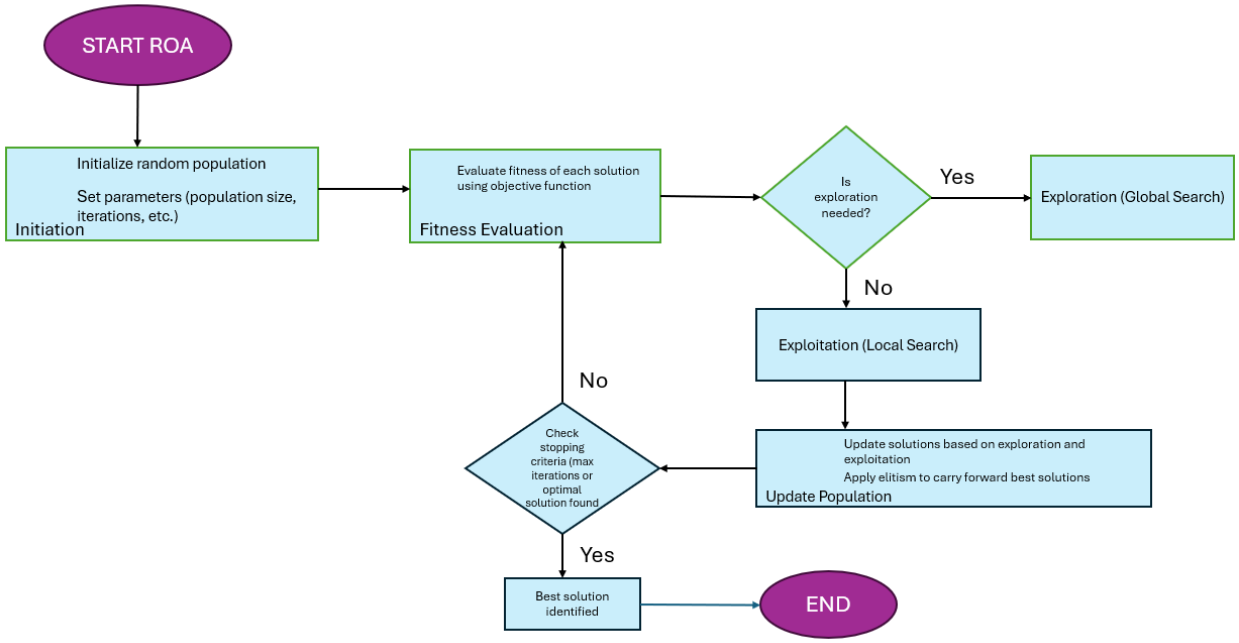

Figure 1: ROA flowchart

ROA modelling steps

To solve complicated real-world problems that call for a systematic quantitative approach optimization algorithm such as the Rhino Optimizer Algorithm (ROA) 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 models assessment. This methodical approach is necessary for ROA to operate effectively and dependably in challenging optimization tasks.

1. Initialization Phase

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

$$x_i^0 = xmin + r \cdot (xmax - xmin)\ for\ \ i = 1,2,\ldots,N$$

Here, r is a random vector uniformly distributed between 0 and 1, and x_{MIN} and x_{MAx} represent the lower and upper bounds of the solution space.

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

2. Fitness Evaluation and Solution Update

- Fitness Function Evaluation: Evaluate the fitness of each solution xi using the objective function $f(xi)$

$$f(xi) = Objective\ Function\ Value\ at\ xi$$

- Selection: Select the best-performing rhinoceros, i.e., the one with the lowest (or highest for maximization problems) objective function value:

$$x_{b\epsilon St}^{t+1} = \min\{f(x_1^{t+1}), f(x_2^{t+1}), \ldots, f(x_N^{t+1})\}$$

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

3. Exploration Phase

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

$$x_i^{t+1} = x_i^t + \beta \,.\, R.\, (x_{best}^t - x_i^t) + \gamma . P_{rand}$$

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

4. Exploitation Phase

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

$$x_i^{t+1} = x_i^t + \alpha \,.\, (\, (x_{best}^t - x_i^t) + \zeta \,.\, M \,.\, (\, x_j^t - x_k^t)$$

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

5. Testing the Rhino Optimizer

The steps for testing are as follows.

1. Data Preprocessing:
    - Input: Dataset (Training and Testing sets)
    - Action:
        - Normalize or standardize the dataset.
        - Split the dataset into training and testing sets.
        - Apply necessary preprocessing (handling missing values, encoding categorical variables).
    - Output: Pre-processed dataset
2. 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. 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.
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.
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.
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.

7. Analysis and Conclusion:

- Analyse the improvements in model performance due to hyperparameter tuning with the Rhino Optimizer.
- Draw conclusions regarding the effectiveness of ROA in optimizing hyperparameters for SVM and Neural Networks.

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.