

SINGIDUNUM UNIVERSITY
DEPARTMENT OF POSTGRADUATE STUDIES

**Application of Hybrid
Methods Combining Recurrent
Neural Networks and Modified
Metaheuristics for Time-Series
Forecasting**

(Ph.D. Thesis)

Mentor:

***Prof. Dr / Ph.D. Nebojša
Bačanin-Džakula***

Student:

Mohamed Salb

Belgrade, 2024

Acknowledgment

First and foremost, I would like to express my deepest gratitude to my family, especially my parents, for their unwavering support, love, and sacrifices. They are the reason I am where I am today, and I will never be able to fully repay them for being the greatest parents. I would also like to thank my aunt, Enshirah Mohamed, for her unquestioning support and for always believing in me. I wouldn't be here without her.

To my supervisor and mentor, Prof. Nebojša Bačanin-Džakula, I am profoundly grateful for your invaluable advice, continuous support, and patience. Your immense knowledge and mentorship have been instrumental in my academic journey, sparking my passion for education and research. I have learned so much under your guidance, and I truly appreciate everything you've done for me.

My sincere thanks go to my Bachelor's degree mentor, Prof. Miodrag Živković, whose influence and support have left a lasting impact on my academic life. His wisdom has shaped my understanding, and I am deeply grateful for everything I have learned from him.

To my friend and life mentor, Ali El-Sadai, thank you for being like a second brother to me here in Serbia. Your endless support, wisdom, and guidance have helped me navigate life's challenges with strength and clarity. I will always be grateful for your mentorship and friendship, which have shaped who I am today.

I would also like to thank my friend and mentor from work, Eilian Wyn Jones. Your guidance has helped me grow both personally and professionally, making me a better teacher. Your encouragement and belief in my potential have been constant sources of motivation, and I am deeply thankful for the life lessons you've shared.

Finally, I would like to thank Singidunum University for accepting me as a student and providing me with a great educational experience. The opportunities and study environment here have played a crucial role in my development. I am also grateful to everyone who has supported me in completing my studies.

Abstract

Time-series forecasting is becoming very important in many different industries, including cybersecurity, healthcare, energy, finance and cloud computing. This thesis explores the use of advanced metaheuristic algorithms together with recurrent neural networks (RNNs) to produce hybrid models enhancing the accuracy, stability, and generalizing ability of forecasting. We address problems including nonlinear behavior, seasonality, and noise in time-series data by utilizing the benefits of both Recurrent Neural Networks (RNNs) and metaheuristics.

This work uses an enhanced Moth Flame Optimizer (MFO) to improve the optimization of logistic regression models, which boosts the accuracy of spam email classification. Furthermore, the efficiency of hybrid models in capturing intricate seasonal trends is demonstrated by the use of an Enhanced Sine Cosine Algorithm (ESCA) in conjunction with Long Short-Term Memory (LSTM) networks for wind energy ahead-of-time prediction. With a modified Reptile Search Algorithm, Convolutional Neural Networks (CNNs) and Extreme Gradient Boosting (XGBoost) show potential in hybrid techniques in enhancing the security of IoT networks in the field of cybersecurity.

Moreover, the practical benefits in the field of healthcare are shown by the identification of seizures in electroencephalogram (EEG) data using metaheuristic-optimized recurrent neural networks (RNNs). The study on improving the performance of LSTM networks utilizing the Enhanced Harris Hawks Optimization Algorithm for crude oil price prediction underlines the need to integrate different methodologies in financial modelling. Using modified metaheuristic-optimized LSTM networks for earthquake magnitude prediction emphasizes the several applications of these hybrid approaches in preventing disasters.

Additionally, this study addresses the challenges associated with cloud instance pricing, which are increased by the complex nature of budgeting due to fluctuating demand and changing costs. A modified optimizer is introduced and tested on cloud instance data using multi-headed recurrent architectures, specifically LSTM and Gated Recurrent Unit (GRU) networks. The GRU model, enhanced by this technique, attained remarkable forecasting

precision, validated through statistical analysis and revealed by using artificial intelligence techniques to clarify feature importance.

The thesis significantly improves the current degree of knowledge in time-series data prediction, resulting in new possibilities for the application of hybrid methods across various areas. By addressing the limitations of previous approaches and using the powers of RNNs and metaheuristics, this work helps predictive modelling and its useful applications to advance.

Keywords: *Time-series forecasting, Recurrent Neural Networks, Metaheuristic algorithms, Long Short-Term Memory, Sine Cosine Algorithm, Moth Flame Optimizer, IoT security, Seizure detection, Crude oil price forecasting, Swarm intelligence algorithms, Earthquake forecasting, wind energy forecasting, Spam filtering, Logistic regression, Cloud computing, Gated recurrent units, Particle swarm optimization, Spot price forecasting*

Table of Contents

1. Introduction	- 1 -
1.1. Hypothesis	- 5 -
1.2. Overall Hypothesis	- 6 -
1.3. Objectives	- 6 -
1.4. Overall objectives	- 8 -
1.5. Thesis Structure	- 8 -
2. Process of Optimization	- 9 -
3. Stochastic Optimization Techniques.....	- 15 -
3.1. Preliminaries and Basics.....	- 15 -
3.2. Taxonomy of Stochastic Optimization Methods.....	- 18 -
4. Swarm Intelligence Metaheuristics.....	- 21 -
4.1. Features of Swarm Intelligence Systems.....	- 22 -
4.2. Basic Principles of Swarm Algorithms	- 24 -
4.3. Representatives of Swarm Algorithms.....	- 25 -
4.3.1. Particle Swarm Optimization (PSO)	- 26 -
4.3.2. Ant Colony Optimization (ACO).....	- 27 -
4.3.3. Artificial Bee Colony (ABC)	- 28 -
4.3.4. Bat Algorithm (BA)	- 28 -
4.3.5. Firefly Algorithm	- 30 -
4.3.6. Cuckoo Search(CS).....	- 30 -
4.3.7. Whale Optimization Algorithm (WOA)	- 31 -
5. Swarm Intelligence Applications.....	- 33 -
5.1. The Experiments Environment.....	- 33 -
5.2. Application 1: Enhancing Internet of Things Network Security Using Hybrid CNN and XGBoost Model Tuned via Modified Reptile Search Algorithm	- 34 -
5.2.1. Methods.....	- 36 -
5.2.1.1. Original RSA	- 36 -
5.2.1.2. Modified RSA.....	- 38 -

5.2.2. Experimental Setup	- 40 -
5.2.3. Experimental Outcomes	- 42 -
5.2.3.1. Binary Classification Experiment.....	- 42 -
5.2.3.2. Multi-Class Classification Experiment.....	- 48 -
5.2.3.3. Outcome Statistical Validation	- 53 -
5.2.3.4. Best Model SHAP Interpretation.....	- 55 -
5.3. Application 2: The Long Short-Term Memory Tuning for Multi-Step Ahead Wind Energy Forecasting using enhanced Sine Cosine Algorithm and Variation Mode Decomposition.....	- 57 -
5.3.1. Methods.....	- 58 -
5.3.1.1. Overview of Long Short-Term Memory (LSTM)	- 58 -
5.3.1.2. Variational Mode Decomposition (VMD).....	- 60 -
5.3.1.3. Original Sine Cosine Algorithm (SCA).....	- 61 -
5.3.1.4. Enhanced Sine Cosine Algorithm (ESCA).....	- 62 -
5.3.2. Experimental Setup	- 64 -
5.3.2.1. Datasets.....	- 64 -
5.3.2.2. Experimental Setup.....	- 65 -
5.3.2.3. Experimental Results	- 66 -
5.4. Application 3: Training Logistic Regression Model by Enhanced Moth Flame Optimizer for Spam Email Classification.....	- 70 -
5.4.1. Method	- 71 -
5.4.1.1. Original Moth Flame Algorithm.....	- 71 -
5.4.1.2. Improved Moth Flame Optimization	- 75 -
5.4.2. Experimental Setup	- 77 -
5.4.2.1. Dataset	- 77 -
5.4.2.2. Preprocessing for Data Representation.....	- 77 -
5.4.3. Results and Discussion.....	- 78 -
5.4.4. Comparison of Classifiers	- 79 -
5.5. Application 4: Seizure Detection Via Time Series Classification Using Modified Metaheuristic Optimized Recurrent Networks.....	- 82 -

5.5.1. Methods.....	- 84 -
5.5.1.1. Original Sinh Cosh Optimiser	- 84 -
5.5.1.2. Modified SCHO Algorithm	- 87 -
5.5.2. Experiment Setup	- 88 -
5.5.3. Experiment Outcomes	- 89 -
5.6. Application 5: Long Short-Term Memory Tuning by Enhanced Harris Hawks Optimization Algorithm for Crude Oil Price Forecasting	- 95 -
5.6.1. Methods.....	- 97 -
5.6.1.1. Harris Hawks Optimization Algorithm	- 97 -
5.6.1.1.1. Exploration Phase	- 97 -
5.6.1.1.2. Phase Transition.....	- 98 -
5.6.1.1.3. Exploitation Phase	- 98 -
5.6.1.2. Deawbacks of the Original Algorithm.....	- 101 -
5.6.1.3. HHO with Disputation Operator	- 101 -
5.6.2. Experimental Setup	- 105 -
5.6.2.1. Dataset, Hyperparameters, Solution Encoding	- 105 -
5.6.2.2. Evaluation Metrics, Basic Setup and Opponent Methods	- 106 -
5.6.3. Experimental Results and Discussion	- 108 -
5.6.3.1. Simulations with LSTM	- 108 -
5.6.3.2. Simulations with VMD-LSTM.....	- 110 -
5.6.3.3. Comparison with Other Well-Known Methods.....	- 112 -
5.7. Application 6: Exploring the Potential of Modified Metaheuristic Optimized Long-Short Term Memory Neural Networks for Earthquake Magnitude Forecasting- 117	-
5.7.1. Methods.....	- 118 -
5.7.1.1. Basic SCA.....	- 119 -
5.7.1.2. Hybrid Adaptive SCA.....	- 119 -
5.7.2. Experimental Setup	- 121 -
5.7.3. Simulation Outcomes	- 123 -

5.8. Application 7: Cloud Spot Instance Price Forecasting Multi-Headed Models Tuned Using Modified PSO	- 128 -
5.8.1. Methods.....	- 129 -
5.8.1.1. Modified PSO Algorithm	- 129 -
5.8.2. Experimental Setup	- 133 -
5.8.3. Simulation Outcomes	- 135 -
5.8.3.1. Simulations with LSTM Networks	- 135 -
5.8.3.2. Simulations with GRU Networks	- 137 -
5.8.3.3. Comparison between the Best LSTM and GRU Models.....	- 139 -
5.8.4. Outcomes Statistical Validation	- 141 -
5.8.5. Best-Performing Model Interpretation	- 142 -
6. Conclusion	- 144 -
7. References.....	- 147 -

List of Figures

Figure 1 Optimization Methods.....	- 16 -
Figure 2 Categories of evolutionary computation.	- 19 -
Figure 3 Bridge made by Red Ants	- 21 -
Figure 4 Birds Flocks	- 23 -
Figure 5 Flowchart of the introduced framework.....	- 41 -
Figure 6 Binary classification evaluation outcome distribution.....	- 44 -
Figure 7 Objective and indicator function convergence plots.....	- 45 -
Figure 8 Objective and indicator function distribution plots for multi-class classification. ..	- 49 -
Figure 9 Objective and indicator function convergence plots for multi-class classification..	- 50 -
Figure 10 GIRSA XGBoost best performing multi-class classification model.....	- 52 -
Figure 11 Objective function KDE plots for binary and multiclass classification.....	- 55 -
Figure 12 SHAP analysis outcomes for best CNN model.....	- 56 -
Figure 13 SHAP analysis outcomes for best XGBoost model.....	- 57 -
Figure 14 Wind farm dataset virtualization with data split.	- 65 -
Figure 15 Decomposition mode visualizations for each input feature.	- 66 -
Figure 16 Objective and R^2 convergence and distribution plots.....	- 69 -
Figure 17 Block diagram of the AVMFO-LR classifier.	- 77 -
Figure 18 Objective and indicator function distributions for over 30 experimental executions.....	- 91 -
Figure 19 Objective and indicator function convergence plots for the best-performing models.....	- 92 -
Figure 20 Best constructed model confusion matrix.....	- 94 -
Figure 21 Train/validation/test of daily Brent oil prices used in experiments.	- 106 -
Figure 22 LSTM Comparative analysis visualizations.	- 113 -
Figure 23 Five steps ahead predicted vs actual price values cast by all tested algorithm.	- 114 -
Figure 24 Five steps ahead predicted vs actual price values presented by the LSTM-HHO- DA algorithm.....	- 114 -
Figure 25 VMD-LSTM Comparative analysis visualizations.....	- 115 -

Figure 26 Five steps ahead predicted vs actual price values cast by all tested algorithm.	115 -
Figure 27 Five steps ahead predicted vs actual price values presented by the VMD-LSTMHHO-DA algorithm.	116 -
Figure 28 Dataset Visualization.	121 -
Figure 29 Objective and indicator distribution plots.	124 -
Figure 30 Objective and indicator function warm plots.	125 -
Figure 31 Objective and indicator function convergence.	127 -
Figure 32 Introduced optimization framework flowchart.	135 -
Figure 33 Objective function distribution plots for LSTM optimizations.	137 -
Figure 34 Objective function distribution plots for GRU optimizations.	139 -
Figure 35 Convergence diagrams for GRU optimizations.	141 -
Figure 36 Best Performing GRU model SHAP interpretations.	143 -

List of Tables

Table 1 variations between deterministic and stochastic optimization.	- 17 -
Table 2 Objective function overall outcomes.....	- 43 -
Table 3 Indicator function overall outcomes.....	- 43 -
Table 4 Binary classification detailed metrics.....	- 46 -
Table 5 Control parameter selections made by each metaheuristic for respective best-performing.	- 47 -
Table 6 Objective function outcome for multi-class classification.	- 48 -
Table 7 Indicator function outcome for multi-class classification.	- 48 -
Table 8 Detailed comparison between best-performing models for multi-class classification.	- 51 -
Table 9 Control parameter selections made by each metaheuristic for respective best-performing	- 53 -
Table 10 Shapiro–Wilk normality tests.....	- 54 -
Table 11 Wilcoxon signed-rank test values exhibiting p-values for all three experiments (GIRSA vs. Others.).	- 54 -
Table 12 Overall objective function performance metrics for each tested approach.	- 66 -
Table 13 Detailed metrics for each prediction step of best-performing models of every evaluated approach.	- 67 -
Table 14 Selected parameters for best-performing models by beach metaheuristic.	- 69 -
Table 15 The highest and lowest classification performance comparison on the CSDMC2010 dataset.	- 79 -
Table 16 AVMFO-LR’s performance and comparison on the Enron 1 dataset.....	- 80 -
Table 17 Overall objective function outcomes over 30 experimental executions.....	- 89 -
Table 18 Overall indicator function outcomes over 30 experimental executions.	- 90 -
Table 19 Detailed metrics for the best-performing models constructed by each optimizer...-	93 -
Table 20 Metaheuristic optimized parameters for best-performing models.....	- 94 -
Table 21 Control parameter values used for the HHO-DO algorithm.	- 103 -
Table 22 LSTM control parameter boundaries.	- 106 -
Table 23 Objective function metrics for LSTM application.	- 109 -

Table 24 The R2, MAE, MSE, and RMSE metrics of the best performing LSTM approach.	- 109 -
Table 25 Objective function metrics for VMD-LSTM application.....	- 111 -
Table 26 The R2, MAE, MSE, and RMSE metrics of the best performing VMD-LSTM approach.....	- 111 -
Table 27 LSTM optimizer parameters and their respective ranges.....	- 122 -
Table 28 Objective function comparisons in terms of the best worst mean and median constructed models.	- 123 -
Table 29 Overall outcomes for three-step ahead forecasts.....	- 124 -
Table 30 Detailed metrics for the highest-achieving structures for one, two and three-step ahead steps.....	- 125 -
Table 31 Parameter selection for the highest-achieving networks made by each optimizer..	- 127 -
Table 32 Outcomes for LSTM simulations.	- 136 -
Table 33 Outcomes for each optimizer included in comparative analysis.	- 138 -
Table 34 Comparison between the best performing GRU and LSTM model.	- 140 -
Table 35 Statistical evaluation outcomes.	- 142 -

List of Equations

Equation 1 Update rules used in the Bat Algorithm.....	- 29 -
Equation 2 Population of agents.....	- 37 -
Equation 3 Generating Population.....	- 37 -
Equation 4 Exploration Mechanism.	- 37 -
Equation 5 Sensitivity.....	- 38 -
Equation 6 Exploitation with two Distinct Hunting Strategies.	- 38 -
Equation 7 Cohen’s Kappa Statistic.	- 42 -
Equation 8 Forgetting gate.	- 59 -
Equation 9 Determine whether information should be contained.	- 59 -
Equation 10 Generate new cell state.....	- 59 -
Equation 11 Output gate.	- 59 -
Equation 12 constrained variational problem.....	- 60 -
Equation 13 quadratic penalty alongside Lagrangian multiplies.....	- 60 -
Equation 14 attained modes.....	- 60 -
Equation 15 wiener filtering.	- 61 -
Equation 16 Sin and Cos algorithms	- 61 -
Equation 17 Incorporated Sin & Cos algorithms	- 61 -
Equation 18 Update r_1 value.	- 62 -
Equation 19 Reptile search algorithm.	- 63 -
Equation 20 Mean absolute error.....	- 65 -
Equation 21 Mean Squared Error.	- 65 -
Equation 22 Root Mean Squared Error.	- 65 -
Equation 23 coefficient of determination.	- 65 -
Equation 24 MFO matrix of moths population.	- 72 -
Equation 25 Relevant objective value of moths.	- 72 -
Equation 26 MFO matrix of flame population.	- 72 -
Equation 27 Relevant objective value of flames.	- 73 -
Equation 28 MFO three different functions.	- 73 -
Equation 29 Random population function (I).....	- 73 -
Equation 30 move the moth and explore the search space function (P).....	- 74 -
Equation 31 spiral function (S).....	- 74 -

Equation 32 Space between moth and flame.....	- 74 -
Equation 33 Decreases the number of flames during the duration of the iterations.	- 75 -
Equation 34 Every one of the unique terms throughout all documents.....	- 78 -
Equation 35 Calculation of the weights.....	- 78 -
Equation 36 Binary function indicating the membership of a document vector in a category.	- 78 -
Equation 37 Random population generated.....	- 84 -
Equation 38 Position of agents.	- 84 -
Equation 39 Balance between exploration and exploitation.	- 85 -
Equation 40 Agents update during exploration phase.	- 85 -
Equation 41 Weighted coefficient for the particular agent.....	- 85 -
Equation 42 The other strategy applied in exploration.....	- 85 -
Equation 43 W_2 weight coefficient.....	- 86 -
Equation 44 Agents update during exploitation phase.	- 86 -
Equation 45 W_3 weight coefficient.....	- 86 -
Equation 46 The other strategy applied in exploitation.....	- 86 -
Equation 47 quasi-reflective learning (QRL).	- 87 -
Equation 48 Error rate.	- 89 -
Equation 49 Cohen's kappa.....	- 89 -
Equation 50 position of a search agent in the upcoming iteration.....	- 97 -
Equation 51 average position of the entire population.	- 98 -
Equation 52 energy of a prey.....	- 98 -
Equation 53 Soft besiege.	- 99 -
Equation 54 hard besiege.....	- 99 -
Equation 55 soft besiege with progressive rapid dives.....	- 99 -
Equation 56 Assess the possible results of taking this direction to previous attack patterns. -	100 -
Equation 57 Levy flight function.	- 100 -
Equation 58 Final strategy for updating the positions of hawks in the soft besiege phase. ...	100 -
Equation 59 agents try to reduce the distances between their average location and prey.	- 100 -
Equation 60 social network search algorithm.	- 101 -
Equation 61 AF values in each iteration.....	- 102 -

Equation 62 Overall MSE metric.	- 107 -
Equation 63 Mechanism of firefly.....	- 119 -
Equation 64 Improve the computational performance of Firefly.	- 120 -
Equation 65 Index of alignment.	- 122 -
Equation 66 Generate random agent positions and velocities.	- 129 -
Equation 67 The present speed of the particle's.....	- 130 -
Equation 68 Chaotic elite learning to enhance individual particles.	- 130 -
Equation 69 Location update for the optimum individual.....	- 131 -
Equation 70 Lévy flight procedure to enhance updates of the positions.....	- 131 -
Equation 71 Update rule for a particle's position.	- 131 -
Equation 72 Lévy flight distribution.	- 132 -
Equation 73 Gamma function.....	- 132 -
Equation 74 The FA's search procedure.	- 132 -
Equation 75 Index of Agreement (IoA).....	- 134 -

1. Introduction

Time-series forecasting plays an essential part in a variety of domains, including finance, energy, healthcare, and natural disaster management. Predicting potential outcomes by using past data is necessary for effective decision-making and planning. Nevertheless, time-series forecasting creates a major obstacle due to the complexity and non-linearity that exist in real-world data which makes it an NP-hard problem. Traditional methods tend to fail to identify the complex patterns and relationships in time-series data, resulting in the need for more sophisticated techniques.

Time-series forecasts in finance play an important role in guiding investment decisions, managing risks, and conducting analysis of markets. Accurate forecasts of the generation and consumption of energy play an important role as well in optimizing electrical grid performance, incorporating renewable energy sources, and arranging scheduled repairs in the energy sector.

Time-series forecasting also plays a crucial role in healthcare by anticipating pandemics, effectively administering resources in hospitals, and closely observing the condition of patients via wearable devices. precise forecasts have the potential to result in immediate actions, more effective distribution of resources, and ultimately, improved health outcomes for patients. When it comes to dealing with natural disasters, time-series forecasts play a key part in providing early warnings for incidents like hurricanes, floods, and earthquakes. These forecasts make it possible to quickly evacuate and to take preparation measures in advance.

Nevertheless, precise forecasting of time-series data can be very difficult due to its complex nature, which includes non-linearity, seasonality, and noise. Standard methods of analyzing data, like Autoregressive Integrated Moving Average (ARIMA), regularly struggle with identifying these details, causing less-than-ideal predictions. Machine learning models, especially RNNs, provide an adaptable and reliable technique for time-series forecasting. However, The efficacy of such models is highly reliant upon choosing the right hyperparameters.

Recurrent Neural Networks (RNNs) are a particular kind of artificial neural networks that performs well at detecting trends in sequences of data, particularly time-series data. RNNs excel in operations that require an awareness of temporal patterns, such as speech detection, language modelling, and time-series forecasting. RNNs have a unique ability to hold onto a hidden state that records data from past inputs, enabling them to effectively model temporal dependencies.

Training RNNs, specifically its more advanced versions such as LSTM and GRU networks, can be quite challenging to compute and demands precise adjustment of various hyperparameters. The selection of these hyperparameters, including the learning rate, number of layers, and number of neurons in each layer, has a major impact on the network's ability to perform. Conventional techniques for hyperparameter optimization, such as grid search or random search, can be quite inefficient and can fail to produce the most optimum set of parameters.

Metaheuristic techniques were employed to optimize the hyperparameters of RNNs in order to tackle these challenges. Metaheuristics are optimization techniques what get inspiration from natural phenomena and biological mechanisms. They leverage randomness to effectively discover the search space and provide reliable solutions. Metaheuristics are capable of finding solutions that are near the global optimum within a reasonable time, compared to standard heuristic algorithms that often remain stuck in local optima. Their outstanding efficacy allows them to be ideal for tackling challenging problems such as time-series forecasting.

Swarm intelligence algorithms like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and more recent techniques such as the Moth Flame Optimizer (MFO) and Sine Cosine Algorithm (SCA) have demonstrated significant potential among the various metaheuristic algorithms. These algorithms mimic the cooperative actions of decentralized systems, for instance flocks of birds or colonies of ants, to solve complex optimization challenges. Swarm intelligence algorithms, in particular, make use of the combined problem-solving abilities of simple agents to successfully discover and exploit the search space.

Swarm algorithms such as PSO and ACO have gained significant popularity in solving optimization problems because of their straightforwardness and impressive performance. PSO emulates the social behavior of birds flocking or fish schooling to discover optimal solutions, while ACO draws inspiration from the searching habits of ants and their skill to identify the shortest routes to food resources. These algorithms keep improving potential solutions based on specified criteria for superior results, usually involving stages of exploration and exploitation.

After making adjustments to these metaheuristics, their accuracy can be improved, resulting in more precise and reliable outcomes. For example, by adding more mechanisms for coordinating exploration and exploitation, the MFO can greatly enhance its ability to locate optimal solutions in search areas with many dimensions. By incorporating new search strategies, the convergence speed and accuracy of SCA can be enhanced.

Combining RNNs with metaheuristic algorithms shows a promising solution to this challenge. Through the utilization of metaheuristics, the hyperparameters of RNNs can be optimized, resulting in hybrid models that outperform both methods separately. The metaheuristic method enhances the efficacy of exploring the hyperparameter space, while the RNN utilizes its impressive sequence modelling features to make precise predictions.

Spam email classifying is an essential task in cybersecurity that aims to recognize and block unwanted and possibly dangerous emails. Current spam recognition technology, such as rule-based systems and basic machine learning models, frequently fails to keep up with spammer's constantly changing methods. As an outcome, more advanced methods are needed to increase the efficiency of detection and eliminate misleading results.

Sustainable energy forecasting, specifically wind energy forecasting, is critical for improving wind farm operations and supervision. precise estimations of wind energy output aid in balancing supply and demand, lowering operating expenses, and incorporating green energy into the electrical system. Wind energy forecasting, on the other hand, is difficult since wind patterns are naturally variable and unpredictable.

The Internet of Things (IoT) ecosystem, which is defined by linked electronics and massive volumes of data, faces unique security problems. Ensuring the security of IoT

networks is critical for securing sensitive data and preserving the integrity of linked systems. Existing security methods frequently fail because of the dynamic and scattered nature of IoT settings.

Epileptic seizures, which are identified by unexpected and repeated electrical abnormalities in the brain, need precise and quick identification for appropriate care. Current seizure detection approaches, such as manual EEG analysis, are consuming a lot of time and subject to inaccuracies. Automated seizure detection with machine learning is a potential option, however, the efficacy of these models is significantly reliant upon the hyperparameters used.

Many variables impact crude oil prices, such as geopolitical occurrences, supply and demand swings, and market speculation. Correct forecasting of crude oil prices is critical for energy industry decision-making, assisting investors in risk management and making accurate investment decisions. Typical forecasting techniques frequently fail to reflect the complex and dynamic character of oil price changes.

Natural catastrophes, such as earthquakes, have a huge influence on populations and nations, inflicting widespread property damage, disrupting livelihoods, and frequently resulting in casualties. Earthquakes are unpredictable, and their risk of triggering severe aftershocks makes precise forecasting critical for disaster planning and control. Current earthquake forecasting systems frequently depend on historical data and geological studies, but they fail to offer good predictions since earthquake activity is complicated and dynamic.

Cloud computing has become essential for hosting applications that demand significant processing resources, providing accurate cloud spot price forecasting critical for managing expenses and optimal resource allocation. With the increasing demand for cloud infrastructure, companies must anticipate pricing to effectively manage budgets and automatically scale resources. Inaccurate price forecasts may result in excessive or insufficient provisioning, leading to financial disaster and operational inefficiency.

All the previously mentioned examples are applications in time-series forecasting and each one of them has its own set of unique issues that demand specialized models to

effectively tackle them. This thesis explores various applications of hybrid methods in time-series forecasting, showcasing their adaptability and efficacy.

1.1. Hypothesis

The following hypotheses of all the research conducted for this thesis were formulated based on the applications in section 5:

- a) The hybrid CNN-XGBoost model will perform better than traditional models in detecting IoT network security threats. (**Application 1**)
- b) Tuning the model with the modified Reptile Search Algorithm will lead to significant improvements in accuracy and efficiency. (**Application 1**)
- c) The LSTM-ESCA model will outperform traditional forecasting models in accuracy. (**Application 2**)
- d) VMD preprocessing will significantly enhance the forecasting performance by decomposing the wind energy data into more predictable subcomponents. (**Application 2**)
- e) The MFO-optimized logistic regression model will achieve higher accuracy in spam email classification compared to models optimized by traditional methods. (**Application 3**)
- f) The MFO will enhance the convergence speed and stability of the logistic regression training process. (**Application 3**)
- g) Recurrent Networks optimized with a modified metaheuristic algorithm will significantly improve the accuracy and timeliness of seizure detection compared to existing methods. (**Application 4**)
- h) The HHO-DO optimized LSTM model will achieve superior accuracy in crude oil price forecasting compared to models optimized by other algorithms. (**Application 5**)
- i) The introduction of the disputation operator in the HHO-DO will significantly enhance the optimization process. (**Application 5**)
- j) The modified sine-cosine algorithm (HASCO) will outperform other metaheuristic optimization algorithms in tuning LSTM parameters for earthquake magnitude forecasting. (**Application 6**)

- k) LSTM networks, when optimized using metaheuristic algorithms, will provide more accurate earthquake magnitude predictions compared to traditional methods. **(Application 6)**
- l) The modified PSO algorithm will outperform standard PSO and other metaheuristics in optimizing LSTM and GRU models for cloud spot price forecasting. **(Application 7)**
- m) GRU models optimized by the modified PSO will achieve better forecasting accuracy compared to LSTM models and traditional forecasting techniques like XGBoost and ARIMA. **(Application 7)**
- n) The proposed models will show statistically significant improvements in accuracy, as measured by MSE and MAE. **(Application 7)**
- o) SHAP analysis will reveal that specific historical data (lags) are critical to improving the accuracy of cloud spot price forecasts. **(Application 7)**

1.2. Overall Hypothesis

The following are the overall hypotheses of the research conducted for this thesis:

- A higher accuracy can be achieved by hybrid and optimized models such as (CNN-XGBoost, and LSTM-ESCA) in their respective applications compared to traditional models.
- Modified algorithms for optimization will significantly increase the efficacy and precision of machine learning models.
- Preprocessing techniques for instance Variation Mode Decomposition (VMD) can improve the precision and efficacy of forecasting models.

1.3. Objectives

In this thesis, there are multiple goals for each of the proposed applications in section 5: -

1) Application 1 [1]

- a) Proposing a CNN-centric approach for feature reduction in IoT datasets;
- b) Utilizing XGBoost for the classification of intrusion events;
- c) Introducing a modified algorithm specifically designed for optimization;
- d) Implementing the proposed methodology on real-world data, addressing a pressing real-world challenge.

2) Application 2 [2]

- a) Improve multi-step ahead wind energy forecasting using LSTM networks.
- b) Utilize the Enhanced Sine Cosine Algorithm (ESCA) for tuning the LSTM parameters.
- c) Incorporate Variation Mode Decomposition (VMD) to preprocess the input data and enhance prediction accuracy.
- d) Implementing the proposed methodology on real-world data, addressing a pressing real-world challenge.

3) Application 3 [3]

- a) Develop a logistic regression model optimized by the Moth Flame Optimizer (MFO) for spam email classification.
- b) Evaluate the performance of the MFO-optimized model against traditional optimization methods.
- c) Demonstrate the efficiency of the MFO in enhancing the classification accuracy of the logistic regression model.

4) Application 4 [4]

- a) Propose an advanced recurrent network approach for seizure detection based on time series classification.
- b) Optimize the network using a modified metaheuristic algorithm to improve detection accuracy and timeliness.

5) Application 5 [5]

- a) A proposal for an improved HHO developed to further enhance the admirable performance of the original
- b) The use of the VMD technique to tackle the complexity of crude oil time series price data.
- c) Tuning LSTM parameter values with the novel proposed approach to enhance the performance of price forecasting, and
- d) Furthering price forecasting research by applying hybrid methods between LSTM and swarm intelligence

6) Application 6

- a) To propose a robust time-series based approach for earthquake prediction using LSTM networks.
- b) To incorporate metaheuristic optimizer algorithms for tuning LSTM parameters to improve performance.

- c) To introduce a modified version of the sine-cosine algorithm (HASCO) and evaluate its effectiveness compared to other contemporary optimization methods.

7) Application 7 [6]

- a) A modified PSO is proposed to address observed drawbacks.
- b) Integrated into ML framework for cloud spot instance price forecasting.
- c) The performance is assessed using real-world datasets and compared to other sophisticated metaheuristics.
- d) Shapley additive explanations (SHAP) [7] is used to analyze top model findings and evaluate feature usage.

1.4. Overall objectives

The overall objectives of this thesis are outlined below:

- Enhance model performance by developing and optimizing CNN, XGBoost, logistic regression, and LSTM models.
- Using advanced optimization algorithms like Modified Reptile Search, Enhanced Sine Cosine Algorithm (ESCA), Moth Flame Optimizer (MFO), Harris Hawks Optimization (HHO), and the modified sine-cosine algorithm (HASCO).
- Apply methodologies to real-world data in IoT security, wind energy, spam classification, seizure detection, crude oil prices, and earthquake prediction.
- Assess the performance of suggested hybrid models to traditional approaches to ensure accuracy and efficiency improvements, while analyzing feature importance through techniques like Shapley Additive Explanations (SHAP).

1.5. Thesis Structure

This thesis is structured in the following order. In Section 2 you will find a summary of the optimization procedure with some of the well-known optimizations and some of the most common optimization challenges. In Section 3 you will see an overview of the stochastic optimization techniques and evolutionary computation. Section 4 is dedicated to swarm intelligence metaheuristics algorithms, features, basic principles and examples of swarm intelligence algorithms. Section 5 includes the applications of hybrid methods for time-series forecasting. Finally, Section 0 has the ultimate conclusion.

2. Process of Optimization

In computer science and related fields, optimization stands for the procedure of enhancing the performance, efficiency, or effectiveness of techniques, programs, systems, or procedures. It implies fine-tuning many components in order to produce the finest potential result. This might involve reducing the utilization of computing resources such as memory or time spent on processing, increasing execution velocity, boosting code readability, and increasing the entire system functioning. Optimization in computer-related fields, as in other areas of science, seeks to identify the best configuration (setting) or solution that achieves a particular goal while taking boundaries and limits into account.

Significant milestones have occurred in the advancement of unconstrained optimization methods. Considerable developments have transformed the optimization nature, fueled by brilliant brains. Amongst them is the birth of optimization theory, which was inspired by Dantzig's invention of the simplex approach in 1947 to address linear programming problems [8]. Bellman's 1957 formulation of the optimality concept for dynamic programming problems remains a fundamental part of optimization theory [9]. Kuhn and Tucker's essential work in 1951 explaining the required and sufficient criteria for optimum solutions in programming problems established the groundwork for later research into non-linear programming (NLP) [10]. Around the early 1960s, Zoutendijk and Rosen delivered crucial contributions to the field of NLP, cementing their reputations as innovators [11] [12]. At the same time, Duffin, Peterson, and Zener created geometrical programming, while Gomory's seminal work in integer programming lifted the topic to a dynamic and competitive region [13] [14]. This ascendancy is due to the prevalence of real-world applications embedded inside this problem space. In particular, stochastic programming approaches developed by Dantzig, Charnes, Cooper, and Symons overcome challenges with expectations of independence and generally distributed design parameters [15] [16] [17].

In recent years, simulated annealing (SA), genetic algorithms (GAs), and neural network approaches like swarm intelligence algorithms (SIs) emerged into the spotlight, revealing a new world inside mathematical programming. SA, for example, mimics the annealing process in substances, whereas GAs use the theory of natural selection and genetics principles. Furthermore, neural network approaches use linked "neuron" processors

to tackle complex problems, furthering the study of optimization in computer science and its applications.

Each type of optimization poses unique challenges that must be handled with specific techniques, computational advances, and domain understanding. To produce optimal or near-optimal solutions, successful optimization algorithms take these constraints into account.

The following are some of the well-known types of optimizations that are used for optimization challenges:

1. **Continuous Optimization Challenges:** The challenges emerge in continuous optimization as a result of the complicated nature of real-world functions. The optimization landscape may have several local optima and unexpected peaks, making algorithm convergence to the global optimum problematic. It is critical to pick proper optimization methods and tactics. Furthermore, determining gradients correctly for large-scale challenges can be computationally costly, demanding the employment of gradient-free approaches. Convergence must be ensured in the presence of noisy objective functions, which introduces an additional level of difficulty.
2. **Discrete Optimization Difficulties:** Because of the combinatorial character of challenges, discrete optimization presents difficulties. As the scale of the challenge expands, so does the area of search, causing it to be impossible to examine all viable solutions. Algorithms have to depend on heuristics and approximation methods to identify near-optimal answers effectively. Considering the availability of local optima, finding globally optimum solutions might be complicated. It is critical to balance exploration and exploitation during the search process to prevent algorithms from being trapped in suboptimal areas.
3. **Multi-objective Optimization Challenges:** Because of competing objectives, multi-objective optimization poses challenges. Algorithms must manage choices between diverse aims, which leads to a Pareto front set of solutions. Correctly establishing the Pareto front necessitates excellent ways of investigating the multidimensional solution region. Choosing one approach from the Pareto front which satisfies the

needs of decision-makers can be difficult. To give a comprehensive collection of trade-off solutions, the created solutions must balance convergence with diversity.

4. **Stochastic Optimization Challenges:** Because of the unpredictable and disorganized characteristics of the optimization process, stochastic optimization faces obstacles. Algorithms must find a balance between intensification and diversification while taking randomization into account in objective judgments. As algorithms may be fooled by biased tests, maintaining convergence in the absence of noise is crucial. Addressing probabilistic limitations and noisy gradients makes it more challenging. Major issues include selecting adequate sample procedures and reacting to changing arbitrarily environments.
5. **Dynamic optimization Challenges:** The dynamic nature of the problem poses challenges for dynamic optimization. As conditions change, algorithms have to modify their strategies instantly. It is difficult to ensure accurate adaptation at a fast pace. Making wise decisions requires being able to predict how a system will change. It is a difficult task to strike a balance between exploration and exploitation while taking time-varying goals and limitations into account. Additionally, because dynamic issues call for quick adjustments, computational efficiency is crucial.
6. **Black-box Optimization Challenges:** The absence of information regarding the behavior of the objective function makes black-box optimization difficult. Algorithms have to minimize function evaluations and explore the solution space as efficiently as possible. Finding the ideal or nearly ideal solution requires the development of clever sampling techniques. It's difficult to strike a balance between exploring uncharted territory and developing areas with potential. The optimization process is made more difficult by managing noisy evaluations, unknown constraints, and the lack of derivative information.
7. **Global Optimization Challenges:** The intricacy of objective landscapes presents difficulties in global optimization. To ensure that they cover the whole solution space, algorithms must avoid becoming stuck in local optima. Finding a balance between intensification and diversification in search strategy design is essential. Due to the curse of dimensionality, finding global optima becomes more challenging as the

number of variables increases. Additional difficulties include determining trustworthy stopping criteria and evaluating the caliber of solutions found.

8. **Convex Optimization Challenges:** The need for the problem to be convex gives rise to convex optimization challenges. It can be difficult to ensure convexity, and dealing with non-convex constraints can make optimization more difficult. It becomes crucial to choose suitable algorithms that ensure convergence to global optima. It is essential to achieve balance between the complexity of the problem structure and computational efficiency. Reliable solutions also depend on resolving numerical stability and ill-conditioning concerns.

These are some of the most common optimization problems and every kind of optimization has its own set of problems that must be solved with specialized tactics, cutting-edge algorithms, and subject-matter knowledge. Successful optimization techniques take these issues into account to produce optimal or nearly optimal solutions [18].

A few of the well-known optimization challenges are as follows:

1. **Traveling Salesman Problem (TSP):** TSP is a recognized combinatorial optimization challenge with many applications. It involves figuring out the quickest path that stops at each of the specified cities exactly once before returning to the starting location. Finding an efficient way to explore every possible combination is the challenge. The TSP is addressed using techniques such as branch-and-bound algorithms, dynamic programming, and genetic algorithms. Using these techniques, the ideal tour with the shortest overall distance is sought after.
2. **The Knapsack Problem** is choosing a set of items from a given set, each with a given weight and value, in order to maximize the total value while staying within a weight limit. It's an essential optimization problem in inventory control and resource allocation. One popular method for resolving the Knapsack Problem is dynamic programming. Building a table of subproblem solutions and quickly determining the best combination are the main goals of the method.

3. Quadratic Assignment Problem (QAP): QAP is important for network design and facility placement. It entails choosing the best locations for facilities to be assigned to while taking facility interaction costs into account. Reducing overall transportation expenses is frequently the goal. Finding the assignment that minimizes the cost function is the goal of both metaheuristic techniques like Simulated Annealing and Quadratic Programming.
4. Optimal Control Problems: The objective of these problems is to determine, over time, the ideal control inputs for a dynamic system. Applications span the fields of engineering, economics, ecology, and robotics. The goals can be to maximize efficiency, keep the system stable, or consume the least amount of energy possible. A popular mathematical framework for resolving optimal control issues is called optimal control theory. This theory makes use of the calculus of variations to identify control schemes that maximize a specified performance metric.
5. Global numerical optimization: Locating the global optimum of a nonlinear objective function is the focus of this class of problems. These optimization problems frequently involve a wide parameter space and lack straightforward analytical solutions. To explore the solution space and identify the global optimum, metaheuristic algorithms like Differential Evolution, Particle Swarm Optimization, and Genetic Algorithms are frequently employed.
6. Dynamic Resource Allocation: Resource allocation in dynamic environments such as cloud computing or telecommunications needs to adjust to changing circumstances. This entails allocating resources like bandwidth, CPU power, or server instances optimally to satisfy different demands. Dynamic resource allocation problems are approached through the lenses of evolutionary algorithms, dynamic programming, and reinforcement learning. These methods let systems adjust and distribute resources effectively at the moment.

These are a few examples of optimization challenges from a variety of fields and applications, these challenges highlight the diversity of optimization problems [19] [20] [21]. A problem's properties, such as its number of variables, constraints, and particular optimization goals, often influence the optimization technique that is chosen. To successfully

address these issues, a variety of optimization techniques—from precise algorithms to heuristic approaches are used.

3. Stochastic Optimization Techniques

Over the past ten or so years, stochastic optimization algorithms have gained a lot of traction, and several of these techniques are now "industry standard" techniques for handling difficult optimization problems. This chapter lists several well-known stochastic optimization algorithms and gives a summary of the differences between deterministic and stochastic optimization techniques.

3.1. Preliminaries and Basics

In deterministic optimization, all the parameters, constraints, and objectives are known with certainty. This type of optimization falls under mathematical optimization. That is to say, the optimization problem is not characterized by any randomness or uncertainty.

Identifying the optimal solution from a range of acceptable choices often the global optimum is the main objective of deterministic optimization. Diverse domains include engineering, economics, logistics, and operations research use deterministic optimization problems extensively. Linear programming, nonlinear programming, integer programming, network optimization, and other forms of problem-solving can be applied to these situations. The purpose of deterministic optimization is to identify the optimal values for decision parameters that meet all constraints while either minimizing or maximizing an objective function.

On the other hand, stochastic optimization addresses optimization issues containing probabilistic, random, or uncertain components. Random variations may affect one or more parameters, constraints, or the objective function itself in these problems. The objective of stochastic optimization is to determine solutions that work well under various probability distributions or scenarios.

In domains such as finance, environmental planning, energy management, and machine learning, stochastic optimization problems are frequently encountered. Supply chain optimization with unpredictable demand, machine learning parameter tuning, and portfolio optimization in finance are a few examples. The goal of stochastic optimization techniques is to produce solutions that are strong and resistant to different sources of uncertainty while accounting for the problem's probabilistic nature.

Numerous approaches fall under the category of stochastic optimization techniques, such as dynamic programming, stochastic programming, Monte Carlo simulation, and metaheuristic algorithms like simulated annealing and genetic algorithms. These techniques enable decision-makers to take well-informed decisions even in dynamic, uncertain, and complex decision-making environments.

The primary distinction between stochastic and deterministic optimization is that the former is utilized for problems with randomness or uncertainty, while the latter is applied when all problem parameters are known with certainty. The decision between the two relies on the type of problem and the degree of uncertainty involved. Both have useful applications in solving real-world challenges across a variety of domains.

The following

Figure 1 Optimization Methods shows some of the optimization methods that stochastic and deterministic optimization have.

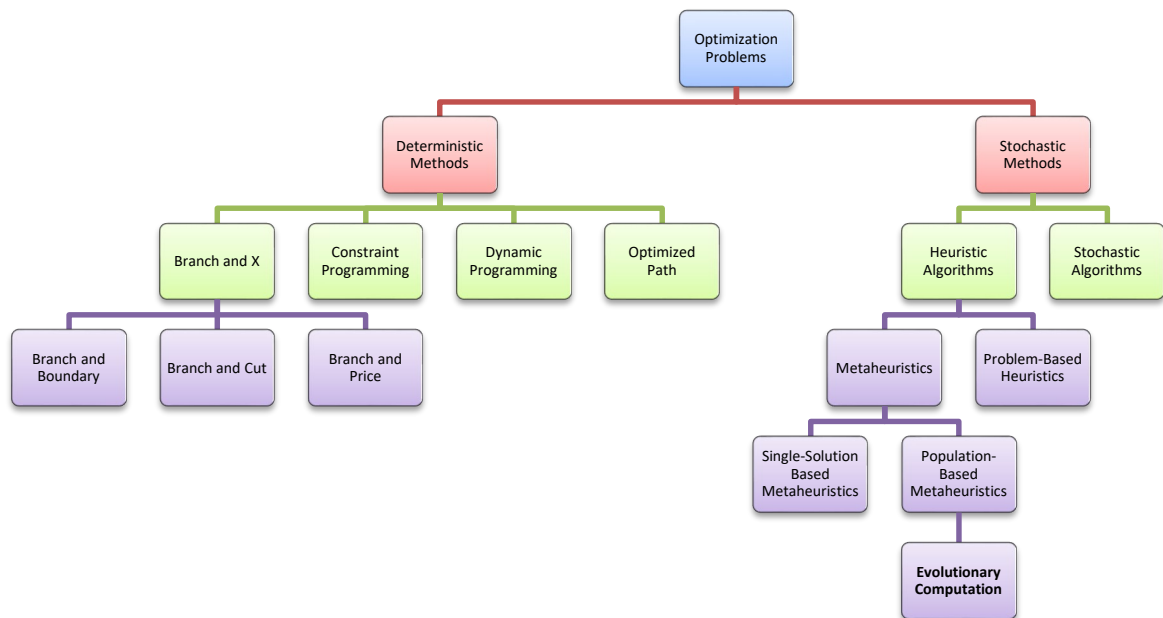


Figure 1 Optimization Methods.

The following Table 1 summarizes the key variations between deterministic and stochastic optimization:

Table 1 variations between deterministic and stochastic optimization.

Aspect	Deterministic Optimization	Stochastic Optimization
Nature of Problem	All parameters are known with certainty.	At least one parameter, constraint, or objective is subject to randomness, uncertainty, or probability distributions.
Objective	Find the best solution, often the global optimum.	Find solutions that perform well under different scenarios or probability distributions.
Constraints	All constraints are deterministic and fixed.	Constraints may involve random variables or uncertain parameters.
Applicability	Commonly used in problems with known data, such as linear programming, integer programming, and network optimization.	Applied to problems with randomness or uncertainty, including finance, machine learning, and environmental planning.
Methods	Typically employs mathematical programming methods (linear programming, nonlinear programming, etc.).	Involves methods such as Monte Carlo simulation, dynamic programming, stochastic programming, and metaheuristic algorithms.
Robustness	Deterministic optimization focuses on finding a single optimal solution.	Stochastic optimization aims to provide robust solutions that can withstand variability and uncertainty.

Examples	Production planning, engineering design, network routing.	Portfolio optimization, machine learning model tuning, supply chain management with uncertain demand.
Decision-making under Risk	May not adequately address risk and uncertainty.	Addresses risk and uncertainty, allowing for better-informed decisions.

3.2. Taxonomy of Stochastic Optimization Methods

A branch of stochastic approaches is known as evolutionary computation, or EC for short. The evolution of EC is one aspect of population-based algorithms, which are responsible for maintaining a group of potential solutions (individuals) and progressing towards good or optimum solutions. To obtain a superior degree of global search capability, EC approaches are widely recognized for their capacity to simultaneously generate many solutions.

The following

Figure 1

Figure 2 highlights categories of evolutionary computation:

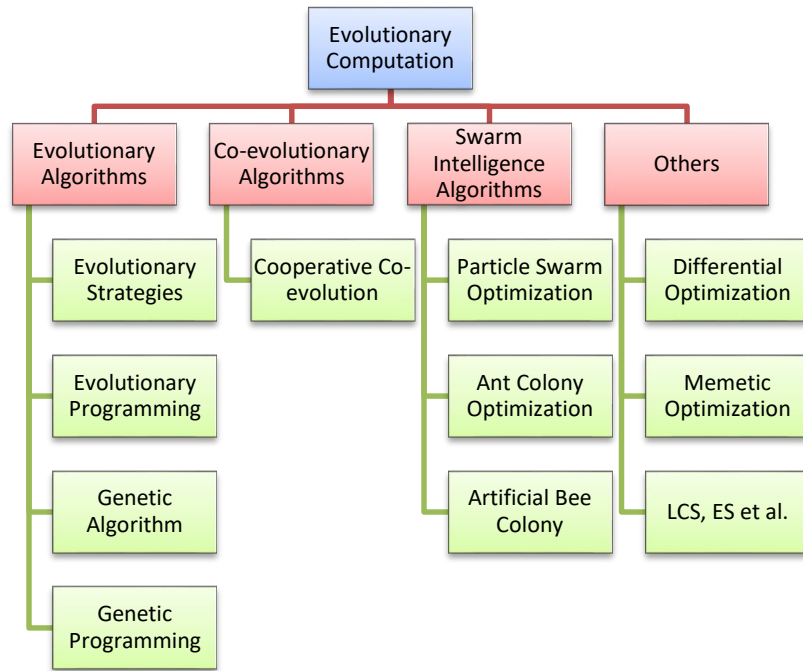


Figure 2 Categories of evolutionary computation.

In general, EC approaches may be classified into multiple categories but the most famous ones are evolutionary algorithms and swarm intelligence [22].

The concepts of Darwinian natural selection are used by evolutionary algorithms in order to efficiently look for optimum solutions. Among the algorithms that fall under this class, genetic algorithms (GAs) and genetic programming (GP) are among the most popular choices. When it comes to the evolution of new individuals, both approaches employ genetic operators like crossover and mutation. In contrast to GAs, which employ a bit string representation of a fixed length, GP is able to operate with structures that are more flexible, for instance trees and graphs with larger or smaller dimensions.

Swarm intelligence approaches, conversely, make use of the social behaviours of animals as their source of inspiration. For example, particle swarm optimization (PSO) and ant colony optimization (ACO), both of which are approaches that resemble birds and ants, respectively, are examples of techniques that fall under this class. ACO operates by replicating a mechanism for communicating between ants that is based on pheromones in order to locate favourable routes to food, whereas PSO makes use of information regarding

the best-found solutions that are transmitted across particles in order to direct search efforts towards more rewarding locations. Aside from artificial immune systems (AIS), learning classifier systems (LCS) and differential evolution (DE), there are additional well-known EC algorithms [22].

EC techniques have demonstrated considerable potential in addressing multi-objective problems involving the simultaneous evolution of multiple solutions and the consideration of optimal solutions despite the presence of multiple conflicting objectives (such as minimizing travel time and cost in flight reservations). A multi-objective technique provides a collection of nondominated (Pareto optimum) solutions that can not be enhanced in a particular objective without causing another objective to be negatively impacted [23]. This is due to the fact that it is very rare that an optimal solution would be found that fulfils both of the competing objectives. Perhaps amongst the most extensively researched EC subjects in recent times is evolutionary multi-objective optimization (EMO), which has seen a huge rise in the number of papers during the course of the past ten years.

Several studies have been conducted on the application of EC for machine learning tasks (EML); For instance, one examined feature selection [24]; another examined classification using GP [25], [26] and [27] discuss technical orientations such as EC and ML or EMO, respectively [23].

4. Swarm Intelligence Metaheuristics

Insect swarms possess remarkable characteristics. They possess the ability to consistently discover resolutions for exceedingly intricate dilemmas, as shown in Figure 3 or the determination of a colony's structure, despite the absence of anybody with the expertise to construct it. Every individual adheres to a prescribed set of uncomplicated guidelines, and there is no centralized authority over the collective group, rendering them highly adaptable and resistant. Insect flocks possess certain characteristics that make them crucial for constructing mathematical models to address certain challenges in computer science [28] [29]. By collaborating with engineers, computer scientists have successfully converted models of collective behavior in insect colonies into algorithms to get the optimal answer among several alternatives.

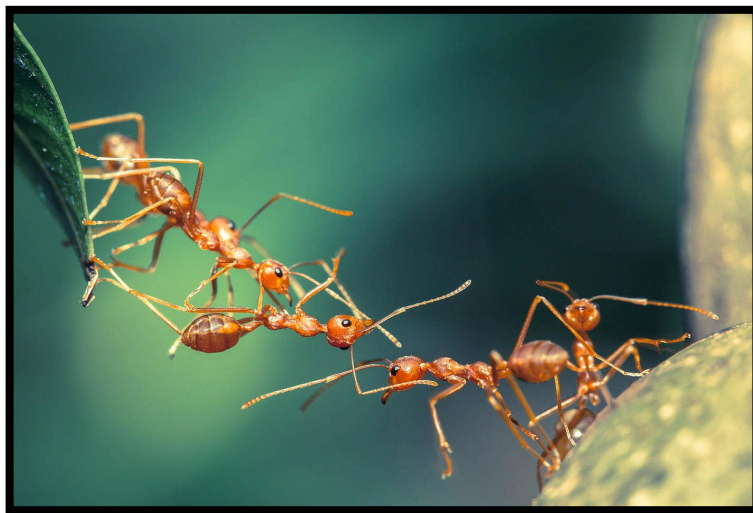


Figure 3 Bridge made by Red Ants

The term Swarm Intelligence was invented by (Gerardo Beni) and (Jing Wang) in 1989 during a seminar he presented in Japan on the robotics community [30] [31]. The definition says: Secret Intelligence is a feature that describes systems consisting of robots with limited (unintelligent) cognitive and intellectual skills that collectively emphasize - in an "unexpected" way - intelligent actions by generating a coordinated pattern of matter. Robot in this sense denotes any unit capable of delivering mechanical or computational labor. The essential term in this definition is "unexpected," since it pertains to instances where the outcome of a system cannot be foreseen, hence indicating the presence of intelligent behavior.

In 1990, Bonabeau, Dorigo, and Theraulaz put up a more encompassing definition of secret intelligence as the pursuit of designing algorithms to address distributed issues (known as Distributed Problem Solving), drawing inspiration from the cooperative behavior in colonies. Entomological and zoological societies. Such as the Ant Colony Optimization (ACO) algorithm which emulates the behavior of ant colonies [28].

In 1995, scientists Russell Eberhart and James Kennedy proposed the "Particle Swarm Optimization (PSO)" method, which essentially relies on mimicking the behavior of bird flocks [32].

The preceding PSO and ACO algorithms marked the inception of Swarm Intelligence (SI), a novel field of evolutionary computation.

4.1. Features of Swarm Intelligence Systems

Algorithms have long played a vital role in calculations [33], but modern algorithms typically incorporate both deterministic and stochastic elements. Swarm intelligence traits are utilized as stochastic elements in nearly all nature-inspired algorithms [34]. Because of the complex nature of swarm intelligence-based algorithms, there are several approaches to evaluate and break down their core components. These methods take into account the algorithms' roots of inspiration in nature and their formulations.

From the standpoint of group behavior systems, algorithms may be examined by considering their primary characteristics and attributes. This is a more advanced analysis that is not reliant on precise algorithmic steps or mathematical formulae. It allows one to depend on the functions of the algorithm and the fundamental behavior of the search process.

The key tenets of collective behavior in bird flocks, as seen in **Error! Reference source not found.**, can be outlined as follows:



Figure 4 Birds Flocks

- Heterogeneity refers to the phenomenon in which every individual bird in a flock exhibits the same behavioral pattern, and there is no leader within the flock.
- The movement of any individual bird within the flock is only affected by the birds that are close to it, and vision is the most important sense for the purpose of arranging the flock efficiently.
- It is important to avoid colliding with a nearby bird in order to avoid collisions.
- In the flock, velocity matching refers to the process of accelerating in a manner that is parallel to the acceleration of surrounding birds.
- The concept of flock centering requires every individual bird to maintain close proximity to its neighbors.

On the basis of the behavior of the flocks of birds, we are able to draw the following conclusions:

- The absence of a supervisor is a characteristic of decentralized control, which means that there is no centralized entity that instructs the squadron on what actions to take.

- Flexibility is a characteristic of the swarm system, which is a dynamic system that undergoes changes and self-regulation in response to the environment that surrounds it, and it also adapts to these changes. Communication with the members of the swarm takes place on a local level.
- Emergent behavior is a phenomenon that arises as a consequence of the individual actions and behaviors of every single person.
- Robustness refers to the ability to complete a task even in the event that one or more individuals fail to do so. The decision-making process does not rely on intricate evidence-gathering tools; it is made up of straightforward humans. Each individual is in possession of a limited quantity of information regarding the system and the many ways in which problems might be solved.
- Stability is a crucial attribute that is needed, as the swarm is incapable of reaching any destination if it lacks this particular quality. It refers to the act of maintaining the swarm in place while it is moving.
- When it comes to self-organization, scientists formerly held the belief that the complexities observed in patterns generated by swarms were a consequence of behavioral and cognitive complexities at the individual level. The perspective, on the other hand, shifted as a result of the implementation of the theory of self-organization, which is used in the engineering and chemical sciences. As a result, it became feasible to assert that the general pattern is comprised of the interaction of individuals who possess simple abilities. This phrase is now utilized in a diverse array of professions, consisting of but not exclusive to computer science, mathematics, biology, and even psychology.

4.2. Basic Principles of Swarm Algorithms

There are two components that are necessary for every metaheuristic algorithm, and they are exploration and exploitation. The search cycle is improved through the utilization of local knowledge, which is referred to as exploitation. Such local knowledge might be gained from the goal or changes in the cost landscape. Both of these ideas are possible. The process of optimization will be sped up by increased exploration; nevertheless, this may result in premature convergence, which often leads to a local optimum or an inaccurate response. Additionally, the probability of uncovering the supreme global best will be diminished. Exploration with the purpose of thoroughly exploring the search region and

contributing to the discovery of new solutions was carried out. The process of discovery of the absolute global best is often slowed down which results in a substantially lower convergence rate when increased exploitation is carried out. However, the chances of discovering the absolute best are increased. The consequence of this is that there is a fine line that separates exploration from exploitation.

During the course of the search, we are required to make use of a suitable process to pick the best approaches. The process that is used the most frequently is called "the fittest survive," which indicates that the idea that has not yet been discovered is continuously improved. Furthermore, other forms of elitism are routinely applied in order to ensure that the most satisfactory ideas are not abandoned and are instead passed down to subsequent generations. Therefore, in order for an algorithm to be successful, it must give a way that properly balances the two primary components that were mentioned earlier. There is a need to acknowledge that the naïve 50/50 balance is not an optimal situation.

4.3. Representatives of Swarm Algorithms

Within the domain of contemporary computational algorithms, a notable classification is inferential algorithms that derive inspiration from the natural world. These algorithms belong to the category of random algorithms, which includes computer techniques focused on iteratively improving problem-solving by creating new solutions and iteratively strengthening existing ones. The appeal of these algorithms rests in their ability to reach optimal solutions efficiently, even for intricate optimization problems.

These algorithms are characterized by their combination of two essential elements: intensification (exploitation) and diversification (exploration). The concentration factor emphasizes a targeted exploration within a certain area, utilizing the local environment to uncover possible solutions. Conversely, diversification entails the process of investigating and exploring various options inside the solution space to discover innovative ideas, aiming to ultimately choose the most suitable solution.

Hard-NP (Non-deterministic Polynomial time) issues are a category of optimization problems that have attracted significant interest, mostly because of their inherent complexity. Some examples of these tasks include the N-Queens problem, the Travelling Salesman

problem, and several others. Several algorithms have been constructed to tackle these challenges, and Swarm Intelligence (SI) Algorithms have emerged as a particularly notable solution.

Swarm Intelligence Algorithms consist of a range of metaheuristic algorithms, each gaining influence from the cooperative actions seen in natural swarms. Notable examples in this area comprise Ant Colony Optimization, Particle Swarm Optimization, Firefly Algorithm, Artificial Bee Colony, Cuckoo Search and many other algorithms.

Furthermore, the following is a concise summary of several of these algorithms, with a particular emphasis on their fundamental principles rooted in Swarm Intelligence.

4.3.1. Particle Swarm Optimization (PSO)

Particle Swarm Optimization is a computational technique that seeks to minimize a challenge by repeatedly attempting to enhance a potential solution based on a specified threshold for outstanding performance [32].

The development of PSO was undertaken by Kennedy and Eberhart in 1995. PSO is an algorithm that solves a challenge by utilizing a group of potential solutions, referred to as particles. These particles are manipulated within the field of search using straightforward mathematical equations that update their positions and velocities. The path of each particle is controlled by its current optimal location, while it is also directed towards the global optimal locations throughout the search field, which are modified when other particles discover more favorable locations. This is anticipated to propel the swarm toward the optimal options.

The PSO algorithm begins by initializing a set of random particles (solutions) and proceeds to search for optimal solutions by iteratively adjusting the population. During each repetition, every particle is updated by considering two optimal values. The first solution has shown the highest performance efficiency. The fitness value is recorded as well. The term used to refer to this value is "*pbest*". The particle swarm algorithm also keeps note of the best value achieved by any particle in the group, which is considered the optimal value. The term "*gbest*" refers to the global best value, which is considered the most optimal value

overall. When a particle selects a subset of the entire population as its topological neighbors, the optimal value inside this subset is referred to as the local best and is denoted as "lbest".

4.3.2. Ant Colony Optimization (ACO)

Ant Colony Optimization is a stochastic method used to solve challenges that involve finding optimal directions in graphs. This algorithm is a part of swarm intelligence approaches. It is used for metaheuristic optimizations [35].

The initial method, developed by Marco Dorigo in 1992 in his Doctorate thesis, aimed to find the optimum route in a graph. It was inspired by the movements of ants as they looked for a route to link their colony and food sources. The initial concept has been expanded to address a broader range of numerical issues, leading to the emergence of multiple challenges that incorporate different elements of ant behavior.

In the natural environment, ants exhibit a behavior of random exploration, and upon discovering food, they navigate back to their colony by depositing pheromone trails. This idiosyncratic behavior serves as the primary source of inspiration for the ACO algorithm. Upon encountering these trails infused with pheromones, other ants tend to forsake aimless investigation and, rather, prefer to stick to the indicated trail. Afterwards, if these subordinate ants similarly locate food, they strengthen the pheromone trail, establishing a self-reinforcing process that increases the probability of successive ants identifying and following the most efficient route.

ACO has been applicable beyond its original scope of graph-based challenges. It is now utilized in several numerical problem-solving fields. The adaptability of ACO rests in its capacity to utilize different aspects of ant behavior to tackle intricate optimization issues. This encompasses situations where the most effective solution entails complex routes or networks, similar to the foraging trails created by ants. ACO's ability to replicate the combined intelligence of ant colonies has elevated it to a prominent position among metaheuristic algorithms, rendering it a significant asset for academics as well as practitioners in the field of optimization.

4.3.3. Artificial Bee Colony (ABC)

The Artificial Bee Colony is an optimization algorithm that was developed by Karaboga in 2005 [36]. It is inspired by the complex foraging activity displayed by honey bees. The algorithm is based on a complex model that closely resembles the foraging behaviors of honey bee populations. This model includes both employed and unemployed honey bees, as well as different sources of food. Essentially, ABC classifies bees in a colony into three crucial roles: worker bees, observer bees, and explorer bees.

The working bees are tightly connected to a specific food supply that they are constantly utilizing. The bees transport vital information about the location back to the hive, and they communicate with other bees through a dance-like behavior to impart particular information. Observer bees, stationed at the hive, watch and analyze the dances of worker bees. They utilize the knowledge obtained from these dances to make judgments about food sources, which might be seen as a fascinating type of "exploitation" in the algorithm.

In addition to these responsibilities, explorer bees are crucial in this process of "exploration". Explorer bees have the responsibility of finding novel sources of food, imitating the innate behavior of bees to investigate their environment for potential sources of food. The ABC algorithm's fundamental dynamics are based on the organized distribution of tasks among worker, observer, and explorer bees.

The adaptability of Karaboga's ABC method is demonstrated by its ability to tackle optimization issues in both continuous and discrete domains. Although ABC has demonstrated effectiveness in addressing various issues, it is essential to take into account the specific attributes of the present challenge when determining if it is the most suitable option. The algorithm's distinctive methodology, derived from the complicated behavior of honey bees, introduces a biological aspect to computational problem-solving, rendering it an intriguing field for research and practical implementation in optimization problems.

4.3.4. Bat Algorithm (BA)

The Bat Algorithm introduced by Yang in 2010, is an advanced method of optimization that takes inspiration from the complex echolocation ability of microbats in

nature. This technique, inspired by biology, adds a group of simulated bats inside the computing field. Every bat represents a possible solution in the vast problem-solving space [37].

The Bat Algorithm is based on the fundamentals of frequency-tuned dynamics. It optimizes by altering two important parameters: the rates at which pulses are emitted and a measure of loudness. During its initialization, the algorithm establishes the framework by randomly assigning numbers in predefined regions—*loudness min*, *loudness max*, and *rate min*, *rate max*—to these crucial parameters. This imparts every bat with a distinct combination of traits that determine its methods for intensification and diversification.

During each cycle, the bats adapt their speed and location in accordance with the changing environment of possible solutions. The intricate interaction is captured in the sophisticated equations:

$$v[i][j] = v[i][j] + (x[i][j] - gbest[j]) \times f[i]$$

$$x[i][j] = x[i][j] + v[i][j]$$

Equation 1 Update rules used in the Bat Algorithm.

The characters $v[i][j]$ represent the speed of the i -th bat at dimension $x[i]$. The symbol $[j]$ indicates the location of the i -th bat at dimension $gbest[j]$. This reflects the current global optimal solution at dimension j . The variable $f[i]$ indicates the frequency of the i -th bat.

The Bat Algorithm is remarkable not only for its mathematical elegance but also for its ability to imitate the natural sonar strategies used by microbats. Microbats in their environment employ echolocation as a means of orienting themselves in darkness. They produce pulses and interpret the resulting echoes to discern food, evade obstacles, and find places to rest. When applied in the field of computation, the Bat Algorithm proves to be a very effective and efficient technique influenced by biological processes. It showcases its capacity to adjust and optimize various environments.

4.3.5. Firefly Algorithm

The Firefly Algorithm was created by Xin-She Yang in 2007, taking inspiration from the luminous habits and patterns of fireflies. FA is a metaheuristic optimization technique that emulates the social behaviours of fireflies to address optimization challenges [38].

FA is based on three fundamental principles that dictate the actions of fireflies:

- Fireflies exhibit equal interest, regardless of gender. This principle exemplifies the monosexual characteristic of fireflies.
- Light-Dependent Attractiveness: In the presence of two fireflies emitting light, the firefly with a lower level of illumination will exhibit movement in the direction of the firefly with a higher level of illumination. When a firefly is more luminous than its surrounding fireflies, it will remain stationary; alternatively, it will travel in a random manner which will incorporate an aspect of diversification.

The fundamental principle of the Firefly Algorithm is based on a set of principles that collaboratively direct the fireflies to navigate the area of interest and discover the most effective solutions for optimization issues. The method effectively manages the trade-off between exploring new possibilities and exploiting known information, making it well-suited for a wide range of optimization problems.

The Firefly Algorithm has been applied in various domains, including as engineering, economics, and biology, demonstrating its adaptability and efficacy in resolving intricate issues.

4.3.6. Cuckoo Search(CS)

The Cuckoo Search algorithm was developed by Yang and Deb in 2009 as an optimization method. CS is derived from the habits of certain cuckoo organisms, specifically those that engage in egg extortion [39]. Cuckoos are renowned for their distinctive breeding method, in which certain species deposit their eggs in communal nests, such as those of Ani and Guira cuckoos. This act is the deliberate destruction of other eggs in the nest in order to

enhance the chances of safely hatching their eggs. Within the framework of optimization, every cuckoo egg symbolizes a prospective solution, while each nest denotes a solution area.

The core concept of Cuckoo Search is to use novel and presumably superior solutions (symbolized by cuckoos) in order to supplant suboptimal solutions in nests. At its fundamental state, every nest has a solitary egg, representing a resolution. Nevertheless, the concept can be expanded to encompass more intricate cases in which every nest can house multiple eggs, hence signifying a multitude of possible solutions.

The process of optimization in computer science entails the following fundamental stages: Cuckoos produce novel solutions, which are subsequently assessed. If a novel solution surpasses the one currently present, it supplants the preexisting answer. This replicates the procedure of natural selection, in which only the most adapted solutions are preserved. The objective is to progressively enhance solutions across generations, resulting in the identification of optimum or nearly optimum solutions to sophisticated challenges.

The Cuckoo Search algorithm has been utilized in diverse domains such as engineering, finance, and data science. The efficiency of its exploration of solution areas allows it to be well-suited for addressing challenges that have several optimal solutions and intricate search landscapes. CS is highly regarded by researchers and practitioners for its straightforwardness, adaptability, and efficiency in tackling optimization difficulties in various fields.

4.3.7. Whale Optimization Algorithm (WOA)

The Whale Optimization Algorithm was proposed by Mirjalili and Andrew in 2016. This algorithm is derived from the feeding strategy employed by humpback whales [40]. WOA is classified as a swarm-based intelligence algorithm that has been developed specifically for handling challenges related to continuous optimization. Despite certain alternative optimization algorithms, the WOA stands out for its limited number of adjustable parameters. The optimization procedure mostly revolves around adjusting a solitary parameter, known as the time interval.

Within the framework of the WOA, the collective population of humpback whales engages in the systematic exploration of a search area that encompasses several dimensions. In this context, the locations of the whales serve as representations of decision variables. The aim of the function's value might be likened to the proximity of whales and their desired food source, and the algorithm's goal is to reduce this number.

The procedure for optimization in the WOA is delineated by three key phases that are associated with the foraging habits of humpback whales:

- Whales employ a strategy called "Shrinking Surrounding Hunt" to optimize their exploration of the neighboring search environment by dynamically adjusting their placements.
- Seeking Hunt (diversification) Whales engage in the act of searching for food sources by systematically examining a wide range of potential areas.
- The Bubble-Net Attacking Approach (intensification) is the phase where the identified areas are utilized to achieve the best possible solutions.

WOA stands out for its bio-inspired methodology, which involves converting the intricate and collaborative feeding habits of humpback whales into a mathematical algorithm. The algorithm's efficacy stems from its capacity to achieve a harmonious equilibrium between intensification and diversification, enabling it to navigate intricate terrains with utmost efficiency.

WOA has diverse applications across multiple areas, such as engineering design, machine learning, and optimization challenges in science and industry. WOA is highly valued by academics for its straightforwardness, accessibility in application, and superior results when viewed alongside alternative optimization techniques. The ability to adapt WOA to many varieties of issues and its efficacy in identifying optimal solutions render it a significant asset among the optimization algorithms

5. Swarm Intelligence Applications

In this section, the experiments will take place to overcome some of the most popular machine learning challenges, which are (Enhancing Internet of Things Network Security, Oil Price Forecasting, Seizure Detection, Wind Energy Forecasting, spam email classification, and Spot instance price forecasting). Also, an enhanced version of the originally proposed algorithms will be used. These experiments are as follows: -

1. Enhancing Internet of Things Network Security Using Hybrid CNN and XGBoost Model Tuned via Modified Reptile Search Algorithm.
2. The Long Short-Term Memory Tuning for Multi-Step Ahead Wind Energy Forecasting using enhanced Sine Cosine Algorithm and Variation Mode Decomposition
3. Training logistic regression model by moth flame optimizer for spam email classification
4. Seizure Detection Via Time Series Classification Using Modified Metaheuristic Optimized Recurrent Networks
5. Long Short-Term Memory Tuning by Enhanced Harris Hawks Optimization Algorithm for Crude Oil Price Forecasting
6. Exploring the Potential of Modified Metaheuristic Optimized Long-Short-Term Memory Neural Networks for Earthquake Magnitude Forecasting
7. Cloud Spot Instance Price Forecasting Multi-Headed Models Tuned Using Modified PSO

5.1. The Experiments Environment

In the following two conducted experiments, the cloud computing environments were used. For more clarity, the cloud machine learning services were used for the experiments, specifically the Amazon SageMaker, which is known to be the most extensive machine learning platform for developers and data analyzers. Moreover, SageMaker key features and the reasons behind using this MlaaS platform are as follows: -

- Automatically construct, train and tuning models.
- Automatic workflow for ML.

- The quickest and most straightforward approach to arrange data for machine learning.
- Protect your source code.
- Training quicker high-end models.
- Cloud deployment with a single click.
- Increase model quality on edge equipment.
- Increase efficiency with the first fully Machine learning IDE.
- High understanding of the predictions and error detection.
- By more than 70% decrease the expenses of data labelling.

5.2. Application 1: Enhancing Internet of Things Network Security Using Hybrid CNN and XGBoost Model Tuned via Modified Reptile Search Algorithm

The Internet of Things (IoT) has ushered in a new era of connectivity, transforming various industries by enabling faster sensor and data access. This enhanced networking capability has been pivotal in facilitating real-time monitoring, which is indispensable for further process optimization across sectors. Real-time data acquisition has significantly improved healthcare by enabling timely patient monitoring and emergency notification systems [41]. In healthcare, IoT devices range from blood pressure and heart rate monitors to advanced devices capable of monitoring specialized implants. The Internet of Medical Things (IoMT) has even led to the creation of automated systems used to analyze health statistics [42]. The industrial application of the IoT includes asset management, predictive maintenance, and manufacturing process control. Overall, the IoT's integration into various domains is revolutionizing the way we live and work, providing more efficient, cost-effective, and intelligent solutions [43].

In the manufacturing sector, the IoT has streamlined operations, ensuring efficient production and quality control. The integration of IoT devices has revolutionized the way machines and systems communicate and interact with each other, forming the backbone of smart factories. Through IoT, manufacturers can monitor and control their production processes in real-time, with sensors embedded in machines, equipment, and products collecting data on various parameters such as temperature, pressure, vibration, and

performance metrics. This data can be transmitted and analyzed instantly, providing valuable insights into operations. IoT also helps manufacturers achieve greater visibility and transparency across the entire supply chain, optimizing inventory management, reducing stockouts, and improving order fulfillment. Technological advancements continue to benefit manufacturers by opening new revenue streams, improving industrial safety, and reducing operational costs, reshaping the way manufacturers operate in the digital revolution [44].

The widespread adoption of the IoT is not without challenges. Devices often grapple with limited battery lifetimes, the need to function in remote locations, and demanding transceiver operations. Among these challenges, security stands out as the most daunting. While a device running out of battery is an observable setback, a data breach, often clandestine, can wreak havoc. IoT devices must communicate with each other and central systems, requiring complex transceiver operations. This can lead to high energy consumption and potential interference with other devices, complicating the network [45].

Security is indeed one of the most critical challenges in the IoT ecosystem. The interconnected nature of these devices means that a breach in one device can potentially compromise an entire network. Issues such as weak authentication, lack of encryption, and insecure interfaces can lead to unauthorized access and data theft [46]. Radio Frequency (RF) attacks have become a prevalent attack vector within the IoT ecosystem. Attackers can exploit vulnerabilities in wireless communication protocols to intercept, modify, or disrupt the RF signals between devices. This can lead to data leakage, device malfunction, or even taking control of the devices [47]. Message Queuing Telemetry Transport (MQTT) is one of the standard application layer protocols emerging in the IoT ecosystem [48]. As an emerging technology, it is a popular target for malicious actors seeking new vulnerabilities [49].

To counter these threats, several network security measures, such as blocklists and firewalls, have been implemented [50] [51]. However, artificial intelligence (AI) algorithms aim to address security challenges without the constraints of predefined rules or continuous manual intervention [52]. A pivotal factor influencing AI's performance is the judicious selection of hyperparameters that steer the algorithm. With the burgeoning complexity of emerging algorithms, the conventional trial-and-error approach for hyperparameter tuning is becoming increasingly untenable. This optimization challenge can be equated to NP-hard problems, which are notoriously difficult to resolve using discrete methods [53]. A potential

respite from this optimization conundrum lies in metaheuristic algorithms. While not always possible to pinpoint the absolute optimal solution, their iterative nature enhances the probability of identifying a near-optimal solution. Often, in practical scenarios, a "good enough" solution is more valuable than an elusive perfect one. Additionally, it is important to explore and address emerging challenges that accompany developments in the field.

In the scope of this research, a refined methodology designed to confront the security challenges inherent in the IoT is introduced. The introduced approach incorporates convolutional neural networks (CNNs) to effectively manage feature sizes within IoT MQTT dataset and employs extreme gradient boosting (XGBoost) for intrusion identification and detection. A distinctive aspect of the introduced methodology lies in the integration of a modified version of a well-established algorithm, specifically tailored for hyperparameter optimization in the unique context of IoT security [54]. This integration represents a notable combination of concepts and techniques, embodying both typical combination novelty and incremental novelty. This research was published in [1].

5.2.1. Methods

This section delves into the original reptile search algorithm (RSA) [55] and a proposed modified version. The objective is to elucidate the mathematical foundations, operational mechanics, and strategies that underpin these algorithms. Moreover, a critical evaluation of their strengths, weaknesses, and potential for further development is presented. The section aims to provide the reader with a deep understanding of the algorithms, thereby setting the stage for the experimental results and discussions that follow.

5.2.1.1. Original RSA

The RSA, like many optimization metaheuristics, employs global as well as local search to effectively locate promising areas within the search space. This algorithm draws inspiration from nature, mathematically modeling the hunting strategies of crocodiles. As it is a gradient-free population-based method, it can effectively tackle complex challenges.

During the initialization process of the RSA, a population of agents is generated based on stochastic techniques. The population is then evaluated and the best solution is considered near-optimal. The population P can be represented as:

$$\mathbf{P} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,n} \end{bmatrix}$$

Equation 2 Population of agents.

where N denotes P size, n the dimensionality of a given challenge, and x is a potential solution. The set population is generated in accordance with:

$$x_{i,j} = rand * (B_{lower} - B_{upper}) + B_{lower}, \quad j = 1, 2, \dots, n$$

Equation 3 Generating Population.

where B_{lower} and B_{upper} define the lower and upper bounds of the search space, and $rand$ is an arbitrary value.

Once a population is established, the algorithm can proceed with optimization. The utilized strategy is highly dependent on the number of remaining iterations of the optimization. For the exploration mechanism, two behaviors are distinctly simulated. The first one simulates crocodile high walking. The second strategy simulates belly walking. These are mathematically modeled as:

$$x_{i,j} = \begin{cases} B_j(t) \cdot -(\eta_{i,j}(t)) \cdot \beta - R_{i,j}(t) \cdot rand, & t \leq \frac{T}{4} \\ B_j(t) \cdot x_{r1,j} \cdot EX(t) \cdot rand, & t \leq 2\frac{T}{4} \text{ and } t > \frac{T}{4} \end{cases}$$

Equation 4 Exploration Mechanism.

where, in this context, $B_j(t)$ symbolizes the j -th component of a given best candidate. Randomness is introduced using the $rand$ value selected from $[0, 1]$. The iterations are tracked using t and T , which denote the current and maximum iterations, and sensitivity is defined by β . Specialized values η , R , and ES are defined in accordance with the following:

$$\eta_{i,j} = B_j(t) - PD_{i,j}$$

$$R_{i,j} = \frac{B_j(t) - x_{r2,j}}{B_j(t) + \epsilon}$$

$$ES(t) = 2 \cdot r3 \cdot \left(1 - \frac{1}{T}\right)$$

Equation 5 Sensitivity.

where the parameter η defines the hunter operator. The role of R is to reduce the search space, while ES defines the evolutionary sense. Random values are denoted as $r2$ and $r3$, and PD describes the percentage difference between the current and best solution. A small value is also added by ϵ to avoid math errors.

Exploitation similarly employs two distinct hunting strategies, hunting cooperation and coordination. Which technique is utilized is highly dependent on the number of remaining iterations in the optimization.

$$x_{i,j} = \begin{cases} B_j(t) \cdot PD_{i,j}(t) \cdot rand, & t \leq 3\frac{T}{4} \text{ and } t > 2\frac{T}{4} \\ B_j(t) \cdot \eta_{i,j} \cdot \epsilon - R_{i,j} \cdot rand, & t \leq T \text{ and } t > 3\frac{T}{4} \end{cases}$$

Equation 6 Exploitation with two Distinct Hunting Strategies.

where, in this context, $B_j(t)$ symbolizes the j -th component of a given best candidate. Randomness is introduced using the $rand$ value selected from $[0, 1]$. the iterations are tracked using t and T , which denote the current and maximum iterations, and sensitivity is defined by β . Specialized values η , R , and ES are defined previously.

5.2.1.2. Modified RSA

While the original RSA demonstrates admirable performance, certain deficiencies can be observed using evaluations using standard CEC [56] evaluation functions. As a relatively novel algorithm, the RSA has significant potential for improvement through hybridization. This work attempts to address some of the observed issues associated with the

original algorithm through hybridization. Mechanisms inspired by the genetic algorithm (GA) [57] are introduced to formulate the genetically inspired RSA (GIRSA).

The introduction is activated following each iteration. An arbitrary agent is selected and spliced by the best attained solution resulting in a combined solution. The parameters are uniformly combined. The crossover is governed by the control parameter pc . Empirically, the optimal value for this parameter has been determined to be $pc = 0.1$.

An additional modification incorporates parameter mutation. Once a mutation is triggered, an arbitrary value is selected from a given parameter constraint. One-half of the selected value is subtracted or added to said parameter. The decision of whether addition or subtraction is used is determined by the mutation direction md parameter. Once again, the value of md is empirically determined as $md = 0.1$.

Once a new solution is generated, the worst-performing solution is removed from the solution and replaced by the new agent. The performance of the new solution is not conducted until the subsequent iteration of the algorithm. Taking this approach maintains the computational complexity of the original algorithm. Finally, the pseudocode for the described algorithm is presented in Algorithm 1.

Algorithm 1 Pseudocode of the introduced GIRSA.

```
Formulate a population  $P$ 
while  $t < T$  do
    Evaluate  $P$  based on objective function
    Update  $ES$  value
    for Solution  $X$  in  $P$  do
        . Update agent locations applying RSA search
        . Generate new solution  $NS$  utilizing genitally inspired mechanism
        . Mutate  $NS$  parameters
        . Replace worst solution with  $NS$ 
    end for
end while
return Best attained solution in  $P$ 
```

It should be emphasized that once a new agent is generated and mutated, it is not evaluated until the next iteration of the optimization. This way the complexity of the metaheuristic remains consistent with the original version of the algorithm.

5.2.2. Experimental Setup

To evaluate the potential of the introduced approach for both detecting and identifying intrusion within IoT networks, a public real-world IoT MQTT dataset [58] (accessed on 24 October 2023)¹ is utilized. However, due to the extensive computational demands of model optimization, the reduced version of this dataset is utilized. The dataset is comprised of a total of 6 classes: legitimate traffic, SlowITe, Bruteforce, Malformed data, looking, and DoS attacks. A total of 34 features are present in the dataset. Further details can be accessed in the original work that introduced the dataset [58]. An additional dataset is formulated from the existing dataset for the needs of thread detection. Classes are separated into legitimate data and anomalous activities.

Due to the large number of features present in the dataset that can result in important features blending into noise, a CNN is utilized for reduction. The reduction network consists of a convolution layer of 128 neurons total and a kernel size of 2 and utilizes a rectified linear unit (relu) activation function. This is followed by an output layer of 6 fully connected neurons that are utilized as an output. The network is trained through 10 epochs to acceptable levels of accuracy (82.7%). Following training, the network output is used as an input for the XGBoost algorithm to further improve accuracy. A graphical representation of the introduced framework for feature reduction and classification can be seen in Figure 5.

Once the features have been reduced and an intermediate dataset formulated, XGBoost hyperparameters are subjected to optimization with the use of metaheuristic algorithms. XGBoost parameters chosen for optimization and associated constraints include learning rate [0.9, 0.1], minimum child weight [10, 1], subsample [1, 0], colsample [10, 1], max depth [10, 1] and gamma [0.0, 0.8]. These parameters were chosen due to their high

¹ <https://www.kaggle.com/datasets/cnriiit/mqttset>

influence on XGBoost model performance, and their respective constraints have been empirically determined.

Several contemporary optimization metaheuristics have been included in a comparative analysis in order to determine an optimal approach. These include the original RSA [55] and GA [57] as the source of inspiration for the introduced GIRSA. Alongside these, established well-performing optimizers are also included, initialized with suggested parameters suggested in the works that originally proposed them. The included algorithms include the SSA [59], FA [38], PSO [32], WOA [40], and the COLSHADE [60] algorithm.



Figure 5 Flowchart of the introduced framework.

To provide a comprehensive assessment of the constructed models in comparison to those constructed by other contemporary optimizers a battery of standard classification

metrics, including accuracy, precision, recall, and f1-score [61], is utilized, with accuracy being the objective function chosen to guide the optimization. Further metrics include Cohen’s kappa [62], described in Equation (10), which gives a more complete easement in cases when unbalanced datasets are utilized.

$$k = \frac{p_o - p_e}{1 - p_e}$$

Equation 7 Cohen’s Kappa Statistic.

in which p_o and p_e represent observed and expected classification values.

It is important to note that the exact execution times of experimentation can vary depending on the specific hardware simulations the experimentation is carried out on.

5.2.3. Experimental Outcomes

The following section describes the outcomes of two independent experiments. The first experiment involves optimizing XGboost models for anomalous traffic detection. The second experiment involves the exact classification of the type of malicious activity. Following the presentation of the results, the outcomes are meticulously statistically validated. Finally, the best-performing model is analyzed to determine feature importance, providing an advantageous starting point for future feature reduction research.

5.2.3.1. Binary Classification Experiment

Binary classification outcomes based on the objective function evaluations over 30 independent runs are shown in terms of best, worst as well as mean and median in Table 2. Alongside these outcomes, the standard deviation and variance are provided in order to asses algorithm stability.

Additional evaluations using the indicator function outcomes over 30 runs are provided in Table 3 with relative stability indicators.

Table 2 Objective function overall outcomes.

Method	Best	Worst	Mean	Median	Std	Var
CNN-XG-GIRSA	0.497019	0.497351	0.497172	0.497109	0.000130	1.69×10^{-8}
CNN-XG-RSA	0.497301	0.497633	0.497464	0.497472	0.000106	1.13×10^{-8}
CNN-XG-GA	0.497190	0.497694	0.497480	0.497563	0.000183	3.37×10^{-8}
CNN-XG-PSO	0.497109	0.497573	0.497369	0.497482	0.000184	3.38×10^{-8}
CNN-XG-FA	0.497140	0.497663	0.497524	0.497623	0.000195	3.78×10^{-8}
CNN-XG-SSA	0.497180	0.497532	0.497412	0.497502	0.000142	2.02×10^{-8}
CNN-XG-WOA	0.497170	0.497754	0.497414	0.497402	0.000221	4.89×10^{-8}
CNN-XG-COLSHADE	0.497049	0.497563	0.497305	0.497291	0.000195	3.82×10^{-8}

An observation can already be made from the presented results. Models struggle with binary classification problems. Despite the poor performance shown by all models, the introduced algorithm attained the best outcome in comparison to other optimization algorithms even when limited performance is observed. Algorithm stability can be observed in **Error! Reference source not found.**

Table 3 Indicator function overall outcomes.

Method	Best	Worst	Mean	Median	Std	Var
CNN-XG-GIRSA	0.006218	0.005539	0.005908	0.006036	0.000265	7.02×10^{-8}
CNN-XG-RSA	0.005650	0.004983	0.005342	0.005262	0.000244	5.95×10^{-8}
CNN-XG-GA	0.005905	0.004858	0.005321	0.005241	0.000368	1.35×10^{-7}
CNN-XG-PSO	0.006036	0.005157	0.005576	0.005535	0.000348	1.21×10^{-7}

CNN-XG-FA	0.005973	0.004978	0.005245	0.005062	0.000368	1.36×10^{-7}
CNN-XG-SSA	0.005892	0.005151	0.005400	0.005199	0.000287	8.22×10^{-8}
CNN-XG-WOA	0.005913	0.004775	0.005479	0.005714	0.000464	2.15×10^{-7}
CNN-XG-COLSHADE	0.006158	0.005177	0.005632	0.005640	0.000394	1.55×10^{-7}

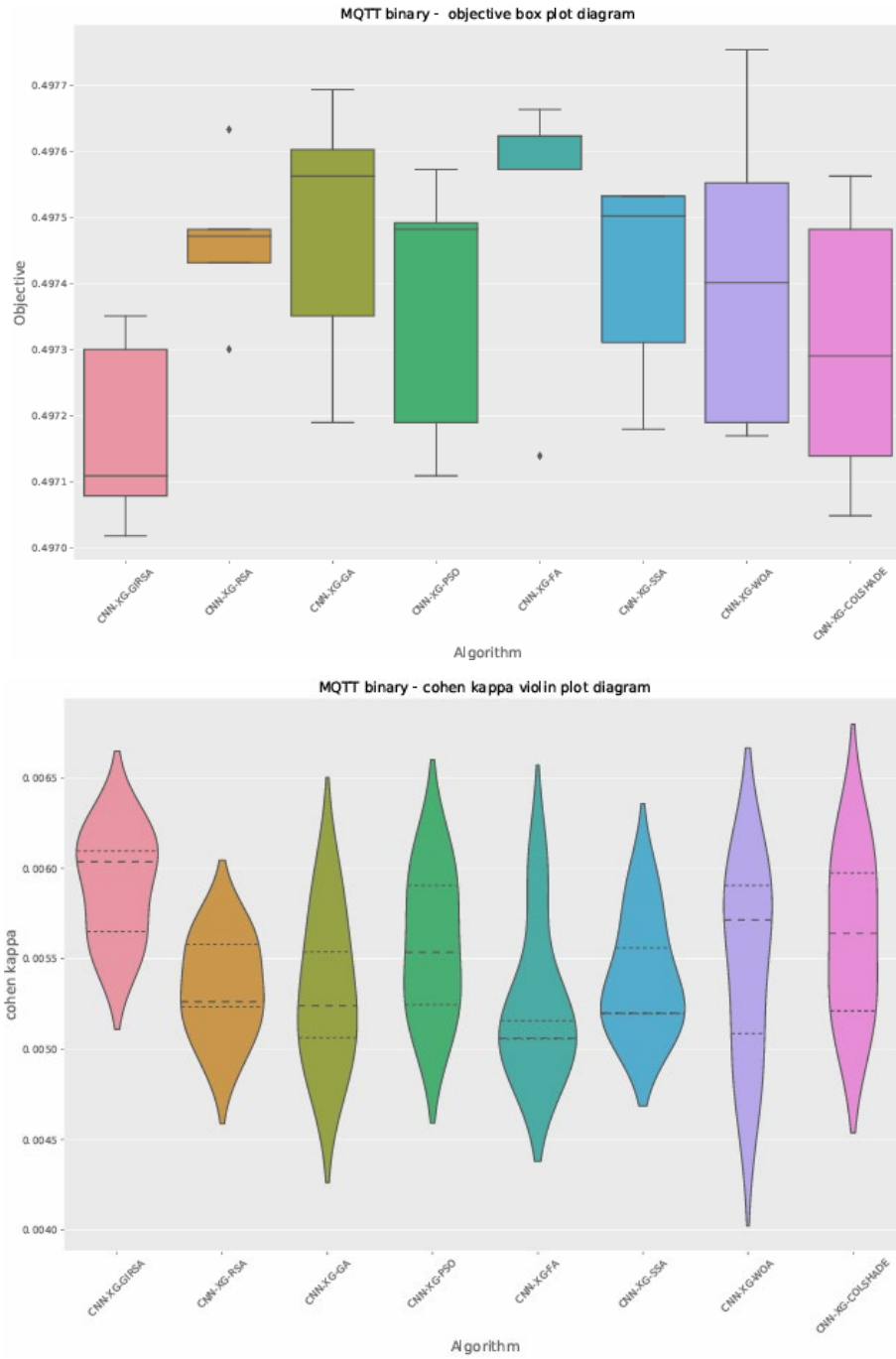


Figure 6 Binary classification evaluation outcome distribution.

An obvious advantage of the introduced algorithm is the overall better performance in comparison to other algorithms. However, a high rate of stability of the original RSA algorithm and FA needs to be noted despite inferior results in both indicator and objective evaluations. Convergence rates for each metaheuristic are tracked and plots can be observed in **Error! Reference source not found..**

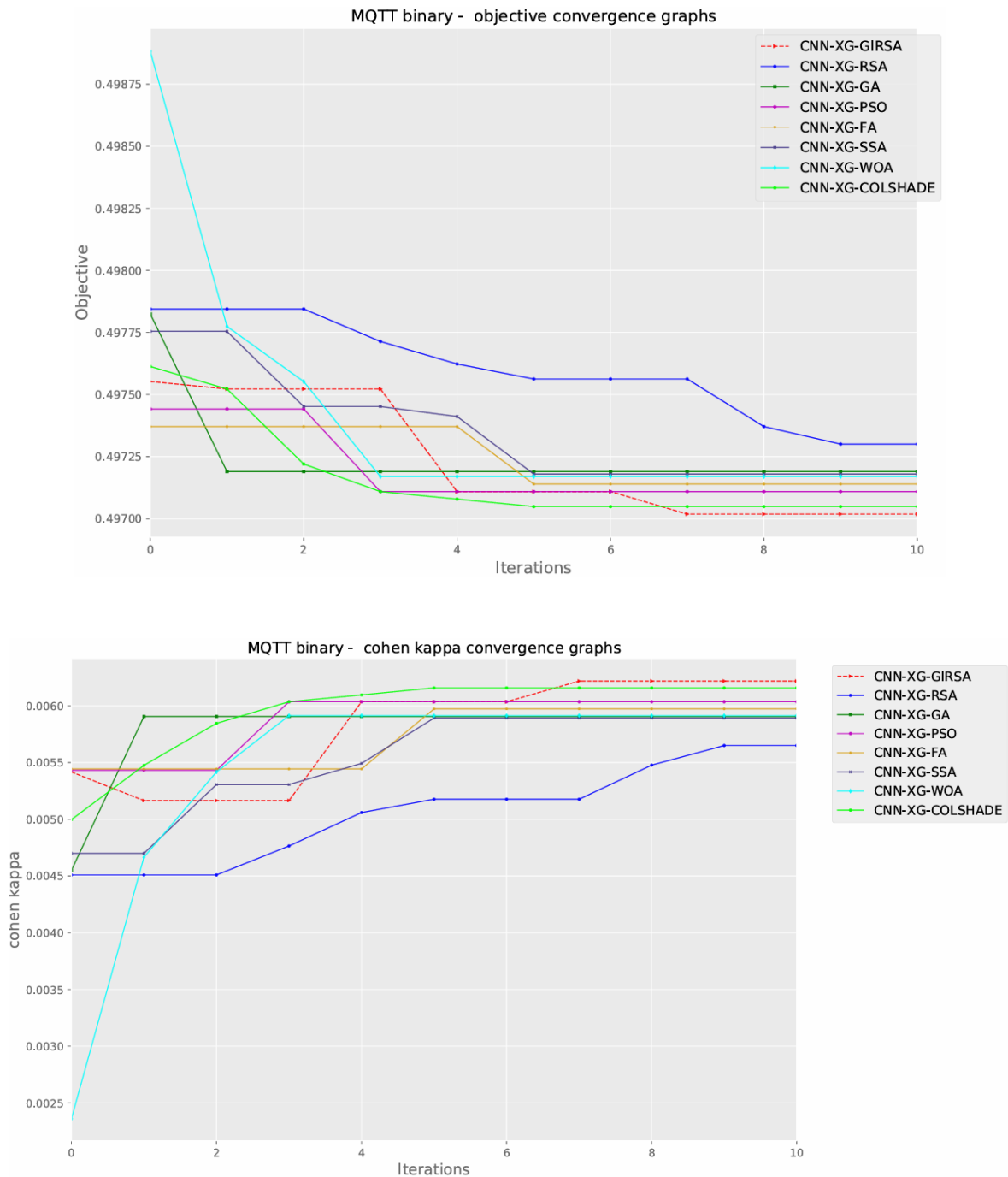


Figure 7 Objective and indicator function convergence plots.

Improvements to the convergence rate can be noted, with the introduced algorithm no longer dwelling in less promising areas, and displaying a better exploration in compassion to the original, as well as competing metaheuristics. A comparison of all constructed models in detail is presented in Table 4.

Table 4 Binary classification detailed metrics.

Method	Metric	Benign	Malicious	Accuracy	Macro Avg	Weighted Avg
CNN-XG-GIRSA	precision	0.501968	0.504592	0.502981	0.503280	0.503282
	recall	0.617113	0.389107	0.502981	0.503110	0.502981
	f1-score	0.553617	0.439388	0.502981	0.496502	0.496438
CNN-XG-RSA	precision	0.501744	0.504207	0.502699	0.502975	0.502976
	recall	0.614874	0.390777	0.502699	0.502826	0.502699
	f1-score	0.552578	0.440303	0.502699	0.496441	0.496378
CNN-XG-GA	precision	0.501790	0.504527	0.502810	0.503158	0.503160
	recall	0.630301	0.375606	0.502810	0.502954	0.502810
	f1-score	0.558751	0.430624	0.502810	0.494688	0.494616
CNN-XG-PSO	precision	0.501896	0.504470	0.502891	0.503183	0.503185
	recall	0.616649	0.389389	0.502891	0.503019	0.502891
	f1-score	0.553386	0.439521	0.502891	0.496454	0.496389
CNN-XG-FA	precision	0.501874	0.504419	0.502860	0.503147	0.503148
	recall	0.615580	0.390395	0.502860	0.502987	0.502860
	f1-score	0.552942	0.440142	0.502860	0.496542	0.496478
CNN-XG-SSA	precision	0.501842	0.504365	0.502820	0.503104	0.503105
	recall	0.615318	0.390576	0.502820	0.502947	0.502820
	f1-score	0.552817	0.440236	0.502820	0.496527	0.496463
CNN-XG-WOA	precision	0.501849	0.504383	0.502830	0.503116	0.503118
	recall	0.615802	0.390113	0.502830	0.502957	0.502830
	f1-score	0.553016	0.439949	0.502830	0.496483	0.496419
	precision	0.501943	0.504556	0.502951	0.503249	0.503251

CNN-XG-COLSHADE	recall	0.617294	0.388865	0.502951	0.503080	0.502951
	f1-score	0.553674	0.439220	0.502951	0.496447	0.496383
	support	49,589	49,701			

As can be observed from the attained outcomes, implemented models struggle with the challenging task of handling intrusion detection through binary classification within an MQTT IoT system. However, the introduced optimizer demonstrated the best outcomes when constructing an XGBoost binary classification model with a clear potential for hyperparameter optimization. Differentiating legitimate and malicious data can be challenging with a reduced feature space and number of classes, especially with observed minority classes in the dataset.

To enforce experiment repeatability, the hyperparameters selected for the respective best-performing model are provided in Table 5

Table 5 Control parameter selections made by each metaheuristic for respective best-performing.

Method	Learning Rate	Min Child Weight	Subsample	Colsample	Max Depth	Gamma
CNN-XG-GIRSA	0.546941	1	1.000000	0.179374	10	0.180448
CNN-XG-RSA	0.687255	1	1.000000	0.010368	10	0.363584
CNN-XG-GA	0.520576	2	1.000000	0.238351	10	0.717737
CNN-XG-PSO	0.566877	5	1.000000	0.024852	10	0.800000
CNN-XG-FA	0.645137	1	1.000000	0.322124	10	0.800000
CNN-XG-SSA	0.621581	1	1.000000	0.015040	10	0.000000
CNN-XG-WOA	0.625413	10	1.000000	0.010000	10	0.800000
CNN-XG-COLSHADE	0.558799	4	1.000000	0.226316	10	0.166346

5.2.3.2. Multi-Class Classification Experiment

Experimentation with multi-class classification is carried out under identical test conditions as the binary classification experiment. However, a total of six classes are present in the dataset. Objective function outcomes are shown in Table 6. Indicator function outcomes are demonstrated in Table 7.

Table 6 Objective function outcome for multi-class classification.

Method	Best	Worst	Mean	Median	Std	Var
CNN-XG-GIRSA	0.140236	0.140880	0.140544	0.140487	0.000226	5.15×10^{-8}
CNN-XG-RSA	0.140679	0.142804	0.141478	0.140941	0.000824	6.78×10^{-7}
CNN-XG-GA	0.140769	0.141595	0.141164	0.141092	0.000288	8.27×10^{-8}
CNN-XG-PSO	0.140649	0.141283	0.140880	0.140830	0.000233	5.43×10^{-8}
CNN-XG-FA	0.140477	0.140659	0.140606	0.140659	0.000072	5.13×10^{-9}
CNN-XG-SSA	0.140900	0.141112	0.141005	0.141011	0.000074	5.50×10^{-9}
CNN-XG-WOA	0.140498	0.140659	0.140624	0.140659	0.000064	4.04×10^{-9}
CNN-XG-COLSHADE	0.140649	0.141112	0.140961	0.141051	0.000178	3.16×10^{-8}

Table 7 Indicator function outcome for multi-class classification.

Method	Best	Worst	Mean	Median	Std	Var
CNN-XG-GIRSA	0.755896	0.754757	0.755369	0.755456	0.000400	1.60×10^{-7}
CNN-XG-RSA	0.755252	0.751125	0.753567	0.754486	0.001533	2.35×10^{-6}
CNN-XG-GA	0.754908	0.753264	0.754114	0.754263	0.000577	3.33×10^{-7}
CNN-XG-PSO	0.755196	0.753870	0.754698	0.754655	0.000475	2.26×10^{-7}
CNN-XG-FA	0.755564	0.754996	0.755164	0.754996	0.000226	5.10×10^{-8}
CNN-XG-SSA	0.754713	0.754328	0.754451	0.754439	0.000154	2.38×10^{-8}
CNN-XG-WOA	0.755336	0.754996	0.755095	0.754996	0.000135	1.81×10^{-8}
CNN-XG-COLSHADE	0.755119	0.754416	0.754548	0.754510	0.000323	1.04×10^{-7}

By observing the outcome, a clear superiority of the introduced algorithm can be observed, with the GIRSA attaining the best outcomes in both indicator and objective function metrics. It is also important to note the admirable stability of the WOA despite this technique not demonstrating the optimal outcomes. Algorithm stability comparisons can also be observed in **Error! Reference source not found.**, where all tested metaheuristics are graphically compared in terms of objective and indicator functions.

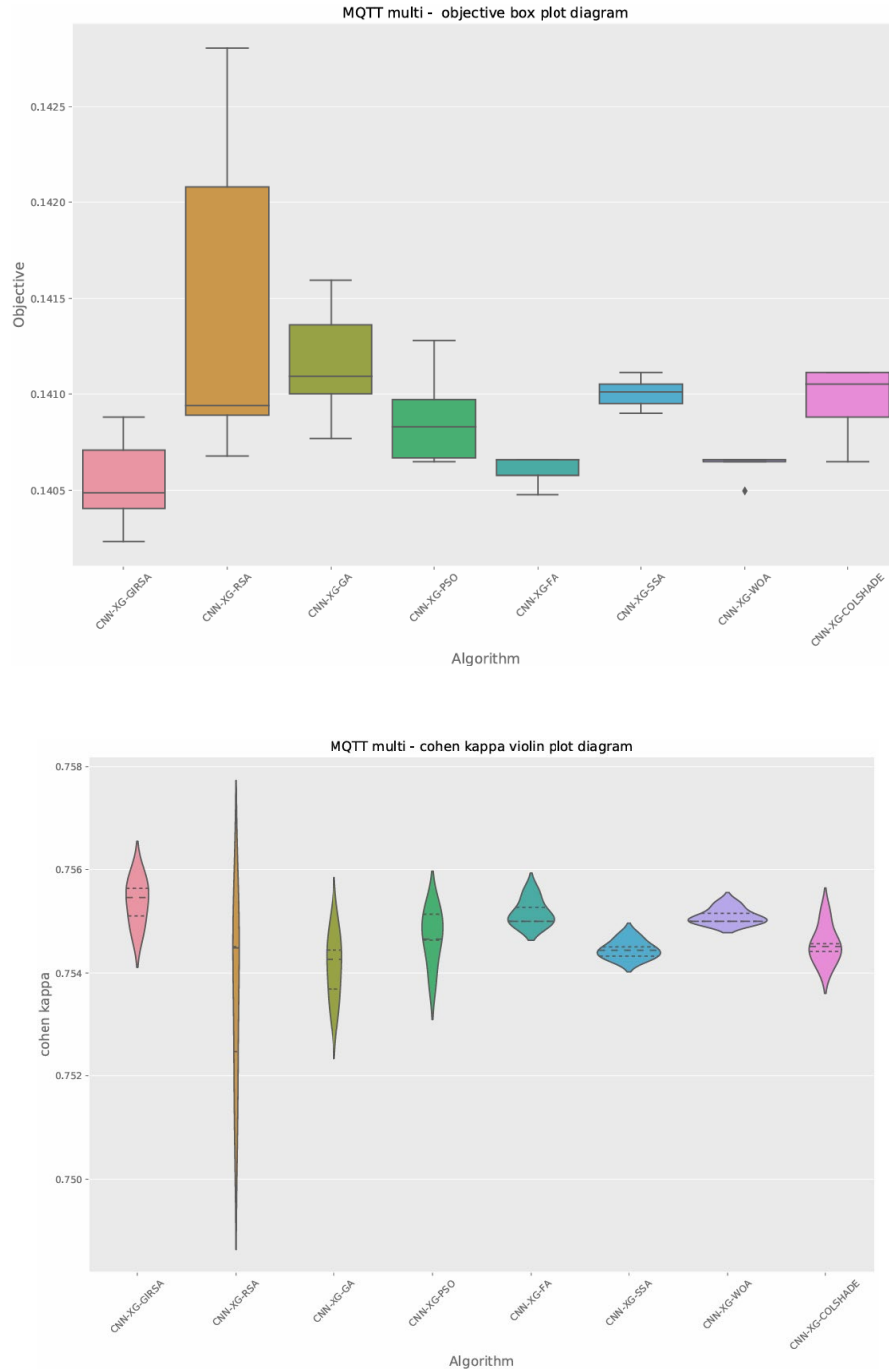


Figure 8 Objective and indicator function distribution plots for multi-class classification.

A significant stability improvement in terms of stability as opposed to the original RSA can be observed for the introduced GIRSA. Further improvements in algorithm convergence can be observed in **Error! Reference source not found.**

As can be observed in the convergence graphs, the introduced algorithm once again demonstrates an improvement in the exploration of the search space, locating a better solution and avoiding local solutions in favor of a global solution demonstrating better outcomes.

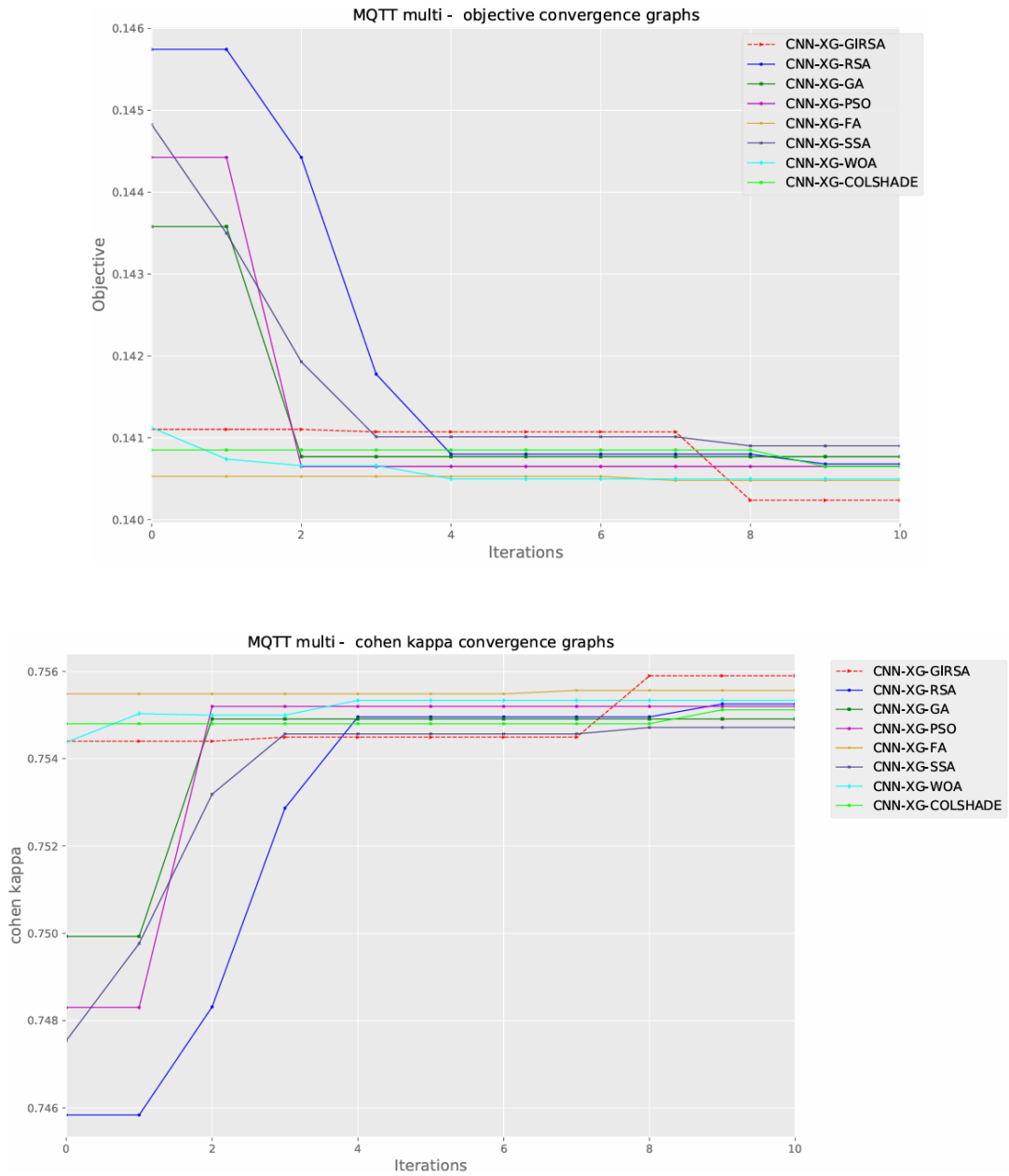


Figure 9 Objective and indicator function convergence plots for multi-class classification.

Detailed comparisons of the best-performing models generated by each metaheuristic are shown in Table 8.

Table 8 Detailed comparison between best-performing models for multi-class classification.

Method	Metric	Benign	DOS	Malformed	Bruteforce	SlowITe	Flood	Accuracy	Macro Avg	Weighted Avg
CNN-XG-GIRSA	precision	0.878533	0.867624	0.728448	0.616408	0.794778	0.967742	0.859764	0.808922	0.855635
	recall	0.950442	0.853392	0.154667	0.638934	0.551250	0.163043	0.859764	0.551955	0.859764
	f1-score	0.913074	0.860449	0.255159	0.627469	0.650984	0.279070	0.859764	0.597701	0.849664
CNN-XG-RSA	precision	0.879449	0.866883	0.662338	0.616214	0.787848	0.967742	0.859321	0.796746	0.853417
	recall	0.949254	0.854416	0.155583	0.630660	0.554147	0.163043	0.859321	0.551184	0.859321
	f1-score	0.913019	0.860604	0.251976	0.623353	0.650649	0.279070	0.859321	0.596445	0.849402
CNN-XG-GA	precision	0.878810	0.866539	0.656366	0.619365	0.795276	0.769231	0.859231	0.764264	0.852741
	recall	0.950281	0.854032	0.161989	0.618938	0.548714	0.163043	0.859231	0.549500	0.859231
	f1-score	0.913149	0.860240	0.259848	0.619152	0.649378	0.269058	0.859231	0.595138	0.849346
CNN-XG-PSO	precision	0.879615	0.865647	0.659119	0.621212	0.789446	0.967742	0.859351	0.797130	0.853170
	recall	0.948972	0.855567	0.159854	0.621926	0.552698	0.163043	0.859351	0.550343	0.859351
	f1-score	0.912978	0.860577	0.257304	0.621569	0.650192	0.279070	0.859351	0.596948	0.849456
CNN-XG-FA	precision	0.879857	0.866169	0.648559	0.622165	0.785898	0.967742	0.859523	0.795065	0.853091
	recall	0.948931	0.855618	0.178462	0.611584	0.553060	0.163043	0.859523	0.551783	0.859523
	f1-score	0.913089	0.860862	0.279904	0.616829	0.649235	0.279070	0.859523	0.599831	0.850136
CNN-XG-SSA	precision	0.879857	0.864857	0.632794	0.618892	0.798845	1.000000	0.859100	0.799208	0.852331
	recall	0.948790	0.856181	0.167175	0.608366	0.550887	0.163043	0.859100	0.549074	0.859100
	f1-score	0.913025	0.860497	0.264479	0.613584	0.652090	0.280374	0.859100	0.597341	0.849390
CNN-XG-WOA	precision	0.879515	0.865361	0.664269	0.623851	0.787410	0.967742	0.859502	0.798024	0.853237
	recall	0.949697	0.855772	0.169005	0.608366	0.552698	0.163043	0.859502	0.549764	0.859502
	f1-score	0.913260	0.860540	0.269455	0.616011	0.649500	0.279070	0.859502	0.597973	0.849721
CNN-XG-COLSH ADE	precision	0.879789	0.864590	0.694805	0.619640	0.789802	0.967742	0.859351	0.802728	0.853961
	recall	0.948911	0.856028	0.163209	0.617789	0.549801	0.163043	0.859351	0.549797	0.859351
	f1-score	0.913043	0.860288	0.264328	0.618713	0.648302	0.279070	0.859351	0.597291	0.849429
	support	49,639	39,077	3278	4351	2761	184			

Various models show differing quality of performance facing different classification challenges. This is to be expected as per the NFL theorem. Notable outcomes are shown by

the GIRSA algorithm, which has demonstrated a clear dominance in terms of optimal outcomes. However, notable results are also shown by the FA and the SSA algorithm. The confusion matrix of the best performing model for multi-class classification can be observed in Figure 10.

As can be observed, the algorithm struggles to identify slowITe and flood attacks often confusing slowing for legitimate data and flood attacks for denial-of-service attacks. However, flood and denial-of-service attacks are fairly similar in practice, so detection is still within acceptable margins. Additionally, malformed data can often be classified as DOS attacks. It can be deduced that the introduced approach performs significantly better when tackling the problem of anomalous traffic detection as a multi-class classification challenge, rather than a simple binary challenge. This is likely due to the confusion between flood and slowITe attacks with legitimate data. Nevertheless, the introduced method shows great potential for real-world implementation. Furthermore, the introduced optimization metaheuristic demonstrates improvements over existing techniques as well as the original base algorithm.

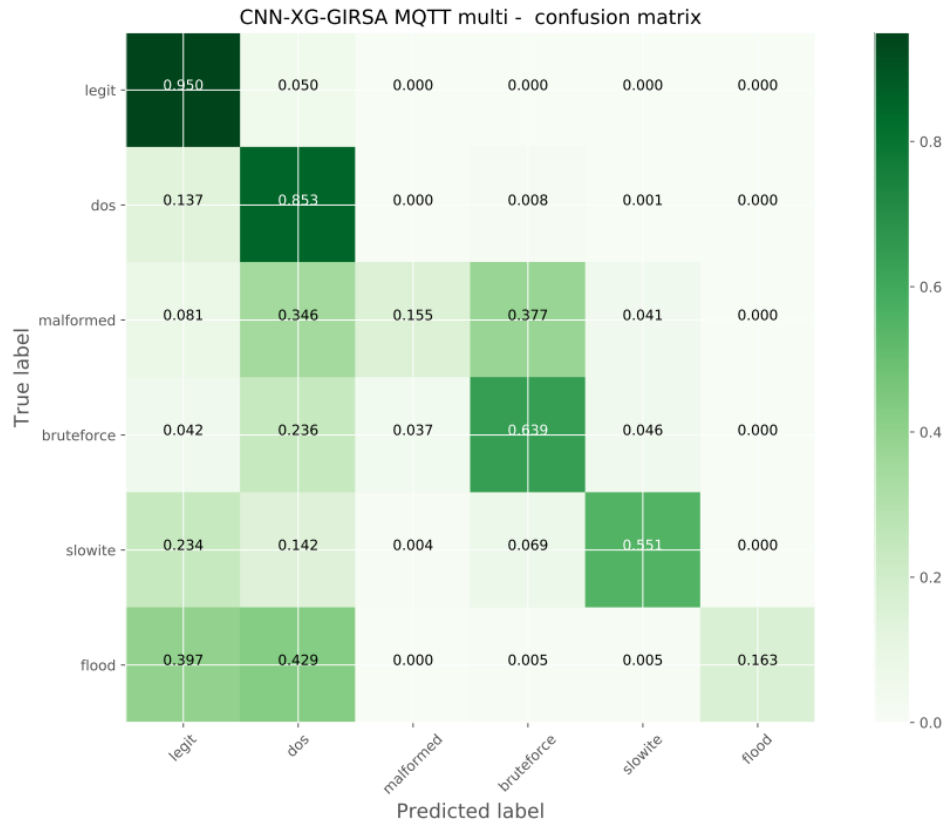


Figure 10 GIRSA XGBoost best performing multi-class classification model.

To encourage experimental repeatability, hyperparameter selections made by each algorithm are shown in Table 9.

Table 9 Control parameter selections made by each metaheuristic for respective best-performing

Method	Learning Rate	Min Child Weight	Subsample	Colsample	Max Depth	Gamma
CNN-XG-GIRSA	0.900000	6	0.484839	1.000000	6	0.123762
CNN-XG-RSA	0.894472	10	0.835341	1.000000	6	0.000000
CNN-XG-GA	0.682788	5	0.654867	1.000000	8	0.439790
CNN-XG-PSO	0.293758	10	0.890094	1.000000	10	0.348853
CNN-XG-FA	0.873667	10	1.000000	1.000000	10	0.800000
CNN-XG-SSA	0.900000	10	0.528148	0.761996	6	0.062693
CNN-XG-WOA	0.578212	10	0.863927	0.915171	10	0.800000
CNN-XG-COLSHADE	0.696242	9	0.589664	0.895006	8	0.381628

5.2.3.3. Outcome Statistical Validation

Modern optimization research demands optimization results be meticulously statistically validated in order to establish the statistical significance of the demonstrated improvements. The preferred approach for validating outcomes is the use of parametric tests. However, the safe utilization of these tests needs to be established first. Three criteria need to be met: independence, normality, and homoscedasticity [63]. The first condition is fulfilled by utilizing an independent random seed for each execution. The normality condition is assessed using the outcomes of the Shapiro–Wilk test shown in Table 10 as well

as through the visual observations of objective function outcome distributions shown in **Error! Reference source not found..**

Table 10 Shapiro–Wilk normality tests.

Problem	GIRSA	RSA	GA	PSO	FA	SSA	WOA	COLSHADE
XG-Binary	0.032	0.041	0.046	0.018	0.021	0.033	0.035	0.040
XG-Multi	0.032	0.041	0.046	0.018	0.021	0.033	0.035	0.040

The resultant *p-values* were all less than 0.05, suggesting that the null hypothesis (H0) may be rejected. As a result, we may infer that the outcomes produced in all three simulations do not follow a normal distribution. These outcomes are further reinforced in **Error! Reference source not found..**

Parametric tests were not applicable since the normality assumption was not fulfilled. As a result, the non-parametric Wilcoxon signed-rank [64] test was used in the following stage. This test can be applied to the same data series consisting of the best values achieved in every run of each metaheuristic.

The created algorithm is used as the control algorithm in this test, and the Wilcoxon signed-rank test was run on the specified data series. The calculated *p-values* in all three observed cases were less than 0.05. Considering the significance level of $\alpha = 0.1$, these findings show that the introduced algorithm outperformed all competing approaches statistically significantly. Table 11 shows the overall results of the Wilcoxon signed-rank test.

Table 11 Wilcoxon signed-rank test values exhibiting *p-values* for all three experiments (GIRSA vs. Others.).

Problem/ <i>p</i> -Values	RSA	GA	PSO	FA	SSA	WOA	COLSHADE
XG-Binary	0.042	0.003	0.018	0.037	0.04	0.027	0.031
XG-Multi	0.042	0.003	0.018	0.037	0.04	0.027	0.031

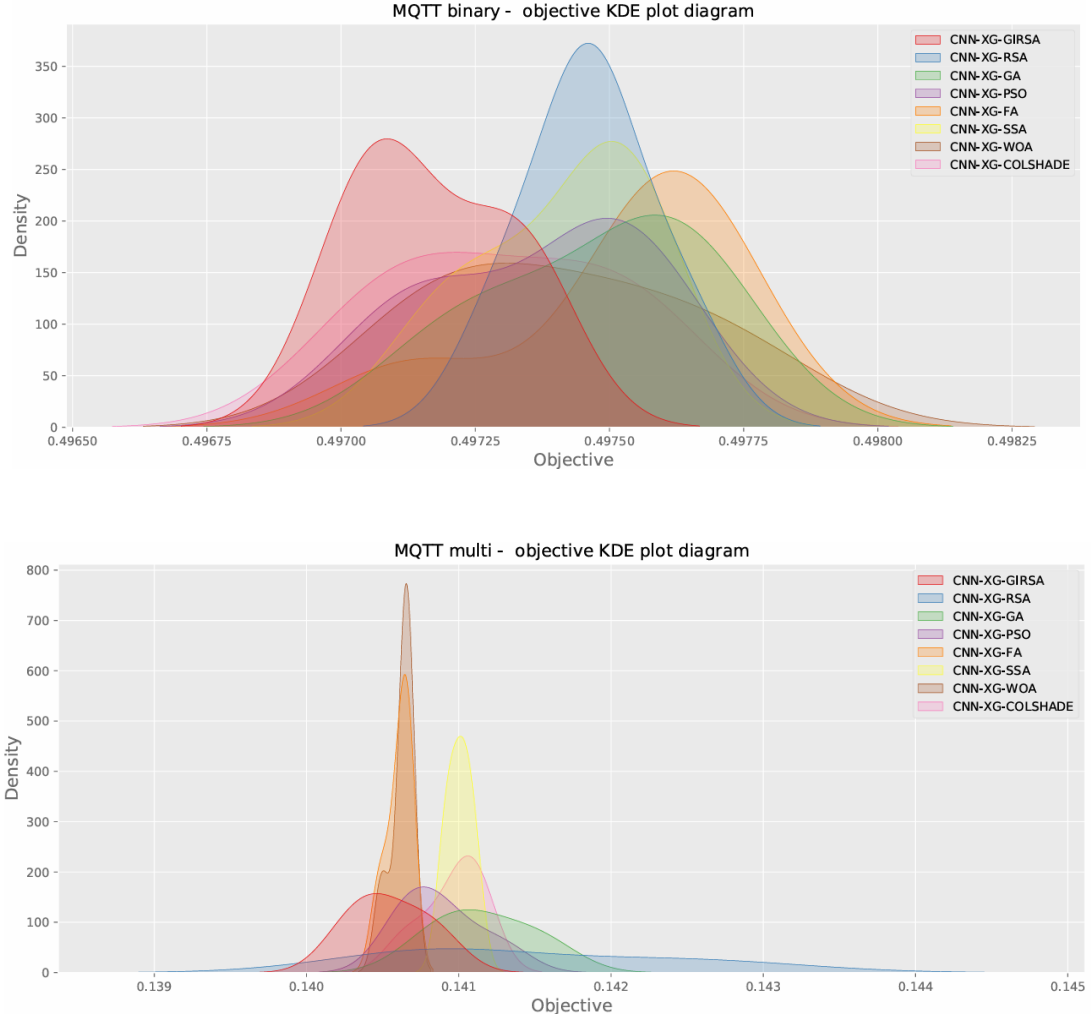


Figure 11 Objective function KDE plots for binary and multiclass classification.

5.2.3.4. Best Model SHAP Interpretation

The attained best-performing model has been subjected to analysis through the use of Shapley additive explanations (SHAP) [7] to determine feature impacts on model decisions. The feature reduction CNN models as well as the best constructed XGBoost model have been interpreted using the kernel and tree explainer techniques. SHAP utilized a game theory-based approach to determine the impact each feature poses on model decisions. The analysis outcomes are graphically presented in Figure 12 for the CNN model and in Figure 13 for the XGBoost model.

From the feature importance analysis of the CNN feature reduction model, it can be deduced that the tcp.time_delta presents the highest influence on model decisions, closely followed by mqtt.conack.flags.reserved. The third-highest importance is shown by the

mqtt.msgtype feature. The remaining features that have a significant influence are mqtt.kalive and mqtt.retain. Following these features, a significant reduction in importance can be observed. Accordingly, a set of six features have been maintained as outputs for the CNN and inputs for the XGBoost model. The importance of these synthetic features is shown in Figure 13.

Since the synthetic features have no direct interpretation of the real-world dataset, they are simply assigned a number. Synthetic feature 3 has the highest impact on the model decision, followed by features 1 and 5. Finally, a small impact can be observed in feature 0. Features 4 and 2 do not seem to notably influence model decisions, suggesting that further feature reduction may be performed while maintaining computational complexity.

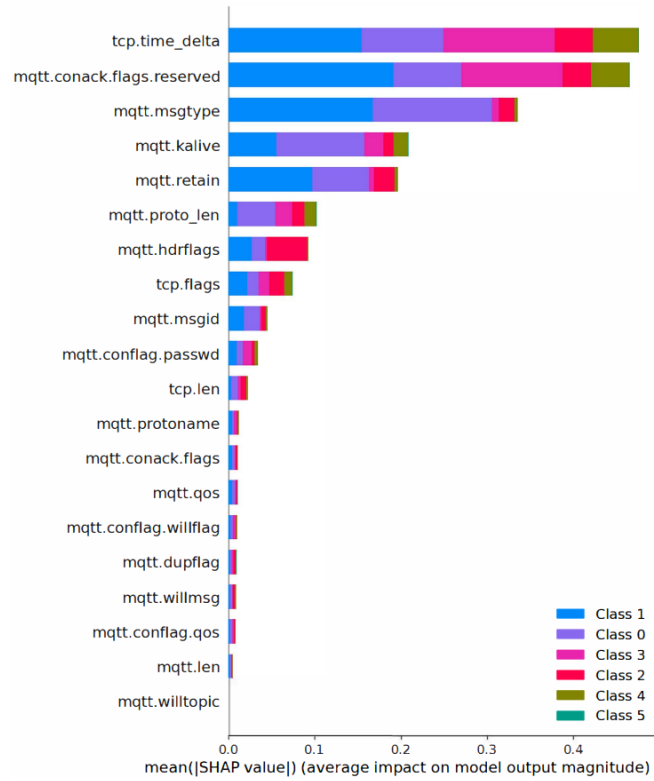


Figure 12 SHAP analysis outcomes for best CNN model.

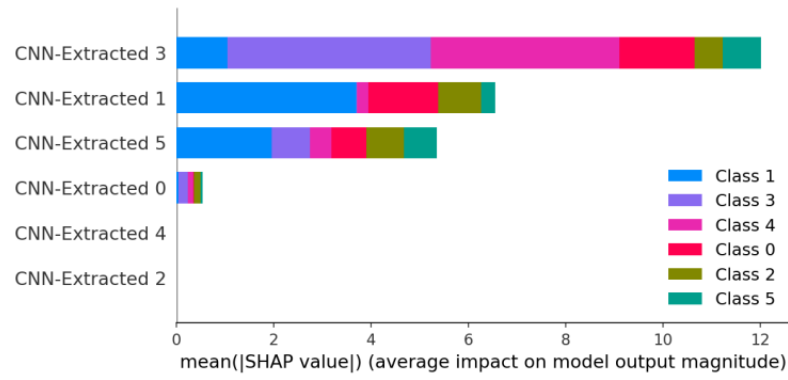


Figure 13 SHAP analysis outcomes for best XGBoost model.

5.3. Application 2: The Long Short-Term Memory Tuning for Multi-Step Ahead Wind Energy Forecasting using enhanced Sine Cosine Algorithm and Variation Mode Decomposition

There are several challenges to forecasting energy consumption accurately, including the complexity of the generated data, which includes multiple sources of energy, various end-use sectors, and diverse demand patterns. Energy consumption is also influenced by various factors such as weather, economic conditions, population growth, and technological advancements, which can make it difficult to accurately predict future energy usage. To address these challenges and improve forecasting, it is necessary to find better models and methods that can take into account multiple factors. One approach for addressing data complexity is through the use of signal decomposition methods. A goal of applying signal decomposition is to extract separate signal components from composite signals. By utilizing decomposition techniques, trends in energy data can be observed and separated, making the prediction process more robust and resilient to noise. This research utilizes one such method, variational mode decomposition (VMD) [65] to address data complexity inherent to energy data time series.

An approach that has seen great success in recent years applied to forecasting is the use of machine learning (ML) algorithms. By formulating energy data as a time-series forecasting can be systematically tackled. One notably well-performing approach for time series is the use of Long short-term memory (LSTM) [66]. However, like many algorithms, LSTM networks present several adjustable parameters, called hyperparameters that require

adequate adjustments to ensure favorable performance. With the increasing complexity of algorithms methods for automating hyperparameters selection are required. Therefore, a popular approach is to formulate parameter selection as an optimization problem.

Swarm intelligence algorithms simulated individual behaviors of agents obeying sets of predefined rules. This allows algorithms to tackle complex tasks, and even NP-hard problems with relative ease, using realistic computational power and within reasonable time frames. Swarm intelligence algorithms excel at optimization problems. Therefore, this work explores the potential of a notably well-performing sine cosine algorithm (SCA) [67] for optimizing LSTM network hyperparameters to improve performance. Furthermore, this work introduces an enhanced variation of the SCA that builds on the success of the original approach. This research was published in [2].

5.3.1. Methods

This section gives an overview of Long short-term memory (LSTM) and Variational mode decomposition (VMD) after that it delivers the original Sine Cosine Algorithm (SCA) and a modified version of it.

5.3.1.1. Overview of Long Short-Term Memory (LSTM)

For handling long-term relationships in the time domain, Hochreiter and Schmidhuber [66] created the LSTM neural network, a form of recurrent neural network (RNN). It has nonlinear gated units and memory cells. The LSTM can handle vanishing/exploding gradient difficulties [68].

Where $X_t = [x_t^1, x_t^2, x_t^3, \dots, x_t^N]$ denotes the count of entries. $H_t = [h_t^1, h_t^2, h_t^3, \dots, h_t^K]$ represent the K hidden neurons, and $C_t = [c_t^1, c_t^2, c_t^3, \dots, c_t^K]$ represent the note status of the LSTM network during iteration t , and f_t represents the forgotten gate, i_t represents the input gate, and O_t represents the output gate. On every iteration, t , the input X_t is delivered to three gates, together with the previously hidden state H_{t-1} , to calculate the next hidden state H_t and to adjust the prior cell state C_{t-1} to calculate the next cell state C_t . The following is the mathematical formula of several LSTM procedures:

In the first step, the forgetting gate f_t in the LSTM layer determines whether information from the previous cell state C_{t-1} is to be deleted expressed as:

$$f_t = \sigma (W_f X_t + U_f H_{t-1} + b_f)$$

Equation 8 Forgetting gate.

The LSTM units then determine whether information should be contained in the cell state C_t . This procedure consists of two steps: firstly, the candidate vector $\hat{C}_t$ collects all of the data from the prior phase H_{t-1} and given input X_t . Secondly, the input gate scans the data to be entered in the cell state C_t carefully. These procedures are calculated as follows:

$$\hat{C}_t = \tanh(W_c X_t + U_c H_{t-1} + b_c)$$

$$i_t = \sigma (W_i X_t + U_i H_{t-1} + b_i)$$

Equation 9 Determine whether information should be contained.

In the third stage, a new cell state C_t is generated based on the outcomes of the preceding phases as follows:

$$C_t = f_t \times C_{t-1} + i_t \times \hat{C}_t$$

Equation 10 Generate new cell state.

Finally, the output gate O_t decided how much information should be delivered to calculate the outcome H_t at the following iteration:

$$O_t = \sigma (W_o X_t + U_o H_{t-1} + b_o)$$

$$H_t = O_t \times \tanh(C_t)$$

Equation 11 Output gate.

In which W_f, W_i, W_c, W_o are the weights arrays related to the input X_t while U_f, U_i, U_c, U_o are the repeated weights arrays linked with previous hidden phase H_{t-1} and b_f, b_i, b_c , and b_o are the biases variables for the four gates. $\sigma(x) = 1/(1+e^{-x})$ is the log-sigmoid

objective function, while $\tanh(x) = (e^x - e^{-x}) / (e^x + e^{-x})$ is the hyperbolic tangent objective function. The log-sigmoid objective function provides values ranging from 0 to 1 that describe what proportion of each quantity goes across. A "0" denotes that nothing will flow across, while a "1" indicates that everything may go across, values between 0 and 1 define what percentage of the number goes across.

5.3.1.2. Variational Mode Decomposition (VMD)

A relatively novel yet powerful method for decomposing a provided by is VMD [65]. This method deconstructs a complex input signal f into a discrete set of bandlimited sub-signals u_k . These are characterized by bandwidth, estimated by H^1 Gaussian smoothness of the shifted signal, and center pulsation w_k . This form of decomposition is a constrained variational problem as per Equation 12.

$$\max_{\{u_k\}, \{w_k\}} \left\{ \sum_{k=1}^k \left\| \partial_t \left[(\delta(t) + \frac{j}{\pi t} \times u_k(t)) \right] e^{j\omega_k t} \right\|_2^2 \right\}$$

Equation 12 constrained variational problem.

Where Equation 12 is subjected to: $\sum_{k=1}^k u_k = f$, and can be tackled by incorporating a quadratic penalty alongside Lagrangian multiplies as given in Equation 13.

$$L(\{u_k\}, \{w_k\}, \lambda) = \alpha \sum_{k=1}^k \left\| \partial_t \left[(\delta(t) + \frac{j}{\pi t} \times u_k(t)) \right] e^{j\omega_k t} \right\|_2^2 + f(t) - \sum_{k=1}^k u_k(t) \left\|_2^2 + \langle \lambda(t), f(t) - \sum_{k=1}^k u_k(t) \rangle \right.$$

Equation 13 quadratic penalty alongside Lagrangian multiplies.

where parameter α is applied to balance data fidelity. The attained modes can be formulated as per Equation 14.

$$\hat{u}_k(w) = \frac{\hat{f}(w) - \sum_{i \neq k} \hat{u}_i(w) + (\frac{\hat{\lambda}(w)}{2})}{1 + 2\alpha(w - w_k)^2}$$

Equation 14 attained modes.

in which w_k represents the calculated center of the appropriate modes' intensified spectrum. Additionally, wiener filtering was incorporated to increase resilience to sampling and noise as shown in Equation 15.

$$w_k = \frac{\int_0^\infty w |\hat{u}_k(w)|^2 dw}{\int_0^\infty |\hat{u}_k(w)|^2 dw}$$

Equation 15 wiener filtering.

A full description of the VMD algorithm in detail can be explored in [65].

5.3.1.3. Original Sine Cosine Algorithm (SCA)

Originally introduced in 2016 the SCA [67] algorithm is a novel mathematics-inspired population-based optimization algorithm. The proposed method modifies agent locations via a mechanism inspired by trigonometry. Upcoming formulas describe how solutions are upgraded in SCA:

$$X_i^{t+1} = X_i^t + r_1 \cdot \cos(r_2) \cdot |r_3 P_i^t - X_i^t|$$

$$X_i^{t+1} = X_i^t + r_1 \cdot \sin(r_2) \cdot |r_3 P_i^t - X_i^t|$$

Equation 16 Sin and Cos algorithms

The above-mentioned are combined to produce equations:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \cdot \sin(r_2) \cdot |r_3 P_i^t - X_i^t|, & r_4 < 0.5 \\ X_i^t + r_1 \cdot \cos(r_2) \cdot |r_3 P_i^t - X_i^t|, & r_4 \geq 0.5 \end{cases}$$

Equation 17 Incorporated Sin & Cos algorithms

wherein X_i^t denotes the present individual i in the d -th dimension at iteration t , P_i^t denotes an ideal agent's positioning in the d - th dimension at iteration t , and r_1, r_2, r_3 and r_4

are arbitrary factors. These factors are used to prevent local optima and to harmonize exploring and exploiting search tendencies.

The value of r_1 influences should an agent adjusts its location towards the global optimal ($r_1 < 1$) or away from it ($r_1 > 1$). It harmonizes intensification and diversification search behaviors, r_1 declines linearly from a fixed constant (a) to 0 [67]. It is updated utilizing the given formula:

$$r_1 = a - t \frac{a}{T_{max}}$$

Equation 18 Update r_1 value.

in which a represents a constant, t denotes the ongoing iteration, and T_{max} is the highest iterative count.

Factor r_2 given in the region $[0, 2\pi]$ defines the intensity and direction of agent movement regarding the attained ideal agent, objective. An additional arbitrary factor, r_3 , gives the location an arbitrary weighting. This enables stressing ($r_3 > 1$) or reducing the stress ($r_3 < 1$) the influence of the endpoint of other solutions' location updates. r_3 from the $[0, 2]$ range. The arbitrary factor r_4 from a range $[0, 1]$ acts as a switch to select which function is used in equation Equation 17.

5.3.1.4. Enhanced Sine Cosine Algorithm (ESCA)

The original SCA metaheuristic demonstrates respectable performance. However, extensive testing using standard CEC functions has indicated that in some executions the algorithm tends to dwell in sub-optimal regions of the search space. This shortcoming may result in decreased overall performance. This work, therefore, proposes an enhanced SCA (ESCA) that tackles the shortcomings of the original SCA to further improve on the admirable performance.

The introduced alteration incorporates the search mechanism from the novel reptile search algorithm [69] shown in Equation 19.

$$\begin{aligned}
X_i^{t+1} &= \begin{cases} X_{best}^t - \eta_i \times \beta - R_i^t \times rand, & t \leq \frac{T}{4} \\ X_{best}^t \times X_{rand}^t \times ES \times rand, & t \leq \frac{T}{2} \end{cases} \quad \text{and } t > \frac{T}{4} \\
\eta_i &= X_{best}^t \times P_i, \\
R_i &= \frac{X_{best}^t - X_i^t}{X_{best}^t + \epsilon}, \\
ES &= 2 \times r_1 \times \left(1 - \frac{1}{T}\right), \\
P_i &= \alpha + \frac{X_i^t - M(X_i^t)}{X_{best}^t \times (UB - LB) + \epsilon}
\end{aligned}$$

Equation 19 Reptile search algorithm.

in which X_{best}^t represents the current best solution, the ongoing iteration is shown as t while the highest iterative count is T . Further β defines a constant defining the speed of exploration and has a value of 0.1, X_{rand}^t is an arbitrarily selected agent, ES denotes a random decreasing value between $[-2, 2]$. A minimal value is defined by η to ensure that the denominator cannot become 0. A random value r_1 is chosen from a range of $[-1, 1]$, α defines a constant of 0.1. Finally, $rand$ represents an arbitrary value from a range $[0, 1]$.

To allow both algorithms to contribute to the exploration an additional control parameter μ is also introduced. During every iteration, a random value DS is drawn from a uniform distribution in the range $[0, 1]$. Should the value of $DS > \mu$ standard SCA search is conducted. However, should the value of $DS < \mu$ then RSA search is utilized. The value of $\mu = 0.3$ has been empirically determined through extensive experimentation.

Algorithm 2 presents the pseudo-code of the introduced ESCA.

Algorithm 2 Pseudo-code of the ESCA.

Initialization. Chaotically initialized solutions $X_i(1, 2, 3, \dots, n)$.

Initialize the maximum count of iterations T .

Initialize DS value to 0.3

while $t < T$ **do**

Generate a random value of μ

if $DS > \mu$ **then**

for Every X in the generated candidates **do**

```

.      .      Calculate objective value.
.      .      if  $f(X)$  is greater than  $f(P)$  then
.      .      .      Update the positioning of the ideal agent ( $P = X^*$ ).
.      .      end if
.      end for
.      Redefine  $r_1$  using Equation 18.
.      Redefine  $r_2$ ,  $r_3$  and  $r_4$  factors.
.      Redefine the locations of the ideal solution by Equation 17.
else
.      for Every  $X$  in the generated candidates do
.      .      Calculate objective value.
.      .      Update positions using reptile search mechanism from Equation 19 .
.      end for
end if
end while
return  $P$  as the ideal agent.

```

5.3.2. Experimental Setup

In this section, the dataset is used during experimentation followed by expectations of encoding schemes and framework details. Furthermore, the evaluation metrics will be covered, as well as the algorithms utilized for comparative analysis and simulation test conditions.

5.3.2.1. Datasets

For this research, a real-world dataset covering an anonymous wind farm in Chain has been utilized. The dataset is publicly available ². For this work data concerning wind farm number 2 has been used date from 01.07.09 to 31.12.2010. A total of 70% of the

² <http://blog.drhongtao.com/2016/07/gefcom2012-load-forecasting-data.html>

available data was used to train the models, the following 10% was used to validate the approach, with the latter 20% reserved for testing as shown in Figure 14.

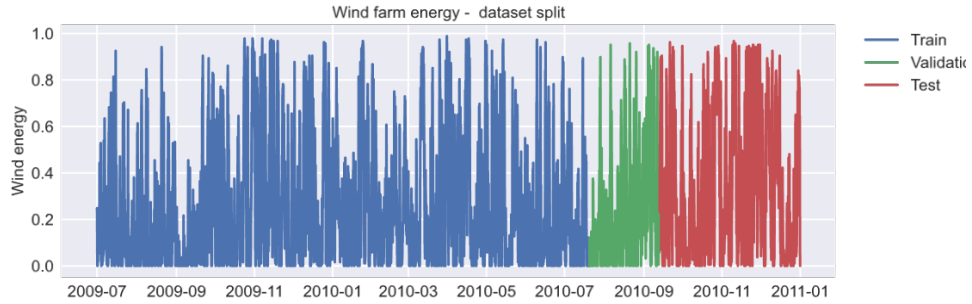


Figure 14 Wind farm dataset virtualization with data split.

5.3.2.2. Experimental Setup

The experimental procedure involves comparing several state-of-the-art metaheuristic algorithms tasked with selecting optimal LSTM parameters. The experimental procedure involves comparing several state-of-the-art metaheuristic algorithms tasked with selecting optimal LSTM parameters. The results are then evaluated using MAE , MSE , $RMSE$ and R^2 metrics shown in Equations Equation 20Equation 21Equation 22Equation 23 respectively.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Equation 20 Mean absolute error.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Equation 22 Root Mean Squared Error.

$$MSE = \frac{1}{n} \sum_{i=1}^n \frac{(y_i - \hat{y}_i)^2}{y_i}$$

Equation 21 Mean Squared Error.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Equation 23 coefficient of determination.

For testing purposes, VMD and all evaluated algorithms have been independently implemented for this research using Python accompanied by standard libraries including TensorFlow, Pandas, NumPy, Sklearn, and VMD-Python, while seaborn was utilized for

visualization. A total of 3 ($k = 3$) modes have been extracted from original input features using VMD, with one additional mode representing the residuals also provided. A visualization of each decomposed feature can be seen in Figure 15.

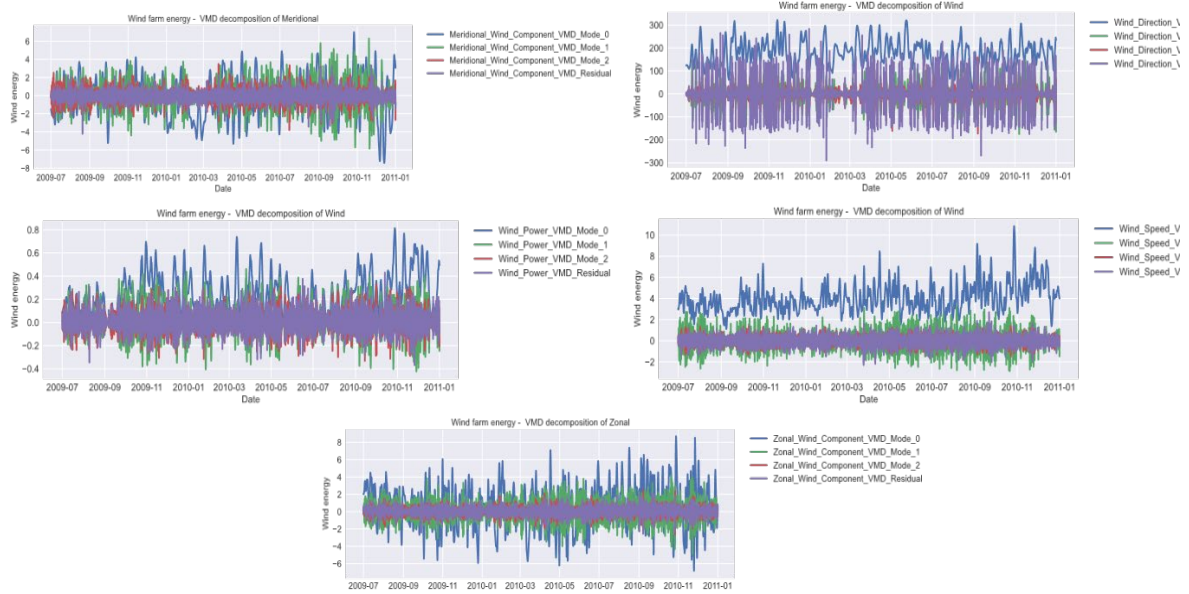


Figure 15 Decomposition mode visualizations for each input feature.

Each evaluated LSTM network has been tasked with casting predictions three steps ahead based on the input of six preceding steps. The metaheuristics tasked with optimizing the performance of these networks have been provided with a population of five agents and allocated eight iterations to improve results. Additionally, due to excessive computational demands, these experiments have been carried out over eight individual executions to account for randomness intrinsic to metaheuristic algorithms.

5.3.2.3. Experimental Results

During testing, all metaheuristics were provided with decomposed signal inputs as evaluated on their prediction performance using the described metrics. Overall objective function performance results averaged over 8 independent runs for each tested metaheuristic as demonstrated in Table 12.

Table 12 Overall objective function performance metrics for each tested approach.

Method	Best	Worst	Mean	Median	Std	Var
--------	------	-------	------	--------	-----	-----

LSTM-ESCA	0.006784	0.006973	0.006879	0.006879	9.47E-05	8.97E-09
LSTM-SCA	0.006864	0.006923	0.006893	0.006893	2.95E-05	8.68E-10
LSTM-PSO	0.007032	0.007281	0.007157	0.007157	1.25E-04	1.55E-08
LSTM-ABC	0.006869	0.006928	0.006901	0.006928	2.94E-05	8.63E-10
LSTM-WOA	0.007014	0.007126	0.007063	0.007014	5.52E-05	3.05E-09

As demonstrated in Table 12, the introduced novel LSTM-ESCA approach attained the best objective results in the word, mean, and median cases, only slightly being outdone by the original SCA in the worst-case scenario.

Detailed metrics for the best execution of each metaheuristic are provided in Table 13

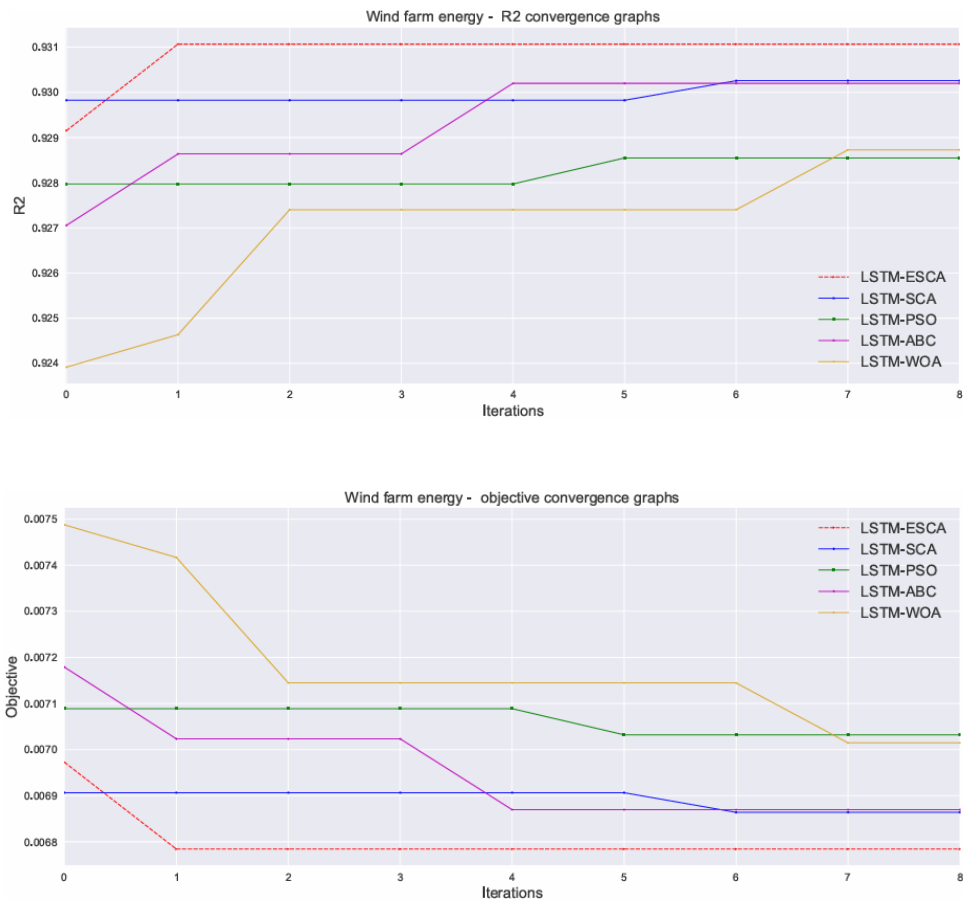
Table 13 Detailed metrics for each prediction step of best-performing models of every evaluated approach.

	Error indicator	LSTM-ESCA	LSTM-SCA	LSTM-PSO	LSTM-ABC	LSTM-WOA
One-step ahead	R ²	0.879276	0.872990	0.872271	0.877037	0.872033
	MAE	0.075739	0.077739	0.078073	0.076589	0.077883
	MSE	0.011881	0.012500	0.012570	0.012101	0.012594
	RMSE	0.109000	0.111801	0.112118	0.110006	0.112222
Two-step ahead	R ²	0.947839	0.948728	0.946276	0.948175	0.947074
	MAE	0.050406	0.050028	0.051666	0.050812	0.050734
	MSE	0.005133	0.005046	0.005287	0.005100	0.005209
	RMSE	0.071648	0.071035	0.072713	0.071417	0.072171
Three-step ahead	R ²	0.966084	0.969051	0.967094	0.965383	0.967067
	MAE	0.041545	0.040221	0.041549	0.042781	0.041488
	MSE	0.003338	0.003046	0.003238	0.003407	0.003241
	RMSE	0.057774	0.055189	0.056907	0.058368	0.056930
Overall Results	R ²	0.931066	0.930256	0.928547	0.930198	0.928725
	MAE	0.055897	0.055996	0.057096	0.056727	0.056702

	MSE	0.006784	0.006864	0.007032	0.006869	0.007014
	RMSE	0.082366	0.082848	0.083857	0.082882	0.083753

As demonstrated in Table 13, the introduced LSTM-ESCA approach attained the optimal prediction results from one-step-ahead predictions, being slightly outdone by the original LSTM-SCA approach for two and three-step-ahead forecasts. However, the proposed approach still attained the best performance in overall evaluations, outperforming all competing algorithms.

To visually demonstrate the improvements made to with the novel introduced metaheuristic convergence and distribution graphs of the objective and R^2 functions are shown in Figure 16



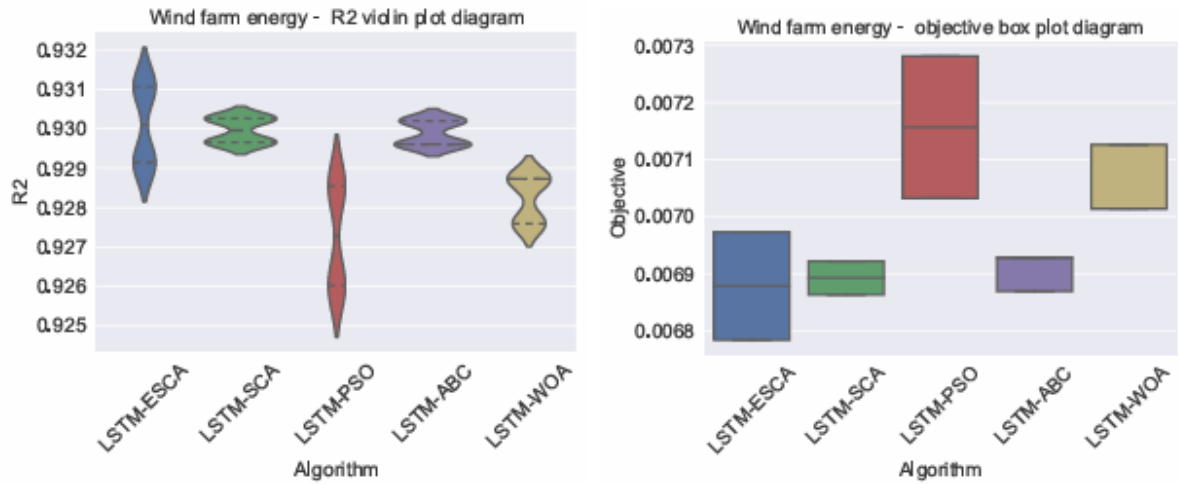


Figure 16 Objective and R^2 convergence and distribution plots.

As shown in Figure 16, significant improvements have been made over the original SCA algorithm, and a faster convergence rate has been achieved. Finally, the best parameter value selected from their respective ranges: total number of hidden network layers [1,2], neuron count in first layer [100,200], neuron count in second layer [100,200], training epochs [300,600], dropout rate [0.05,0.2] by each evaluated method are shown in Table 14.

Table 14 Selected parameters for best-performing models by beach metaheuristic.

Method	Neurons layer 2	learning rate	Training epochs	dropout	Number of layers	Neurons layer 2
LSTM-ESCA	188	0.010000	600	0.172817	2	100
LSTM-SCA	100	0.009183	600	0.192692	2	121
LSTM-PSO	118	0.010000	514	0.078699	2	100
LSTM-ABC	151	0.007756	533	0.085550	2	152
LSTM-WOA	112	0.010000	598	0.101423	2	149

5.4. Application 3: Training Logistic Regression Model by Enhanced Moth Flame Optimizer for Spam Email Classification

The growing number of e-commerce businesses has led to a rise in the quantity of commercial ads and emails. On a regular basis, consumers receive emails from unknown senders attempting to steal private information, passwords or bank account details. Spam refers to unwelcome commercial emails that have been sent in a random fashion.

Radicati stated that [70] the number of expected sent and received emails will approach 347 billion by the end of 2023. According to [71] spam emails made up 55% of global email traffic in 2017 and 2018. Total spam levels in 2015 were at 52.7%, in 2016 at 53.4%, and in 2017 at 54.6%. Spam emails annoy receivers, and consume network bandwidth, storage space, and computer resources, while potentially damaging files by inserting virus items in attached files.

Various strategies for preventing spam have been proposed. Bhowmick and Hazarika [72] classified spam email mitigation strategies into eight categories: machine-learning-based techniques, signature schemes, whitelisting, blacklisting, heuristic filters, collaborative spam filtering, challenge-response systems, and honey pots. A further classification splits the approaches into two major categories: dynamic techniques and static techniques.

Filtration techniques based on specified blacklists, and whitelists are examples of static techniques. However, these prove unsuccessful, as spammers utilize multiple email addresses. Additionally, people with compromised devices could transmit spam while ignorant of the virus. To determine if an email is spam, dynamic techniques often evaluate its message using text classification built with a statistical methodology or machine learning (ML) algorithms [73] [74].

Throughout this section, a novel spam filtering technique that benefits from logistic regression (LR) was proposed. Featuring quick classification in real-world applications, and effective methods for avoiding convergence of local minima by applying the moth flame optimization algorithm (MFO) [75]. As a nature-inspired swarm intelligence metaheuristic algorithm the MFO imitates moth flying behavior and attitude. It should be highlighted that

this work presents the first spam filter that employs the MFO as an LR training algorithm. The MFO method works very well with high-dimensional challenges [76], as it has to balance between intensification and diversification, making it a strong choice for spam detection. Using incorporation with the term frequency and inverse document frequency (*tf* and *idf*) approach, the suggested method was improved and employed with LR to minimize additional restriction for this classifier, linked with its vulnerability to identical feature weights.

Comparison between the proposed method's classification performance to that of (Gaussian naive Bayes (NB), multinomial NB, linear support vector machine (SVM), radial basis SVM, and LR classifiers). On the publicly available datasets CSDMC2010, experimental findings reveal that AVMFO-LR outperforms the other approaches. The Enron dataset was utilized to evaluate AVMFO-LR performance to that of earlier approaches, which shows that AVMFO-LR methodology surpasses state-of-the-art techniques that have been evaluated on the Enron dataset in the overall classification and false-negative (FN) rate. This research was published in [3].

5.4.1. Method

In this section, the original and the improved version of the moth flame algorithm is presented.

5.4.1.1. Original Moth Flame Algorithm

Moths are a species closely related to butterflies, that developed to fly at night by utilizing moonlight for navigation using a system known as transverse orientation. A moth flies by keeping a steady angle regarding the moon, which is a very efficient way of traveling vast distances in a straight line. However, artificial lights confuse moths, mainly due to the inefficiencies of the transverse orientation mechanism. When moths detect a man-made light, they strive to fly in a straight path by maintaining a consistent angle, often resulting in a pointless or lethal spiral flight route. This behavior is mathematically represented to create the MFO approach [75].

Solutions in MFO are represented by moths, and the factors in the equation are their positions. As a result, the moths may fly in one-dimensional or multidimensional space by modifying positional variables. Because the MFO method is a population-based approach, it is displayed in the matrix in Equation 24

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,d} \\ m_{2,1} & m_{2,2} & \dots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \dots & m_{n,d} \end{bmatrix}$$

Equation 24 MFO matrix of moths population.

where d is the number of dimensions and n is the number of moths in the search space.

There's also an array for recording the relevant objective value, as shown in Equation 25:

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_N \end{bmatrix}$$

Equation 25 Relevant objective value of moths.

where the number of moths in the search space is n .

Every moth's position vector from the M matrix is given to the fitness function, and the fitness function's return is allocated to the associated moth as its fitness value in the OM matrix.

Flame is also a control parameter in the MFO algorithm, and it can be represented as in Equation 26:

$$F = \begin{bmatrix} f_{1,1} & f_{1,2} & \dots & f_{1,d} \\ f_{2,1} & f_{2,2} & \dots & f_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n,1} & f_{n,2} & \dots & f_{n,d} \end{bmatrix}$$

Equation 26 MFO matrix of flame population.

where d is the number of diminutions and n is the number of moths in the search space. There's also an array for recording the relevant objective value, demonstrated by Equation 27:

$$OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix}$$

Equation 27 Relevant objective value of flames.

with the number of moths in the search space represented by n .

It must be highlighted that both moths and flames are possible solutions. The distinction is in how to handle and modify them on every cycle. Moths are the main search workers that travel around the solution area, while flames represent the optimum location of moths that have so far been obtained. Using this approach will guarantee that the moth's finest solution will never be lost.

The MFO model is given, as a three-tuple as in Equation 28, which corresponds to the global optimum of optimization problems:

$$MFO = (I, P, T)$$

Equation 28 MFO three different functions.

where T is the function that will result in true or false depending on the termination criterion. P is the function that moves the moth around the search space using the M as an input. I is the function that generates random moths and corresponding fitness values.

The random population function (I) can be represented as Equation 29:

$$M(i,j) = (ub(i) - lb(i)) \times rand() + lb(i)$$

Equation 29 Random population function (I).

where ub is the upper bound, and lb is the lower bound.

After generating the random population, the P function is used to move the moth and explore the search space. It can be represented as follows:

$$M_i = S(M_i, F_j)$$

Equation 30 move the moth and explore the search space function (P).

where S is the spiral function, M_i is the i -th moth, and F_j is the j -th flame. The spiral can be represented mathematically as follows:

$$S(M_i, F_j) = D_i \times e^{bt} \times \cos(2\pi t) + F_j$$

Equation 31 spiral function (S).

where t is a randomly generated number between (-1 and 1), it specifies how near the moth's following location will be to the flame ($t = -1$ indicates the nearest location to the flame, whereas $t = 1$ indicates the furthest), b is a constant number to help with shaping the spiral, and D_i represents the space between the i -th moth and the j -th flame. And D can be defined as follows:

$$D_i = |F_j - M_i|$$

Equation 32 Space between moth and flame.

Because of the spiral formula, a moth can fly around a flame rather than just in the area between them. As a result, the intensification and diversification of the solution area can be verified.

To highlight intensification, it is presumed that t is a randomly generated variable between (r and 1), with r decreasing linearly from (1 to 2) throughout the length of the cycle. It is worth noting that r is referred to as the convergence factor. Moths use this strategy to exploit their associated flames more precisely proportionate to the number of cycles. To avoid being trapped in the local minimum in Equation 30 each moth location updates its location with correspondence to only one flame. The first moth adjusts its location in relation

to the greatest flame in the array, while the final moth adjusts its location in relation to the lowest flame in the array.

An additional issue to consider is that the location update of moths with n various places in the solution area might limit the diversification of the most viable solution. To address this issue, an adjustable system is developed that decreases the number of flames during the duration of the iterations. This method is shown as follows:

$$FlameNum = round(N - I \times \frac{N - 1}{T})$$

Equation 33 Decreases the number of flames during the duration of the iterations.

whereby N the maximum number of flames is represented, I the current number of iterations, and T the maximum number of iterations.

5.4.1.2. Improved Moth Flame Optimization

By analyzing the original MFO algorithm certain deficiencies were noticed. Specifically, in early phases of execution in some runs, due to the stochastic nature, occurs that the right region of the search space is not found and consequently in later iterations the search process is unable to converge around the optimum parts of the search region. Also, it is noticed that the diversity of the population is not on a satisfactory level throughout the whole run, leading consequently to just minor solutions' improvements over subsequent iterations. On the other hand, by conducting rigid empirical analysis, it is concluded that the basic MFO's search expressions are highly efficient, yet its potential is not fully employed due to the above-mentioned drawbacks.

To address these deficiencies and to better exploit the potential of MFO, the idea of enhanced search based on average fitness (AV) is introduced. After regular solutions are updated by the basic MFO search mechanism, the average fitness (fit_{avg}) of the population is calculated. Afterward, random solution M_{rw} whose fitness is below the average ($fit_r < fit_{avg}$) is selected and discarded from the population. Such a solution is then replaced by the offspring (M_o) individual generated by applying a uniform crossover operator between the current best solution (M_{best}) and randomly chosen individual M_{rb} whose fitness is greater than the average ($fit_r > fit_{avg}$). In this implementation simple fitness representation, where

fitness is reverse proportional in the case of minimization problems, and directly proportional to the objective when maximization problems are concerned, is utilized. A simple mutation operator with a fixed mutation probability of 0.001 (p_m) is then applied to each parameter on such generated solution as in [77].

The proposed method is named average fitness MFO (AVMFO) and according to experimental findings, it exhibits better convergence, as well as exploration than the basic MFO approach. The AVMFO pseudo-code is shown in Algorithm 3.

Algorithm 3 Pseudo-code for the AVMFO.

Initialization. Produce the N random moths M inside solution area bounds using Equation 29, and measure their fitness. Initialize T as the max number of iterations. Update the number of flames using Equation 33 Do For all M of the moth produced do . For all parameters do . Update r and t. . Calculate D using Equation 32 with respect to the corresponding moth. . Update $M(i,j)$ using Equations, and Equation 31 with respect to the corresponding moth . End for . Calculate objective and fitness values . Update flames End for Calculate average fitness Discard M_{rw} individual from population Perform uniform crossover between M_{best} and M_{rb} and generate M_o Apply mutation operator and introduce to M_o into population While ($t < T$)

5.4.2. Experimental Setup

In this section, an overview of the experimental setup for evaluating the proposed AVMFO for LR training (AVMFO-LR), along with a comparative analysis with other well-known ML approaches is given. The complete workflow of the proposed AVMFO-LR classifier is given in Figure 17.

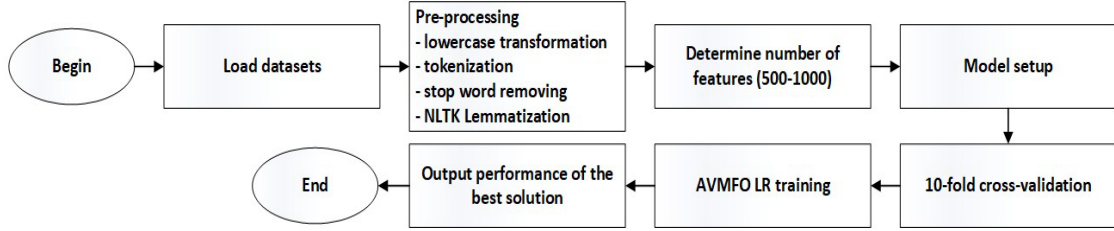


Figure 17 Block diagram of the AVMFO-LR classifier.

5.4.2.1. Dataset

Datasets used for validation include the CSDMC2010 dataset consisting of 4327 English emails, amongst which 2949 (68.15%) are valid emails, and 1378 (31.85%) are spam emails, as well as the Enron-1 dataset [78] comprised of 5172 emails, 3672 valid (71%), and 1500 spam (29%). The CSDMC2010 and Enron datasets each have (82,148) and (47,939) unique words.

5.4.2.2. Preprocessing for Data Representation

In this classification problem a series of emails $D = \{d_1, d_2, d_3, \dots, d_m\}$ were given and two categories $C = \{c_{spam}, c_{valid}\}$. The difficulty is that you need to allocate an email address $d_i \in D$ to one of the two categories.

When handling emails, the content and subject are collected, and the HTML elements are eliminated. Words are split into sets of substrings, and all characters are changed to lowercase and tokenized. With the exception of exclamation marks, stop words and punctuation are eliminated. The NLTK stemming package [79] is used on the dataset to eliminate morphological prefixes and suffixes from phrases and keep just the phrase stem. Non-informative frequent stop-words are removed, and the bag of words (BOW) paradigm

is produced regardless of word grammar or position. The BOW paradigm is a syntax (word collection) that contains every one of the unique terms throughout all documents. As shown in the following Equation 34

$$B = \{t_1, t_2, t_3, \dots, t_B\}$$

Equation 34 Every one of the unique terms throughout all documents.

where B is the term set and $|B|$ is the total number of unique terms gathered in all documents. Following the creation of the word collection, every document $d_i \in D$ is defined by a vector $d_i = [w_1, w_2, w_3, \dots, w_{|B|}]$, in which $w_1, w_2, w_3, \dots, w_{|B|}$ are the weights corresponding to the terms $t_1, t_2, t_3, \dots, t_{|B|}$.

The weights are calculated with tf and idf , and can be calculated with the following Equation 35

$$tf - idf(t_j, d_i) \leq w_j, d_i \geq tf(t_j, d_i) \times \log \frac{|D|}{|D_{t_j}|}$$

Equation 35 Calculation of the weights.

when $tf(t_j, d_i)$ indicates the overall number of words t_j in document d_i , $|D|$ represents the overall number of documents, and $|D_{t_j}|$ indicates the overall number of documents containing the term t_j .

$$\phi(\vec{d}_i^l, c_j) = \begin{cases} 1, & \text{if } \vec{d}_i^l \in c_j \\ 0, & \text{otherwise} \end{cases}$$

Equation 36 Binary function indicating the membership of a document vector in a category.

5.4.3. Results and Discussion

The classification performance of Gaussian NB, Multinomial NB, linear SVM, radial basis SVM, LR, and the proposed model on the CSDMC2010 dataset was evaluated via comparing, in a similar way as it was proposed in [80]. It is important to mention that the authors have implemented independently standard machine learning methods, ML and ABC-LR as well, to verify the results from [80]. Moreover, in this research, the termination condition for all tested methods was set according to the total amount of FFEs (fitness function evaluations) used, and to provide similar conditions as in [80], the number of evaluations was set to 160000 for all conducted experiments.

Furthermore, the classification accuracy of the model on the Enron dataset was evaluated by comparing it to other techniques from earlier research, and to avoid the influence of the arranging and dividing mechanism on the data, tenfold cross-validation was used in all experimental studies. As a consequence, ten classification outcomes were achieved, and the mean of these outcomes is measured as the research's ultimate classification results.

5.4.4. Comparison of Classifiers

Table 15 displays the highest and lowest classification performance achieved by six classifiers on the CSDMC2010 dataset with vector sizes of 500 and 1000, and the best-achieved results are marked in bold. According to Table 15, the LR classifier with the AVMFO algorithm made the greatest outcome, with the highest obtained accuracy of 98.81 % on the CSDMC2010 dataset with a feature size of 1000, and 98.67 % with a feature size of 500.

Table 15 The highest and lowest classification performance comparison on the CSDMC2010 dataset.

<i>Classification techniques</i>	<i>Feature vector size = 500</i>		<i>Feature vector size = 1000</i>	
	Highest	Lowest	Highest	Lowest
AVMFO-LR	98.67%	94.65%	98.81%	96.03%
MFO-LR	98.45%	76.52%	98.52%	77.68%
Gradient descent with LR [80]	98.25%	73.82%	98.54%	66.74%
SVM with RBF kernel [80]	98.57%	68.22%	98.61%	68.23%
Linear SVM [80]	98.11%	92.32%	98.41%	94.11%
Multinomial NB [80]	90.92%	88.92%	94.52%	92.03%
Gaussian NB [80]	93.73%	93.73%	95.94%	95.94%

Table 16 displays the detection accuracy of the spam detection. The top spam detection performance was presented and associated FN and FP rates, as well as the mean duration in seconds, in this subsection. In Table 16, the STD was displayed for every outcome acquired using 10-fold cross-validation.

In classification performance and FN rate, the findings show that AVMFOLR beat the other methodologies on the Enron dataset. On the Enron dataset, AVMFO-LR likewise generated much higher FN rates compared to the rest of the approaches. When FP rates were compared, AVMFO-LR outperformed all other methods on the Enron dataset excluding the (FDA + NB, random forest, and AdaBoost methods). Nevertheless, due to their relatively weak detection effectiveness, these three approaches should not be used for spam detection (FN rate). The basic MFO-LR approach also obtained decent performances, finishing in third place, behind the ABC-LR approach.

It is important to note that all demonstrated methods were implemented and tested in order to independently validate the results for this research paper.

Table 16 AVMFO-LR's performance and comparison on the Enron 1 dataset.

Methodology	Performance — STD	FN — STD	FP — STD
AVMFO-LR	98.97% — 0.85	0.0005 — 0.0010	0.0209 — 0.0143
MFO-LR	98.83% — 0.88	0.0007 — 0.0013	0.0213 — 0.0184
ABC-LR [80]	98.92% — 0.86	0.0006 — 0.0012	0.0212 — 0.0178
DBB-RDNN-Rel [81]	98.74% — 0.58	0.0016 — 0.0019	0.0213 — 0.0102
CNN [81]	97.49% — 0.88	0.0346 — 0.0194	0.0216 — 0.0078
KNN [81]	91.35% — 1.36	0.0377 — 0.0156	0.1064 — 0.0178
MLP [81]	96.31% — 2.83	0.0502 — 0.0824	0.0315 — 0.0359
AIRS2Parallel [81]	71.36% — 7.65	0.1220 — 0.1679	0.3535 — 0.1519
LR [81]	94.56% — 1.03	0.0664 — 0.0202	0.0498 — 0.0126
AdaBoost [81]	78.76% — 1.15	0.6838 — 0.0340	0.0200 — 0.0800
Random forest [81]	98.03% — 0.58	0.0179 — 0.0123	0.0202 — 0.0074
Voting [81]	97.20% — 1.06	0.0247 — 0.0118	0.0294 — 0.0130
IL-C4.5 [81]	93.35% — 1.29	0.0608 — 0.0207	0.0688 — 0.0159
FDA + SVM [81]	96.64% — 0.85	0.0462 — 0.0175	0.0285 — 0.0093

FDA + NB [81]	91.30% — 1.21	0.1211 — 0.0162	0.0044 — 0.0059
MDL [81]	95.67% — 1.13	0.0107 — 0.0105	0.0566 — 0.0165

Additionally, with a higher number of features, accuracy increases, however, it also introduces some penalties because more LR coefficients need to be determined. The time complexity of the process does not scale linearly. During testing, it was determined that when the feature size is increased from 500 to 1000, the metaheuristics take approximately 30% more time to execute.

Methodology	Performance
AVMFO-LR	98.97%
MFO-LR	98.83%
ABC-LR [80]	98.92%
DBB-RDNN-ReL [81]	98.74%
Boosted NB + SVM [82]	95.60%
Incremental SVM [83]	96.86%
Natural language toolkit NB [84]	94.70%
k means + SVM [85]	97.35%
Relief + NB [86]	96.30%
RF [87]	96.39%
Enhanced genetic programming [88]	94.10%
Bagged RF [89]	97.75%
Minimum description length [90]	95.56%
Distinguishing feature selector [91]	94.35%
Multivariate Bernoulli NB [92]	94.79%
AIS [93]	90.00%
Deep belief networks [94]	97.43%

5.5. Application 4: Seizure Detection Via Time Series Classification

Using Modified Metaheuristic Optimized Recurrent Networks

Seizure disorder, also known as epilepsy, is a neurological condition in which seizures are caused by sudden, hectic electrical activity in the brain [95]. The effects of these electrical discharges on the individual differ based on which part of the brain is affected, however, general symptoms include tremors and jerking movements of the body, confusion, loss of consciousness, heightened negative emotions, and stiff musculature. Rarer symptoms include behavioral changes, hallucinations in the forms of light auras or tastes and smells, and brief loss of vision. Many dangers are associated with these seizures as they are hard to predict, especially in the beginning stages, and may cause a person to hurt themselves by falling or losing consciousness in risky situations. Another danger is damage to the brain caused by the electrical discharges. The more epileptic seizures a person has the higher the chance of significant permanent damage to the brain. Last, but not least, epileptic disorder is often followed by psychological difficulties such as anxiety, depression, and substance abuse connected to a lower quality of life due to the disorder.

Epilepsy has a slightly varying presentation depending on the area of the brain affected, the main distinct groups being Temporal lobe epilepsy, Frontal lobe epilepsy, Occipital lobe epilepsy, and a generalized version that has no distinct localization. Determining the area most affected allows for better-suited treatment and a higher understanding of the potential development of the disorder. The cause of the disorder, though well researched, is not yet definitely determined. However, its beginning in many patients has been linked to the following factors: genetics that make a person more vulnerable to the disease, head trauma, prenatal injuries, infections, developmental conditions, and co-morbid conditions.

While there is no cure for the disorder, medication that lowers the probability of seizures, known as anti-epileptic medication, is often prescribed as treatment. Considering the risks to immediate safety, brain function, and long-term mental health brought on by seizures, early detection of the disorder is crucial. If the disorder is diagnosed early in its development, treatment via medication and a lifestyle change may maintain a high quality of life despite the illness.

Usual diagnosis includes the use of neurodiagnostic tools such as Electroencephalography (EEG), patient medical and family history, and information provided by witnesses to the seizure. EEG, as the crucial confirmational tool, is a technique utilizing electrodes placed along the scalp that record an electrogram of the brain's spontaneous activity [96]. The benefit of such an approach is its non-invasive nature. While there are versions that are surgically implanted into the brain, most often the basic EEG can be done with minimal preparation and discomfort. Besides diagnosis made with the help of EEG scans showing abnormal neural activity, EEG also helps localize the seizures in the brain and track the success of the treatment [97].

Yet, even with EEG, epilepsy diagnosis can present a challenge. This is due to both external confounding factors of a poorly controlled environment and human error. Patients may have a hard time sitting still while the data is being recorded, and even the slightest movements contribute to noise in the recorded data. This is particularly hard to distinguish in patients actively having seizure-induced spasms during the recording. Additionally, the interpretation of the results varies based on the knowledge, expertise, and experience of the medical professional [98].

Artificial intelligence emerges as a promising tool to advance epilepsy diagnosis through EEG results analysis [99]. Its ability to learn on its own and find patterns in data elusive to the human eye, as well as its capability to take into account enormous amounts of data simultaneously makes it a great addition to the tools available for medical diagnosis. Previous research in the field confirms the value of AI for medical diagnostics for various diseases [100]. Due to the nature of brain activity as a temporally dependant variable, an approach based on time series classification would be appropriate. Considering that Recurrent neural networks (RNN) have been specifically developed to handle time series classification, they will be utilized in this work [101]. A model based on RNNs will be applied to a publicly available dataset containing EEG scans of epilepsy patients, active seizures, and healthy individuals, with the task of detecting anomalous brain behavior.

With the higher complexity models needed for such demanding tasks as medical diagnosis comes an inherent challenge: the hyperparameter selection also grows in complexity, requiring meticulous fine-tuning to enable an accurate model. Considering the choice of hyperparameters is an NP-hard problem, traditional techniques of trial and error

are costly in time and resources. A more effective approach is using metaheuristic algorithms due to their capacity to tackle NP-hard problems [102]. For this purpose, these algorithms will be included in the work with the goal of choosing the best possible hyperparameters and thus accuracy of the model. This research was published in [4].

5.5.1. Methods

This section describes the original optimizer that serves as an inspiration for the modified version of the algorithm. The observed drawbacks of the original are also described. This is then followed by a description of a proposed mechanism for overcoming these shortcomings followed by the introduction of the modified version of the algorithm that is capable of dealing with some of the observed challenges observed with the original algorithm.

5.5.1.1. Original Sinh Cosh Optimiser

Newly developed, the Sinh cosh optimiser (SCHO) draws inspiration from mathematical principles [103]. The beginning stages are characterized by a created population with high randomness as visible in Equation 37.

$$A = \begin{bmatrix} a_{1,1} \dots a_{1,j} \dots a_{1,D} \\ a_{1,2} \dots a_{2,j} \dots a_{2,D} \\ \vdots \\ a_{N,1} \dots a_{N,j} \dots a_{N,D} \end{bmatrix}$$

Equation 37 Random population generated.

in the equation, a population of agents is defined by P . The dimensional space of the solution and the number of agents is marked with D , and N , respectively. The position of each agent's A_i, j is defined by the following equation:

$$a = rnd(N, D) \times (ub, lb) + lb$$

Equation 38 Position of agents.

in which rnd stands for a random number, and ub and lb stand for the upper and lower boundaries of the search space.

After the initialization, a harmony between exploration and exploitation needs to be made, in order to direct the agents to the preferred areas of the search space. The equation pictured next governs the balance and the two strategies that divide the exploration.

$$S = \text{floor}\left(\frac{T}{ct}\right)$$

Equation 39 Balance between exploration and exploitation.

here the T stands for the maximum number of allotted iterations while ct marks the control coefficient selected to be 3.6 based on previous research.

While the exploration phase is going on, the agents are updated according to Equation 40:

$$A_{(i,j)}^{t+1} = \begin{cases} A_{best}^{(j)} + r_1 \times W_1 \times A_{(i,j)}^t & r_2 > 0.5 \\ A_{best}^{(j)} - r_1 \times W_1 \times A_{(i,j)}^t & r_2 < 0.5 \end{cases}$$

Equation 40 Agents update during exploration phase.

in this context, t represents the iteration number, $A_{t+1}(i, j)$ characterizes the j -th agent and i -th agents, and $A_{best}^{(j)}$ indicates the best agent in the j dimension. Random values within the range of $[0, 1]$ are chosen for r_1 and r_2 . The parameter W_1 represents a weighted coefficient for the particular agent and can be determined accordingly using the equation:

$$W_1 = r_3 \times b_1 \times (\cosh r_4 + \mu \times \sinh r_4 - 1)$$

Equation 41 Weighted coefficient for the particular agent.

the b_1 value is progressively diminished during the iteration, while r_3 and r_4 present randomly chosen values from between $[0, 1]$. A parameter marking sensitivity is also included as μ . The other strategy applied in exploration uses Equation 42

$$A_{(i,j)}^{t+1} = \begin{cases} A_{best}^{(j)} + |\epsilon \times W_2 \times A_{best}^{(j)} - A_{(i,j)}^{(t)}| & r_5 > 0.5 \\ A_{best}^{(j)} - |\epsilon \times W_2 \times A_{best}^{(j)} - A_{(i,j)}^{(t)}| & r_5 < 0.5 \end{cases}$$

Equation 42 The other strategy applied in exploration.

here, e is stable at 0.003 per the original work advice. The W_2 weight coefficient is calculated using:

$$W_2 = r_6 \times b_2$$

Equation 43 W_2 weight coefficient.

in which r_6 marks an arbitrary value between 0 and 1, while b_2 stands for a gradually decreasing value.

Another significant phase in the optimization process is exploitation, where agents concentrate on promising regions within the search space, making more precise movements toward optima. The metaheuristic employs two strategies during this stage. The initial phase involves the utilization of Equation 44

$$A_{(i,j)}^{t+1} = \begin{cases} A_{best}^{(j)} + r_7 \times W_3 \times A_{(i,j)}^t & r_8 > 0.5 \\ A_{best}^{(j)} - r_7 \times W_3 \times A_{(i,j)}^t & r_8 < 0.5 \end{cases}$$

Equation 44 Agents update during exploitation phase.

here the values of r_7 and r_8 get selected from a 0 – 1 range, while W_3 is calculated according to:

$$W_3 = r_9 \times b_1 \times (\cosh r_{10} + \mu \times \sinh r_{10})$$

Equation 45 W_3 weight coefficient.

in this context, r_9 and r_{10} denote randomly chosen values within the range of [0, 1]. The second approach follows Equation 46

$$A_{(i,j)}^{t+1} = A_{(i,j)}^t + r_{11} \times \frac{\sinh r_{12}}{\cosh r_{12}} |W_2 \times A_{best}^t - A_{i,j}^t|$$

Equation 46 The other strategy applied in exploitation.

where r_{11} and r_{12} mark once again randomly selected values in the [0,1] range.

5.5.1.2. Modified SCHO Algorithm

The CEC standard evolution function testing has shown that the original SCHO algorithm has done well when contrasted against other contemporary optimizers [104]. However, there are still aspects of it that can benefit from continued development. In order to promote this development two new mechanisms are implemented.

The resulting algorithm is named the modified SCHO (MSCHO). The first innovation stems from the classic ABC algorithm [105]. If exhausted solutions fail to show improvement, they are substituted with newly made ones. Within the confines of the restricted experimental iterations, solutions that do not contribute are replaced after two iterations.

This approach has demonstrated its efficacy in improving exploration. The next new mechanism is called quasi-reflective learning (QRL) [106]. The aim of this technique is to create extra solutions, thereby augmenting the exploration. Moreover, during the initial stages of the algorithm, this mechanism is employed to generate potential solutions, with the quasi-reflected component 'z' of the solution A location being determined as:

$$A_z^{qr} = rad\left(\frac{lb_z + ub_z}{2}, a_z\right)$$

Equation 47 quasi-reflective learning (QRL).

lb and *ub* mark the search spaces' lower and upper bounds, while *rad* stands for a random value in a specific interval. The pseudocode of the optimizer may be seen in Algorithm 4.

Algorithm 4 Pseudocode for the described MSCHO algorithm.

Initialize starting parameter values

Populate and set up initial conditions using the QRL mechanism

While *T* is greater than *t* **Do**

 Employ a suitable SCHO strategy to update the positions of agents based on the current time *t*

 Evaluate agent fitness using an objective function

For each individual p in the <i>Population</i> Do . If no improvement has been observed for 2 consecutive iterations by p then . . Generate a fresh solution using the QRL mechanism and substitute it for p . End if End for End while Return The optimal solution found within the population

5.5.2. Experiment Setup

To validate the proposed approach a publicly available real-world EEG dataset is used [107]. The dataset contains clinical EEG recordings of a healthy patient and a patient going through an active seizure. The dataset is formulated as a time series and combined into a unified dataset. The data was labeled according to patient outcomes 0 for normal brain activity and 1 for seizure. The dataset was split into training data (consisting of the initial 70% of samples) while the remaining data remained re-served for testing (20%) and validation (10%). Each time-series input consists of 15 individual samples represented as gals.

Due to the high dependence of RNN on parameter settings are optimized by a set of state-of-the-art metaheuristics alongside the introduced optimizer. The algorithms included are the original recently proposed SCHO [103], GA [57], PSO [32], FA [38] and SCA [67]. Agent parameters represent encoded values of RNN hyperparameters such as learning rate, dropout, and number of training epochs with respective ranges of [0.0001, 0.01], [0.05, 0.2], and [30, 60]. Further architecture parameters were also tuned including the number of layers, and neurons per layer from respective ranges of [1, 2] and [15, 5]. All algorithm implementations are done independently in Python specifically for the needs of this study. Additional support libraries are used to support the research including pandas, numpy, and TensorFlow. Optimizations are carried out under identical conditions with six agents in the population and eight iterations allocated to improve solutions. To account for randomness associated with this class of optimizer, optimizations are carried out over 30 indented executions to provide a valid comparison.

As the detection of epilepsy-induced abnormal signals from time series samples can be considered a classification task, several standard metrics are included in the evaluation process. The utilized metrics include precision, recall, f1-score, and accuracy. The objective function used to guide the optimization process is error rate determined as:

$$\text{Error rate} = 1 - \text{accuracy}$$

Equation 48 Error rate.

An additional metric is included due to the observed imbalance in the analyzed data. The metric often used by researchers for these challenges is Cohen's kappa which can be calculated as:

$$\kappa = \frac{V_o - V_e}{1 - V_e}$$

Equation 49 Cohen's kappa.

where V_o represents the observed and V_e the expected values for a given prediction.

5.5.3. Experiment Outcomes

Simulation outcomes in terms of overall objective function values for the best, worst, mean and median executions are provided in Table 17 with var and std provided for algorithm stability assessment. Similarly, outcomes in terms of the objective function are presented in Table 18.

Table 17 Overall objective function outcomes over 30 experimental executions.

Method	Best	Worst	Mean	Median	Std	Var
RNN-MSCHO	0.001231	0.001846	0.001600	0.001846	0.000301	9.09E-08
RNN-SCHO	0.001231	0.003077	0.001969	0.001846	0.000718	5.15E-07
RNN-GA	0.001231	0.001846	0.001477	0.001231	0.000301	9.09E-08
RNN-PSO	0.001231	0.001846	0.001354	0.001231	0.000246	6.06E-08
RNN-FA	0.001231	0.004308	0.002462	0.002462	0.001030	1.06E-06

RNN-SCA	0.001231	0.003077	0.002092	0.001846	0.000628	3.94E-07
---------	----------	----------	----------	----------	----------	----------

Table 18 Overall indicator function outcomes over 30 experimental executions.

Method	Best	Worst	Mean	Median	Std	Var
RNN-MSCHO	0.996072	0.994114	0.994892	0.994114	0.000963	9.28E-07
RNN-SCHO	0.996072	0.990214	0.993720	0.994100	0.002280	5.20E-06
RNN-GA	0.996072	0.994114	0.995286	0.996072	0.000962	9.26E-07
RNN-PSO	0.996072	0.994100	0.995677	0.996072	0.000789	6.22E-07
RNN-FA	0.996072	0.986201	0.992128	0.992143	0.003304	1.09E-05
RNN-SCA	0.996072	0.990191	0.993324	0.994100	0.001999	4.00E-06

As demonstrated in Table 17 and Table 18 metaheuristics attained admirable performance when tackling the challenge of epilepsy detection with an algorithm matching the best performance across both metrics. However, the introduced algorithms attained the highest stability on both objective and indicator functions.

Graphical comparisons between competing algorithms in terms of stability are provided in Figure 18.

As demonstrated in Figure 18 the introduced algorithm demonstrates an improved level of stability over the base algorithm in both indicator and objective outcomes.

Convergence comparisons between optimizers are shown in terms of objective and indicator functions in Figure 19.

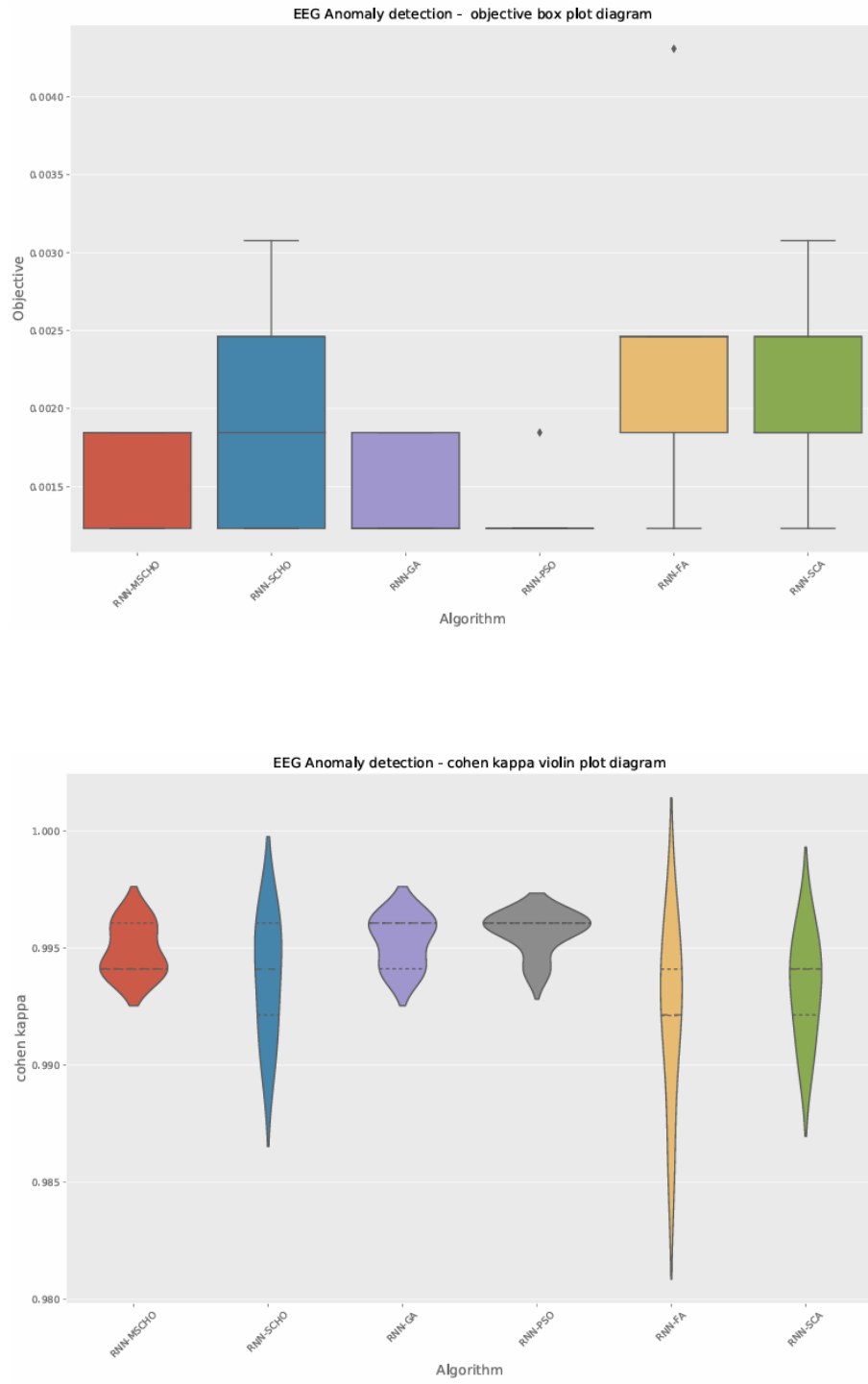


Figure 18 Objective and indicator function distributions for over 30 experimental executions.

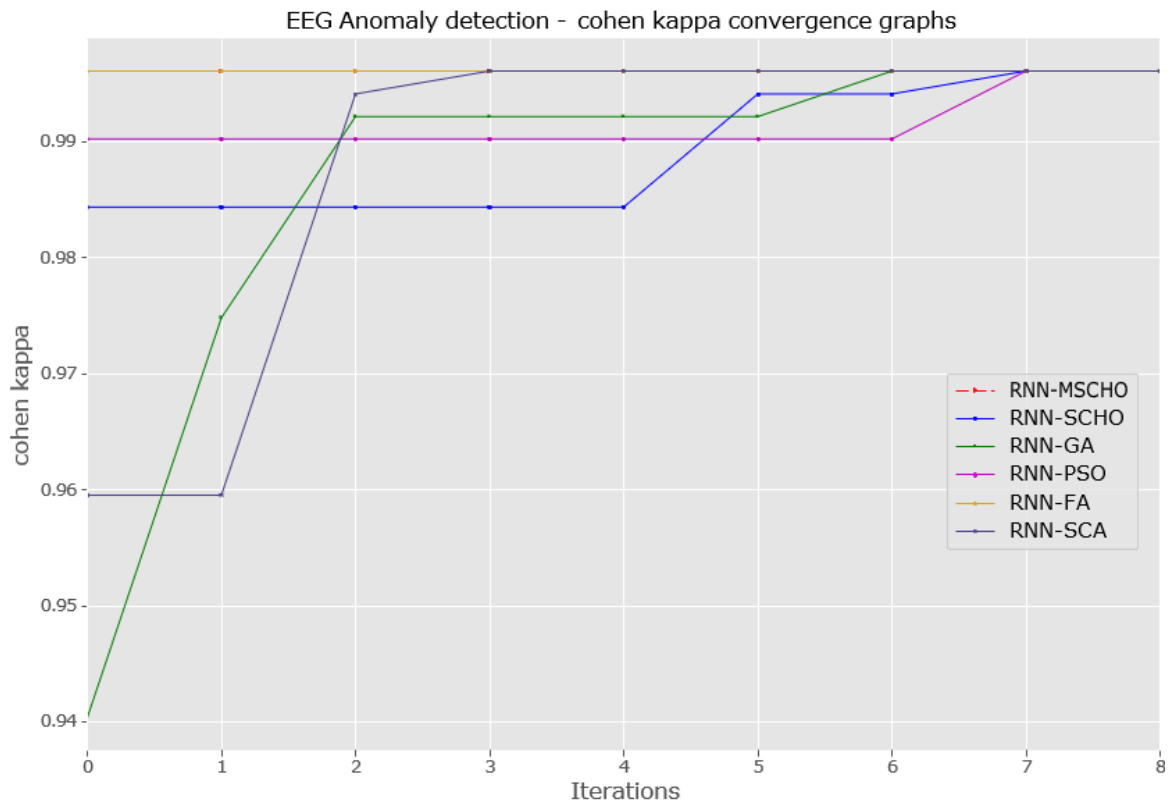
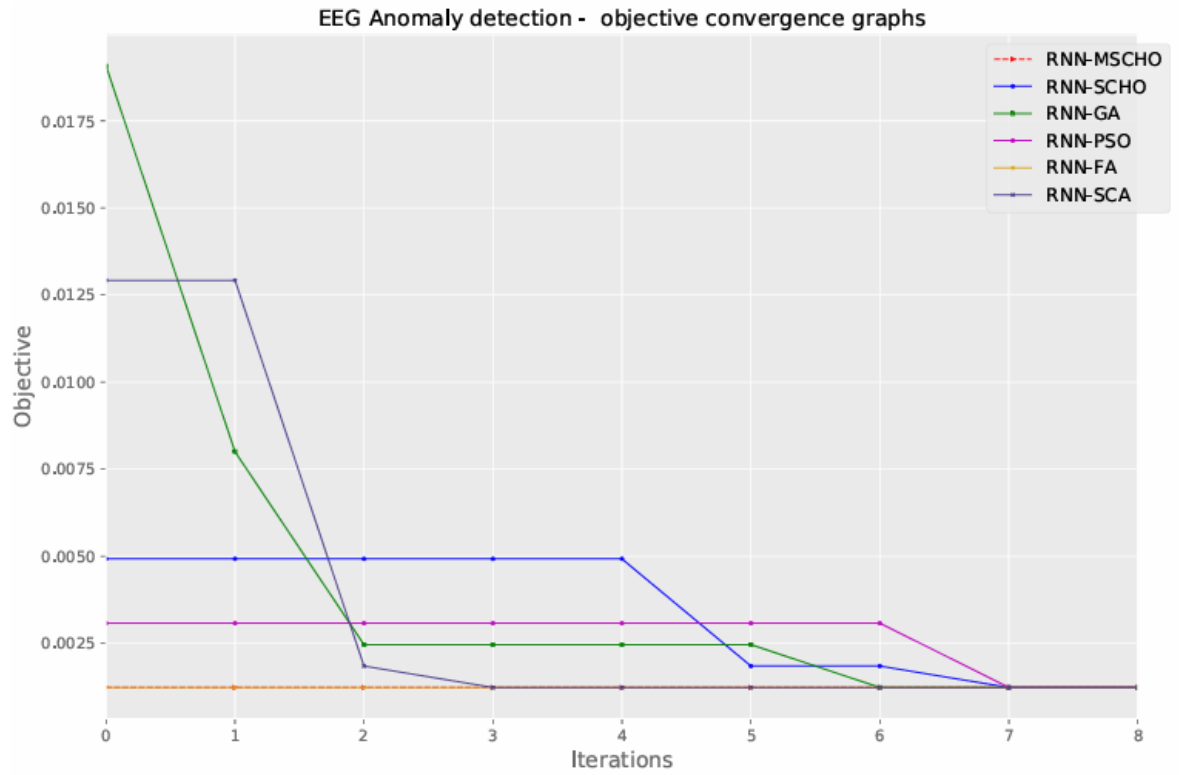


Figure 19 Objective and indicator function convergence plots for the best-performing models.

Detailed comparisons between the best-performing models constructed by each algorithm are provided in Table 19.

Table 19 Detailed metrics for the best-performing models constructed by each optimizer.

Method	Metric	Normal	Seizure	Accuracy	macro avg	weighted avg
RNN-MSCHO	precision	0.996835	0.999236	0.998769	0.998036	0.998769
	recall	1.000000	0.999236	0.998769	0.998036	0.998769
	f1-score	0.996835	0.999246	0.998769	0.998036	0.998769
RNN-SCHO	precision	0.195303	1.000000	0.198769	0.597651	0.843517
	recall	1.000000	0.005348	0.198769	0.502674	0.198769
	f1-score	0.326784	0.010638	0.198769	0.168711	0.072116
RNN-GA	precision	0.996835	0.999236	0.998769	0.998036	0.998769
	recall	0.996835	0.999236	0.998769	0.998036	0.998769
	f1-score	0.996835	0.999236	0.998769	0.998036	0.998769
RNN-PSO	precision	0.996835	0.999236	0.998769	0.998036	0.998769
	recall	0.996835	0.999236	0.998769	0.998036	0.998769
	f1-score	0.996835	0.999236	0.998769	0.998036	0.998769
RNN-FA	precision	0.195545	1.000000	0.200000	0.597772	0.843564
	recall	1.000000	0.006875	0.200000	0.503438	0.200000
	f1-score	0.327122	0.013657	0.200000	0.170390	0.074614
RNN-SCA	precision	0.996835	0.999236	0.998769	0.998036	0.998769
	recall	0.996835	0.999236	0.998769	0.998036	0.998769
	f1-score	0.996835	0.999236	0.998769	0.998036	0.998769
	support	316	1309			

As can be seen in Table 19 the introduced algorithm shows decent performance across multiple metrics, managing to tackle imbalanced data with high precision for both normal as well as seizure detection, while the original algorithm struggles with seizure detection, attaining a very low score for detection in terms of precision.

A confusion matrix of the best-performing model constructed by the introduced optimizer is shown in Figure 20.

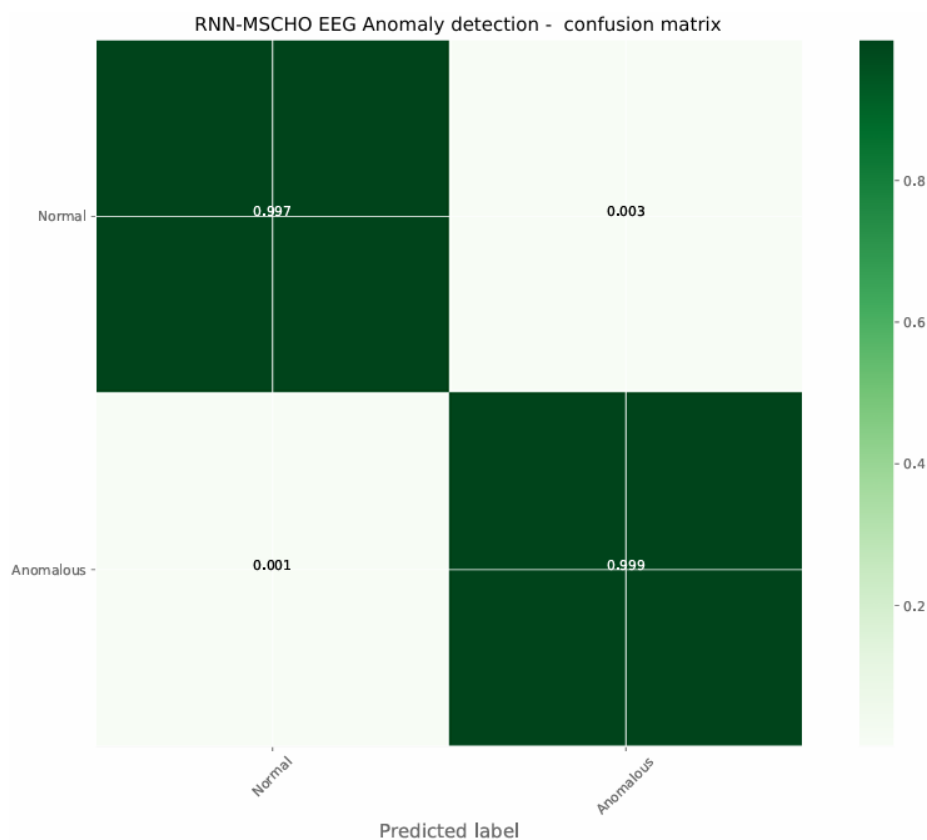


Figure 20 Best constructed model confusion matrix.

To facilitate experimental repeatability, hyperparameter selections made by each optimizer are provided in Table 20.

Table 20 Metaheuristic optimized parameters for best-performing models.

Method	Learning Rate	Dropout	Epochs	Layers	Neurons L1	Neurons L2
RNN-MSCHO	0.008964	0.132481	48	1	12	N/a
RNN-SCHO	0.010000	0.200000	60	2	12	5
RNN-GA	0.006064	0.050000	59	1	8	N/a
RNN-PSO	0.009642	0.150645	54	1	13	N/a
RNN-FA	0.009699	0.177742	53	2	14	12
RNN-SCA	0.006965	0.150773	60	1	15	N/a

5.6. Application 5: Long Short-Term Memory Tuning by Enhanced Harris Hawks Optimization Algorithm for Crude Oil Price Forecasting

The susceptibility of even the most robust global economies to spikes in oil prices has been made more apparent due to major events in recent years. It has become apparent that the crude oil supply and demand-driven economic model show great significance for evaluating development. Investment decisions, both on an individual and corporate level are influenced by changes in crude oil prices, significantly affecting economic development. The inherent investment risks associated with crude oil volatility can be somewhat alleviated through price forecasting.

Accurate price forecasting is a challenging but potentially very lucrative task. It is important to consider all relevant factors and predictors that influence prices. Additionally, crude oil prices form a highly volatile nonlinear time-series. The ever-changing environment, makes the use of traditional methods difficult, as accuracy becomes insufficient. To attain a more refined level of accuracy, advanced techniques relying on modern technologies are needed.

Price forecasting is often formulated as a time-series forecasting problem, where the outcomes of an evaluation don't depend only on the input value but on their order as well. Time-series encapsulate complex and non-linear relations in input data, and the interpretation process is challenging, often relying on patterns and trends in the observed data. It is important to note that pronounced patterns are often easy to notice, and less pronounced trends are more difficult to notice. A similar challenge is often tackled in signal processing when tackling signal decomposition. Accordingly, various methods have been developed to adequately decompose signals into sub-components. By treating time-series data as signals, decomposition techniques are utilized to improve prediction results. A notably interesting method for signal decomposing is the use of variational mode decomposition (VMD) [65], which improves on the preceding algorithm with a solid mathematical foundation. By applying VMD to a time-series, the dataset is transformed into a series of component data with a simpler shape, making predicting on an individual level easier as opposed to working with one highly complex single signal.

As an emerging and very promising field of computer science, artificial intelligence (AI) has been used to tackle multiple complex nonlinear problems with promising results [102] [59] [76]. By mathematically modeling learning mechanisms observed in humans, AI is capable of adapting to changing environments. Methods have been developed for replicating learning mechanisms to address complex tasks and improve how they are tackled through machine learning (ML), often considered a sub-field of artificial intelligence. Various models have been developed to deal with domain-specific problems using ML [81]. An interesting emerging approach for tackling time-series prediction problems is through the use of a specialized type of neural network known as a long-short-term memory (LSTM) network [66]. This type of network is capable of storing and acting on information stored within the network, making it a promising choice for doing the task at hand.

Great efforts have gone into optimization, through the development of various algorithms and hybrid methods. A notably well-performing group of meta-heuristic algorithms is swarm intelligence, due to their ability to tackle complex and even NP-hard problems considered impossible to solve through the use of traditional deterministic approaches. Various algorithms have been created to address specific problems with certain strengths and weaknesses. A suitable algorithm can be selected through a comparative analysis such as in [108]. It is also important to note that according to the no-free lunch theorem [109], no single approach works best for all cases. Some notable examples include algorithms inspired by nature such as the gray wolf optimizer GWO [110], firefly algorithm FA [38], and whale optimization algorithm WOA [40]. however, approaches inspired by more abstract concepts exist such as the arithmetic optimization algorithm AOA [111], and the sine cosine algorithm [67].

It is important to note that to attain the desired performance from applied ML algorithms adequate parameter settings are required. To be able to address various tasks, ML algorithms need to attain certain flexibility. This is done through the introduction of control parameters, otherwise known as hyperparameters, that dictate the internal functioning of certain algorithm mechanisms. The process of selecting these values has traditionally been done by hand, through a process of trial and error. However, with emerging algorithms proposing an ever-increasing number of control parameters, newer methods for selection are needed. The process of determining optimal control parameters can be formulated as an optimization problem. Hence an optimization algorithm is tasked

with selecting optimal values in a process known as hyperparameter tuning. Several proposals for selecting parameters can be found in [112].

A notable gap in modern research is present considering the use of time-series prediction and price forecasting through the application of ML techniques. Additionally, research considering data decomposition techniques applied to the same task is equally scarce. Researchers have proposed the use of VMD combined with kernel extreme learning machines and demonstrated promising results [113]. Their works serve as inspiration for conducting this work. This research was published in [5].

5.6.1. Methods

This section gives an overview of the Harris Hawks optimization algorithm during the different phases with the drawbacks of the algorithm and our proposed changes to improve its performance.

5.6.1.1. Harris Hawks Optimization Algorithm

Feeding strategies are a popular inspiration for optimization algorithms, and a notably well-performing population-based optimization algorithm inspired by specialized feeding techniques of a species of hawk is the HHO algorithm [114]. This algorithm can be considered a global optimization algorithm as it supports both exploitation as well as exploration.

5.6.1.1.1. Exploration Phase

Hawks rely on their highly developed sense of sight during hunting. However, prey isn't necessarily always imminently easily observable. Hawks, hence, assume a random position and visually search for prey. The algorithm mimics this behavior with two strategies, with condition q deciding as described in Equation 50

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB - r_4(UB - LB)) & q < 0.5 \end{cases}$$

Equation 50 position of a search agent in the upcoming iteration.

where $X(t+1)$ represents the position of a search agent in the upcoming iteration, X_{rabbit} denotes the position of prey, and $X(t)$ is the current agent position. Additionally, r_1, r_2, r_3, r_4 and 1 are random values in the range (0,1) selected in each iteration, LB and UB represent lower and upper bounds respectively, X_r and (t) represent a random agent selected for the population, while the average position of the entire population is represented as X_m determined according to Equation 51

$$X_m(t) = \frac{1}{N} \sum_{i=0}^N X_i(t)$$

Equation 51 average position of the entire population.

5.6.1.1.2. Phase Transition

It is important to note that the HHO smoothly transitions from exploitation to exploration strategies based on the energy of the simulated prey. Prey energy is gradually decreased during an escape according to Equation 52

$$E = 2E_0(1 - \frac{t}{T})$$

Equation 52 energy of a prey.

with E being the prey escape energy, T representing maximum iterations, and E_0 initial energy. It is important to note that in the algorithm implementation, E_0 randomly shifts in the range $(-1,1)$ in each successive iteration. Prey energy plays an important role in the algorithm, as when $|E| \geq 1$ the exploration phase will be carried out, while when $|E| < 1$ exploitation is performed.

5.6.1.1.3. Exploitation Phase

During the exploitation phase, agents will attempt to capture prey based on the location detected in the preview iteration. Prey will attempt to avoid capture, using one of four possible strategies.

The first strategy is called soft besiege which is applied when $r \geq 0.5$ and $|E| \geq 0.5$. Agents circularly approach prey softly, slowly exhausting it, then perform a surprise pounce. Mathematically this behavior is modeled according to Equation 53

$$X(t + 1) = \Delta X(t) - E|J X_{rabbit}(t) - X(t)|$$

$$\Delta X(t) = X_{rabbit}(t) - X(t)$$

Equation 53 Soft besiege.

in which $\Delta X(t)$ represents the distance between the rabbit position in the current iteration t , random jump strength is represented by J and calculated according to $J = 2(1 - r_5)$, with r_5 being a random value from a range $(1,0)$.

The second strategy is hard besiege in cases where $r \geq 0.5$ and $|E| < 0.5$. In this case, hawks will perform surprise pounces. During this strategy, the search is performed according to Equation 54

$$X(t + 1) = X_{rabbit}(t) - E|\Delta X(t)|$$

Equation 54 hard besiege.

The third strategy combines soft besiege with progressive rapid dives in cases where $|E| \geq 0.5$ and $r < 0.5$. In this case, the prey has enough energy to escape so an advanced strategy is used. To model, this behavior leapfrog movements are used and the levy flight (LF) is incorporated for the zigzag patterns taken by prey. To select the best possible route agents apply Equation 55

$$Y = X_{rabbit}(t) - E|J X_{rabbit}(t) - X(t)|$$

Equation 55 soft besiege with progressive rapid dives.

Following this, agents assess the possible results of taking this direction to previous attack patterns to assess the chances of success. The pattern of prey motion is assumed to be according to an LF-based pattern that follows Equation 56

$$Z = Y + X \times LF(D)$$

Equation 56 Assess the possible results of taking this direction to previous attack patterns.

where D is the dimension of the problem and S is a random vector. The levy flight function can be determined using Equation 57

$$LF(x) = 0.001 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}$$

$$\sigma = \left(\frac{\Gamma(1 + \beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{\frac{1}{\beta}}$$

Equation 57 Levy flight function.

in which β represents a default constraint set to 1.5, and u and v represent random values in the range of (0,1).

With all this in mind, the strategy taken by agents in this phase can be summered in Equation 58

$$X(t+1) = \begin{cases} Y & \text{if } F(Y) < F(X(t)) \\ Z & \text{if } F(Z) < F(X(t)) \end{cases}$$

Equation 58 Final strategy for updating the positions of hawks in the soft besiege phase.

with Z and Y calculated according to Equation 55 and Equation 56 respectively.

The final strategy is taken in cases where $|E| < 0.5$ and $r < 0.5$ is hard to besiege with progressive rapid dives. This strategy is assumed once the prey is exhausted and close enough to be pounced on by surprise. The strategy is similar to the previous, however, agents try to reduce the distances between their average location and prey. This process follows rules set in Equation 58. However, parameter Y is determined using Equation 59 in this step.

$$Y = Y_{rabbit}(t) - E|JX_{rabbit}(t) - X_m(t)|$$

Equation 59 agents try to reduce the distances between their average location and prey.

with $X_m(t)$ determined according to Equation 52

5.6.1.2. Deawbacks of the Original Algorithm

Despite the admirable performance of the original HHO algorithm, following exhaustive testing using standard CEC2013 benchmark functions, certain deficiencies have been observed in the original algorithm, signifying that further room for improvement is present. Specifically, the HHO can greatly benefit from an improvement in exploration, as well as an intensification-diversification trade-off.

Optimization algorithms can, as a rule, be improved through minor alterations, such as variations in search equations, through the introduction of various search mechanics. Furthermore, the hybridization of algorithms often yields improved performance but at the cost of increasing implementation and computational complexity. In this work, the original HHO algorithm is augmented by adding a novel mechanism as well as hybridization with a notably powerful optimization algorithm.

5.6.1.3. HHO with Disputation Operator

The proposed improved HHO algorithm mitigates the inherent drawbacks of the original by integrating a disputation operation, originally introduced by the original social network search (SNS) algorithm [115]. By interdicting this operator, a subgroup of solutions can be guided toward exploring certain promising areas. Originally described as a process of information exchange between social network users discussing and expressing defining opinions. It allows a subgroup to establish a subgroup of a population and to elaborate on certain subjects, it is mathematically modeled in Equation 60

$$\begin{aligned}
 x_{i \text{ new}} &= x_i + rand(0, 1) \times (M - AF \times x_i) \\
 M &= \frac{\sum_t^{N_r} x_t}{N_r} \\
 AF &= 1 + round(rand)
 \end{aligned}$$

Equation 60 social network search algorithm.

where x_i depicts the relocation vector of a certain view of the $i - th$ user, $rand(0,1)$ represents a random value between 0 and 1, while W stands for the mean of the views held by the users of this subgroup. The value of AF denotes a factor of admission, used to gauge the persistence of user-held opinions the value of which can vary in value between 1 and 2. Additionally, $rand(0,1)$ denotes a random value selection function that ranges between 0 and 1, with the *round* function rounding the value to the closest integer value. The number of agents within a subgroup is denoted by the N_r parameter with a range between 1 and N users, with N being the maximum number of individuals in a given population.

The disputation operator implementation in this work is somewhat modified. In the original SNS, the first value in AF is assumed to be locked to 1, accordingly, the value of AF can only be 1 or 2. Following extensive testing using the original SNS algorithm, it can be deduced that it is preferable to define a larger initial step size, shifting the focus toward exploration, then gradually decreasing it through iteration once exploitation becomes preferable. Additionally, by treating AF as a continuous value fine-tuning becomes an option. With this in mind, the approach proposed by this work applies an additional dynamic group search control parameter (gso) and makes use of Equation 61 to calculate AF values in each iteration

$$AF = gsp + round(rand)$$

Equation 61 AF values in each iteration.

in which the value of gsp is initially set at 2 and significantly decreased in subsequent iterations. By doing so the algorithm is capable of shifting focus from exploration towards exploitation in later iterations where it plays a more important role.

The proposed method further implements a two-mode group search mechanism. The initial group searches with a total of N_r arbitrarily selected agents, while the second group centers around the optimal N_b solutions from the swarm. Both groups are randomly selected in each iteration from a range of 1 to N . While Equation 60 is used in both groups each subsequent step is determined according to Equation 61. The algorithm is configured to use the second mode in a later iteration, where exploitation plays a more significant role under the assumption that the algorithm is converging toward more promising areas in the search

region. The transition from each mode is determined by a control parameter named the change mode trigger (cmt).

In accordance with carried out empirical experimentation, the original HHO can efficiently converge toward promising search areas in cases when agents have been positioned near promising solutions in the initial stages. To improve the chances of the base HHO search mechanics and reduce computational demands the proposed group search mechanism is withed in early execution stages. Should the search condition be met, new individuals x_{new} are introduced. Following this, a greedy selection is done, between the newly introduced solutions and the randomly selected latter 50% worse performing agents $x_{rnd\ worst}$.

The novel method proposed in this work is named the HHO with disputation operator (HHO-DO), to suitably credit the introduced search approach. Besides the introduced operator, the methods introduce three additional parameters that rely on the termination condition T. The values used for these parameters have been determined empirically, and the selected values are shown in Table 21.

Table 21 Control parameter values used for the HHO-DO algorithm.

Parameter	Expression	Description
gsp	$gsp = gsp - \frac{t}{T}$	dynamic group search parameter, initially 2
gss	$gss = \frac{T}{3}$	group search start
cmt	$cmt = gss + \frac{T}{3}$	change mode trigger

The novel proposed algorithm's pseudo-code can be seen in Algorithm 5.

Algorithm 5 Pseudocode for the novel HHO-DO metaheuristic.

Initialize random population $X_i(i = 1, 2, \dots, N)$

While Exit criteria not met **Do**

Evaluate agent fitness

Set X_{rabbit} as best location

For each agent X_i **Do**

```

.      Update E, E0 and J
.      If  $|E| \geq 1$  then
.          .      Exploration phase
.      Else
.          .      Exploitation phase
.      If  $r \geq 0.5 \ \&\& \ |E| \geq 0.5$  then
.          .      Adopt soft besiege strategy
.      Else if  $r \geq 0.5 \ \&\& \ |E| < 0.5$  then
.          .      Adopt hard besiege strategy
.      Else if  $r < 0.5 \ \&\& \ |E| \geq 0.5$  then
.          .      Adopt soft besiege with progressive rapid dives strategy
.      Else if  $r < 0.5 \ \&\& \ |E| < 0.5$  then
.          .      Adopt hard besiege with progressive rapid dives strategy
.      End if
.      Update location vector end if
End for
If  $t > gss$  then
.      If  $t < cmt$  then
.          .      Generate new agent  $x_{new}$  via group search mode 1 operator
.      Else
.          .      Generate new agent  $x_{new}$  via group search mode 2 operator
.      End if
.      Perform greedy selection between  $x_{new}$  and  $x_{rnd}$  worst
End if
Rank all solutions to find the current best agent
End while
Return Best attained solution

```

5.6.2. Experimental Setup

The conducted research covers two extensive experiments. The first experiment makes use of LSTM networks to tackle crude oil price forecasting from the original time-series data. The second experiment uses VMD to decompose data into subrends, which are provided as inputs to a LSTM network with the approach dubbed VMD-LSTM. During both experiments, 6 data points were provided as inputs for forecasting. A total of three simulations have been carried out in both experiments. The first simulation predicts one, the second three, and the final simulation forecasts prices five steps(days) ahead.

The conducted experiments are set up similarly to the preceding research [113], which serves as inspiration for the proposed work.

In this section, the dataset is used during experimentation followed by expectations of encoding schemes and framework details. Furthermore, the evaluation metrics will be covered, as well as the algorithms utilized for comparative analysis and simulation test conditions.

5.6.2.1. Dataset, Hyperparameters, Solution Encoding

The conducted research is done using a real-world price dataset. Namely, Brent crude oil trading prices were used, acquired from a public source provided by the United States Energy Information Administration (EIA)³. The available data covers trading days from 20.05.1987. to 03.10.2022. excluding weekends. For experimental purposes, the available data has been split into three subsets, with 70% of the available data used for model training, 10% for validation, and the final 20% used for testing as shown in Figure 21.

For the two conducted experiments the primary goal was to employ the proposed HHO-DO algorithm to attain the most accurate predictions of Brent crude oil prices via a tuned LSTM network. Accordingly, each potential solution of the metaheuristic represented a set of LSTM control parameters. The boundaries were determined through extensive empirical testing and are shown in Table 22.

³ <https://www.eia.gov/petroleum/gasdiesel/>

Table 22 LSTM control parameter boundaries.

Parameter	Boundaries	Data type
number of neurons in the LSTM layer (nn)	lb = 20, ub = 200	integer
learning rate (lr)	lb = 0.0001, ub = 0.01	double
dropout rate (dr)	lb = 0.001, ub = 0.01	double
number of training epochs (epochs)	lb = 100, ub = 300	integer

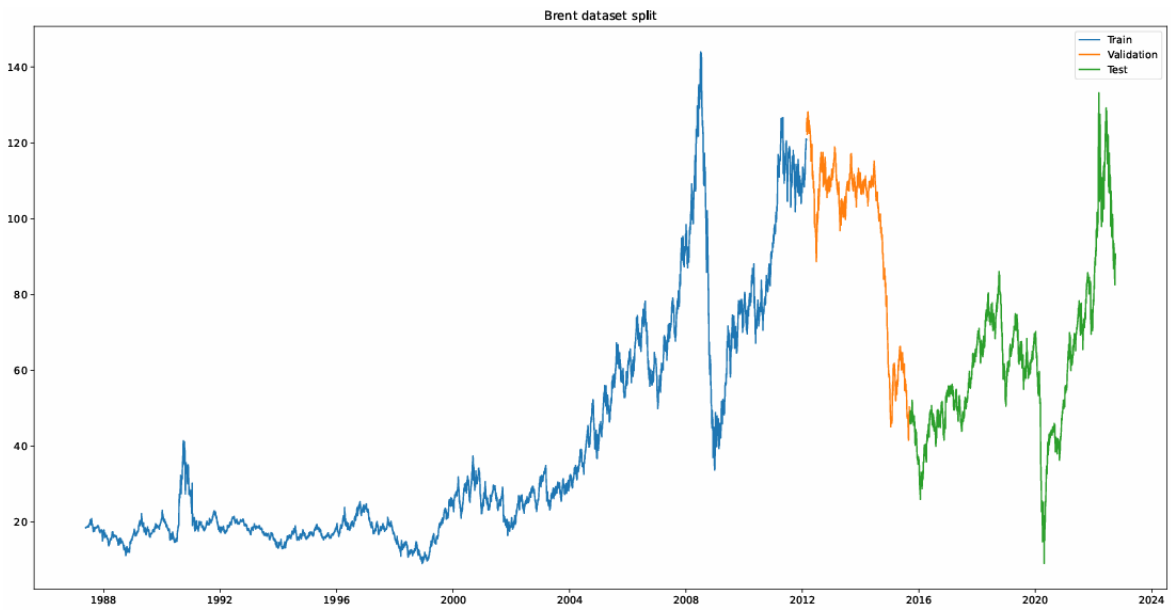


Figure 21 Train/validation/test of daily Brent oil prices used in experiments.

For the second conducted experiment, VMD is applied to the data. The number of the data points is therefore extended to $D = 4 \cdot (K + 1)$, where K represents the number of VMD modes. Since the decomposition process results in one additional series of data that represents residual noise, the remainder of decomposition is presented as an additional series, resulting in $K + 1$.

5.6.2.2. Evaluation Metrics, Basic Setup and Opponent Methods

To assess, the performance of the suggested method four evaluation metrics are utilized. These include: : root mean square error ($RMSE$) calculated using Equation 22, mean absolute error (MAE) determined according to Equation 20, mean square error (MSE)

computed using Equation 21 and coefficient of determination (R^2) determined using Equation 23 in which y_i and $\hat{y}_i$ represent actual prediction values for a given iteration, $\bar{y}$ represents the mean value of real data, and n represents the number of data samples.

When conducting multiple steps ahead forecasts, LSTM networks provide several output neurons, each representing a single day. Accordingly, for each step ahead, each metric is evaluated independently for every step. When multiple steps are provided as output an overall metric is introduced. However, when a single output neuron is provided, in the case of a single step ahead prediction, the metrics are identical.

When performing multiple steps ahead predictions, the initial *step-1* predictions provide less than the sufficient number of output data points to determine an appropriate error rate. Therefore, in cases when for instance a 3-step ahead prediction is made, the first two predictions for the first and second days ahead are disregarded, and metrics are only calculated after the 3rd day ahead. For this reason, the metrics are not the same as simple arithmetic mean.

The prediction task can be formulated as a minimization problem guided by an objective function. This research makes use of Equation 62 that relies on the overall MSE metric.

$$Obj = \min(MSE)$$

Equation 62 Overall MSE metric.

Due to the heavy computation demands of the described methods, for both experiments, testing has been conducted using a population size of 6 individual agents and through 5 iterations. Additionally, to provide valid groups for comparative analysis, due to the random nature innate to metaheuristic algorithms the results shown are averaged over 10 independent runs.

Additionally, an early stopping condition is introduced for both simulations as $patience = \frac{epochs}{3}$ the recurrent dropout rate for the LSTM networks has been kept at the

suggested default 0.01 rate, the utilized optimizer is *adam* with loss function *mse*, the *batchsize* is set to 16, and the applied activation function being *relu*.

To ensure a valid assessment, several contemporary metaheuristics have been put into a comparative analysis to demonstrate the improvements made. The algorithms tested include the novel HHO-DO, the original HHO [114], BA [37], SCA [67], FA [38], and TLB [116], implemented independently for this study, and tested under identical conditions as the HHO-DO algorithm, with control parameters set to their default values recommended in the papers that initially introduced the algorithms.

5.6.3. Experimental Results and Discussion

This section presents the results attained during experimentation and provides a discussion of the outcomes. The first results attained from the LSTM simulation are covered, which use the original dataset without decomposition methods. Following these results from the VMD-LSTM method are presented along with a comparison with contemporary metaheuristics to present the improvements made and validate enactments made with the use of the HHO-DO algorithm.

Furthermore, the presented results are shown using normalized data points in the range of (0,1). Subsequently, results attained might differ from those shown in previous studies due to differences in scales.

5.6.3.1. Simulations with LSTM

The following section covers findings and results as well as a comparative analysis of the LSTM approach applied to data prediction. In Table 23 the worst, mean, median, standard deviation, and variance of the objective function (MSE) are shown and cover 10 independent runs for five-step ahead predictions.

The second Table 24 covers performance indicators *R2*, *MAE*, *MSE*, and *RMSE* attained during the best-performing execution for each prediction step individually.

Objective function metric results shown in Table 23 strongly indicate that the proposed LSTM-HHO-DO approach on average and median obtains better results compared to other contemporary algorithms, only being outperformed by the LSTM-HHO-FA algorithm in the worst case, suggesting that further improvement is possible and enforcing the no free lunch theorem.

Further, this observation is solidified by results shown in Table 24 where predictions for each step are shown. The proposed approach performed the best overall. However, when casting three and five-step ahead predictions it has been marginally outperformed by the LSTM-FA.

Table 23 Objective function metrics for LSTM application.

Method	Best	Worst	Mean	Median	Std	Var
LSTM-HHO-DO	2.346E-04	2.463E-04	2.392E-04	2.380E-04	4.700E-06	2.210E-11
LSTM-HHO	2.467E-04	2.722E-04	2.658E-04	2.722E-04	1.110E-05	1.230E-10
LSTM-SCA	2.478E-04	2.619E-04	2.528E-04	2.507E-04	5.460E-06	2.980E-11
LSTM-FA	2.357E-04	2.456E-04	2.406E-04	2.405E-04	3.520E-06	1.240E-11
LSTM-BA	2.421E-04	2.559E-04	2.503E-04	2.515E-04	5.730E-06	3.280E-11
LSTM-TLB	2.410E-04	3.010E-04	2.607E-04	2.505E-04	2.450E-05	6.000E-10

Table 24 The R2, MAE, MSE, and RMSE metrics of the best performing LSTM approach.

	Error indicator	LSTM-HHO-DO	LSTM-HHO	LSTM-SCA	LSTM-FA	LSTM-BA	LSTM-TLB
One-step ahead	R2	0.989239	0.988916	0.988825	0.988771	0.989125	0.988837
	MAE	0.010868	0.010978	0.011016	0.010907	0.010914	0.011149
	MSE	0.000238	0.000249	0.000246	0.000243	0.000242	0.000244
	RMSE	0.015428	0.015780	0.015682	0.015580	0.015552	0.015634
Two-step ahead	R2	0.989706	0.989070	0.988966	0.989129	0.988986	0.989205
	MAE	0.010667	0.010907	0.010953	0.010719	0.011032	0.010983
	MSE	0.000228	0.000243	0.000243	0.000237	0.000243	0.000236

	RMSE	0.015113	0.015602	0.015604	0.015391	0.015575	0.015378
Three-step ahead	R2	0.989688	0.988703	0.988883	0.989659	0.988945	0.989041
	MAE	0.010665	0.011282	0.011076	0.010631	0.011160	0.011100
	MSE	0.000230	0.000247	0.000245	0.000227	0.000242	0.000239
	RMSE	0.015180	0.015720	0.015656	0.015064	0.015571	0.015475
Four-step ahead	R2	0.989455	0.989119	0.988461	0.989176	0.989311	0.989165
	MAE	0.010822	0.010953	0.011376	0.010934	0.010899	0.011055
	MSE	0.000235	0.000241	0.000253	0.000235	0.000235	0.000239
	RMSE	0.015317	0.015522	0.015906	0.015344	0.015334	0.015447
Five-step ahead	R2	0.989063	0.988567	0.988498	0.989196	0.988729	0.988721
	MAE	0.011102	0.011317	0.011349	0.010945	0.011178	0.011290
	MSE	0.000242	0.000253	0.000251	0.000236	0.000248	0.000246
	RMSE	0.015543	0.015900	0.015852	0.015370	0.015760	0.015680
Overall Results	R2	0.989431	0.988876	0.988728	0.989188	0.989019	0.988994
	MAE	0.010825	0.011087	0.011154	0.010827	0.011037	0.011116
	MSE	0.000235	0.000247	0.000248	0.000236	0.000242	0.000241
	RMSE	0.015317	0.015705	0.015740	0.015351	0.015559	0.015523

5.6.3.2. Simulations with VMD-LSTM

In the second experiment, the original dataset has been subjected to VMD, and the outputs have been piped as inputs to a LSTM network. Results are grouped similarly to the previous section, with Table 25 showing the worst, mean, median, standard deviation, and variance of the objective function (MSE), and Table 26 covering comparative analysis results of the proposed approach and other contemporary algorithms for individual prediction steps.

The improvements introduced by the application of VMD can be seen in Table 25 where the proposed VMD-LSTM-HHO-DO outperformed all other contemporary

algorithms according to all evaluation metrics emphasizing the accuracy improvements made by subjecting data to decomposition methods.

The proposed VMD-LSTM-HHO-DO algorithm outperformed all contemporary algorithms on average and overall metrics as shown in Table 26. However, when casting one and four-step ahead predictions, the VMD-LSTM-SCA algorithm performed slightly better, further affirming the no-free-lunch theorem.

Table 25 Objective function metrics for VMD-LSTM application.

Method	Best	Worst	Mean	Median	Std	Var
VMD-LSTM-HHO-DO	1.750E-04	1.750E-04	1.750E-04	1.750E-04	0	0
VMD-LSTM-HHO	1.845E-04	2.037E-04	1.884E-04	1.845E-04	7.680E-06	5.890E-11
VMD-LSTM-SCA	1.970E-04	2.145E-04	2.057E-04	2.053E-04	7.820E-06	6.120E-11
VMD-LSTM-FA	2.056E-04	2.305E-04	2.226E-04	2.236E-04	9.120E-06	8.310E-11
VMD-LSTM-BA	1.914E-04	2.027E-04	1.958E-04	1.914E-04	5.360E-06	2.880E-11
VMD-LSTM-TLB	1.939E-04	2.254E-04	2.035E-04	2.022E-04	1.160E-05	1.340E-10

Table 26 The R2, MAE, MSE, and RMSE metrics of the best performing VMD-LSTM approach.

	Error indicator	VMD-LSTM-HHO-DO	VMD-LSTM-HHO	VMD-LSTM-SCA	VMD-LSTM-FA	VMD-LSTM-BA	VMD-LSTM-TLB
One-step ahead	R2	0.991313	0.991233	0.992671	0.990029	0.991466	0.992343
	MAE	0.009602	0.008802	0.008981	0.010450	0.009816	0.009138
	MSE	0.000192	0.000169	0.000161	0.000221	0.000191	0.000169
	RMSE	0.013866	0.012685	0.012670	0.014865	0.013824	0.012991
Two-step ahead	R2	0.993067	0.992557	0.990262	0.989065	0.991466	0.991418
	MAE	0.008436	0.009051	0.010879	0.010966	0.009838	0.009761
	MSE	0.000157	0.000169	0.000213	0.000241	0.000187	0.000192
	RMSE	0.012513	0.012702	0.014611	0.015520	0.013682	0.013842

Three-step ahead	R2	0.992323	0.992101	0.988879	0.991197	0.991039	0.991445
	MAE	0.008892	0.009802	0.011990	0.009958	0.009931	0.009645
	MSE	0.000173	0.000183	0.000239	0.000194	0.000200	0.000189
	RMSE	0.013158	0.013215	0.015473	0.013935	0.014151	0.013742
Four-step ahead	R2	0.991394	0.991174	0.993357	0.991559	0.991979	0.991835
	MAE	0.009561	0.009455	0.008705	0.009693	0.009437	0.009492
	MSE	0.000190	0.000172	0.000148	0.000186	0.000178	0.000182
	RMSE	0.013785	0.012796	0.012166	0.013648	0.013346	0.013497
Five-step ahead	R2	0.992753	0.992142	0.989867	0.991673	0.991084	0.989215
	MAE	0.008641	0.009694	0.011544	0.009523	0.010058	0.011082
	MSE	0.000163	0.000186	0.000224	0.000186	0.000200	0.000238
	RMSE	0.012763	0.013319	0.014956	0.013630	0.014154	0.015429
Overall Results	R2	0.992177	0.991823	0.991023	0.990707	0.991406	0.991253
	MAE	0.009027	0.009361	0.010420	0.010118	0.009816	0.009824
	MSE	0.000175	0.000176	0.000197	0.000206	0.000191	0.000194
	RMSE	0.013228	0.012946	0.014037	0.014339	0.013835	0.013924

5.6.3.3. Comparison with Other Well-Known Methods

Five-step ahead predictions cast some of the tested state-of-the-art algorithms used to improve a LSTM network as well as real price data are shown in Figure 23, while predictions made by the best performing LSTM-HHO-DO model, alongside real-world data, can be seen in **Error! Reference source not found..**

The best prediction made by the best-performing LSTM-HHO-DO model, alongside real-world data, can be seen in **Error! Reference source not found..**

It is interesting to observe the rapid convergence rates of the proposed approach as opposed to other metaheuristics shown in Figure 22 for algorithms used to optimize the LSTM approach, as well as those used in combination with the VMD decomposition methods shown in Figure 25.

Five-stem ahead predictions cast some of the tested state-of-the-art algorithms as opposed to real data are shown in Figure 26, while predictions made by the best performing VMD-LSTM-HHO-DO model, alongside real-world data, can be seen in Figure 27.

Aside from the previous deductions, it is important to note that research presents implications concerning policy. Namely, governing policymakers need to remain vigilant of oil price changes and adjust economic measures as needed to reduce the impact of price variation. This requires swift actions from monetary agencies to reduce the secondary impact as well as price shocks. The suggested approach may help mitigate the effects of price shocks and the resulting aftereffects thereof. Furthermore, the proposed approach has great potential to improve economic growth by stimulating better-informed investments, while enhancing the resistance of local economies.

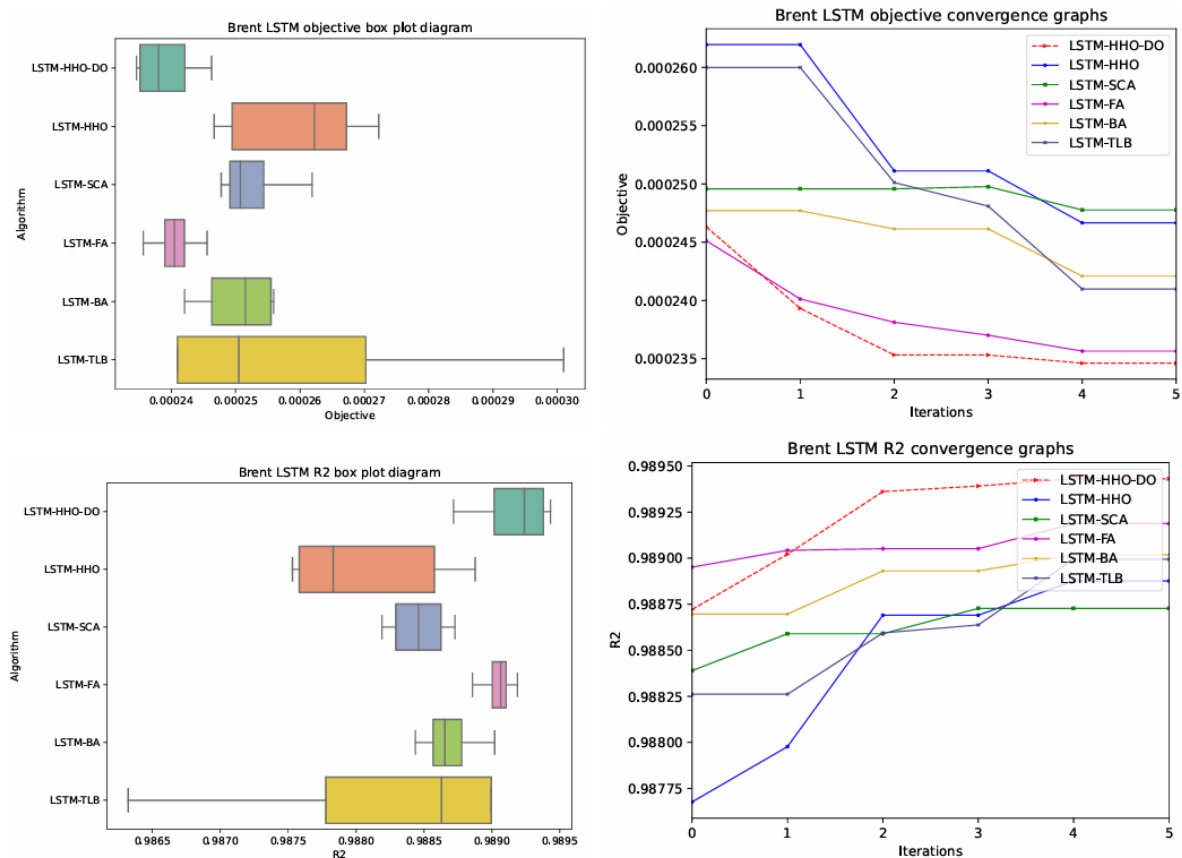


Figure 22 LSTM Comparative analysis visualizations.

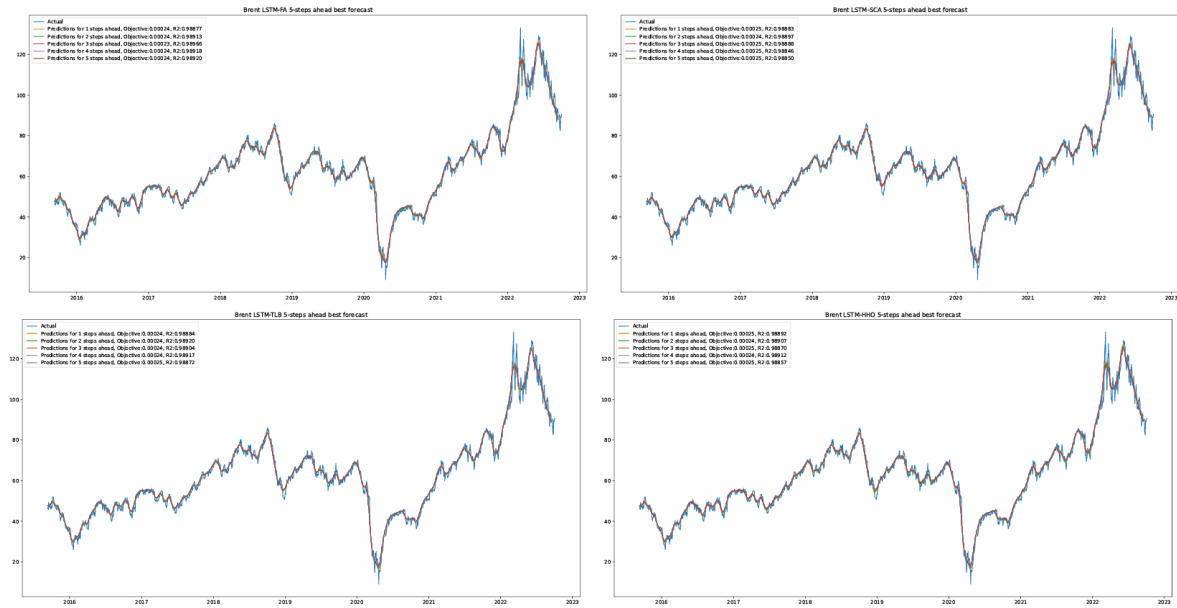


Figure 23 Five steps ahead predicted vs actual price values cast by all tested algorithm.



Figure 24 Five steps ahead predicted vs actual price values presented by the LSTM-HHO-DA algorithm.

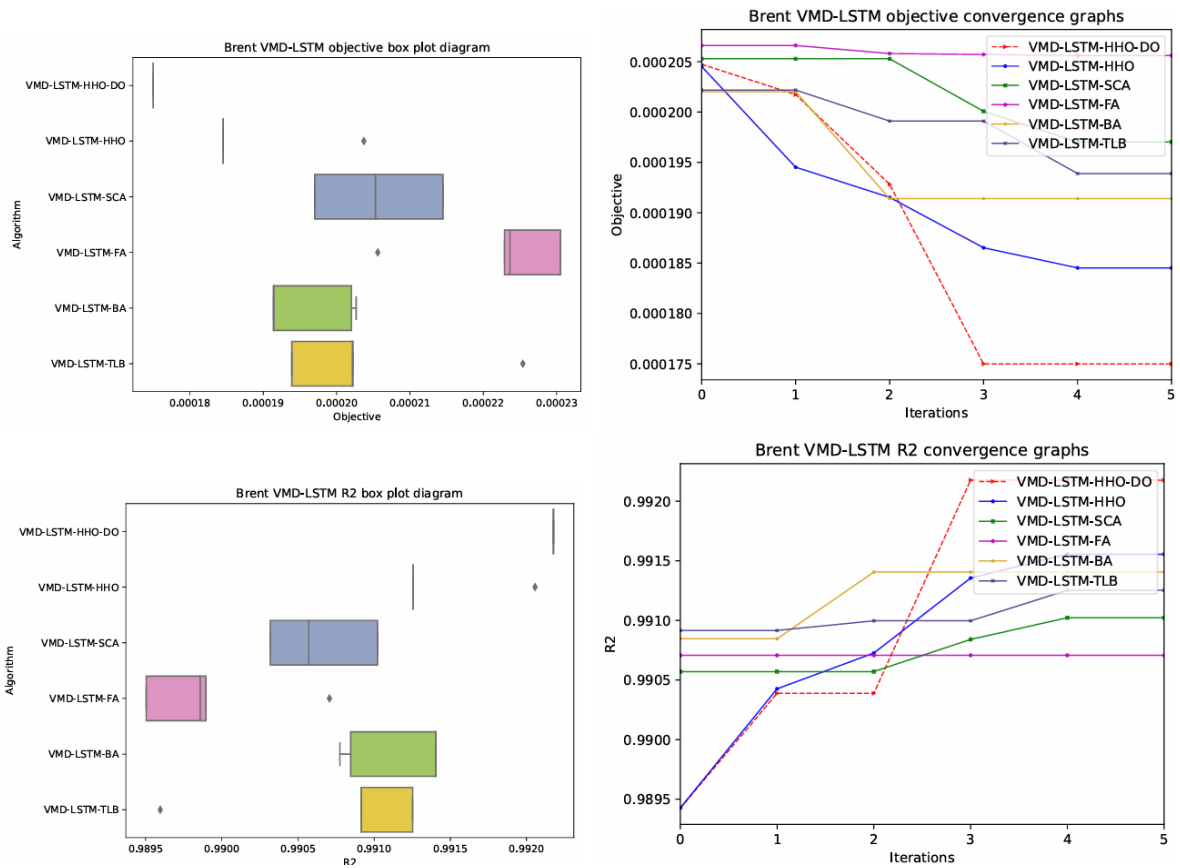


Figure 25 VMD-LSTM Comparative analysis visualizations.

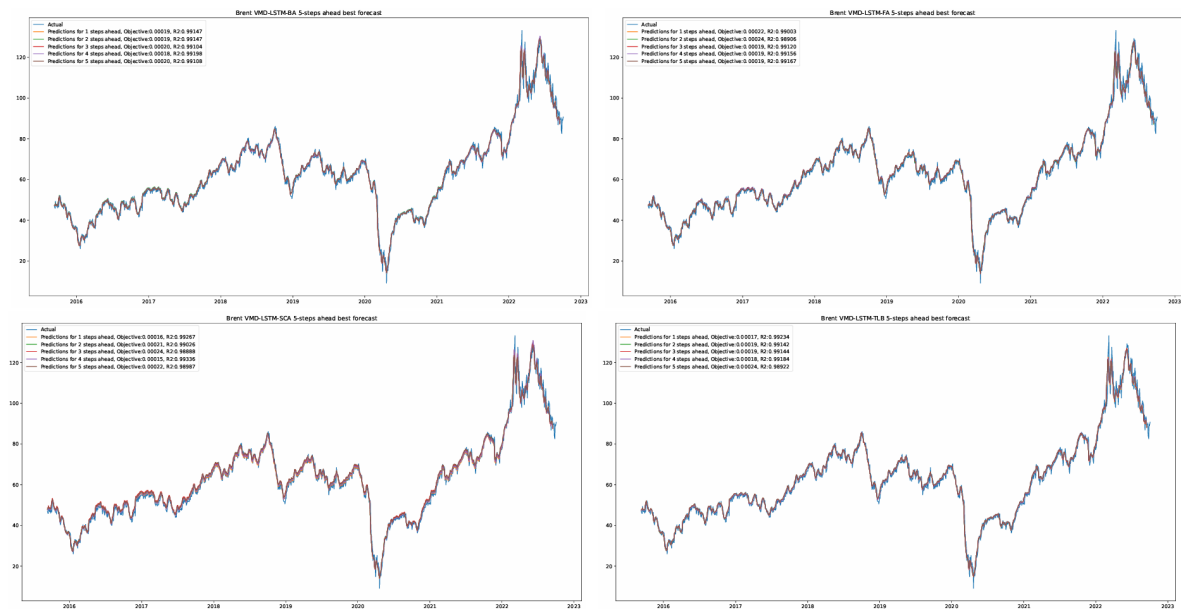


Figure 26 Five steps ahead predicted vs actual price values cast by all tested algorithm.

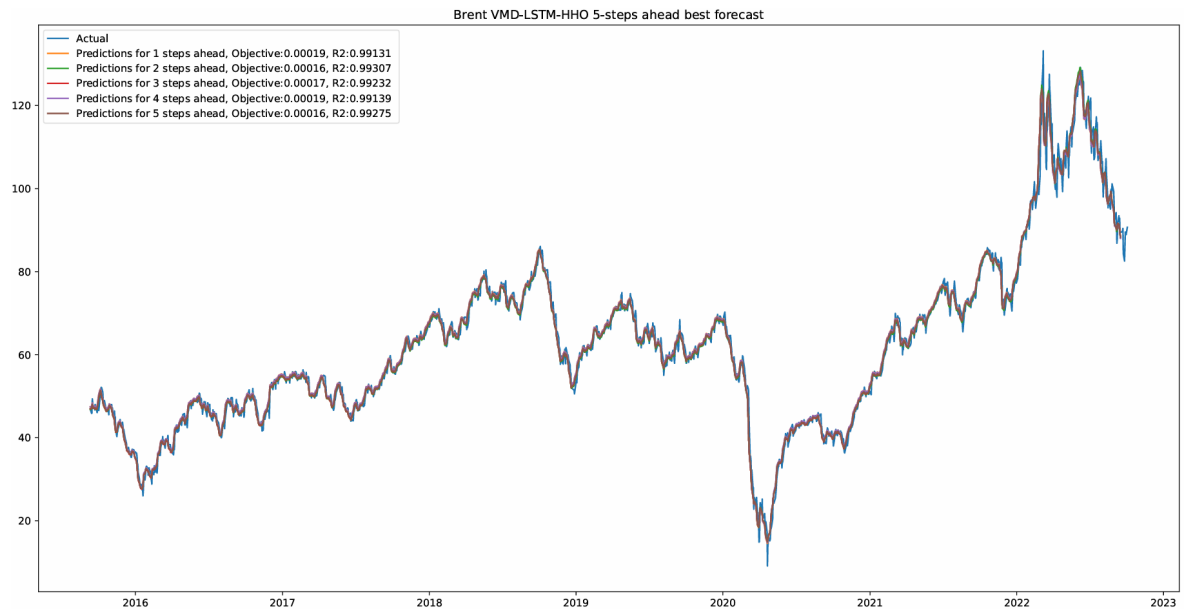


Figure 27 Five steps ahead predicted vs actual price values presented by the VMD-LSTMHHO-DA algorithm.

5.7. Application 6: Exploring the Potential of Modified Metaheuristic Optimized Long-Short Term Memory Neural Networks for Earthquake Magnitude Forecasting

The intricate interplay of stress and strain occurring between tectonic plates, coupled with various geological phenomena, culminates in the occurrence of earthquakes, a natural disaster of profound impact. These seismic events are particularly pronounced in regions where tectonic plates converge or interact, resulting in a higher frequency of such occurrences. The specialized field of seismology is dedicated to comprehensively studying earthquakes, focusing on both the seismic waves they generate and the underlying geological processes. Seismologists play a crucial role in monitoring and analyzing seismic events globally, meticulously recording the intensity and geographical locations of these occurrences. The amassed wealth of data provides a fertile ground for the application of data-driven methodologies, offering the promise of developing robust earthquake forecasting techniques [117].

However, despite the abundance of seismic data, challenges persist in achieving uniformity in data formats, collection methods, and presentation standards. This lack of consistency poses a hurdle in accurately forecasting earthquake events, adding complexity to an already intricate task. The urgency for an effective forecasting system becomes evident considering the devastating impact of earthquakes [118]. Nevertheless, the inherently nonlinear nature of this challenge further complicates efforts to devise a comprehensive and reliable forecasting methodology. Addressing these complexities is essential for advancing our ability to predict and mitigate the destructive consequences of earthquakes.

Advancements in artificial intelligence (AI) in recent times have created new opportunities to tackle the problems related to earthquake forecasting. AI in seismology has the potential to significantly increase forecast efficiency and accuracy, which will improve our capacity to lessen the effects of seismic occurrences. Since earthquake precursors are frequently subtle and intricate [119], advanced pattern detection skills are needed. Machine Learning (ML) algorithms may be trained to recognize minute signals or abnormalities in seismic data that can be signs of an impending earthquake. Incorporating AI and time-series forecasting models for seismic activity can help scientists and emergency response workers

make more informed decisions. Additionally, the detection of aftershocks [120] can prove vital to preserving lives in disaster scenarios.

The challenge of proper hyperparameter selection further complicates the application of machine learning algorithms in earthquake forecasting. Many algorithms are initially designed with a focus on achieving good general performance across various tasks. However, adapting these algorithms to specific applications with desired outcomes requires careful adjustment of hyperparameters [121]. The task of hyperparameter tuning becomes particularly daunting due to the nature of emerging algorithms, which introduce an increasing number of parameters. This process can be considered NP-hard, and traditional approaches often involve a trial-and-error method. In recent years, researchers have increasingly turned to metaheuristic algorithms to address the complexity of hyperparameter selection. These innovative approaches aim to optimize the algorithm's performance by exploring the hyperparameter space more efficiently, providing a promising avenue for overcoming challenges in the customization of machine learning models for earthquake prediction.

This work examines the prospect of long short-term memory (LSTM) [122] neural networks for earthquake magnitude forecasting. Recognizing that the performance of LSTM networks heavily relies on appropriate hyperparameter choice, an altered variant of the sine-cosine algorithm [67] is proposed specifically tailored to meet the requirements of this study. The proposed approach is evaluated using real-world, publicly available seismological data, and a comparative analysis is executed with several state-of-the-art optimization methods. This research has not been published yet while writing this thesis.

5.7.1. Methods

This section sheds light on the elementary version of the sine cosine algorithm and discusses the drawbacks of this basic variant. Afterward, a modified variant of the algorithm is suggested to address these constraints.

5.7.1.1. Basic SCA

The Basic sine cosine algorithm was mentioned previously in Original Sine Cosine Algorithm (SCA) pages - 61 -

5.7.1.2. Hybrid Adaptive SCA

The SCA algorithm performs better than many other algorithms that are currently in use. Even though it's a relatively new addition to optimization techniques, trials are essential to find possible enhancements. Its efficiency has been increased by a number of changes. To improve the variety, the main change consists of integrating quasi-reflexive learning (QRL) [123] initialization. QLR initialization helps in improving convergence rate and solution quality, by allowing the algorithm to commence close to the promising areas of search space, and consequently enhances the variety of solutions and overall algorithmic efficacy and performance. To clarify, the first half of the population is normally initialized, while the second half is created according to the implementation Equation 47.

where rad represents a random value inside the specified interval and lb and ub designate the search space's lower and upper bounds. Previous research has demonstrated that the QRL mechanism enhances exploration.

Hybridization in an adaptable way using the FA [124] handles additional boosts to convergence, as FA is well renowned for its powerful search. Hybridization allows taking the best of both SCA and FA, while cancelling out their individual drawbacks. The mechanism of firefly search is explained by Equation 63

$$X_i(t+1) = X_i(t) + \beta e^{-\gamma r_{ij}^2} (X_j(t) - X_i(t)) + \alpha \epsilon_i(t)$$

Equation 63 Mechanism of firefly.

where β_0 represents the attractiveness at $r = 0$. However, Equation 63 is commonly swapped for Equation 64 to improve computational performance.

$$\beta(r) = \frac{\beta_0}{(1 + \gamma \times r^2)}$$

Equation 64 Improve the computational performance of Firefly.

here r_{ij} indicates the current location of agent j during the corresponding iteration t , and $X_i(t)$ indicates the current position of agent i at a particular iteration t . The distance between agents indexed as i and j , which serves as a measure of their mutual attraction, is indicated by the parameter β . The agent attraction coefficient is β ; the light absorption coefficient is γ ; the degree of randomness is α ; and a stochastic vector is $\epsilon_i(t)$. In this hybridization, the FA's function is to increase exploitation in the last few optimization rounds.

An empirically determined value is chosen for an introduced parameter $\theta = 0.8$, and a random value between 0 and 1 is picked for value ψ in each optimization iteration. The FA search is performed if $\psi > \theta$; if not, the SCO search is performed. After every iteration, θ is reduced by 0.4, which raises the likelihood that FA will be utilized. With this strategy, the updated version of the algorithm may profit from the early exploration boost from QRL while concentrating on convergence in subsequent iterations.

The updated method is formulated using hybridization and an adaptive approach, resulting in an optimizer known as the hybrid adaptive SCO (HASCO). Algorithm 6 presents the pseudocode for this optimizer.

Algorithm 6 Pseudo-code for the proposed HASCO.

Initialize $\frac{1}{2}$ of the population P

Apply QRL to generate the second half of agents.

Set $\theta = 0.8$

While $T > t$ **Do**

Evaluate population fitness Generate a random value for ψ

If $\psi > \theta$ **then**

. Apply FA search

Else

. Apply SCO search

End if

$\theta = \theta - 0.4$

End while

5.7.2. Experimental Setup

To validate the suggested methodology, a publicly available seismology dataset is utilized that was last accessed at the Earthquake Hazards Program website ⁴on the 7th of February 2024. The datasets covers all earthquakes occurring over a 30-day basis from January 5th to February 4th. The intensity of earthquakes is logged in the Richter local magnitude scale. The data is converted into a time-series format with 6 lats (samples provided as inputs) in hopes of predicting the magnitude one, two and three earthquakes ahead. Such an approach provides a good basis for detecting aftershocks and allowing communities and individuals to plan disaster relief and recovery. A visualization of the dataset is provided in Figure 28.

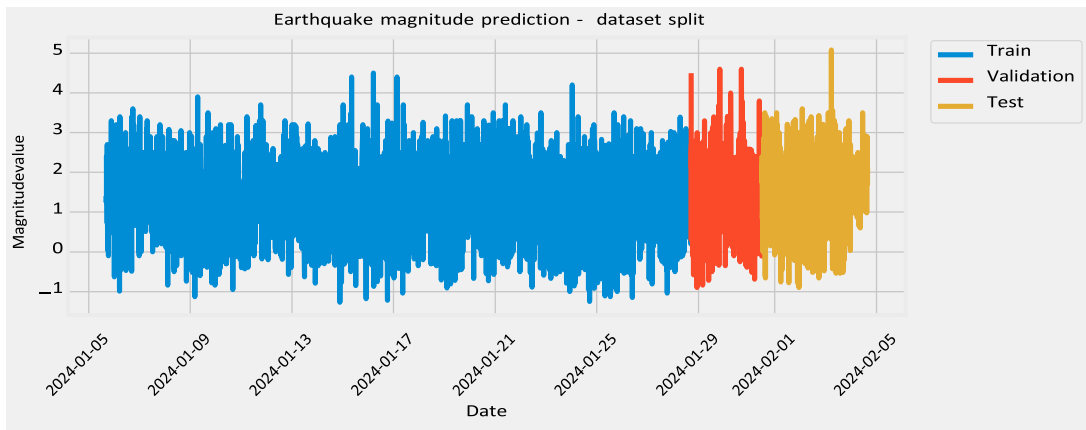


Figure 28 Dataset Visualization.

⁴ <https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php>

Models are constructed and trained with the initial 70% of the available data, with an additional 10% serving for validation and the last 20% withheld for testing. Several metaheuristic optimizers are included in a comparative analysis with the introduced HASCA algorithm. The algorithms included in the evaluations are the original SCA [67], GA [57], BA [37], ABC [105] and FDO [125]. Optimization algorithms are independently implemented in Python, using the default hyperparameters suggested in the works that originally introduced each respective algorithm. Each algorithm is allocated a population size of 5 agents and allowed a total of 5 iterations to enhance the results. Additionally, simulations are carried out through 30 independent runs to ensure a fair comparison is presented despite the randomness associated with the heuristic approach. Optimizers are tasked with selecting optimal network architectures and training parameter form ranges shown in Table 27

Table 27 LSTM optimizer parameters and their respective ranges.

Parameter	Lower Bound	Upper Bound
Learning rate	0.0001	0.01
Dropout	0.05	0.2
Epochs	300	600
Layers	1	3
Layer neurons	100	200

For evaluating the performance of the proposed method several metrics were monitored during the experiments. These include the mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), coefficient of determination (R^2) and the index of alignment (IoA)

$$IA = 1 - \frac{\sum_{i=1}^n (b_i - \hat{b}_i)^2}{\sum_{i=1}^n (|b_i - \bar{b}| + |b_i - \hat{b}_i|)^2}$$

Equation 65 Index of alignment.

where b_i denotes the observed values, $\hat{b}_i$ the forecast value, and $\bar{b}$ the mean of the observed values. Finally, n denotes the number of observed samples. In both experiments, the mean

value of MSE across all three stations is utilized as the objective function. This is done to attain models with a good capacity for generalization.

5.7.3. Simulation Outcomes

Outcomes in terms of objective function results for the best worst mean and median performing models are provided in Table 28. As evident the best-constructed model according to the objective function is the model with parameters optimized by the introduced HASCA algorithm. Nevertheless, the original SCA showcases decent outcomes attaining the best outcomes for the worst-case execution as well as demonstrating admirable stability. The BA algorithm showcases good outcomes in the mean and median executions despite not attaining the best outcomes.

Table 28 Objective function comparisons in terms of the best worst mean and median constructed models.

Method	Best	Worst	Mean	Median	Std	Var
LSTM-HASCA	0.018787	0.020068	0.019428	0.019428	0.000640	4.10E-07
LSTM-SCA	0.019071	0.019094	0.019083	0.019094	0.000011	1.26E-10
LSTM-GA	0.018864	0.020623	0.019626	0.018864	0.000872	7.60E-07
LSTM-BA	0.018828	0.019271	0.019020	0.018828	0.000220	4.84E-08
LSTM-ABC	0.019300	0.019735	0.019518	0.019518	0.000217	4.72E-08
LSTM-FDO	0.019110	0.019546	0.019328	0.019328	0.000218	4.74E-08

Stability comparisons can further be compared in Figure 29 where the admirable consistency of the SCA is evident despite the algorithm not attaining the best outcomes. Additionally, swarm plots for the best agents in each iteration are shown in Table 29.

Detail metric comparisons for the best-performing models in overall performance for all three prediction steps are provided in Table 29. As evident the best outcomes in terms of R2, MSE and MAE are showcased by the models optimized by the introduced HASCA algorithm. The best outcomes in terms of MSE are showcased by the BA and the best IA is attained by the ABC algorithm.

An additional consideration when assessing optimizers is the convergence rates demonstrated during optimization. Comparisons in terms of convergence are demonstrated in Figure 31. A clear improvement over the original SCA is evident with the original algorithm showcasing a slower convergence rate resulting in a local minima being located. The introduced algorithm showed a rapid convergence towards the best-attained outcome, suggesting that the introduced modifications demonstrate a sufficient basis for improvement. Finally, to support easier reproducibility of the experiments, the parameter selections obtained by every metaheuristic for the respective highest-achieving models are showcased in Table 31.

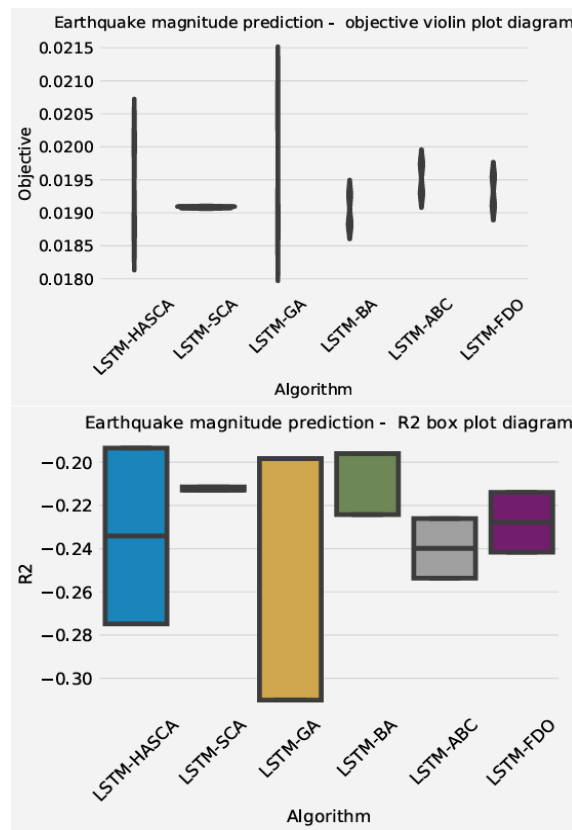


Figure 29 Objective and indicator distribution plots.

Table 29 Overall outcomes for three-step ahead forecasts.

Method	R2	MSE	RMSE	MAE	IA
LSTM-HASCA	-0.193475	0.117033	0.018787	0.137067	0.408592

LSTM-SCA	-0.211495	0.118319	0.019071	0.138098	0.414423
LSTM-GA	-0.198330	0.117429	0.018864	0.137346	0.411955
LSTM-BA	-0.196023	0.116942	0.018828	0.137213	0.405988
LSTM-ABC	-0.226061	0.118951	0.019300	0.138926	0.415379
LSTM-FDO	-0.213977	0.118428	0.019110	0.138240	0.413007

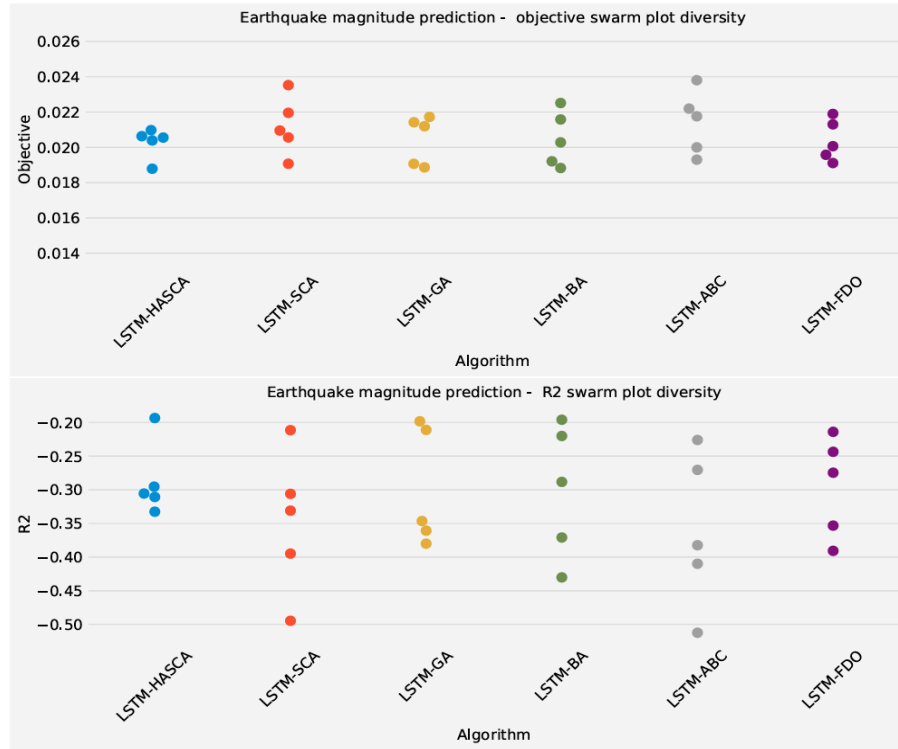


Figure 30 Objective and indicator function warm plots.

Table 30 Detailed metrics for the highest-achieving structures for one, two and three-step ahead steps.

Method	Metric	One step ahead	Two steps ahead	Three steps ahead
LSTM-HASCA	R^2	-0.217610	-0.178719	-0.184096
	MSE	0.118555	0.116061	0.116484
	RMSE	0.019167	0.018555	0.018640
	MAE	0.138446	0.136217	0.136528
	IA	0.412749	0.405384	0.407643

LSTM-SCA	R^2	-0.218881	-0.230806	-0.184799
	MSE	0.118864	0.119448	0.116643
	RMSE	0.019187	0.019375	0.018651
	MAE	0.138518	0.139194	0.136568
	IA	0.416257	0.416156	0.410856
LSTM-GA	R^2	-0.227290	-0.191407	-0.176294
	MSE	0.119258	0.117014	0.116015
	RMSE	0.019320	0.018755	0.018517
	MAE	0.138995	0.136948	0.136077
	IA	0.415218	0.411282	0.409366
LSTM-BA	R^2	-0.211080	-0.186479	-0.190510
	MSE	0.117886	0.116367	0.116572
	RMSE	0.019065	0.018677	0.018741
	MAE	0.138074	0.136665	0.136897
	IA	0.406530	0.405559	0.405876
LSTM-ABC	R^2	-0.254984	-0.219057	-0.204143
	MSE	0.120689	0.118527	0.117636
	RMSE	0.019756	0.019190	0.018955
	MAE	0.140555	0.138528	0.137678
	IA	0.419106	0.414577	0.412453
LSTM-FDO	R^2	-0.222958	-0.208962	-0.210012
	MSE	0.119000	0.118058	0.118226
	RMSE	0.019252	0.019031	0.019048
	MAE	0.138750	0.137954	0.138014
	IA	0.414286	0.411495	0.413241

Table 31 Parameter selection for the highest-achieving networks made by each optimizer.

Approach	Learning rate	Dropout	Epochs	Layers	Neurons L1	Neurons L2
LSTM-HASCA	0.003497	0.166512	600	1	145	N/a
LSTM-SCA	0.001278	0.193595	304	1	139	N/a
LSTM-GA	0.002494	0.119767	493	2	111	134
LSTM-BA	0.003723	0.129873	438	2	195	144
LSTM-ABC	0.004034	0.139697	372	2	165	137
LSTM-FDO	0.001730	0.123893	320	1	100	N/a

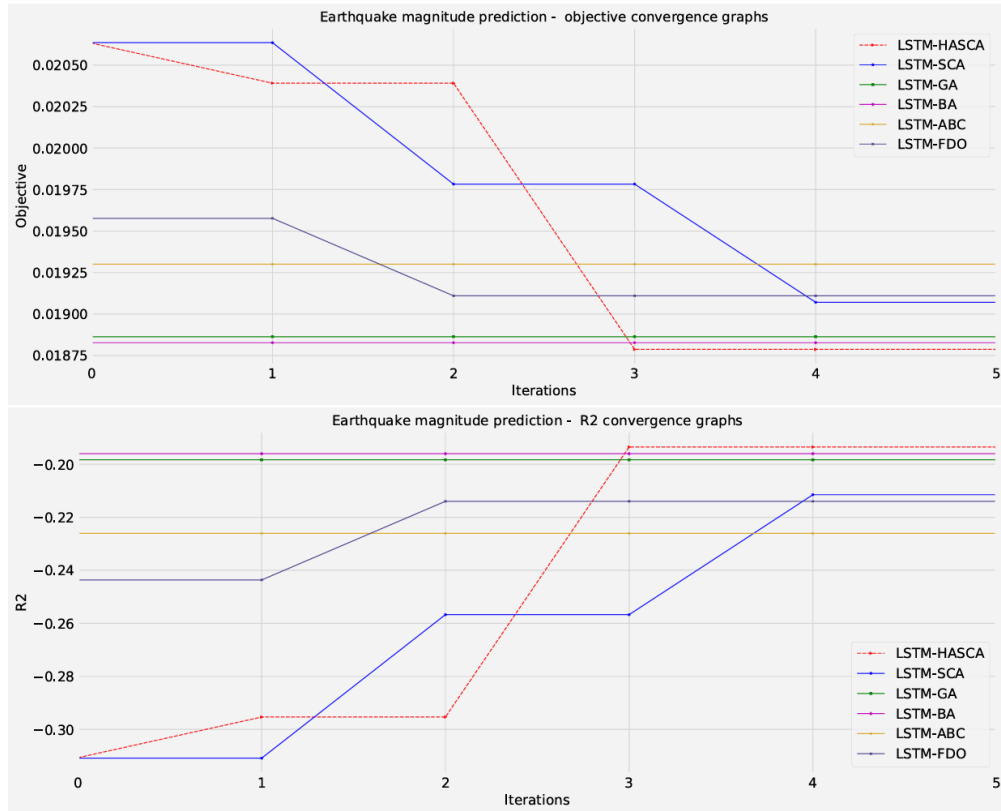


Figure 31 Objective and indicator function convergence.

The simulation outcomes show considerable potential for LSTM models tuned by the metaheuristics algorithms, which may be integrated into earthquake detection and early warning systems in the future. Described LSTM networks may enhance the accuracy and responsiveness of these systems by providing the capability to analyze complex seismic data

in real-time. Seismic data typically includes a variety of time-series signals originating from different sensors, that can be fed to the pre-trained models deployed for real-time earthquake tracking. These models can then process incoming data, identify potential earthquakes, and trigger early warning systems.

Challenges that need to be addressed before employing these models in real-time systems include quality of data (which can be noisy and affect the performance of the model), additional computational resources required for real-time data processing, and integration into the existing infrastructure which may necessitate considerable modifications.

5.8. Application 7: Cloud Spot Instance Price Forecasting Multi-Headed Models Tuned Using Modified PSO

Cloud computing has risen in demand for hosting applications that deploy a large number of processing resources. Customers may face significant leasing rates as a result of this. Precise pricing forecasting is critical for consumers of cloud services [126]. By predicting the demand, Artificial intelligence (AI) and machine learning (ML) can aid in scaling resources up or down automatically to deal with different loads properly [127] [128]. ML models can also be used for cost prediction and management, allowing better budget management.

A challenge all ML models face is the necessity to tune their hyperparameters, which is widely regarded as an NP-hard challenge. Metaheuristics algorithms have proven to be capable of reaching satisfactory solutions. The research proposed in this article aims to improve cloud spot price prediction accuracy by tuned multi-headed gated recurrent units (GRU) [129] and long short-term memory (LSTM) [130] deep learning models, as part of a developed ML framework. These architects are chosen due to their unique ability to address time-series forecasting well due to intrinsic memory retention mechanisms. A modified variant of particle swarm optimization (PSO) [32] is also introduced.

A multi-headed artificial neural network [131] is a uniquely designed network architecture with several branches or "heads". Multi-headed networks are best for processing multiple data sources at once. Sequential data is commonly handled by LSTMs. It solves

fading gradients with memory cells and gate controllers. Three gates (input, forget, and output) formed the LSTM design. These gates control unit information flow. The LSTM design helps it preserve long-term dependencies better than RNNs.

A GRU network is a recurrent architecture variation with fewer parameters and computational complexity than LSTM networks. Yet, it performs similarly in many applications. Only reset and update gates were used in the GRU architecture. This approach lets the GRU capture sequential data dependencies without the expense of LSTMs. In multi-headed designs, LSTM and GRU cells allow network heads to handle complex temporal relationships in data discreetly, reducing computational expenses and making the network more resistant to overfitting. This research was published in [6].

5.8.1. Methods

This section has the modified version of the PSO algorithm.

5.8.1.1. Modified PSO Algorithm

The initialization equation for PSO typically generates random agent positions and velocities. Equation representation:

$$x_{i,d}(0) = rnd() \times (max - min) + min$$

Equation 66 Generate random agent positions and velocities.

the symbol x represents the starting coordinates of particle i in dimension d , denoted as $x_{i,d}(0)$. The function $rnd()$ produces a random decimal number inside the bounds of 0 to 1. The terms max and min provide the upper and lower limits for the search area in dimension d .

Particles measure speed and move in every iteration. The particle's ideal prior location is indicated by $p_{i,d}$, while the particle's global optimal location is denoted by $p_{g,d}$.

The present speed of the i -th particle's d -th dimension at time t may be determined using this equation:

$$v_{i,d}(t) = w \times v_{i,d}(t-1) + c_1 \times rnd_1 \times (p_{i,d}(t-1) - x_{i,d}(t-1)) + c_2 \times rnd_2 \times (p_{g,d} - x_{i,d}(t-1))$$

$$v_{i,d} \in [-v_{max}, v_{max}]$$

Equation 67 The present speed of the particle's.

where this equation has several parts. The rnd function demonstrates an arbitrary number produced within the range of $[0,1]$. Individual and social learning features are represented by the variables c_1 and c_2 . The inertia weight is represented by w , which is employed to balance local and global intensification. The particle's speed is restricted to a range of $[-v_{max}, v_{max}]$, where v_{max} is a specified perimeter number. This constraint guarantees that the particle's travel stays inside a specific region. The size of the searching areas within the present and destination coordinates is determined by the value of v_{max} .

The basic PSO [32] performs well for different optimization problems. The basic algorithm constraints have been found through thorough experimentation with a standard collection of CEC benchmark functions [126]. By increasing solution diversity during initialization and algorithm runs, these modifications aim to address limited exploration. There were three basic PSO alterations.

The initial alteration incorporates chaotic elite learning to enhance individual particles by generating novel solutions nearby, which ensures population diversity and helps escape sub-optimal regions. A logistic map, as stated in Equation 68, is the focus of this modification:

$$C_{i+1} = 4 \times C_i \times (1 - c_i)$$

Equation 68 Chaotic elite learning to enhance individual particles.

in which i represents the integration number, c_i denotes a chaotic value in round i , while the initial value of c_0 is synthesized randomly inside $[0,1]$. The optimum individual X_i updates its location by utilizing Equation 69:

$$X'_{i,j} = X_{i,j} + rand \times (c_i - 1)$$

Equation 69 Location update for the optimum individual.

where $X'_{i,j}$ represents the j -th part of the best solution.

Further on, the algorithm employs the Lévy flight procedure to enhance updates of the positions. By providing each solution the opportunity arises to occasionally execute long flight jumps. This facilitates escaping from suboptimal areas and improves the general performance of the individual particle. While the algorithm converges to the promising region, this long flight distance is dynamically decreased.

$$X'_{i,j} = X_{i,j}^{best} \times L \times e$$

$$e = \frac{\vec{a}}{2}$$

Equation 70 Lévy flight procedure to enhance updates of the positions.

where $X_{i,j}$ represent a individual's components that are modified, $X_{i,j}^{best}$ denotes the best solution, while e corresponds to the scaling factor and $\vec{a}$ is obtained with respect to the following:

$$\vec{a} = 1 + \cos\left(\pi + \frac{t}{T}\right)$$

Equation 71 Update rule for a particle's position.

where t marks the ongoing round, and T represents the maximum count of rounds. The value of L is the Lévy flight distribution which can be obtained by:

$$\phi = \left(\frac{\Gamma(1 + \tau) \times \sin\left(\frac{\pi \times \tau}{2}\right)}{\Gamma\left(\frac{1+\tau}{2}\right) \times \tau \times 2^{\frac{\tau-1}{2}}}\right)^{\frac{1}{\tau}}$$

Equation 72 Lévy flight distribution.

here, the Γ value is calculated by:

$$\Gamma(1 + \tau) = \int_0^{\infty} x^{\tau} e^{-x} dx$$

Equation 73 Gamma function.

Lastly, the third modification incorporates a low-level hybrid mechanism with the firefly algorithm (FA) [124]. The FA's search procedure given by Equation 74 is merged with the modified metaheuristics to further improve the exploration phase.

$$X_i^{t+1} = X_t^i + \beta_0 \cdot e^{-\gamma R_{ij}^2} (X_j^t - X_i^t) + \alpha^t (\kappa - 0.5)$$

Equation 74 The FA's search procedure.

where α is the randomization parameter, κ is a pseudo-random value drawn from the Gaussian distribution. The parameter γ represents the properties of the environment, β_0 denotes the maximal attractiveness of a single solution, and R_{ij} marks the range among solutions.

To allow fair metaheuristics contributions, it was necessary to implement a supplementary search control variable θ . If the value of θ is larger than the current iteration t , every individual randomly selects which algorithm will be employed. A random number is generated from a uniform distribution within $[0,1]$. If this value is larger than 0.5 the FA search procedure is employed. For this research, a θ value was empirically established, and the best outcomes were obtained at $\theta = 3$.

The formulated metaheuristics were labeled PSO with dynamic strategy change (PSO-DSC). The pseudocode of the PSO-DSC is explained in Algorithm 7.

```
Initialize parameters
Synthesize a populace  $X$  by employing chaotic initialization
while ( $t < T$ ) do
    Calculate fitness of all solutions and note the best
    for ( each solution in  $X$  ) do
        . if  $t > \theta$  then
        . . Utilize PSO search procedure
        . else
        . . Produce random number  $rnd$ 
        . . if  $rnd < 0.5$  then
        . . . Utilize PSO search procedure
        . . else
        . . . Utilize FA search procedure
     $t = t + 1$ 
Return the best solution from  $X$ 
```

5.8.2. Experimental Setup

The suggested methods are evaluated by employing real-world spot pricing data from Amazon EC2's 'us-west' area. The dataset covers huge Linux VMs. Inputs include 'c38xlarge', 'm44xlarge', and 'r38xlarge', with the final price forecasting one step ahead. The dataset analysis starts from 24.03.2017.

Due to the huge dataset sample size, a smaller sub-sample is used. Kaggle hosts the original dataset⁵. The available data is a 10-lag time-series. Data is split into training validation and evaluation subsets for investigation. An initial 70% (1200 samples) are allotted for training, 10% for validation and the remainder for evaluation.

⁵ <https://www.kaggle.com/datasets/berkayalan/aws-ec2-instance-comparison>

Several metaheuristic optimizers are applied to the optimization of both LSTM and GRU networks in two separate experiments. A comparative analysis is carried out between the introduced modified algorithm and the original PSO [32]. Additional algorithms included in the comparisons are "genetic algorithm" [57] (GA) and COLSHADE [60] optimizes. Well-established swarm optimizes such as the "firefly algorithm" [38] (FA), "artificial bee colony" [105] (ABC) and "Harris hawk optimizes" [114] (HHO) are also included. The "variable neighborhood search" [132] (VNS) algorithm is also used in the comparative analysis. Each optimizer is given a population size of five and five iterations to enhance population quality. To accommodate for randomness, simulations are conducted over 30 times independently. To prevent overtraining an early stopping criterion is used based on the equal to one-third of training epochs. Parameter ranges from both LSTM and GRU are optimized from a constrained including learning rate [0,0001,0.01], dropout [0.05,0.2], epochs [10,30], neurons per layer [10,30].

Evaluation of models involves standard regressing measures such as "root mean square error" (RMSE), "mean absolute error" (MAE), "mean square error" (MSE), and "coefficient of determination" (R^2). The objective metric is MSE, and R^2 is the indicator function. An additional metric, the index of agreement (IoA), was used to evaluate synthesized models (Eq. (13)).

$$IoA = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (|y_i - \bar{y}| + |y_i - \hat{y}_i|)^2}$$

Equation 75 Index of Agreement (IoA).

where y_i and $\hat{y}_i$ one more time denote the real and predicted values and $\bar{y}$ is the average value of real data. If this index has a significant value the model's performance is good, suggesting strong agreement.

A flowchart of the proposed experimental framework describing dataset preparation, and the optimization process is provided in Figure 32.

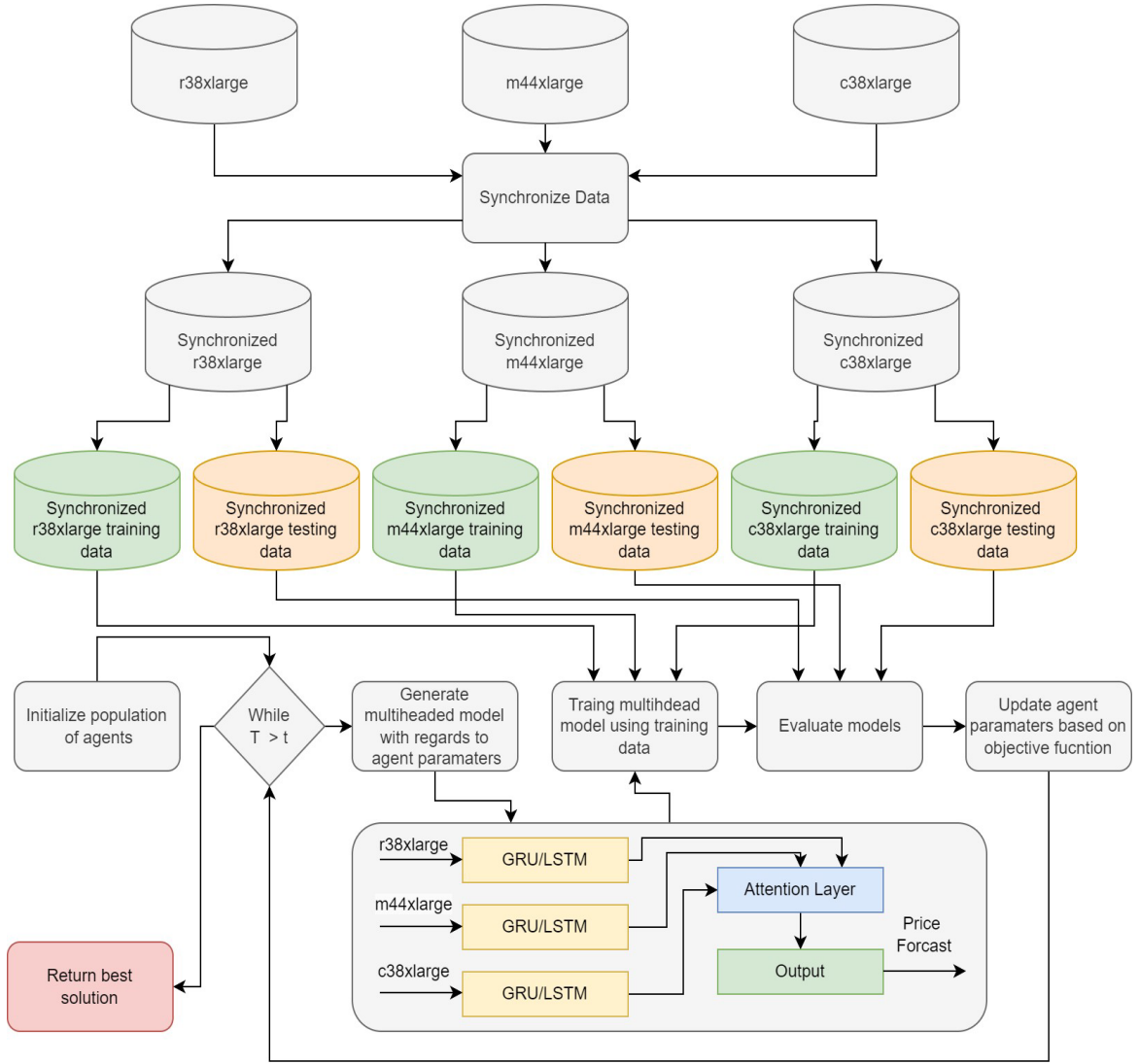


Figure 32 Introduced optimization framework flowchart.

5.8.3. Simulation Outcomes

This section covers the outcomes of conducted simulations with GRU and LSTM and a comparison between the Best LSTM and GRU models.

5.8.3.1. Simulations with LSTM Networks

The objective (MSE) function outputs for the best, worst mean, and median LSTM simulations are presented in Table 32, followed by comprehensive evaluations within the most promising models. The proposed optimizer demonstrates impressive outcomes, outperforming all competing metaheuristics demonstrating an MSE of 1.33E-06 in the best-

case execution. Furthermore, the proposed optimizer achieved the highest outcome for the median and mean cases demonstrating an objective function score of 1.48E-06 and 1.33E-06 respectively. The FA algorithm demonstrates adorable stability, with more stability evaluations shown in Figure 33. As showcased in the stability diagrams, the introduced algorithm outperforms the original algorithm in all cases as well as competing metaheuristics demonstrating better results. While the stability is comparatively lower than some optimizers such as the original PSO, GA, and FA, the results attained by these optimizers are below even the worst results attained by the introduced optimizer.

Table 32 Outcomes for LSTM simulations.

Overall Comparisons						
Method	Best	Worst	Mean	Median	Std	Var
MHLSTM-PSO-DSC	1.33E-06	1.66E-06	1.48E-06	1.33E-06	1.67E-07	2.77E-14
MHLSTM-PSO	1.54E-06	1.64E-06	1.59E-06	1.59E-06	5.03E-08	2.53E-15
MHLSTM-GA	1.53E-06	1.62E-06	1.57E-06	1.53E-06	4.51E-08	2.04E-15
MHLSTM-VNS	1.71E-06	1.74E-06	1.72E-06	1.71E-06	1.44E-08	2.08E-16
MHLSTM-ABC	1.67E-06	1.78E-06	1.73E-06	1.78E-06	5.55E-08	3.08E-15
MHLSTM-FA	1.69E-06	1.70E-06	1.69E-06	1.70E-06	2.17E-09	4.71E-18
MHLSTM-HHO	1.41E-06	1.68E-06	1.53E-06	1.41E-06	1.35E-07	1.81E-14
MHLSTM-COLSHADE	1.67E-06	1.89E-06	1.79E-06	1.89E-06	1.07E-07	1.15E-14

Detailed Comparisons					
Method	R2	MAE	MSE	RMSE	IoA
MHLSTM-PSO-DSC	0.191541	0.000908	1.33E-06	0.001152	0.531649
MHLSTM-PSO	0.061297	0.000990	1.54E-06	0.001241	0.510089
MHLSTM-GA	0.066289	0.001044	1.53E-06	0.001238	0.472012

MHLSTM-VNS	-0.043414	0.001118	1.71E-06	0.001308	0.142093
MHLSTM-ABC	-0.020013	0.001119	1.67E-06	0.001294	0.314240
MHLSTM-FA	-0.030752	0.001149	1.69E-06	0.001300	0.200151
MHLSTM-HHO	0.139422	0.000971	1.41E-06	0.001188	0.433447
MHLSTM-COLSHADE	-0.018482	0.001127	1.67E-06	0.001293	0.115080

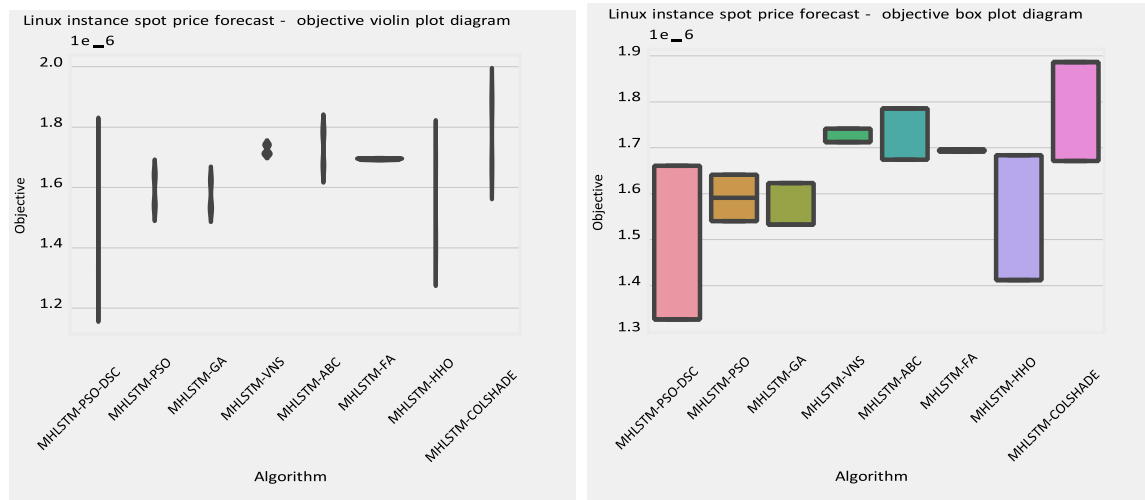


Figure 33 Objective function distribution plots for LSTM optimizations.

5.8.3.2. Simulations with GRU Networks

Outcomes in terms of objective (MSE) function for the best, worst mean and median GRU simulations followed by detailed comparisons between the best models are provided in Table 33. The introduced optimizer demonstrates impressive outcomes, outperforming all competing metaheuristics, producing an MSE of $1.16\text{E} - 06$ in the best-case execution, and outperforming LSTM models as well. Furthermore, the introduced optimizer attained the best outcome for the worst, median, and mean cases demonstrating an objective function score of respectively. The highest rate of stability is showcased by the original PSO algorithm despite this optimizer not matching the performance of the modified algorithm. Further comparisons in terms of stability are provided in Figure 34. As showcased in the stability diagrams, the introduced algorithm outperforms the original algorithm in all cases as well as competing metaheuristics demonstrating better results. While the stability is comparatively lower than some optimizers such as the original PSO, GA, and FA, the results

attained by these optimizers are below even the worst results attained by the introduced optimizer. The best-performing model was optimized by the introduced modified metaheuristic and demonstrates the best outcomes across all metrics for the given challenge.

Further details on optimizer capabilities to balance exploration and exploitation while avoiding local minimum traps are provided in Figure 35. The introduced optimizer manages to avoid a local trap in the third iteration of the optimization locating a more promising area of the search space by the 3rd iteration, suggesting that the introduced modifications help the modified algorithm locate more promising areas within the search space.

Parameter selections for the best GRU models are as follows: a learning rate of 0.000194, dropout of 0.050000, 30 training epochs neurons in the first head 11, 30 in the second and 14 in the third heads. A total of 28 is utilized in the dense concatenation layer.

Table 33 Outcomes for each optimizer included in comparative analysis.

Overall Comparisons						
Method	Best	Worst	Mean	Median	Std	Var
MHGRU-PSO-DSC	1.16E-06	1.23E-06	1.20E-06	1.23E-06	3.41E-08	1.16E-15
MHGRU-PSO	1.37E-06	1.38E-06	1.38E-06	1.37E-06	3.86E-09	1.49E-17
MHGRU-GA	1.64E-06	2.05E-06	1.90E-06	2.05E-06	1.97E-07	3.89E-14
MHGRU-VNS	1.32E-06	1.33E-06	1.32E-06	1.33E-06	5.86E-09	3.44E-17
MHGRU-ABC	1.41E-06	1.63E-06	1.52E-06	1.52E-06	1.09E-07	1.20E-14
MHGRU-FA	1.47E-06	1.48E-06	1.47E-06	1.47E-06	5.92E-09	3.51E-17
MHGRU-HHO	1.30E-06	1.62E-06	1.47E-06	1.62E-06	1.63E-07	2.65E-14
MHGRU-COLSHADE	1.69E-06	1.96E-06	1.81E-06	1.69E-06	1.32E-07	1.76E-14
Detailed Comparisons						
Method	R2	MAE	MSE	RMSE	IoA	
MHGRU-PSO-DSC	0.290191	0.000801	1.16E-06	0.001079	0.678492	
MHGRU-PSO	0.162884	0.000985	1.37E-06	0.001172	0.389012	

MHGRU-GA	0.000602	0.001105	1.64E-06	0.001281	0.056272
MHGRU-VNS	0.197970	0.000963	1.32E-06	0.001147	0.592289
MHGRU-ABC	0.139642	0.000987	1.41E-06	0.001188	0.435323
MHGRU-FA	0.107040	0.001065	1.47E-06	0.001210	0.435001
MHGRU-HHO	0.209853	0.000944	1.30E-06	0.001139	0.560155
MHGRU-COLSHADE	-0.030043	0.001084	1.69E-06	0.001300	0.179363

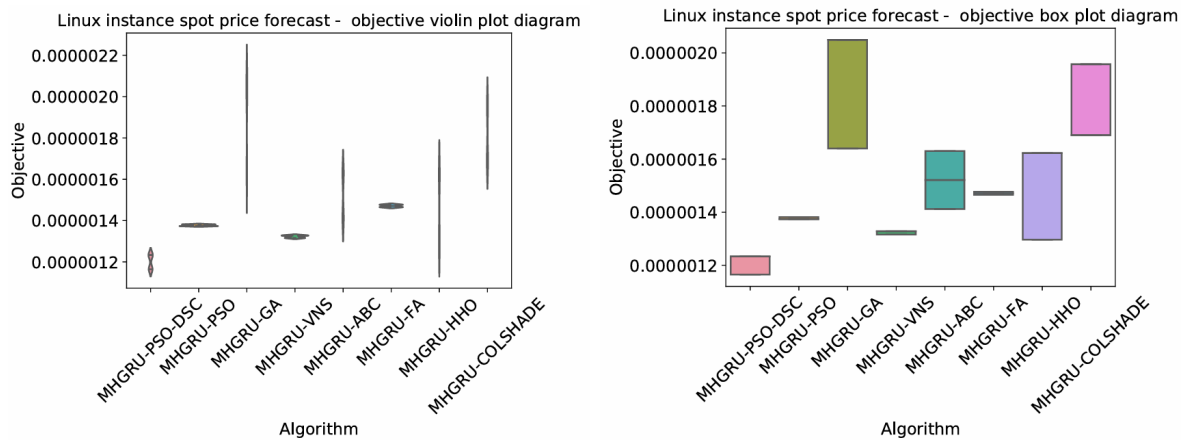


Figure 34 Objective function distribution plots for GRU optimizations.

5.8.3.3. Comparison between the Best LSTM and GRU Models

Both LSTM and GRU-optimized models have similar objective and indicator functions, with the modified PSO-DSC algorithm optimizing the best models. A comparison between the best performing GRU, LSTM, simple feed-forward artificial neural networks (ANN), and other contemporary regression forecasting models such as extreme gradient boosting (XGBoost), Support Vector Machines (SVM), decision trees and random forest models, as well as statistical forecasting techniques, such as autoregressive integrated moving average (AIRMA), is provided in Table 34 in terms of detailed metrics. The best-performing GRU model outperforms the best-performing LSTM across all metrics, confirming that the simple GRU architecture is better at forecasting cloud instance prices. This challenge benefits from the lower computational costs of GRU models in comparison to LSTM.

Table 34 Comparison between the best performing GRU and LSTM model.

Method	R2	MAE	MSE	RMSE	IoA
MHGRU-PSO-DSC	0.290191	0.000801	1.16E-06	0.001079	0.678492
MHLSTM-PSO-DSC	0.191541	0.000908	1.33E-06	0.001152	0.531649
Linear Regression	-1.933209	0.001710	4.68E-06	0.002163	0.446984
Decision Tree Regression	-3.268749	0.002200	6.81E-06	0.002609	0.393007
Random Forest Regression	-1.049327	0.0015143	3.27E-06	0.001808	0.435379
XGBoost	-0.989057	0.001508	3.17E-06	0.001781	0.444499
SVR	-0.048330	0.001143	1.67E-06	0.001293	0.264830
Feedforward ANN	-4.708735	0.002499	9.10E-06	0.003017	0.367674
ARIMA	-7.810849	0.003529	1.40E-05	0.003748	0.000000

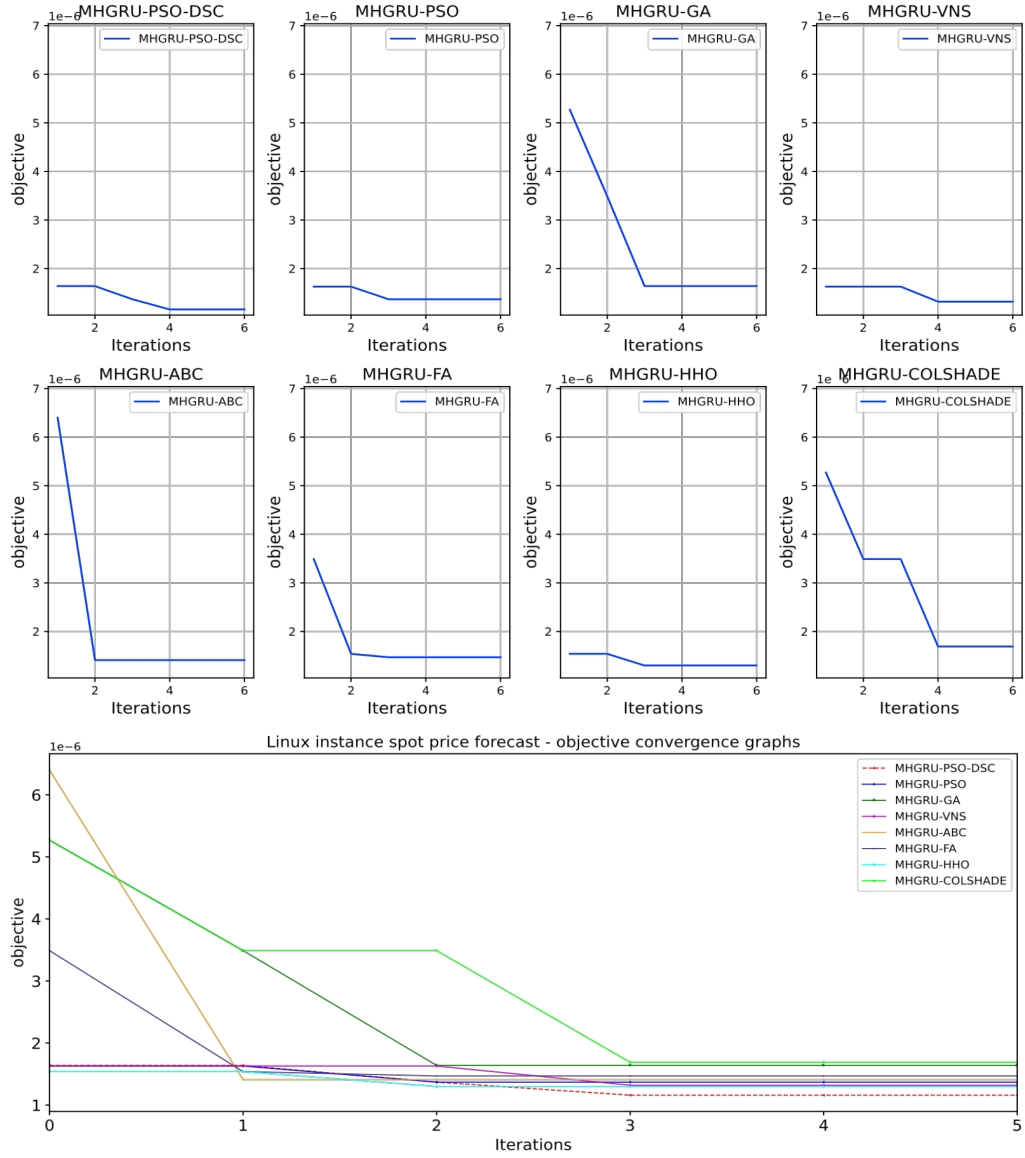


Figure 35 Convergence diagrams for GRU optimizations.

5.8.4. Outcomes Statistical Validation

Methods with randomness, repeatability, and reliability must be evaluated. Parametric tests have to meet independence, normality, and homoscedasticity [63] criteria to ensure safe utilization. By using independent random seeds the independence criterion is fulfilled. The homoscedasticity criterion was evaluated through the application of "Levene's test" [133], which yielded the p -value of 0.68 for each case, therefore this condition is also met. To confirm normality Shapiro-Wilk's single problem analysis is utilized [134] with outcomes shown in Table 35. Shapiro-Wilk's p -values were determined separately for each regarded approach all obtained p -values are below 0.05 and H_0 is rejected. Since the

conditions for parametric tests were not met, validation is conducted via non-parametric tests. The "Wilcoxon signed-rank" test was performed with the objective function outcomes (MSE). The outcomes of this test are provided in Table 35. As the p -values are in each case below threshold limit $\alpha = 0.05$, it means that the outcomes attained by the proposed PSO-DSC approach are statistically significantly superior to other methods regarded in comparative analysis.

Table 35 Statistical evaluation outcomes.

Shapiro-Wilk outcomes for PSO-DSC								
Approach	PSO-DSC	PSO	GA	VNS	ABC	FA	HHO	COLSHADE
MHGRU	0.024	0.028	0.028	0.033	0.035	0.030	0.028	0.040
MHLSTM	0.023	0.027	0.034	0.030	0.028	0.030	0.030	0.041
Wilcoxon signed-rank test outcomes for MHLSTM simulations								
PSO-DSC vs. others		PSO	GA	VNS	ABC	FA	HHO	COLSHADE
MHGRU		0.033	0.023	0.021	0.023	0.021	0.028	0.040
MHLSTM		0.037	0.022	0.020	0.023	0.022	0.032	0.040

5.8.5. Best-Performing Model Interpretation

Understanding the factors that influence a model's decisions is crucial. SHAP [7] is useful for obtaining insights. The best-constructed forecasting GRU model is subjected to SHAP interpretation. A high importance is given to r38xlarge samples 3 lags back followed by the same feature six months back. This is then followed by the m44xlrage one lag back. While their contribution is modest, the cumulative contributions contribute towards more accurate forecasting being made by the best-constructed model.

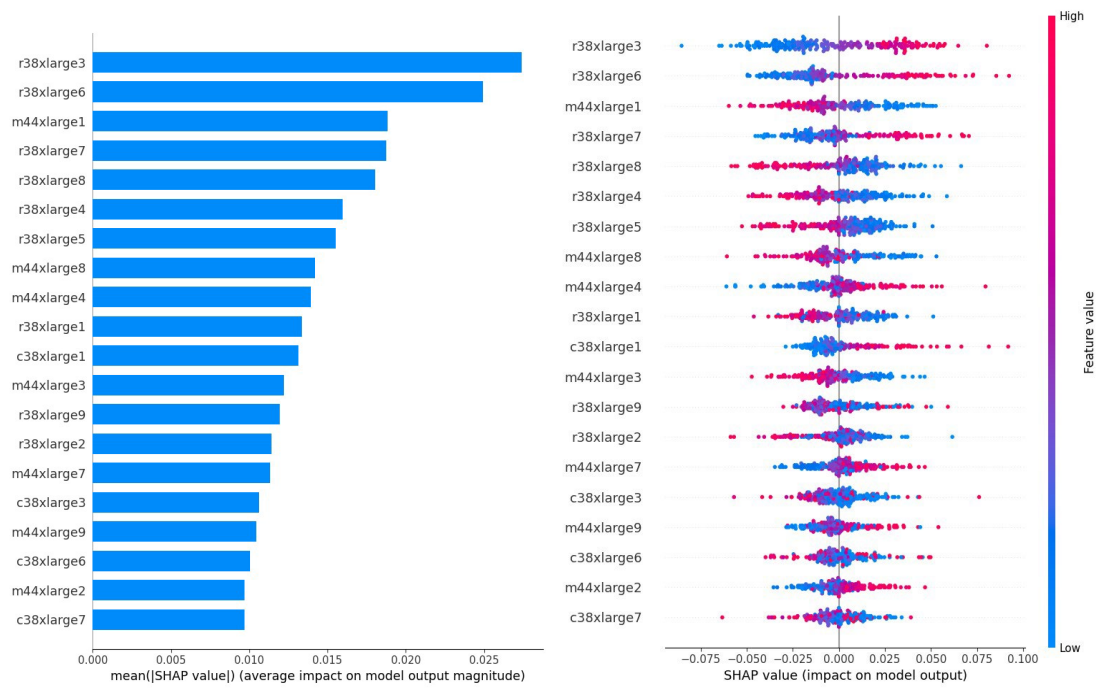


Figure 36 Best Performing GRU model SHAP interpretations.

6. Conclusion

This study investigated the combination of Recurrent Neural Networks (RNNs) with modified metaheuristic algorithms to tackle the difficulties of predicting time-series data in different practical scenarios. The outcomes of each application show the efficacy of these hybrid models in enhancing the accuracy and reliability of forecasting, consequently confirming the hypotheses put forward at the beginning of this thesis.

Our first study focused on the critical need for security in IoT systems. We have suggested a hybrid methodology that integrates Convolutional Neural Networks (CNNs) with Extreme Gradient Boosting (XGBoost) using a modified Reptile Search Algorithm (GIRSA) to enhance the optimization of hyperparameters. The model obtained an exceptional overall accuracy of 85.97%, exceeding previous models. It showed promising results in enhancing classification performance for identifying abnormal behaviors and certain sorts of attacks, especially in situations involving numerous classifications. [1]

The second application was centered on the prediction of wind energy. The hybrid model we used involved the integration of an Enhanced Sine Cosine Algorithm (ESCA) with Long Short-Term Memory (LSTM) networks. We also employed Variation Mode Decomposition (VMD) for data preprocessing. The hybrid ESCA-LSTM model attained an overall R^2 rate of 0.931, effectively capturing intricate, non-linear characteristics in the wind energy data. Consequently, this led to a notable enhancement in the accuracy of the forecasts. The suggested technique achieved superior performance in comprehensive evaluations, outperforming all other algorithms. Furthermore, it displayed the capacity of deep learning methodologies to improve the efficiency of contemporary power systems, therefore establishing its value as an important tool for overseeing renewable energy. [2]

Our work on spam email classification demonstrated the efficacy of the enhanced Moth Flame Optimizer (MFO) in improving logistic regression models, which was the third application in the field of cybersecurity. The AVMFO-LR approach demonstrated superior performance compared to other methods, achieving an accuracy rate of 98.81% on the CSDMC2010 dataset with a feature vector size of 1000. This indicates that it is a reliable and effective technique for spam identification. [3]

We utilized recurrent neural networks (RNNs) tuned with advanced metaheuristics in our fourth application, which focused on detecting seizures in patients with epilepsy. The RNN-MSCHO technique outperformed previous methods and greatly improved the performance of the RNN model. It achieved an outstanding accuracy of 99.88%, with precision, recall, and f1-scores all above 99.68%. This technique presents a hopeful resolution for immediate medical diagnostics, improving the standard of living for people with epilepsy. [4]

In our fifth application, we utilized an advanced optimization algorithm called the Enhanced Harris Hawks Optimization Algorithm (HHO-DO) to improve Long Short-Term Memory (LSTM) networks to predict crude oil prices in financial forecasting. In addition to Variational Mode Decomposition (VMD), this hybrid model exhibited greater performance in comparison to other modern metaheuristics. The model scored an R^2 value of 0.992 and an RMSE of 0.013228, offering significant insights for investors in the energy industry and effectively handling the intricacies of crude oil price data. [5]

The sixth application we developed for predicting earthquake magnitude utilized LSTM models that were tuned using a modified Sine Cosine Algorithm. The technique has shown positive results in forecasting earthquake magnitudes, emphasizing its worth for disaster management. The model's R^2 values for one-step, two-step, and three-step forward predictions were -0.217610, -0.178719, and -0.184096 correspondingly, suggesting that more refining is necessary. Given these difficulties, the model's ability to accurately represent the intricate and ever-changing characteristics of seismic data highlights its importance in reducing the effects of natural disasters.

Finally, our study into cloud instance price forecasting tackled a complex and significant issue within the domain of cloud computing. As the need for cloud services grows, a consistent and predictable pricing model is crucial for organizations to effectively manage spending and enhance operations. This study investigated multi-headed recurrent architectures, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, enhanced by a modified Particle Swarm Optimization (PSO) algorithm for parameter tuning. The GRU models optimized via this technique achieved optimal performance, demonstrated by a Mean Absolute Error (MAE) score of 0.000801. The adjustments made to the PSO enhanced convergence rates and reduced premature stagnation,

allowing more efficient exploration of the hyperparameter space. The results were rigorously analyzed statistically, and the optimal models were assessed using explainable AI methods to identify key features influencing forecasts. [6]

Every hypothesis proposed in this thesis was proven correct. The combination of powerful machine learning models and modified optimization techniques resulted in exceptional performance across a range of practical applications.

The results of this thesis have important consequences, providing novel approaches for sophisticated prediction challenges. The combination of Recurrent Neural Networks (RNNs) with metaheuristics improves the accuracy and reliability of forecasting, establishing a basis for future advancements in time-series forecasting. The approaches proposed here give useful tools for ongoing study and application as the problems in data analysis and prediction rise.

Further studies will be focused on the evaluation of more adapted algorithms for diverse real-life scenarios, including climate forecasting, stock market prediction, and personalized healthcare. Our objective is to enhance predictive modeling and its practical applications by improving these models and utilizing them on various datasets.

7. References

- [1] M. Salb, L. Jovanovic, N. Bacanin, M. Antonijevic, M. Zivkovic, N. Budimirovic i L. Abualigah, „Enhancing internet of things network security using hybrid CNN and xgboost model tuned via modified reptile search algorithm,“ *Applied Sciences*, t. 46, br. 23, p. 12687, 2023.
- [2] M. Salb, L. Jovanovic, N. Bacanin, G. Kunjadic, M. Antonijevic, M. Zivkovic i V. K. Devi, „The Long Short-Term Memory Tuning for Multi-step Ahead Wind Energy Forecasting Using Enhanced Sine Cosine Algorithm and Variation Mode Decomposition,“ *International Conference on Paradigms of Communication, Computing and Data Analytics*, pp. 31-43, 2023.
- [3] M. Salb, L. Jovanovic, M. Zivkovic, E. Tuba, A. Elsadai i N. Bacanin, „Training logistic regression model by enhanced moth flame optimizer for spam email classification,“ *Computer networks and inventive communication technologies: Proceedings of fifth ICCNCT 2022*, pp. 753-768, 2022.
- [4] S. Kozakijevic, M. Salb, A. Elsadai, J. Mani, K. Devi, A. D. Sharko i S. Muthusamy, „Seizure detection via time series classification using modified metaheuristic optimized recurrent networks,“ *Theoretical and Applied Computational Intelligence*, t. 1, br. 1, pp. 82-94, 2023.
- [5] L. Jovanovic, M. Antonijevic, M. Zivkovic, M. Dobrojevic, M. Salb, I. Strumberger i N. Bacanin, „Long short-term memory tuning by enhanced Harris hawks optimization algorithm for crude oil price forecasting,“ 2024.
- [6] M. Salb, L. Jovanovic, A. Elsadai, N. Bacanin, V. Simic, D. Pamucar i M. Zivkovic, „Cloud spot instance price forecasting multi-headed models tuned using modified PSO,“ *Journal of King Saud University-Science*, p. 103473, 2024.
- [7] S. M. Lundberg i S.-I. Lee, „A unified approach to interpreting model predictions,“ *Advances in neural information processing systems*, t. 30, 2017.
- [8] K. G. Murty, „A new interior variant of the gradient projection method for linear programming,“ 1985.

- [9] R. Bellman, „Dynamic programming, princeton, nj: Princeton univ,“ *iversity Press. BellmanDynamic Programming*1957, 1957.
- [10] H. W. Kuhn i A. W. Tucker, *Nonlinear programming, paper presented at Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, Calif, 1951.
- [11] J. B. Rosen, „The gradient projection method for nonlinear programming. Part I. Linear constraints,“ *Journal of the society for industrial and applied mathematics*, t. 8, p. 181–217, 1960.
- [12] G. Zoutendijk, „Methods of feasible directions,“ *1960*, 1960.
- [13] R. E. Gomory, „Solving linear programming problems in integers,“ *Combinatorial Analysis*, t. 10, p. 211–215, 1960.
- [14] R. J. Duffin i C. Zener, „Optimization of engineering design by geometric programming,“ u *Transactions of the Conference of Arsenal Mathematicians*, 1967.
- [15] A. Charnes, W. W. Cooper i G. H. Symonds, „Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil,“ *Management science*, t. 4, p. 235–263, 1958.
- [16] A. Charnes i W. W. Cooper, „Chance-constrained programming,“ *Management science*, t. 6, p. 73–79, 1959.
- [17] G. B. Dantzig, „Linear programming under uncertainty,“ *Management science*, t. 1, p. 197–206, 1955.
- [18] S. S. Rao, *Engineering optimization: theory and practice*, John Wiley & Sons, 2019.
- [19] X.-S. Yang, „Nature-inspired optimization algorithms: Challenges and open problems,“ *Journal of Computational Science*, t. 46, p. 101104, 2020.
- [20] T. Wei, S. Wang, J. Zhong, D. Liu i J. Zhang, „A review on evolutionary multitask optimization: Trends and challenges,“ *IEEE Transactions on Evolutionary Computation*, t. 26, p. 941–960, 2021.
- [21] F. Peres i M. Castelli, „Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development,“ *Applied Sciences*, t. 11, p. 6449, 2021.

- [22] T. Bäck, D. B. Fogel i Z. Michalewicz, „Handbook of evolutionary computation,“ t. 97, br. 1, p. B1, 1997.
- [23] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan i Q. Zhang, „Multiobjective evolutionary algorithms: A survey of the state of the art,“ *Swarm and evolutionary computation*, t. 1, br. 1, pp. 32-49, 2011.
- [24] B. Xue, M. Zhang, W. N. Browne i X. Yao, „A survey on evolutionary computation approaches to feature selection,“ *IEEE Transactions on evolutionary computation*, t. 20, br. 4, pp. 606-626, 2015.
- [25] P. G. Espejo, S. Ventura i F. Herrera, „A survey on the application of genetic programming to classification,“ *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, t. 20, br. 2, pp. 121-144, 2009.
- [26] F. Neri i V. Tirronen, „Recent advances in differential evolution: a survey and experimental analysis,“ *Artificial intelligence review*, t. 33, pp. 61-106, 2010.
- [27] J. Zhang, Z.-h. Zhan, Y. Lin, N. Chen, Y.-j. Gong, J.-h. Zhong, H. S. Chung, Y. Li i Y.-h. Shi, „Evolutionary computation meets machine learning: A survey,“ *IEEE Computational Intelligence Magazine*, t. 6, br. 4, pp. 68-75, 2011.
- [28] E. Bonabeau, D. d. R. D. F. Marco, M. Dorigo, G. Théraulaz, G. Theraulaz i others, „Swarm intelligence: from natural to artificial systems,“ *Oxford university press*, t. 1, 1999.
- [29] P. Miller, „The genius of swarms,“ *National Geographic*, t. 212, br. 1, pp. 126-147, 2007.
- [30] W. J. Beni G, „Swarm Intelligence,“ u *7th Ann Meeting of the Robotics Society of Japan*, Japan, 1989.
- [31] G. Beni i J. Wang, „Swarm intelligence in cellular robotic systems,“ *Robots and biological systems: towards a new bionics?*, pp. 703-712, 1993.
- [32] J. Kennedy i R. Eberhart, „Particle swarm optimization,“ *Proceedings of ICNN'95-international conference on neural networks*, t. 4, pp. 1942-1948, 1995.
- [33] É. Barbin, „A history of algorithms,“ u *the pebble to the microchip*, t. 23, J. Chabert, Ur., Berlin: Springer, 1999.

- [34] X.-S. Yang, S. Deb, S. Fong, X. He i Y.-X. Zhao, „From swarm intelligence to metaheuristics,“ *Computer*, t. 49, br. 9, pp. 52-59, 2016.
- [35] M. Dorigo, M. Birattari i T. Stutzle, „Ant colony optimization,“ *IEEE computational intelligence magazine*, t. 1, br. 4, pp. 28-39, 2006.
- [36] D. Karaboga, „An idea based on honey bee swarm for numerical optimization,“ *Technical report-tr06*, t. 200, pp. 1-10, 2005.
- [37] X.-S. Yang, „A new metaheuristic bat-inspired algorithm,“ *Nature inspired cooperative strategies for optimization (NISCO 2010)*, pp. 65-74, 2010.
- [38] X.-S. Yang, „Firefly algorithms for multimodal optimization,“ *International symposium on stochastic algorithms*, pp. 169-178, 2009.
- [39] X.-S. Yang i S. Deb, „Engineering optimisation by cuckoo search,“ *International Journal of Mathematical Modelling and Numerical Optimisation*, t. 1, br. 4, pp. 330-343, 2010.
- [40] S. Mirjalili i A. Lewis, „The whale optimization algorithm,“ *Advances in engineering software*, t. 95, pp. 51-67, 2016.
- [41] A. Ghubaish, T. Salman, M. Zolanvari, D. Unal, A. Al-Ali i R. Jain, „Recent advances in the internet-of-medical-things (IoMT) systems security,“ *IEEE Internet of Things Journal*, t. 8, br. 11, pp. 8707-8718, 2020.
- [42] D. Gupta, O. Kayode, S. Bhatt, M. Gupta i A. S. Tosun, „ierarchical federated learning based anomaly detection using digital twins for smart healthcare,“ u *2021 IEEE 7th International Conference on Collaboration and Internet Computing (CIC)*, IEEE, 2021, pp. 16-25.
- [43] C. Turcu i C. Turcu, „Improving the quality of healthcare through internet of things,“ *arXiv preprint arXiv:1903.05221*, 2019.
- [44] K. Valtanen, J. Backman i S. Yrjölä, „Blockchain-powered value creation in the 5G and smart grid use cases,“ *IEEE Access*, t. 7, pp. 25690-25707, 2019.
- [45] H. Okuhara, A. Elnaqib, M. Dazzi, P. Palestri, S. Benatti, L. Benini i D. Rossi, „A Fully Integrated 5-mW, 0.8-Gbps Energy-Efficient Chip-to-Chip Data Link for Ultralow-Power IoT End-Nodes in 65-nm CMOS,“ *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, t. 29, br. 10, pp. 1800-1811, 2021.

- [46] Z. Luo, W. Wang, J. Qu, T. Jiang i Q. Zhang, „ShieldScatter: Improving IoT security with backscatter assistance,“ u *Proceedings of the 16th ACM conference on embedded networked sensor systems*, 2018, pp. 185-198.
- [47] P. Gupta, V. Dedeoglu, K. Najeebullah, S. S. Kanhere i R. Jurdak, „Energy-aware demand selection and allocation for real-time IoT data trading,“ u *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, 2020, pp. 138-147.
- [48] F. Azzedin i T. Alhazmi, „Secure data distribution architecture in IoT using MQTT,“ *MDPI*, t. 13, br. 4, p. 2515, 2023.
- [49] A. J. Hintaw, S. Manickam, M. F. Aboalmaaly i S. Karuppayah, „MQTT vulnerabilities, attack vectors and solutions in the internet of things (IoT),“ *IETE Journal of Research*, t. 69, br. 6, pp. 3368-3397, 2023.
- [50] M. Kodyš, L. K. W. F. Zhi i L. T. Vrizlynn, „Intrusion detection in internet of things using convolutional neural networks,“ u *2021 18th International Conference on Privacy, Security and Trust (PST)*, IEEE, 2021, pp. 1-10.
- [51] V. Ayumi, L. R. Rere, M. I. Fanany i A. M. Arymurthy, „Optimization of convolutional neural network using microcanonical annealing algorithm,“ u *2016 International Conference on Advanced Computer Science and Information Systems (ICACISIS)*, IEEE, 2016, pp. 506-511.
- [52] J. Li, Z. Zhao, R. Li i H. Zhang, „Ai-based two-stage intrusion detection for software defined iot networks,“ *IEEE Internet of Things Journal*, t. 6, br. 2, pp. 2093-2102, 2018.
- [53] X. Xiao, M. Yan, S. Basodi, C. Ji i Y. Pan, „Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm,“ *arXiv preprint arXiv:2006.12703*, 2020.
- [54] Z. Guo i Y. Cao, „SA-CNN: Application to text categorization issues using simulated annealing-based convolutional neural network optimization,“ u *Proceedings of the 2022 6th International Conference on Electronic Information Technology and Computer Engineering*, 2022, pp. 1932-1939.
- [55] L. Abualigah, M. Abd Elaziz, P. Sumari, Z. W. Geem i A. H. Gandomi, „Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer,“ *Expert Systems with Applications*, t. 191, p. 116158, 2022.

- [56] S. Jiang, S. Yang, X. Yao, K. C. Tan, M. Kaiser i N. Krasnogor, „Benchmark functions for the cec'2018 competition on dynamic multiobjective optimization,“ Newcastle University, 2018.
- [57] S. Mirjalili, „Genetic algorithm,“ *Evolutionary algorithms and neural networks: Theory and applications*, pp. 43-55, 2019.
- [58] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli i E. Cambiaso, „MQTTset, a new dataset for machine learning techniques on MQTT,“ *Sensors*, t. 20, br. 22, p. 6578, 2020.
- [59] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris i S. M. Mirjalili, „Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems,“ *Advances in engineering software*, pp. 163-191, 2017.
- [60] J. Gurrola-Ramos, A. Hernández-Aguirre i O. Dalmau-Cedeño, „COLSHADE for real-world single-objective constrained optimization problems,“ u *2020 IEEE congress on evolutionary computation (CEC)*, IEEE, 2020, pp. 1-8.
- [61] M. Hossin i M. N. Sulaiman, „A review on evaluation metrics for data classification evaluations,“ *Academy & Industry Research Collaboration Center (AIRCC)*, t. 5, br. 2, p. 1, 2015.
- [62] M. J. Warrens, „Five ways to look at Cohen's kappa,“ *Journal of Psychology & Psychotherapy*, t. 5, 2015.
- [63] A. LaTorre, D. Molina, E. Osaba, J. Poyatos, J. Del Ser i F. Herrera, „A prescription of methodological guidelines for comparing bio-inspired optimization algorithms,“ *Swarm and Evolutionary Computation*, t. 67, p. 100973, 2021.
- [64] F. Wilcoxon, „Individual comparisons by ranking methods,“ u *Breakthroughs in statistics: Methodology and distribution*, 1992, pp. 196--202.
- [65] K. Dragomiretskiy i D. Zosso, „Variational mode decomposition,“ *IEEE transactions on signal processing*, t. 62, br. 3, pp. 531-544, 2013.
- [66] S. Hochreiter i S. Jürgen, „Long short-term memory,“ *Neural computation*, t. 9, br. 8, pp. 1735-1780, 1997.
- [67] S. Mirjalili, „SCA: a sine cosine algorithm for solving optimization problems,“ *Knowledge-based systems*, t. 96, pp. 120-133, 2016.

- [68] S. Hochreiter i S. Jürgen, „LSTM can solve hard long time lag problems,“ *Advances in neural information processing systems*, p. 9, 1996.
- [69] Abualigah, Laith and Abd Elaziz, Mohamed and Sumari, Putra and Geem, Zong Woo and Gandomi, Amir H, „Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer,“ *Expert Systems with Applications*, t. 191, p. 116158, 2022.
- [70] S. Radicati i J. Levenstein, „Email statistics report, 2010,“ *The Radicati Group, Inc. Retrieved*, 2010.
- [71] T. M. Ma, K. Yamamori i A. Thida, „A comparative approach to Naïve Bayes classifier and support vector machine for email spam classification,“ u *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, IEEE, 2020, pp. 324--326.
- [72] A. Bhowmick i S. M. Hazarika, „E-mail spam filtering: a review of techniques and trends,“ *Advances in Electronics, Communication and Computing: ETAEERE-2016*, pp. 583-590, 2018.
- [73] K. Kottursamy, „A review on finding efficient approach to detect customer emotion analysis using deep learning analysis,“ *Journal of Trends in Computer Science and Smart Technology*, t. 3, br. 2, pp. 95-113, 2021.
- [74] D. Sarma, T. Mitra, R. M. Bawm, T. Sarwar, F. F. Lima i S. Hossain, „Comparative analysis of machine learning algorithms for phishing website detection,“ *Inventive Computation and Information Technologies: Proceedings of ICICIT 2020*, pp. 883-896, 2021.
- [75] S. Mirjalili, „Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,“ *Knowledge-based systems*, t. 89, pp. 228-249, 2015.
- [76] I. Strumberger, M. Sarac, D. Markovic i N. Bacanin, „Moth search algorithm for drone placement problem,“ *International Journal of Computers*, t. 3, p. 3, 2018.
- [77] N. Bacanin i M. Tuba, „rtificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators,“ *Studies in Informatics and Control*, t. 21, br. 2, pp. 137-146, 2012.
- [78] V. Metsis, I. Androutsopoulos i G. Paliouras, „Spam filtering with naive bayes-which naive bayes?,“ u *CEAS*, CA, Mountain View, 2006, pp. 28-69.

- [79] S. Bird, E. Klein i E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, O'Reilly Media, Inc., 2006.
- [80] B. K. Dedetürk i B. Akay, „Spam filtering using a logistic regression model trained by an artificial bee colony algorithm,“ *Applied Soft Computing*, t. 91, p. 106229, 2020.
- [81] A. Barushka i P. Hajek, „Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks,“ *Applied Intelligence*, t. 48, pp. 3538-3556, 2018.
- [82] S. K. Trivedi i S. Dey, „A combining classifiers approach for detecting email spams,“ *2016 30th international conference on advanced information networking and applications workshops (WAINA)*, pp. 355-360, 2016.
- [83] G. Sanghani i K. Kotecha, „Personalized spam filtering using incremental training of support vector machine,“ *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, pp. 323-328, 2016.
- [84] K. Chhogyal i A. Nayak, „An empirical study of a simple naive bayes classifier based on ranking functions,“ u *AI 2016: Advances in Artificial Intelligence: 29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016, Proceedings 29*, Springer, 2016, pp. 324-331.
- [85] D. Hassan, „Investigating the effect of combining text clustering with classification on improving spam email detection,“ u *Intelligent Systems Design and Applications: 16th International Conference on Intelligent Systems Design and Applications (ISDA 2016) held in Porto, Portugal, December 16-18, 2016*, Springer, 2017, pp. 99-107.
- [86] S. K. Trivedi i S. Dey, „A comparative study of various supervised feature selection methods for spam classification,“ u *Proceedings of the second international conference on information and communication technology for competitive strategies*, 2016, pp. 1-6.
- [87] R. Mishra i R. Thakur, „Analysis of random forest and Naive Bayes for spam mail using feature selection categorization,“ *International Journal of Computer Applications*, t. 80, br. 3, pp. 42-47, 2013.

- [88] S. K. Trivedi i S. Dey, „An enhanced genetic programming approach for detecting unsolicited emails,“ u *2013 IEEE 16th International Conference on Computational Science and Engineering*, IEEE, 2013, pp. 1153-1160.
- [89] R. Shams i R. E. Mercer, „Personalized spam filtering with natural language attributes,“ *2013 12th International conference on machine learning and applications*, t. 2, pp. 127-132, 2013.
- [90] T. A. Almeida i A. Yamakami, „Occam’s razor-based spam filter,“ *Journal of Internet Services and Applications*, t. 3, pp. 245-253, 2012.
- [91] A. K. Uysal i S. Gunal, „A novel probabilistic feature selection method for text classification,“ *Knowledge-Based Systems*, t. 36, pp. 223-235, 2012.
- [92] T. A. Almeida, J. Almeida i A. Yamakami, „Spam filtering: how the dimensionality reduction affects the accuracy of Naive Bayes classifiers,“ *Journal of Internet Services and Applications*, t. 1, pp. 183-200, 2011.
- [93] A. Abi-Haidar i L. M. Rocha, „Adaptive Spam Detection Inspired by the Immune System.,“ *ALIFE*, pp. 1-8, 2008.
- [94] Tzortzis, Grigorios; Likas, Aristidis, „Deep belief networks for spam filtering,“ *19th IEEE International conference on tools with artificial intelligence (ICTAI 2007)*, t. 2, pp. 306-309, 2007.
- [95] E. Beghi, „The epidemiology of epilepsy,“ *Neuroepidemiology*, t. 54, br. 2, pp. 185-191, 2020.
- [96] S. R. Benbadis, B. Sándor, B. Edward, M. Stephanie i L. M. Solomon, „The role of EEG in patients with suspected epilepsy,“ *Epileptic Disorders*, t. 22, br. 2, pp. 143-155, 2020.
- [97] S. Noachtar i J. Rémi, „The role of EEG in epilepsy: a critical review,“ *Epilepsy & Behavior*, t. 15, br. 1, pp. 22-22, 2009.
- [98] L. Benarous, M. Gavaret, M. S. a. T. J. Diop, S. d. G. de Bourmont, C. Allez, F. Bouzana, M. Gainnier i A. Trebuchon, „Sources of interrater variability and prognostic value of standardized EEG features in post-anoxic coma after resuscitated cardiac arrest,“ *Clinical neurophysiology practice*, t. 4, pp. 20-26, 2019.

- [99] T. Kaur, A. Diwakar, P. Mirpuri, M. Tripathi, P. S. Chandra, T. K. Gandhi i others, „Artificial intelligence in epilepsy,“ *Neurology India*, t. 69, br. 3, pp. 560-566, 2021.
- [100] L. Jovanovic, N. Bacanin, M. Zivkovic, M. Antonijevic, B. Jovanovic, M. B. Sretenovic i I. Strumberger, „Machine learning tuning by diversity oriented firefly metaheuristics for industry 4.0,“ *Expert Systems*, t. 41, br. 2, p. e13293, 2024.
- [101] L. R. Medsker, L. Jain i others, „Recurrent neural networks,“ *Design and Applications*, t. 5, br. 2, pp. 64-67, 2001.
- [102] N. Bacanin, C. Stoean, M. Zivkovic, M. Rakic, R. Strulak-Wojcikiewicz i R. Stoean, „On the benefits of using metaheuristics in the hyperparameter tuning of deep learning models for energy load forecasting,“ *Energies*, t. 16, br. 3, p. 1434, 2023.
- [103] J. Bai, Y. Li, M. Zheng, S. Khatir, B. Benaissa, L. Abualigah i M. A. Wahab, „A sinh cosh optimizer,“ *Knowledge-Based Systems*, t. 282, p. 111081, 2023.
- [104] Jiang, Shouyong and Yang, Shengxiang and Yao, Xin and Tan, Kay Chen and Kaiser, Marcus and Krasnogor, Natalio, „Benchmark functions for the cec'2018 competition on dynamic multiobjective optimization,“ Newcastle University, 2018.
- [105] Karaboga, Dervis; Basturk, Bahriye, „A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,“ *Journal of global optimization*, t. 39, pp. 459-471, 2007.
- [106] N. Bacanin, N. Budimirovic, K. Venkatachalam, H. S. Jassim, M. Zivkovic, S. Askar i M. Abouhawwash, „Quasi-reflection learning arithmetic optimization algorithm firefly search for feature selection,“ *Heliyon*, t. 9, br. 4, 2023.
- [107] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David i C. E. Elger, „Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state,“ *Physical Review E*, t. 64, br. 6, p. 061907, 2001.
- [108] D. Sarkar i A. Biswas, „Comparative performance analysis of recent evolutionary algorithms,“ *Evolution in Computational Intelligence: Proceedings of the 9th*

International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA 2021, pp. 151-159, 2022.

- [109] D. H. Wolpert i W. G. Macready, „No free lunch theorems for optimization,“ *IEEE transactions on evolutionary computation*, t. 1, br. 1, pp. 67-82, 1997.
- [110] S. Mirjalili, S. M. Mirjalili i A. Lewis, „Grey wolf optimizer,“ *Advances in engineering software*, t. 69, pp. 46-61, 2014.
- [111] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz i A. H. Gandomi, „The arithmetic optimization algorithm,“ *Computer methods in applied mechanics and engineering*, t. 376, p. 113609, 2021.
- [112] D. Sarkar, N. Mishra i A. Biswas, „Genetic algorithm-based deep learning models: a design perspective,“ u *Proceedings of the Seventh International Conference on Mathematics and Computing: ICMC 2021*, Springer, 2022, pp. 361-372.
- [113] H. Niu i Y. Zhao, „Crude oil prices and volatility prediction by a hybrid model based on kernel extreme learning machine,“ *Mathematical Biosciences and Engineering*, t. 18, br. 6, pp. 8096-8122, 2021.
- [114] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja i H. Chen, „Harris hawks optimization: Algorithm and applications,“ *Future generation computer systems*, t. 97, pp. 849-872, 2019.
- [115] S. Talatahari, H. Bayzidi i M. Saraee, „ocial network search for global optimization,“ *IEEE Access*, t. 9, pp. 92815-92863, 2021.
- [116] R. V. Rao, V. J. Savsani i D. P. Vakharia, „Teaching--learning-based optimization: a novel method for constrained mechanical design optimization problems,“ *Computer-aided design*, t. 43, br. 3, pp. 303-315, 2011.
- [117] N. Yang, G. Li, P. Zhao, J. Zhang i D. Zhao, „Porosity prediction from pre-stack seismic data via a data-driven approach,“ *Journal of Applied Geophysics*, t. 211, p. 104947, 2023.
- [118] L. Karataş, T. Ateş, A. Alptekin, M. Dal i M. Yakar, „A systematic method for post-earthquake damage assessment: Case study of the Antep Castle, Turkiye,“ *Advanced Engineering Science*, t. 3, pp. 62-71, 2023.

- [119] S. Pulinets, D. Ouzounov, A. Karelin i K. Boyarchuk, *Earthquake Precursors in the Atmosphere and Ionosphere*, Springer, 2022.
- [120] K. Im i J.-P. Avouac, „Cascading foreshocks, aftershocks and earthquake swarms in a discrete fault network,“ *Geophysical Journal International*, t. 235, br. 1, pp. 831-852, 2023.
- [121] B. e. al, „Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges,“ *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, t. 13, br. 2, p. e1484, 2023.
- [122] R. Cahuantzi, X. Chen i S. Güttel, „A comparison of LSTM and GRU networks for learning symbolic sequences,“ *Science and Information Conference*, pp. 771-785, 2023.
- [123] J. Basha, N. Bacanin, N. Vukobrat, M. Zivkovic, K. Venkatachalam, S. Hubálovský i P. Trojovský, „Chaotic harris hawks optimization with quasi-reflection-based learning: An application to enhance cnn design,“ *Sensors*, t. 21, br. 19, p. 6654, 2021.
- [124] X.-S. Yang i X. He, „Firefly algorithm: recent advances and applications,“ *International journal of swarm intelligence*, t. 1, br. 1, pp. 36-50, 2013.
- [125] Abdullah, Jaza Mahmood; Ahmed, Tarik, „Fitness dependent optimizer: inspired by the bee swarming reproductive process,“ *IEEE Access*, t. 7, pp. 43473-43486, 2019.
- [126] R. Cheng, M. Li, Y. Tian, X. Xiang, X. Zhang, S. Yang, Y. Jin i X. Yao, „Benchmark functions for the cec'2018 competition on many-objective optimization,“ 2018.
- [127] A. K. Yazdi i H. Komasi, „Best practice performance of COVID-19 in America continent with artificial intelligence,“ *Spectrum of Operational Research*, t. 1, br. 1, pp. 1-12, 2024.
- [128] R. Younas, H. B. U. Haq i M. D. Baig, „A framework for extensive content-based image retrieval system incorporating relevance feedback and query suggestion,“ *Spectrum of Operational Research*, t. 1, br. 1, pp. 13-32, 2024.
- [129] R. Dey i F. M. Salem, „Gate-variants of gated recurrent unit (GRU) neural networks,“ *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pp. 1597-1600, 2017.

- [130] K. Greff, S. Rupesh K., K. Jan, S. Bas R. i S. Jürgen, „LSTM: A search space odyssey,“ *IEEE transactions on neural networks and learning systems*, t. 28, br. 10, pp. 2222-2232, 2016.
- [131] S.-J. Bu i S.-B. Cho, „Time series forecasting with multi-headed attention-based deep learning for residential energy consumption,“ *Energies*, t. 13, br. 18, p. 4722, 2020.
- [132] N. Mladenovi'c i P. Hansen, „Variable neighborhood search,“ *Computers & operations research*, t. 24, br. 11, pp. 1097-1100, 1997.
- [133] G. V. Glass, „Testing homogeneity of variances,“ *American Educational Research Journal*, t. 3, br. 3, pp. 187-190, 1966.
- [134] S. S. Shapiro i R. Francia, „An approximate analysis of variance test for normality,“ *Journal of the American statistical Association*, t. 67, br. 337, pp. 215-216, 1972.