A Meta-Heuristic Approach for Optimizing Neural Network Model for Heart Disease Prediction

Gummuluri Venkata Ravi Ram[1], Prathipati Jayanth[2],
M Hemanth Kumar[3], Sona Mundody[4], Ram Mohana Reddy Guddeti[5]
National Institute of Technology Karnataka, Surathkal, Karnataka, India
Emails: {toraviram2003, jayanthprathipati2003,
mogilipalemhemanthkumar, sona.mndy}@gmail.com, profgrmreddy@nitk.edu.in

***Abstract*—In heart-disease detection, we have several systems utilizing Machine Learning models to assess the likelihood of a person experiencing heart disease based on input data gathered from health checkups. To deploy these in real-time, we need to constantly update the model with validation data generated from the feedback of doctors and users. One crucial component is hyperparameter tuning, as hyperparameters define the working conditions of a model. Current hyperparameter tuning methods have more time-complexity, which might cause latency. Meta-heuristic algorithms are known to achieve optimal solutions in less time, leveraging accuracy. In this paper, we performed time-complexity analysis for three bio-inspired meta-heuristic algorithms viz. Genetic Algorithm, Ant-Colony Optimization and Swarm-bee optimization for hyperparameter tuning in the context of Artificial Neural Networks in heart disease prediction. We achieved accuracy 93.17 on Cleveland dataset and 95.12 on Cleveland, Hungary, Switzerland, and Long Beach V combined dataset, which outperformed conventional algorithms in less time complexity. We built a web-app and mobile-app based on best models.**

I. Introduction

Heart disease remains a notable global health issue, resulting in a considerable number of fatalities and placing a significant burden on healthcare systems. Accurate heart disease prediction is essential for early diagnosis and timely interventions, enabling healthcare professionals to provide appropriate treatments and preventive measures. However, developing precise prediction models for heart disease is challenging due to the complex and nonlinear relationships among various risk factors and disease outcomes. Numerous approaches are available for heart disease prediction, ranging from traditional statistical methods to machine learning algorithms. However, these methods often exhibit limitations such as suboptimal predictive accuracy, reliance on expert feature engineering and lack of automation in hyperparameter tuning. Manual tuning of hyperparameters for Artificial Neural Networks (ANNs), one of the favoured machine learning approaches.

The core idea of this work is to compare meta-heuristic algorithms to choose the best hyperparameters of ANNs for heart disease prediction. By leveraging the power of meta-heuristic algorithms, we aim to speed up and enhance the hyperparameter tuning process, leading to reduced latency in model updation. As far as our understanding goes, no works have addressed the comparison of meta-heuristic algorithms for optimising hyperparameters in the context of predicting heart disease.

The major contributions of the presented work are summarized as follows:

- Comparative Time complexity analysis of three meta-heuristic algorithms - Genetic Algorithm, Min-Max Ant Colony Optimisation Algorithm and Particle Swarm Optimisation Algorithm for hyperparameter tuning in heart disease prediction.
- Development of the Heart Disease prediction model using ANNs optimized by meta-heuristic algorithms.
- Creating web and android application for heart disease Prediction.

The structure of the rest of this paper is arranged as follows: In Section II, a concise review of existing works is presented. Section III delves into the proposed methodology, offering insights into the optimization algorithms employed for hyperparameter tuning. The experimental setup and results of the proposed research work are discussed in Section IV. Finally, Section V concludes the paper and provides an incisive insight into the potential future directions of the research.

II. Literature Survey

Firdaus et al. [1] introduced a heart disease detection method utilizing a deep neural network. The work incorporated tuning of hyperparameters through grid search, random search, and Bayesian optimization. Notably, Bayesian optimization demonstrated superior accuracy. The deep neural network, optimized by Bayesian methods, achieved impressive metrics with a 91.67% testing accuracy, 95.83% sensitivity, 88.89% specificity, 85.19% precision, 90.20% F1-score, and an AUC value of 0.9514. This underscores the effectiveness of Bayesian optimization for heart disease detection.

Yang et al. [2] developed a multi-objective Genetic Algorithm for hyperparameter tuning and showed that their method performed better than others on several datasets. The study does not compare the proposed method to grid search or other hyperparameter optimization methods. Ayan et al. [3] used a Genetic Algorithm for tuning the hyperparameters of convolutional neural networks and compared the results using the CIFAR-10 dataset. The study considers only one dataset and its performance on other types of problems is unclear.

Lobvithee et al. [4] used the Ant Colony Optimisation algorithm to optimise manually fine-tuning hyperparameters. They showed that Ant Colony Optimization (ACO) offers a notable advantage over cross-validation with reduced computational time. The limitation of the work is the high complexity and high computational time involved in the optimization process. Vincent et al. [5] provided a framework hyperparameter optimization using evolutionary algorithms for Auto-ML systems. The authors, however, did not offer any implementation results for the classification problem.

Jianjun et al. introduced the Improved Particle Swarm Optimization (IPSO) algorithm, demonstrating its efficiency in selecting hyperparameter combinations for improved results using a single dataset [6]. Yaru and Yulai Zhang developed a hyperparameter optimization algorithm based on Particle Swarm Optimization (PSO), highlighting its effectiveness but noting slower operation in high-dimensional spaces [7]. B.H Shekar and Guesh Dagnew presented the Grid-Search based hyperparameter tuning algorithm (GSHPT) for classification [8]. Christopher I et al. employed Random-Search with genetic algorithms and clustering analysis for hyperparameter tuning [9]. Vu Nguyen utilized Bayesian optimization for accelerated hyperparameter tuning in automated design [10]. Kashif Ahmad introduced a Particle Swarm Optimization-based technique for hyperparameter optimization within a Federated Learning framework, specifically focusing on applications in smart city services and industrial Internet of Things (IIoT) services [11]. Yuhua Qi proposed a method based on the Ant Colony Optimization algorithm for test case prioritization, offering two methods: distance-based and index-based [12].

III. Methodology

An arbitrary artificial neural network (ANN) model is used for the purpose of prediction. The model is trained using hyperparameters provided by the different optimization algorithms. The metrics of the model for different hyperparameters are compared. Finally, the optimization algorithm that provides the best hyperparameters (for better prediction accuracy) is chosen and used for training the ANN model. The detailed description of the methodology as in Fig.1 is given in the following sections.

A. Datasets

This research utilizes two datasets. The first dataset is the Cleveland Heart Disease dataset from the UCI Machine Learning Repository [13], encompassing 303 instances. The dataset comprises 303 data instances with 14 attributes, encompassing 13 input features such as age, resting blood pressure, sex, serum cholesterol in mg/dl, chest pain type (4 values), fasting blood sugar if more than 120 mg/dl, exercise induced angina, resting electro-cardiographic results containing values(0,1,2), maximum heart beat rate, old peak (ST depression induced by exercise relative to rest), the slope of the peak exercise ST segment, number of major vessels (0-3) coloured by fluoroscopy, that (0 = normal; 1 = fixed defect; 2 = reversible) defect and one predicted attribute, all expressed in numeric data. The second dataset (Dataset-2) combines four datasets namely: Cleveland, Hungary, Switzerland, and Long Beach V heart disease data. The dataset comprises 1025 data instances and 76 attributes, of which a subset of 14 features are considered. Input features were similar to features in [13]. 80% of the available data is used for training the models, and 20% is used for validation & testing in both datasets.

B. Hyperparameter Tuning Using Conventional Methods

There are many conventional methods for hyperparameter optimization. Of these, we considered Grid-search, Randomized search, and Bayesian optimization. The respective hyperparameter search space is given in Table I.

TABLE I. Hyperparameter Search Space

Hyperparameter	Value
Epoch	[10, 100]
Batch Size	[32, 256]
Hidden Neurons	[8, 64]
Activation Function	['relu', 'sigmoid']
Learning Rate	[0.0001, 0.1]

C. Hyperparameter Tuning using Meta-Heuristic Algorithms

1) Genetic Algorithm: The genetic algorithm evolves a population over several generations, evaluating the fitness of each set of hyperparameters using the neural network model that has been created. The fitness function uses the accuracy of the model as its basis and Algorithm 1 shows the details.

Algorithm 1 initializes a random population of hyperparameters and iteratively evolves it. Parents in each generation are taken from the individuals with higher fitness scores, and new offspring are generated by combining their hyperparameters. Some randomness is introduced through mutation, where hyperparameters may be randomly changed. This process continues for a given number of generations.

Finally, the best set of hyperparameters is determined based on the highest fitness score in the last generation. The neural network is trained and evaluated using these hyperparameters. The genetic algorithm aims to discover optimal hyperparameter configurations for improving the model's performance.

We experimented with an inbuilt genetic algorithm using DEAP and TPOT libraries to optimise hyperparameters. DEAP is utilized to define a genetic algorithm, incorporating hyperparameters and employing crossover and mutation operations. The neural network is trained and evaluated based on these hyperparameters, utilizing accuracy as the fitness metric.

2) Min-Max Ant-Colony Optimization: There are several ant-colony optimization variants, of which the Min-Max Ant System (MMAS) is the most prevalent one for hyperparameter tuning.

The key component driving the search process is the pheromone information denoted as $\tau(i, j)$, where i and j represent specific hyperparameter values within predefined

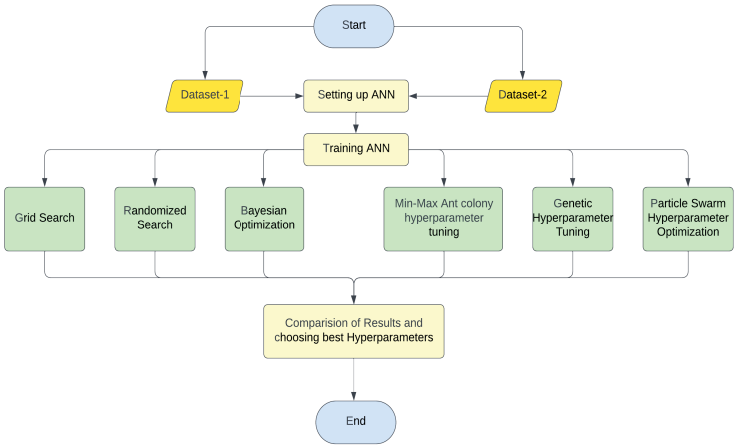

Fig. 1. Proposed Methodology

Algorithm 1 Genetic Algorithm

INPUT

- Population _Size, n
- NUM _OF _GENERATIONS, G

OUTPUT

- Best _Hyperparameters, , Y_{bt}

Initialize population of size n Y_i (1,2,...n)
Define the fitness function
for each i from 1 to G **do**
    Select two individuals based on fitness.
    Perform "crossover" and insert them.
    Perform "mutation" on the offspring and insert.
    Replace the old with the newly generated.
end forreturn best_hyperparameters, Y_{bt}

bounds. This pheromone information is similar to the chemotactic cues in ant colonies, guiding the ants (representing hyperparameter combinations) towards promising regions in the search space. Additionally, the heuristic information $\eta(i, j)$ contributes to the attractiveness of a hyperparameter combination, providing a complementary influence on the ant's decision-making process.

The MMAS algorithm (Algorithm 2) occurs over a predefined number of iterations, each consisting of two main steps: pheromone update and ant behaviour. In the pheromone update phase, the algorithm applies evaporation by reducing the pheromone values by $(1 - \rho)$, ensuring a gradual decay of previously laid pheromones. The ant behaviour phase involves each ant's probabilistic selection of hyperparameter combinations. The probability of selecting a particular combination (i, j) is determined by the ratio of the pheromone product and heuristic information for that combination to the sum of such products for all combinations. Subsequently, the selected hyperparameter combinations are used to train and evaluate an

Algorithm 2 MMAS Algorithm

INPUT (num_ants), (max_iterations), (pheromone_rho), (pheromone_min), (pheromone_max)

OUTPUT (best_accuracy), (best_hyperparameters)

Initialisation

- best_pheromone_train as an array of ones same dimension as pheromone_matrix.
- best_accuracy = 0
- best hyperparameters as empty dictionary.

best_accuracy = 0
LOOP Process
for $i = 1$ to $max\_iterations$ **do**
    ant_accuracies = [] ; ant_hyperparameters = []
    for each ant from 1 to num_ants **do**
        Select $hyperparameters$ probabilistically
    end for
Update $pheromone\_matrix$
    Evaporate pheromone values)
    for each ant from 1 to num_ants **do**
        Calculate pheromone deposit change in $\tau(i, j)$
        Update $pheromone$ in $pheromone\_matrix$.
    end for
    if new best accuracy is found **then**
        Update $best\_accuracy$
        Update $best\_hyperparameters$
    end if
    Update $best\_pheromone\_trail$ & Clip $pheromone\_matrix$ values
end for
Set $best\_accuracy$ as the highest accuracy obtained during the iterations
Set $best\_hyperparameters$ as the hyperparameters corresponding to the $best\_accuracy$
return best_accuracy and best_hyperparameters

ANN model, and the resulting accuracy influences the update of pheromone values. The algorithm continues iteratively until a termination criterion, such as reaching a maximum number of iterations, is met.

To ensure feasibility and stability of the search process, the algorithm incorporates bounds on pheromone values, preventing them from exceeding specified limits. Moreover, using an ant colony-based approach enables the MMAS algorithm to explore the hyperparameter search space adaptively, providing a mechanism for fine-tuning ANNs for optimal performance.

3) Particle Swarm Optimization Algorithm: Particle Swarm Optimization (PSO), introduced by Kennedy and Eberhart [16], is a nature-inspired meta-heuristic widely applied to hyperparameter tuning in machine learning. Utilizing particles as potential hyperparameter sets, the algorithm explores the hyperparameter space by navigating through a combination of personal and global best positions.

The components of the PSO algorithm (Algorithm 3) include particles, positions, velocities, fitness functions, and tracking mechanisms for standard best positions in parameter space. The primary objective in hyperparameter tuning is to maintain balance between exploration and exploitation for gaining optimal performance. This algorithm addresses challenges such as premature convergence or divergence during optimasation.

Algorithm 3 Particle Swarm Optimization

```
Initialize X_i, V_i, iter, part_best and glob_best
Generate a set of diverse random particles (P)
for each Particle i do
    Compute the fitness function f_i
    Update the selected part_best, glob_best values
end for
while iter do
    for each particle in the swarm do
        Update X_i, V_i
        if X_i exceeds a threshold limit then
            Set X_i to the limit
        end if
        Calculate the fitness function f_i
        Update part_best, glob_best
    end for
end while
```

The iterative process of PSO for hyperparameter tuning involves initializing particles with hyperparameter sets, evaluating the fitness of each set based on the model's performance, and iteratively updating the positions and velocities to converge towards optimal hyperparameter configurations. We experimented using Optunity Module for optimisation purposes.

D. Mobile Application

For predicting whether a person is prone to heart disease; we have generated a predictive model (classification) using which we developed a user-friendly Android application is shown in Fig.2. Exported the model with the best cross-validation accuracy as a PICKLE file and its feature names to the environment where we create the app.

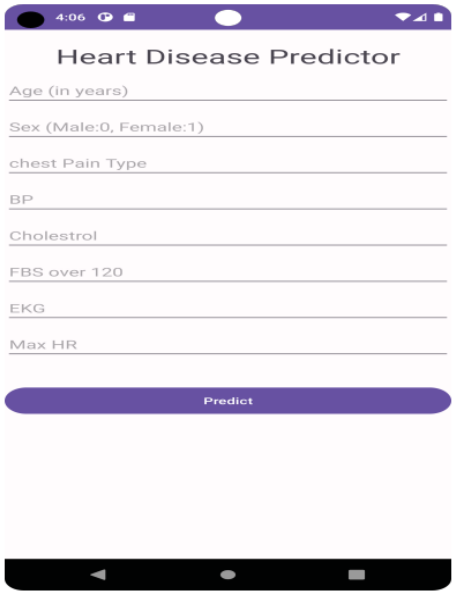

Fig. 2. Mobile-app interface[1]

E. Web Application

The web application design involves converting the model with the best hyperparameters to API by using Pickle. Pickle converts the model to a .sav or a .h5 file, then uses streamlit, third-party software to deploy the heart disease prediction. Fig.3 shows the web application page predicting heart disease.

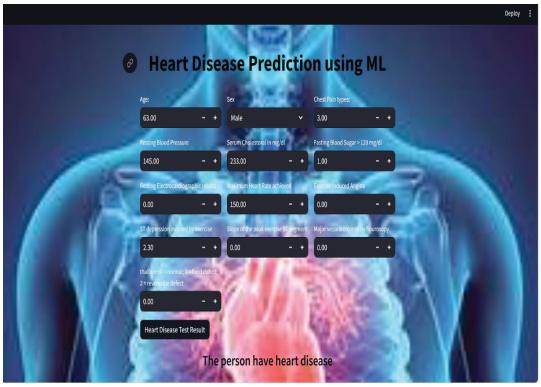

Fig. 3. Web-app Interface[2]

IV. Experimental Results and Analysis

A. Experimental Setup (ANN)

1) Building the ANN Model: The ANN(Artificial Neural Network) Model contains three different layers. The First layer in the network the Input layer that contains the no.of neurons, is equal to the number of features in our dataset. The second layer is the hidden layer, which contains 8 neurons (prefer

TABLE II. COMPARITIVE ANALYSIS OF DIFFERENT HYPER PARAMETER TUNING METHODS FOR DATASET-1

Tuning Algorithm	Epoch	Batch size	Neurons	Activation	Learning Rate	Accuracy	Precision	Recall	Time(in s)
Without Tuning	55	32	16	sigmoid	0.01	77.63	0.59	0.57	20
Grid Search	100	32	64	relu	0.01	83.33	0.62	0.64	402
Randomised Search	75	32	60	relu	0.01	81.48	0.60	0.53	276
Bayesian Optimisation	56	59	59	relu	0.02325	83.33	0.70	0.62	120
Genetic Algorithm	24	65	16	sigmoid	0.0519	**93.17**	0.91	0.94	395
Ant Colony Optimisation	20	32	16	sigmoid	0.0112	90.73	0.90	0.87	354
Particle Swarm Optimisation	25	22	45	relu	0.01	83.12	0.76	0.80	528

TABLE III. COMPARITIVE ANALYSIS OF DIFFERENT HYPER PARAMETERS TUNING METHODS FOR DATASET-2

Tuning Algorithm	Epoch	Batch size	Neurons	Activation	Learning Rate	Accuracy	Precision	Recall	Time(in s)
Without Tuning	55	32	16	sigmoid	0.01	88.79	0.87	0.86	22
Grid Search	100	32	64	relu	0.01	89.67	0.83	0.84	442
Randomised Search	75	32	53	relu	0.01	90.73	0.89	0.87	324
Bayesian Optimisation	57	43	35	relu	0.036	92.57	0.92	0.93	168
Genetic Algorithm	64	189	16	sigmoid	0.0636	95.08	0.97	0.96	426
Ant Colony Optimisation	20	32	16	sigmoid	0.0245	**95.12**	0.93	0.90	418
Particle Swarm Optimisation	36	28	62	relu	0.01	90.42	0.91	0.92	587

the power of 2) and the activation function is ”ReLU”. The third layer is the output layer, which contains only one neuron as it predicts whether the person has heart disease or not by using the same activation function, ”ReLU”.We trained it for 75 epochs and batch size of 32. We utilize metrics such as accuracy, precision, and recall to assess the model's performance for various hyperparameters. A comparative analysis of all experimented hyperparameter tuning methods and their corresponding evaluation metrics is presented in Table II and Table III.

B. Results and Complexity Analysis

1) Complexity Analysis of Genetic Algorithm: The initialization step of a genetic algorithm involves creating a population of size ”n” with random hyperparameters, resulting in O(n) time complexity. Fitness evaluation, conducted by iterating over the population, adds another O(n) to the overall complexity.

The selection of fittest individuals introduces O(n * log(n)) complexity, encompassing sorting and selecting the top individuals. Generating offspring, achieved through combining parental hyperparameters, contributes O(n) to the time complexity.

Mutation, population replacement, and identifying the best set each have O(n) components. These operations collectively emphasize a linear relationship with the population size (n) in the overall time complexity of the genetic algorithm..

Considering these individual complexities, the overall time complexity of evolving the population over ”G” generations is denoted as O(G * n * log(n)).

Overall Time Complexity: O(n + G * n * log(n))

- G: Number of Generations
- n: population size

2) Complexity Analysis of MMAS algorithm: Min-Max Ant colony optimisation algorithm has an O(1) time complexity for the initialization phase, involving the creation of fixed-dimension pheromone matrices. During the Ant Behavior phase, where ants iterate over operations, the probabilistic selection of hyperparameters remains constant for each ant. The evaluation function's time complexity is O(e * b * t), where e is the number of epochs, b is the batch size, and t is the training dataset size.

Other Ant Behavior phase operations, such as updating best accuracy (O(1)), updating the best pheromone trail (O(m)), and pheromone matrix clipping (O(m)), have constant or m-proportional time complexities, where m is the number of elements in the pheromone matrix. These factors contribute to the algorithm's efficiency and scalability in hyperparameter optimization.

Overall Time Complexity: O(iterations*n*(e * b * t)+m)

- iterations: Number of iterations in the MMAS algorithm
- n: Number of ants
- e: Number of epochs
- b: Batch size
- t: Size of the training dataset
- m: Size of the pheromone matrix (number of elements)

3) Complexity Analysis of Particle Swarm Optimization: In the PSO algorithm, initialization involves randomly setting positions and velocities, with a time complexity of O(ND), where N is the number of particles and D is the dimensionality. The evaluation step, dependent on the specific fitness function, has a time complexity of O(num_hyperparameters) in a heart disease prediction context. The update step, utilizing the best positions found, incurs a time complexity of O(ND), capturing the computational workload of updating particle velocities and positions. Together, these steps contribute to the overall time complexity of the PSO algorithm.

Overall Time Complexity: O(T*(N*D+fitness))

- Number of Particles(N): The Time Complexity of PSO algorithm directly depends on the number of particles in

the swarm.

- Dimensionality of the problem(D): The Time Complexity of PSO algorithm directly depends on the dimensionality of the problem as PSO works on different dimensions and the search space accordingly increases.
- Number of Iterations(T): Time Complexity also depends on number of iterations required to terminate the algorithm by meeting the required conditions. Algorithm converges to optimal or near optimal solution finally.

In comparison to the study referenced as [1], which uses the same our Dataset-1 in [13] and reported grid search tuning taking 635.78 seconds, random search tuning requiring 159.32 seconds, and Bayesian optimization lasting 189.48 seconds, our approach achieved more efficient tuning times. Specifically, our built genetic algorithm outperformed [1] by significantly reducing tuning time to 243 seconds, while the ant colony algorithm took 354 seconds, and the particle swarm algorithm took 503 seconds while maintaining similar accuracy. Notably, our enhanced genetic algorithm yielded superior evaluation scores(93% accuracy, 0.92 precision, 0.94 recall) compared to the methodology presented in [1] that provides us a faster and better hyperparameter tuning method. Compared to the investigation cited as [15], which utilizes our Dataset-2, as described in [13], and demonstrates notable achievements with high accuracy, precision, and recall (approximately 93%, 0.87, and 0.84, respectively), our tuned models exhibit superior performance in these metrics.

V. Conclusion and Future Work

In regular approaches, best hyperparameters identified by Grid search, Randomized search, Bayesian Optimization gave an accuracy of 83.33% , 81.48% and 83.3%, respectively. Then we worked on Evolutionary algorithms such as Genetic , which gave an accuracy of 93.17%,and was computationally efficient as mentioned in the time complexity analysis section. In Ant Colony Optimization , we got an accuracy of 85.15%. Particle Swarm Optimization gave an accuracy of 83.12% and the respective time complexity is mentioned in time complexity analysis section.

For the deployment we used streamlit ,a third party software for deploying the machine learning models.Genetic Algorithms (GA's) are commonly preferred for tasks involving extensive search spaces, intricate hyperparameter interactions, and global exploration requirements. They excel at handling computationally demanding models and can be parallelized efficiently. Conversely, Ant Colony Optimization (ACO) can be advantageous in leveraging local information and adapting to dynamic scenarios, potentially resulting in quicker convergence.So we used Genetic algorithm for the deployment. We outperformed [1] and [15] with respect to time taken for hyperparameter tuning with almost same accuracy. As a future work, we would like to explore bio-inspired and metahueristic algorithms for hyperparameter tuning and generalize better hyperparameter tuning algorithms for specific tasks. Explainability and Interpretability can be added for making it more accessible and adding multi-model data.

References

[1] Firdaus, Fathania Firwan, Hanung Adi Nugroho, and Indah Soesanti. "Deep neural network with hyperparameter tuning for detection of heart disease." In 2021 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), pp. 59-65. IEEE, 2021.

[2] Yang, Chuanguang, et al. "Multi-objective pruning for cnns using genetic algorithm." Artificial Neural Networks and Machine Learning–ICANN 2019: Deep Learning: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part II 28. Springer International Publishing, 2019.

[3] Ayan, Enes. "Genetic Algorithm-Based Hyperparameter Optimization for Convolutional Neural Networks in the Classification of Crop Pests." Arabian Journal for Science and Engineering (2023): 1-15.

[4] Lohvithee, Manasavee, et al. "Ant colony-based hyperparameter optimisation in total variation reconstruction in X-ray computed tomography." Sensors 21.2 (2021): 591.

[5] Vincent, Amala Mary, and P. Jidesh. "An improved hyperparameter optimization framework for AutoML systems using evolutionary algorithms." Scientific Reports 13.1 (2023): 4737.

[6] Zhan, Jianjun, et al. "Improved particle swarm optimization algorithm based on grouping and its application in hyperparameter optimization." Soft Computing (2023): 1-13.

[7] Li, Yaru, and Yulai Zhang. "Hyper-parameter estimation method with particle swarm optimization." arXiv preprint arXiv:2011.11944 (2020).

[8] Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data B. H Shekar, Department of Computer Science, Mangalore University, India Guesh Dagnew, Department of Computer Science, Mangalore University, India (2019).

[9] Marrison, Christopher I., and Robert F. Stengel. "The use of random search and genetic algorithms to optimize stochastic robustness functions." In Proceedings of 1994 American Control Conference-ACC'94, vol. 2, pp. 1484-1489. IEEE, 1994.

[10] Nguyen, Vu. "Bayesian optimization for accelerating hyper-parameter tuning." In 2019 IEEE second international conference on artificial intelligence and knowledge engineering (AIKE), pp. 302-305. IEEE, 2019.

[11] Particle Swarm Optimized Federated Learning ForIndustrial IoT and Smart City Services Basheer Qolomany, Kashif Ahmad, Ala Al-Fuqaha, and Junaid Qadir(2020)

[12] Zhang, Weixiang, Yuhua Qi, Xuebo Zhang, Bo Wei, Min Zhang, and Zhaohui Dou. "On test case prioritization using ant colony optimization algorithm." In 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 2767-2773. IEEE, 2019.

[13] R. Detrano, "Heart Disease Data Set," UCI Machine Learning Repository. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Heart+Disease. [Accessed: Dec 10, 2019]

[14] Tang, Kit-Sang, Kim-Fung Man, Zhi-Feng Liu, and Sam Kwong. "Minimal fuzzy memberships and rules using hierarchical genetic algorithms." IEEE Transactions on Industrial Electronics 45, no. 1 (1998): 162-169.

[15] Ismail, Mohammad, V. Harsha Vardhan, V. Aditya Mounika, and K. Surya Padmini. "An effective heart disease prediction method using artificial neural network." International Journal of Innovative Technology and Exploring Engineering 8, no. 8 (2019): 1529-1532.

[16] Bratton, Daniel, and James Kennedy. "Defining a standard for particle swarm optimization." In 2007 IEEE swarm intelligence symposium, pp. 120-127. IEEE, 2007.