Hyperparameter tuning of ANN using Metahueristic Algorithms for Heart Disease Prediction

G V Ravi Ram
Artificial Intelligence
NITK, Suratkal
Suratkal, India
Email: toraviram2003@gmail.com

Mogilipalem Hemanth kumar
Artificial Intelligence
NITK, Suratkal
Suratkal, India
Email: mogilipalemhemanthkumar@gmail.com

Jayanth Prathipati
Artificial Intelligence
NITK, Suratkal
Suratkal, India
Email: jayanthprathipati2003@gmail.com

***Abstract*—This research project focuses on developing a heart disease prediction application by leveraging metaheuristic algorithms for hyperparameter tuning of Artificial Neural Networks (ANN). The study compares the performance, time complexity, and space complexity of three metaheuristic algorithms, namely Genetic Algorithm, Ant Colony Algorithm, and Particle Swarm Optimisation Algorithm , to identify the algorithm that produces the highest accuracy model. Hyperparameter tuning plays a crucial role in optimizing ANN models, and metaheuristic algorithms offer effective approaches to automate this process. By implementing these algorithms, we aim to create an application that accurately predicts heart disease risk based on relevant patient data. The performance metrics, execution time, and memory usage of each algorithm will be assessed to determine the most suitable algorithm for achieving the highest prediction accuracy. The findings of this research can significantly contribute to improving heart disease diagnosis and risk assessment, providing a valuable tool for healthcare professionals.**

Keywords: Ant Colony Algorithm, Artificial Neural Networks, Genetic Algorithm, Healthcare Application, Heart Disease Prediction, Hyperparameter Tuning, Metaheuristic Algorithms, Prediction Accuracy, Swarm-Bee Algorithm,

.

I. Introduction

Heart disease continues to be a significant global health concern, leading to a high number of fatalities and imposing a substantial burden on healthcare systems. Accurate prediction of heart disease 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 have been proposed 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), a popular machine learning technique, is a time-consuming process and does not guarantee optimal performance.

The core idea of this research project is determining metaheuristic algorithms to choose the best hyperparameters of ANNs for heart disease prediction. By leveraging the power of metaheuristic algorithms, we aim to automate and enhance the process of hyperparameter tuning, leading to improved prediction accuracy and model performance. The motivation behind this project is two-fold: (1) to develop a highly accurate heart disease prediction model that facilitates early detection and intervention, and (2) to streamline the model development process by automating hyperparameter tuning, reducing the reliance on manual trial-and-error approaches.

This research project makes several significant contributions to the field of heart disease prediction:

- Development of the Heart Disease prediction model using ANNs optimized by metaheuristic algorithms.
- Comparative analysis of three metaheuristic algorithms - Genetic Algorithm, Ant Colony Algorithm, and Swarm-Bee Algorithm - in terms of their performance, time complexity, and Space complexity for hyperparameter tuning in heart disease prediction.
- Creation of a user-friendly heart disease prediction application that integrates the optimized ANN model, providing accurate risk assessments and supporting healthcare professionals in making informed decisions.

This report contains literature survey folowed by consice problem statement and objectives we have in this project. Then methodology followed by dataset, experimental setup and experimental results and analysis are provided. Finally, complexity analysis, conclusion, individual contribution are provided and concluded with references.

II. Literature Survey

Yang, Chaunguang in [1] implemented a method for multi objective genetic algorithm and shows improved performance compared to other methods on several datasets.The study does not compare the proposed method to grid search or other hyperparameter optimization method. Ayan, Enes in [2] gave an a approach shows improved performance compared to other methods on the CIFAR-10 dataset.The study only evaluates one dataset,and its performance on other types of problems is unclear.

Lobvithee, Manas [3] used Ant Colony Optimisation to propose an algorithm to circumvent the laborious task of manually fine-tuning hyperparameters, utilizing Ant Colony Optimization (ACO) offers a notable advantage over cross-validation: reduced computational time. This feature becomes particularly advantageous when considering the time-consuming nature of cross-validation.. Limitation is the higher level of complexity and longer computational time . Vincent, Amala mary [4] designed The results from the paper can be extended to other models and datasets. But this paper doesn't explain about real-time image classification problems.

Zhan and Jianjun in [5] gave an particle swarm optimization. IPSO can search for satisfactory hyperparameter combinations more quickly and stably considers only one dataset.Yaru and Yulai Zhang [6] implemented a swarm optimisation to obtain maximum results and minimum results, PSO-BO outperforms the other two algorithms under comparison in terms of the accuracy. The algorithm runs slowly in high-dimensional space. B.H shekar and Guesh Dagnew

in their paper [7] proposed a grid search based hypertuning GSHPT for random forest parameters in classifying cancer data using 10-fold cross validation and gini index as criteria in splitting the decision trees."Christopher I. Marrison and Robert F. Stengel" in their paper [8] used random search as search parameter to find global minimum number of monto carlo evaluations and given two different approaches based on genetic algorithm and clustering analysis.Vu Nguyen University of Oxford, United Kingdom, in his paper [9] used bayesian optimisation for devoleping and accelerating Hyper paramters tuning which are used in automated designing.Kashif Ahmad and team in their paper proposed a Particle swarm optimisation based one technique for optimising hyper parameter settings for machine learning models in an FL environment considering smart city services and industrial (IIOT) services. Yuhua QI1 and team in their paper to improve test case prioritisation proposed a method based on ant colony optimisation involving two methods they are distance based and index based.
some others in their paper shared knowledge on emerging technology based on genetic algorithms and ways to integrate it to create a large designed frameworks.

III. Problem Statement

To design, develop and analyze the performance of an efficient heart disease prediction system using ANN whose hyperparameters are tuned using metaheuristic algorithms to enhance accuracy, enable early detection, and facilitate effective healthcare decision-making.

A. Objectives

- Optimize the hyperparameters of ANNs using the selected metaheuristic algorithms to maximize the prediction accuracy and minimize false positives and false negatives.
- Implement and compare the performance of metaheuristic algorithms, such as Genetic Algorithm, Ant Colony Algorithm, and Swarm-Bee Algorithm, for hyperparameter tuning of artificial neural networks (ANNs).
- Develop a full-fledged mobile application to predict chance of user being prone to Heart Disease.

IV. Methodology

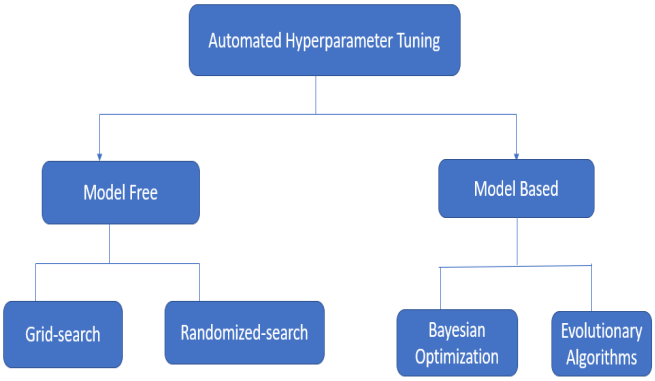

Fig. 1. Automated Hyperparameter Tunning

Figure-1 shows the Automated Hyperparameter Tuning.which is classified as Model Free approach and Model Based Approach.The Model Free , which is also known as conventional methods , is further classified as Grid-search and Randomized Search.The Model Based Approach is also further classified as Bayesian Optimizaion,Gradient Descent and Evolutionary Algorithms such as Gentic, Ant colony and swarm Bee.

A. Hyperparameter tuning using conventional methods

These are many conventional methods for hyperparameter optimization .out of those, we considered Grid-search, Randomized search and Bayesian optimization.

*1) **Grid-Search**:* :
In this project, we conducted hyperparameter tuning for an Artificial Neural Network (ANN) model using Grid Search Cross-Validation. The ANN model was implemented using TensorFlow's Keras wrapper for Scikit-learn. The model architecture comprised of two hidden layers and one output layer, and we optimized the model's performance by exploring various hyperparameters.

*2) **Randomized-Search**:* :
In our project, we employed Randomized Search Cross-Validation to conduct hyperparameter tuning for an Artificial Neural Network (ANN) model. The ANN model was implemented using TensorFlow's Keras wrapper for Scikit-learn. The architecture of the ANN includes two hidden layers and one output layer, with the hyperparameters carefully defined to optimize the model's performance.

*3) **Bayesian-Optimization**:* :
we utilized Bayesian Optimization to optimize the hyperparameters for an Artificial Neural Network (ANN) model. The optimization is performed using the BayesianOptimization class from the bayes_ opt library.

The optimization function, optimize, takes the hyperparameters (learning_ rate, neurons, batch_ size, epochs, patience) as input and builds an ANN model accordingly. The model is trained using the provided hyperparameters and early stopping is employed to prevent overfitting. The validation accuracy is then returned as the optimization metric.

B. Hyperparameter tuning using metaheuristic algorithms

*1) **Genetic Algorithm**:* One can utilize them in a beneficial manner for hyperparameter optimization, which pertains to the task of determining the most advantageous values for a machine learning model's hyperparameters.
The genetic algorithm begins by creating an initial population of potential solutions, where each solution represents a set of hyperparameters for the model. The population is typically generated randomly, covering a wide range of hyperparameter values. At first , we defined the bounds for the hyperparameters , those we considered (learning rate,epochs and Batch size). Then we defined the specific values for the popoulation size and no.of generations by our previous knowledge.
The following operators are used for the implementation of the genetic algorithm.

- Selection:
  There are different types of selection methods used in genetic algorithms such as Fittness-Proportionate (Roulette Wheel),Tournament,Rank-Based ,Elitism. Out of all these selection techniques, we considered "Fitness-proportionate" selection.Fitness-proportionate selection as it sorts the fitness scores of individuals in descending order. By selecting the fittest individuals as parents, it aims to give preference to individuals with better performance (higher fitness) for producing the next generation.
- Cross over
  Among the various types of cross-over techniques, such as Single-Point Crossover, Two-Point Crossover, Uniform Crossover, and Arithmetic Crossover, we specifically chose

TABLE I
SUMMARY OF LITERATURE SURVEY

Authors	Methodology	Merits	Limitations	Additional Details Based on your project (eg. Dataset Used)
Ayan, Enes	Genetic Algorithm on image data	The approach shows improved performance comparedto other methods on the CIFAR-10 dataset	The study only evaluates one dataset, and its performance on other types of problems is unclear.	CIFAR-10
Yang, Chuanguang	Multi objective pruning using genetic algorithm on cnn	The proposed approach uses a multi objective genetic algorithm and shows improved performance compared to other methods on several datasets.	The study does not compare the proposed method to grid search or other hyperparameter optimization method	MNIST
Lohvithee, Manas	Ant colony optimisation	”The proposed algorithm offers, to avoid the tedious process of manual hyperparameter tuning . The computational time for ACO compared to a cross validation is also a significant advantage.	higher level of complexity and longer computational time”	X-ray
Vincent, Amala Mary	Frame work for evolutionary algorithms	”The results from the paper can be extended to other models and datasets	Doesn’t explain about real-time image classification problems	SVHN,CALTECH
Zhan, Jianjun	particle swarm optimization	IPSO can search for satisfactory hyperparameter combinations more quickly and stably”	considers only one dataset	Tenth Teddy Cup Data
Yaru, Yulai Zhang	swarm optimization	”results, PSO-BO outperforms the other two algorithms under comparison in terms of the accuracy	The algorithm runs slowly in high-dimensional space.”	Boston,Digits

the Two-Point Crossover technique. This method allows us to combine genetic material from both parents at two distinct points, facilitating the creation of new offspring with a diverse combination of traits.

- Mutation
  There are different types of Mutation methods used in genetic algorithms such Random Mutation,Non-Uniform Mutation,Gaussian Mutation,Swap Mutation. we used random mutation as the mutation method. This strategy randomly modifies the hyperparameters of the offspring by selecting new values from the predefined bounds or ranges. Random mutation is simple to implement and allows for exploration of different hyperparameter combinations. It is particularly suitable for problems involving discrete hyperparameter values.

The genetic algorithm implementation to find best hyperparameters is as follows:

- Fitness Evaluation :
  The evaluate_ fitness takes a set of hyperparameters as input and assesses the quality of an ann model configured with those hyperparameters. The ANN consists of two hidden layers and one output layer, constructed using the Keras library. The hyperparameters, including the learning rate, batch size, and number of epochs, influence the training process of the model. The function compiles the model with the Adam optimizer, utilizing the provided learning rate, and applies the binary cross-entropy loss function. It proceeds to train the model on the training data (X_ train and y_ train) .
- Genetic operators:
  It initializes a population consisting of random sets of hyperparameters within predefined bounds. The population size, number of generations, and bounds for each hyperparameter are predefined constants. The algorithm evolves the population over multiple generations. In each generation, it evaluates the fitness of each set of hyperparameters by invoking the evaluate fitness for each individual The fitness scores obtained are stored in the fitness scores list.

  1. Selection:
  Next, the fittest individuals are selected as parents to form the subsequent generation using the above mentioned selection method. The parent selection process relies on the fitness scores, where individuals with the highest scores are chosen. The indices of the selected parents are stored in the parent_ indices list.

  2.Cross Over:
  To create offspring, the algorithm combines the hyperparameters of randomly selected parent using the above mentioned cross over technique. For each offspring, a random pair of parents is chosen, and for each hyperparameter, a random decision is made to inherit the value from one parent or the other. The resulting offspring are appended to the offspring list.

  3.Mutation:
  To introduce variation in the offspring, we incorporated a mutation process that randomly modifies their hyperparameters using the mutation technique mentioned earlier. For each offspring, there is a small probability (0.1 in this case) assigned to each hyperparameter for potential mutation. If a mutation occurs, the corresponding hyperparameter is assigned a new value within the predefined bounds. This allows for exploration of different hyperparameter values and adds diversity to the population of potential solutions.

- Best Hyperparameters:
  Finally, the old population is replaced with the new offspring population, and the process continues for the specified number of generations.
  Upon completing the generations, the best set of hyperparameters is selected from the final population

based on the highest fitness score.

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 for
return best_hyperparameters, Y_{bt}

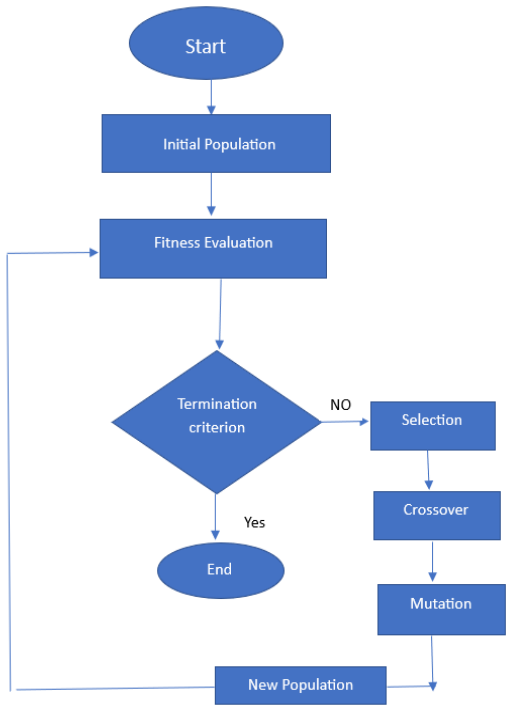

Fig. 2. Process of Genetic Algorithm

Figure-2 2 shows the process of Genetic Algorithm.

*2) **Ant-Colony Optimization**:* The are several ant-colony optimization variants of which MMAS (Min-Max Ant System) is the most prevelant one for hyperparaeter tuning. The following are the steps taken to implement the Ant Colony Optimization (ACO) algorithm ie. MMAS variant for hyperparameter optimization in Artificial Neural Networks (ANNs).

Pheromone Initialization:
The pheromone matrix, denoted as τ, was initialized with a constant value of τ_ 0 for all hyperparameter combinations. This ensured that all hyperparameter combinations had an equal initial pheromone level.

Pheromone Update:

- Evaporation: At the beginning of each iteration, the pheromone values were evaporated using an evaporation rate (ρ). This was done to avoid stagnation and encourage exploration of the search space. The evaporation was performed by reducing the pheromone values by a factor of (1 - ρ).
- Pheromone Deposit: After evaluating the accuracy of the ANN model using a particular hyperparameter combination, the pheromone values were updated. The pheromone deposit, denoted as (i, j), was added to the corresponding pheromone trail based on the accuracy obtained. A higher accuracy resulted in a larger pheromone deposit, which indicated a better quality of the hyperparameter combination.

Pheromone Bounds:
To ensure that the pheromone values remained within predefined bounds, a min-max update was applied after each iteration. The pheromone values were clipped to the minimum (τ_ min) and maximum (τ_max) bounds. This prevented the pheromone values from becoming too small or too large.

Ant Decision Rule:
At each iteration, the ants probabilistically selected hyperparameter combinations based on the pheromone values and a heuristic function.

The probability, denoted as P(i, j), of an ant selecting a hyperparameter combination (i, j) was calculated as the ratio of the product of the pheromone value and the corresponding heuristic information to the sum of the products over all hyperparameter combinations.

The heuristic information, denoted as η(i, j), represents the attractiveness of the hyperparameter combination (i, j) based on some predefined criteria. This information guides the ants towards more promising hyperparameter combinations.

Hyperparameter Bounds and Search Space:
The hyperparameters, including the learning rate, number of epochs, and batch size, were constrained within specified bounds to define the search space for the MMAS algorithm:

MMAS Algorithm Execution:
The MMAS algorithm was executed for a predefined number of iterations to explore the hyperparameter search space. The steps involved in each iteration are as follows:

Pheromone Update:
Evaporation: The pheromone values were evaporated by reducing them by a factor of (1 -ρ).

Ant Behavior:

- Ant Selection:
  Each ant probabilistically selected a hyperparameter combination based on the pheromone values and the heuristic function. The probability of selecting a hyperparameter combination (i, j) was calculated using the formula

  P(i, j) = [τ(i, j) * η(i, j)] / σ[(i', j') * η(i', j')].

– Evaluation:
The selected hyperparameter combinations were used to train and evaluate the ANN model. The accuracy of the model was computed.

– Pheromone Deposit:
The accuracy obtained by each ant was used to update the pheromone values. A higher accuracy resulted in a larger pheromone deposit, indicating a better quality of the hyperparameter combination.

– Pheromone Bounds:
The pheromone values were bounded to ensure they stayed within the specified limits. They were clipped to the minimum (τ_min) and maximum (τ_ max) bounds.
– Termination Criteria:
The iterations continued until the maximum number of iterations was reached.

Figure-3 3 shows the process of Ant Colony Optimisation.

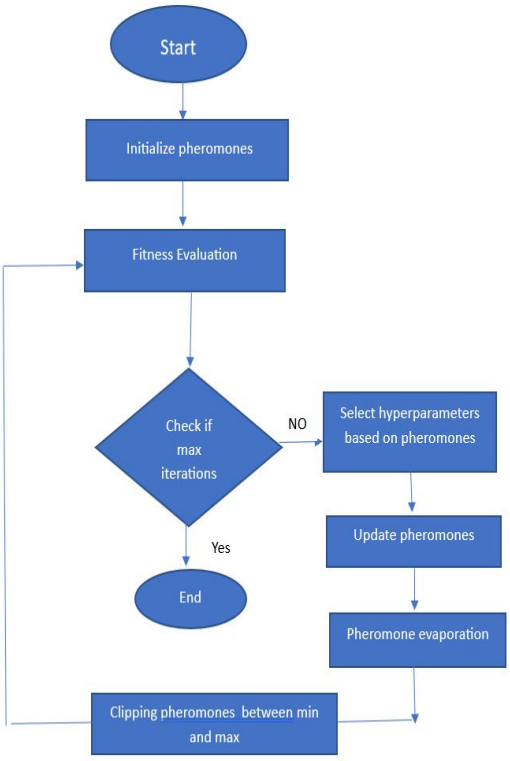

Fig. 3. Process of Ant Colony

Algorithm 2 MMAS Algorithm

INPUT

– Number of ants(num_ants)
– Max number of iterations(max_iterations)
– Evaporation Rate(pheromone_rho)
– Min pheromone value(pheromone_min)
– Max pheromone value(pheromone_max)

OUTPUT

– Best Accuracy(best_accuracy)
– Best Hyperparameters (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
    Evaluate the $accuracy$ of the ANN model using the selected $hyperparameters$
    Append $accuracy$ to $ant\_accuracies$
    Append $hyperparameters$ to $ant\_hyperparameters$
  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

*3) **Particle Swarm Algorithm**:* Particle Swarm optimisation algorithm is a global heuristic method developed based on Swarm intelligence. It is evolutionary algorithm that is extracted from nature's behaviour. The major change in particle swarm optimisation when compared to basic swarm optimisation is the cooperation of each member or individual which we call a particle. The simple and optimised approach obtained is due to individual contribution. Due to this Particle Swarm

Optimisation Algorithm is being widely used in fields of optimisations of certain complex functions, signal processing, neural network training.

Major Principles of particle swarm optimisation:

1) Maintaining inertia.
2) Updating the positions according to the most optimal position of particle in each iteration.
3) Updating the positions according to the most optimal position of swarm in each iteration.

All particles in the swarm maintains position, speed and updates it through each iteration. Process of Particle swarm optimisation is given in fig
The speed V of particle i after k iterations is a_{id}^k and position X of particle i is a_{id}^k. Then the velocity update after k iterations will be:

$$v_{id}^{k+1} = v_{id}^k + c_1 r_1^k (pbest_{id}^k - x_{id}^k) + c_2 r_2^k (gbest_d^k - x_{id}^k)$$

here, $pbest_{id}^k$ refers to optimistic position after k iterations. $gbest_{id}^k$ refers to dimension quantity of swarm at optimal position.

$$x_{id}^{k+1} = k_{id}^x + v_i d^{k+1}$$

The maximum value of v_{dmax}indicates that the optimal solution is very far away form the present position. The minimum value of v_{dmax} indicates that it is the local optimal solution. The speed of the particle lies in the space of $-v_{dmax}$ to v_{dmax}.

Pseudo Code of Particle Swarm Optimisation:

Algorithm 3 Particle Swarm Optimization

```
Initialize X_i, V_i, iteration, pbest, gbest
Generate different random particles (P)
for each Particle i do
    Compute the fitness function f_i
    Update the selected pbest, gbest values
end for
while iteration do
    for each particle in swarm do
        Update X_i, V_i
        if X_i > limit then
            X_i = limit
        end if
        Calculate fitness function f_i
        Update pbest, gbest
    end for
end while
```

Figure-4 4 shows the process of Particle swarm optimisation. Advantages of Particle Swarm Optimisation:

1) Simple math and simple calculations.
2) It doesn't have mutation and overlapping conditions as only the optimistic particles can transmit information to further generations.

Disadvantages of Particle Swarm Optimisation:

1) PSO algorithm work only under the consideration of co ordinate system.

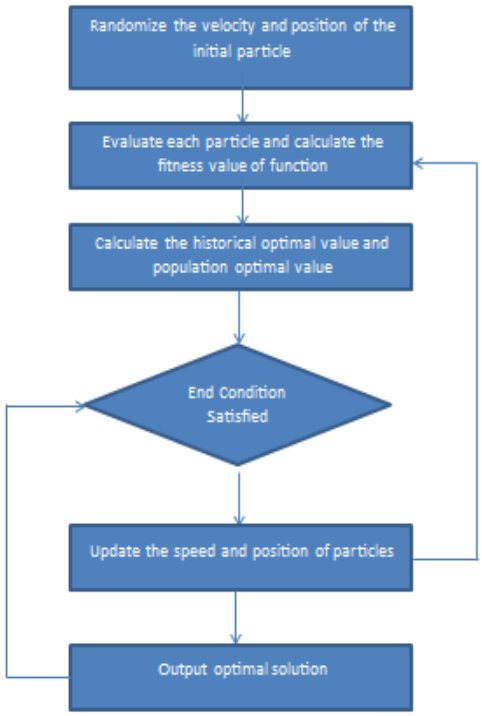

Fig. 4. Process of Particle Swarm Optimisation

C. *Creating an app*

For predicting whether or not a person's cardiovascular functionality is good, and how far he is prone to get a heart disease; we have generated a predictive model (classification) using which we developed a user-friendly mobile application. We preferred mobile application because not all persons have access to laptops or desktops and also is portable.

1) Problems faced and Solution approched: We are creating an app in android studio using Java which works under different environment compared to that of ANN in python and hence to avoid this issue, we build an API through which Java environment can post requests to the model and get result back from it.

2) Exporting the model: We exported the dash model which showed the best cross-validation accuracy as PICKLE file along with its' feature names to the environment where we are creating the app.

3) Creating a flask API: A Flask API was developed to establish communication between the mobile app and the trained model. The Flask framework, being lightweight and suitable for building RESTful APIs, was chosen for its simplicity and flexibility. The API acted as an intermediary, handling incoming requests from the mobile app, preprocessing the data as necessary, and returning the predicted results in a structured format. This facilitated real-time predictions and seamless interaction between the user interface of the app and the model.

4) Building app on Android Studio: The mobile app for heart disease prediction was developed using Android Studio.

Android Studio provided a comprehensive set of tools and libraries to design the app's layout, implement user interactions, and handle data input/output. The app featured an intuitive and user-friendly interface where users could input relevant health parameters. The app seamlessly integrated the Flask API to obtain real-time predictions from the deployed model, providing accurate heart disease risk assessments to the users.

5) Creating Web Application: we first, converted the best accurate model to API by using pickle which convetrs the model to .sav or .h5 file and then using streamlit, a third-aprty software we deployed the heart disease prediction model.

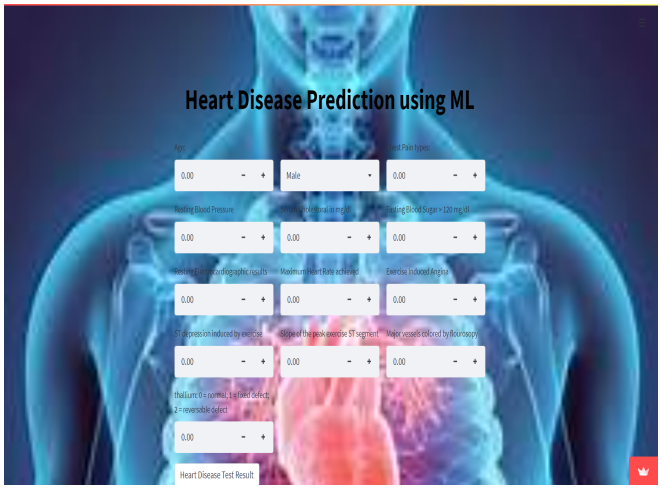

Fig. 5. Web Application

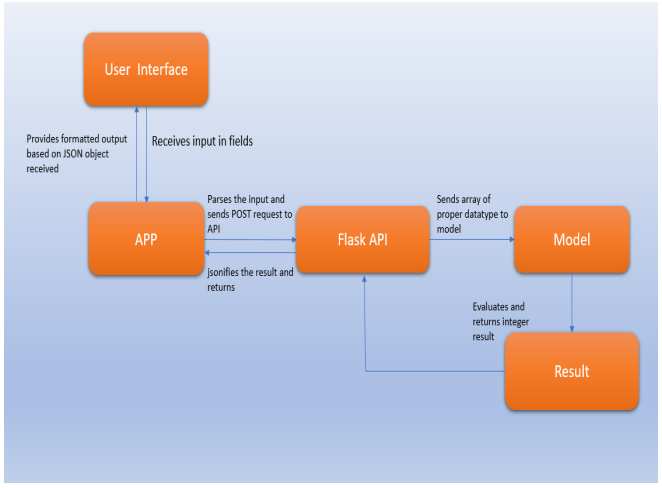

Fig. 6. Work flow of app

Figure-5 6shows the work flow of app.

V. Datasets

we considered the dataset from the kaggle Here is the link to datset: Datset. This dataset contains 1025 rows (data points) and 14 columns (features). The dataset that we considered contains 13 independent features and 1 target feature.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

Fig. 7. Dataset

VI. 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 Hidden layer, which contains 8 neurons (prefer 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 the using the same activation function "relu".

2) Compiling the Model : We then compiled this ANN model using the "adam" optimizer, "Binarycrossentropy" loss function and the metric used is accuracy.

3) Fitting the Model: After Compiling the model, the train data is fit into the model.so that model gets trained using the concept of back propagation , for this purpose , we are using 75 epochs and batch size of 32(prefer power of 2).

4) Evaluating the Model: Now, the model is ready for evaluation.To know the performance of the model , we are using the metric accuracy.

B. Results

We created an application using flask and Postman API which predicts whether the given patient has a heart disease or not based on various input parameters.
The Front page of app looks as given in 8.

C. Discussion and Complexity Analysis

1) Complexity Analysis of Genetic Algorithm: :

Initializing the population:
The time complexity of initializing the population is O(n) since it involves creating a list of size "n" and generating random hyperparameters for each individual.

Comparitive Analysis of different automated hyper parameters						
Tuning Algorithm	Epoch	Batch size	Neurons	Activation	Learning Rate	Accuracy
NA	55	32	16	sigmoid	0.01	79.63
Grid Search	100	32	64	relu	0.01	83.33
Randomised Search	75	32	60	relu	0.01	81.48
Bayesian Optimisation	55.27	58.76	59.82	relu	0.02325	83.33
Genetic Algorithm(Inbuilt)	32	64	16	sigmoid	0.01	81.48
Particle Swarm Algorithm(Inbuilt)	10	32	16	sigmoid	0.046	83
Genetic Algorithm(Custom)	24	65	16	sigmoid	0.0519	93.17
Ant Colony Optimisation	20	32	16	sigmoid	0.0112	90.73
Particle Swarm Optimisation	24.17	22	44.68	relu	0.01	83.12

TABLE II
COMPARITIVE ANALYSIS OF DIFFERENT AUTOMATED HYPER PARAMTERS

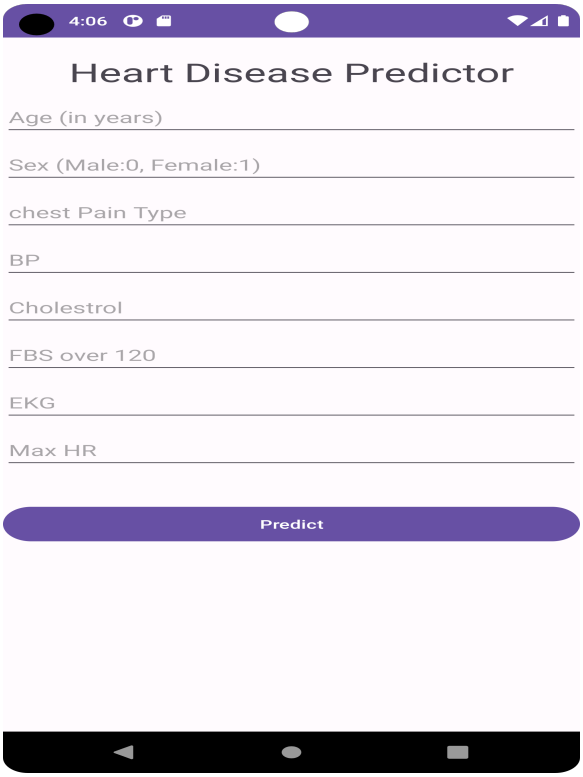

Fig. 8. App model

- Evaluating the fitness of each set of hyperparameters:O(n)
  This step involves iterating over the population and calling the "evaluate_ fitness" function for each individual.

- Selecting the fittest individuals to be parents of the next generation:O(n * log(n))
  This step requires identifying the individuals with the highest fitness scores. Sorting the fitness scores takes O(n * log(n)) time, and selecting the top "n / 2" individuals takes O(n) time.

- Generating offspring by combining the parents' hyperparameters: O(n)
  This step involves creating new individuals by randomly selecting and combining hyperparameters from the selected parents.

- Mutating the offspring by randomly changing hyperparameters:
  This step involves randomly modifying hyperparameters of the offspring individuals with a certain probability. It is done for "n / 2" individuals, resulting in a time complexity of O(n).

- Replacing the old population with the new offspring:O(n)
  This step simply involves assigning the new offspring to the population variable.

- Selecting the best set: O(n)
  Finding the best set from the final population requires iterating over the population to identify the individual with the highest fitness score.

Overall time complexity:

The time complexity of evolving the population over "G generations can be derived as follows:

1) Evaluating the fitness: O(n)
2) Selecting the fittest individuals: O(n * log(n))
3) Generating offspring: O(n)
4) Mutating the offspring: O(n)
5) Replacing the old population: O(n)

Since these steps are repeated for "G" times, the overall time complexity is O(G * n*log(n)).
Selecting the best set of hyperparameters: O(n)
Overall Time Complexity: O(n + G * n*log(n).

2) Complexity analysis of Ant Colony Optimization: :

Initialization: O(1)
The initialization step involves creating the pheromone matrix and best pheromone trail with fixed dimensions. As the dimensions are known beforehand, the time complexity is constant.

Ant Behavior:

- Loop over ants: O(n)

- Selecting hyperparameters probabilistically: O(1)
  This step involves probabilistically selecting hyperparameters based on the pheromone values and a heuristic function. As it does not depend on the number of ants, the time complexity is constant for each ant.

- Evaluation function (evaluate_ fitness): O(e * b * t)

The evaluation function trains and evaluates an ANN model using the selected hyperparameters. The time complexity depends on the number of epochs (e), batch size (b), and the size of the training dataset (t).

- Best Accuracy and Hyperparameters Update: O(1)
  Updating the best accuracy and corresponding hyperparameters if a new best is found can be done in constant time as it involves simple assignment operations.

- Best Pheromone Trail Update: O(m)
  Updating the best pheromone trail by taking the element-wise maximum between the best trail and the current pheromone matrix requires iterating over each element in the matrix. Thus, the time complexity is proportional to the number of elements in the pheromone matrix (m).

- Pheromone Matrix Clipping: O(m)
  Clipping the pheromone matrix values to stay within the specified bounds also requires iterating over each element in the matrix. Therefore, the time complexity is proportional to the number of elements in the pheromone matrix (m).

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 Optimisation: :

1) Initialisation
   Initialisation step involves initializing the positions,velocities randomly in a multidimensional space.Time Complexity for this step will be O(N*D) as we have to generate the random values for all particles in each dimensions.Here N represents the number of particles, D represents the dimensionality of the problem.

2) Evaluation
   Considering all iterations in the algorithm,the fitness function or the objective function is calculated for each particle. So the time complexity in this step depends on the fitness function itself. So Time Complexity of Evaluation step would be O(fitness) where fitness refers to the fitness function or the objective function. Time complexity of fitness function varies as per the problem statement.In considered algorithm the Time complexity achieved is O(num_hyperparameters) i.e number of hyper parameters in the heart disease prediction dataset.

3) Update
   According to the best positions found by swarm in each iteration has to be updated as initial positions in the successive iterations. Time Complexity of this step will be O(N*D) as we need to update velocities and positions of each particle in each dimension.

Considering all the steps above, The final Time complexity of PSO can be concluded as:

O(T*(N*D+fitness))

Factors effecting Time complexity of PSO algorithm:

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

2) 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.

3) Number of Iterations(T):
   Time Complexity also depends on number of iterations required to terminate the algorithm by meeting the required conditions. The algorithm should converge to optimal or near optimal solution finally.

VII. Conclusion

In this project, we worked on hyperparameter optimization using automated techniques such as model free and model based. In model free, we worked on Grid search , which gave an accuracy of 83.33% but the time complexity is O(nK̂) where n is the number of values for each hyperparameter and k is the number of hyperparameters.This grid search uses blind search, as a result exponential time complexity.Then we worked on Randomized search, which gave an accuracy of 81.48%. The time complexity for this is,O(n_iter * (epochs * batch_* num_training_samples)).This technique has drawbacks such as inefficiency with large search spaces, lack of guidance or adaptability, uniform sampling without considering important regions, sensitivity to search space distribution, and limited control over the search process.Considering the draw backs of model free techniques, we now explored model based techniques.

In model Based, we worked on Bayesian Optimization , which gave an accurcay of 83.33%.The time complexity for this is O(n_ * (epochs * num_ training_ samples)).But the drawbacks of Bayesian optimization include the higher computational cost compared to simpler methods like grid or random search, as well as the requirement for careful selection and tuning of acquisition functions and surrogate models to ensure optimal performance.Then we worked on Evolutionary algorithms such as Genetic , which gave an accuracy of 93.17%,and computionally efficient as metioned above in the time complexity analysis section.In Ant Colony Optimization , we got an accuracy of 85.15% and repsective time complexity is mentioned in time complexity analysis section. Then Particle Swarm Optimixzation gave an accuracy of 83.12% and the respective time complexity is mentioned in time complexity analysis section.

Now, for the deployment we used streamlit ,a third party software for deploying the machine learning models.At first , we we converted the best accurcate model which has less time complexity to .sav file , which contains weights of the architecture.Of the above evolutionary techniques, we now wanted to select one best model for deployment.we considered elimination procedure, In which Genetic and Ant Colony are good accurate models,out of these one modrl has to be selected for deployment.Genetic Algorithms (GAs) are commonly preferred for tasks involving extensive search spaces, intricate hyperparameter interactions, and the requirement for global exploration. 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 that we used Genetic algorithm for the deployment.

INDIVIDUAL CONTRIBUTION

- **G V Ravi Ram**
  1) Explored different variants of Ant Colony Optimization ALgorithms.
  2) In Evolutionary Algorithms I focused on AntColony Optimisation and its different real life applications, implementation in machine learning domain.
  3) Preferred Heart disease dataset for implementing ACO algorithm and got an accuarcy of 85.18% and compared the respective result with the basic machine learning ANN model and other conventional algorithms like Grid Search, Random Search, Bayesian Optimisation and other evolutionary algorithms such as Genetic Algorithms, Particle Swarm Optimisation.
  4) Helped mates building the experimental setup.
  5) Implemented Flask REST API which is ready to go live.
  6) Implemented an Android app including front-end and back-end which can go in air once the API goes live.
- **Mogilipalem Hemanth Kumar**
  1) Explored different types algorithms related to Conventional Algorithms.
  2) Literature survey of many recent Research papers related to hyperparameter optimization ,randomized search , Bayesian optimization and genetic algorithm.
  3) Implementing Randomized search for best hyperparameters.
  4) Implementing Bayesian Optimization for best hyperparameters.
  5) Implementing Genetic Algorithm for best hyperparameters and improved the accuracy to 93.17% , Detailed Analysis of the Algorithm with respect to time and space.
  6) Comparison of Genetic Algorithm performance to other Conventional and evolutionary algorithms.
  7) Created Web Application for heart disease prediction using streamlit software. Both front-end (using Python) and Back-end (using API created by Pickle).
- **Jayanth Prathipati**
  1) Explored different types algorithms related to Conventional Algorithms.
  2) In Conventional Algorithms I worked on Grid Search, which gave better accuracy of 83.33% but when considered time complexity performance is not satisfactory.
  3) In Evolutionary Algorithms I focused on Particle Swarm Optimisation and its different real life applications, implementation in machine learning domain.
  4) Preferred Heart disease dataset for implementing PSO algorithm and got an accuarcy of 83.12% and compared the respective result with the basic machine learning ANN model and other conventional algorithms like Grid Search, Random Search, Bayesian Optimisation and other evolutionary algorithms such as Genetic Algorithms, Ant colony Optimisation.
  5) Worked with mates for creating an android app for the heart disease prediction model using postman API, flask technologies.

Gantt Chart is attached below 9:

Fig. 9. Gantt Chart

IMPLEMENTED/BASE PAPER

- **A**uthors: Tani, Laurits, et al
- **P**ublished: The European Physical Journal C 81 (2021): 1-9.
- **D**ataset used: Higgs boson machine learning challenge (HBC) for high energy physics.

In this paper, authors stated that: By fine-tuning the hyperparameters, the sensitivity of data analysis significantly improves, resulting in a 12-13% enhancement in the AMS score compared to using default values. Both PSO and GA demonstrate their effectiveness in minimizing complex functions, such as the Rosenbrock function. Interestingly, randomly exploring different hyperparameter sets and selecting the best one yields comparable results to PSO and GA. The automated nature of PSO and GA streamlines the hyperparameter optimization process, relieving researchers from manual tuning. This automation proves to be invaluable for HEP data analysis, where optimizing hyperparameters is crucial.
In our study, we have utilized a foundational research paper on hyperparameter optimization techniques for high energy physics (HEP) data analysis as a reference for our work on optimizing hyperparameters for a heart disease dataset. Drawing inspiration from the concepts presented in the paper, we focused on applying these methods specifically in predicting heart disease. we automated the process of hyperparameter tuning, eliminating the need for manual adjustments. Our objective was to evaluate the efficiency of these techniques in optimizing hyperparameters and improving the accuracy of heart disease prediction models on our selected dataset.And our work moved one stop forward , that we implemented Ant Colony Optimization algorithm to this heart disease dataset as the part of novelty to the existing methodology.

VIII. APP REVIEW ANALYSIS

The review analysis of our application is given in below figures. From these we will try to improve the user interface and accuracy by applying on several models and different datasets.

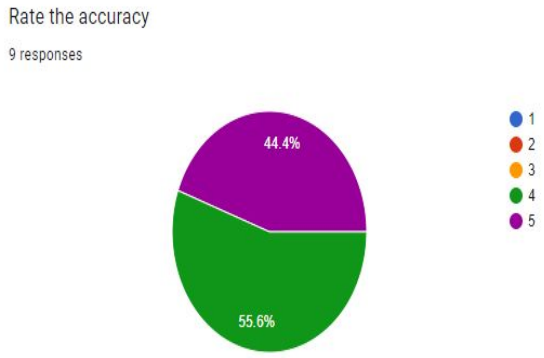

Fig. 10. Speed utility review

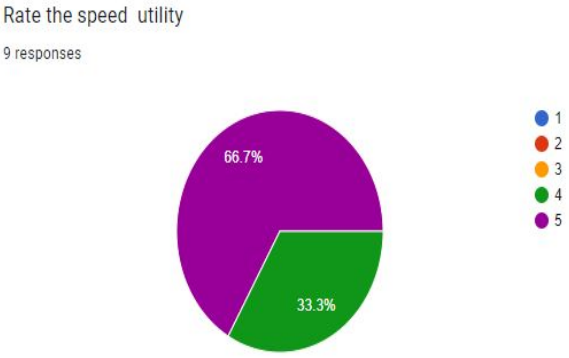

Fig. 11. User interface review

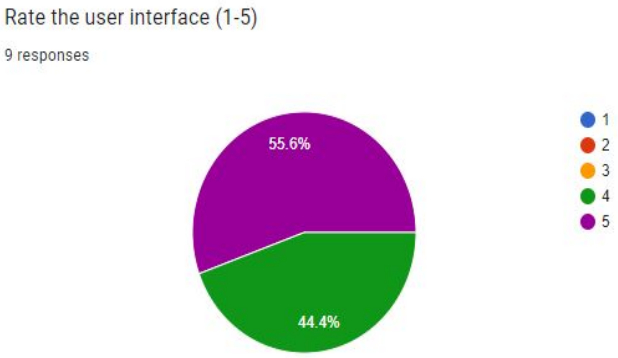

Fig. 12. Accuracy review

REFERENCES

[1] 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.

[2] 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.

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

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

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

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

[7] 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).

[8] The Use of Random Search and Genetic Algorithms to Optimize Stochastic Robustness Functions Christopher I. Marrison* and Robert F. Stengel Princeton University Department of Mechanical and Aerospace Engineering Princeton, NJ 08544

[9] Bayesian Optimization for Accelerating Hyper-parameter Tuning Vu Nguyen University of Oxford, United Kingdom(2019)

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

[11] On Test Case Prioritization Using Ant Colony Optimization Algorithm Weixiang ZHANG1, Yuhua QI1, Xuebo ZHANG2, Bo WEI1, Min ZHANG1 and Zhaohui DOU1

[12] Genetic Algorithms: Concepts and Applications K. F. Man, Member, IEEE, K. S. Tang, and S. Kwong, Member, IEEE