BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 72(2), 2024, Article number: e148942
DOI: 10.24425/bpasts.2024.148942

An improved facial image retrieval using hybrid cheetah optimization algorithm

Balasubramanian C* and **Raja Sekar J**

Department of Computer Science and Engineering, Mepco Schlenk Engineering College, Sivakasi 626005, India

Abstract. With the advent of social media, the volume of photographs uploaded on the internet has increased exponentially. The task of efficiently recognizing and retrieving human facial images is inevitable and essential at this time. In this work, a feature selection approach for recognizing and retrieving human face images using hybrid cheetah optimization algorithm is proposed. The deep feature extraction from the images is done using deep convolutional neural networks. Hybrid cheetah optimization algorithm, an improvised version of cheetah optimization algorithm fused with genetic algorithm is used, to choose optimum features from the extracted deep features. The chosen features are used for finding the best-matching images from the image database. The image matching is performed by approximate nearest neighbor search for the query image over the image database and similar images are retrieved. By constructing a k-NN graph for the images, the efficiency of image retrieval is enhanced. The proposed system performance is evaluated against benchmark datasets such as LFW, MultiePie, ColorFERET, DigiFace-1M and CelebA. The evaluation results show that the proposed methodology is superior to various existing methodologies.

Keywords: deep learning; feature selection; facial image retrieval; cheetah optimization; approximate nearest neighbor.

1. INTRODUCTION

Over the years, numerous contributions have been made to face recognition in the form of different algorithms and their enhancements [1, 2]. However, with the growing demands and technologies, there is still room for advancement in this field. Despite the fact that majority of face recognition algorithms perform effectively in controlled environments, performance of the system may be influenced by various factors [3, 4] such as poor lighting, camera angles, distance, etc. This makes face recognition harder.

In our work, we are proposing a human facial image retrieval system hoping to address the challenges and issues in the current systems using deep convolutional neural networks (DCNN) with hybrid cheetah optimization (HCO). The hidden layers in the convolutional neural network (CNN) are increased in order to extract deep features from the images.

The most popular deep learning method for extracting features from images is CNN [5, 6]. It involves artificial neural networks implementing convolutional operations on an image to extract its unique features. A progressive hierarchy of deep features is created by the DCNN, a deep learning technique that uses convolutional layers and feature extraction. DCNN has layers such as convolution, pooling, ReLU [7], and normalization layers. Convolutional layers and sub-sampling layers are arranged alternatively in the neural network hierarchy, succeeded by a fully connected layer. These layers learn the key image features, and finally the fully connected layer generates the classification vectors.

The cheetah optimization (CO) [8] algorithm is a swarm intelligence meta heuristic algorithm based on the hunting behavior of the cheetahs. Compared to recent algorithms like grey-wolf optimization (GWO), whale optimization algorithm (WOA), marine predators algorithm (MPA), etc., the CO algorithm has demonstrated performance improvements in finding solutions to complex optimization problems. The optimum feature subset is chosen using the proposed HCO method intelligent feature space search by omitting unwanted information from the image. Afterwards, The k-NN graph-based approximate nearest neighbor (ANN) method is used for efficient image retrieval.

The proposed work is claimed to be novel in such a way that it employs a hybrid approach by combining CO with genetic algorithm (GA) towards improving feature selection performance. The image retrieval performance is improved by constructing a k-NN graph offline, which is used for approximate nearest-neighbor search. Contributions of the proposed work is expressed as follows:

1. By adding more hidden layers in DCNN, the deep features of facial images are extracted.
2. With the help of HCO, based on the natural hunting behavior of cheetahs combined with GA, the optimal feature subset is selected.
3. A k-NN graph is constructed to speed up the approximate neighbor search.
4. The proposed methodology effectiveness is compared with that of other optimization methods. The proposed methodology has shown excellent results when comparing with the existing methodologies.

Rest of the paper is organized as follows: Section 2 discusses the literature survey of works related to the proposed domain of study. The proposed methodologies are illustrated in Section 3. The implementation methods, results and comparison study are

*e-mail: balasubramanian@mepcoeng.ac.in

Manuscript submitted 2023-03-10, revised 2023-12-07, initially accepted for publication 2024-01-03, published in March 2024.

presented in Section 4. The concluding remarks are provided in Section 5. The symbols and notations used in this paper are listed in Table 11.

2. RELATED WORK

2.1. Feature extraction

There are many approaches proposed by various researchers to extract facial image features [9]. Belhumeur *et al.* [10] proposed 'Fisherface' method, which considered the image pixels as coordinates in high dimensional space. But the method has limitations over images influenced by lighting and shadow regions. Scale invariant feature transform (SIFT) [11] is another popular method for feature extraction. This algorithm helps in locating 'keypoints' (local features) of an image. However, this method suffers from slowness and consumes more space. Surbhi Gupta *et al.* [12] proposed combination of speeded up robust features (SURF) along with SIFT. SURF method is used to extract local features fast. They used both SURF and SIFT to generate combined feature vector generation of facial images. This method has shown varying performance across different datasets. Bobulksi [13] proposed a 2D hidden Markov Model for face recognition. This work has its own limitations regarding data acquisition. Additionally, the collected images may be deprived of image illumination invariance.

When it comes to the usage of neural networks, backpropagation neural network (BPNN) [14] is widely used architecture. In BPNN, the input data given to the input layer is processed in the hidden layer based on weight values, and the output is passed on to the output layer. Robert Szmurło *et al.* [5] proposed that the performance of classifiers can be improved by creating an ensemble of classifiers using CNN. Hana Ben Fredj *et al.* [15] proposed a CNN-based approach to detect faces from massive noisy data. They achieve this by rigorously training the model with multi-task cascaded CNN. This method has limitations in terms of iteration steps during training. Suleman Khan *et al.* [16] used AlexNet, a variation of CNN with 8 layers for facial recognition. They used this method for applying face recognition in portable smart glasses. The method heavily suffers from light and pose variations related challenges. Ali Elmahmudi *et al.* [17] proposed the idea of recognizing faces with partial data.They used pretrained VGGF model along with linear support vector machine for training the system.

2.2. Feature selection

R. Pati *et al.* [18] used particle swarm optimization (PSO) for face recognition, a computational approach that optimizes a problem by repeatedly attempting to boost the quality of a solution. H. Zhi *et al.* used genetic algorithm [19] for feature dimension reduction while recognizing faces. This method suffers from diminishing efficiency with respect to the total number of iterations. Annamalai [20] used enhanced firefly algorithm for recognizing faces. But the work suffers from limited accuracy. Tajinder Kumar *et al.* [21] used whale optimization algorithm for feature selection in biometric systems. Although this methodology works fine for limited systems, the performance gets degraded when the scope gets broader. Haseena Sikkandar *et al.* [22] used grey wolf optimization algorithm to select optimum features for facial recognition. But the system has inferior accuracy rate when compared with the proposed HCO. Similarly many of the swarm intelligence metaheuristic algorithms such as salp swarm algorithm [23], marine predators algorithm [24], whale optimization algorithm [25], butterfly optimization algorithm [26] etc., are proposed for feature selection. One among them is CO algorithm [8] based on the hunting behavior of cheetahs.

2.3. Image retrieval

Similar matching images for a query image are retrieved based on distance metrics such as Manhattan distance (L1) and Euclidean distance (L2) [27]. The performance of the image matching can be improved with the help of an approximate nearest neighbor search [28]. Wen Li *et al.* [29] applied approximate nearest neighbor search on various algorithms and evaluated its performance. They proposed that graph-based approach in ANN is more suitable for high dimensional data through which stuck up in local minima can be avoided. Xiaoliang Xu *et al.* [30] proposed multiattribute ANN search a small world graph-based approach for nearest neighbor search. This method may not be suitable for situations where attribute length is greater.

3. PROPOSED WORK

The proposed system consists of image pre-processing, deep feature extraction using DCNN, feature selection using HCO, and image extraction using ANN search method.

After selecting the best feature subset using HCO, a k-NN graph is constructed offline. A greedy approximate k-nearest neighbor search [28, 31] for the given query is performed on the k-NN graph, and the most recent top K nearest nodes are returned. The proposed system architecture is illustrated in Fig. 1. The detailed discussion on various components of the system is given below:

3.1. Pre-processing of images

Image pre-processing involves suppressing the background, removing noise and artifacts, and detecting the region of interest. The purpose of pre-processing is to eliminate any undesirable distortions from the image and improve its attributes so that it may be used in subsequent processing. Low-abstraction images are subjected to pre-processing. Background suppression, opening and closing operations, detecting facial landmarks and noise removal form the stages of image pre-processing. Once the landmarks are found, the image is cropped and resized to 112×112.

3.2. Feature extraction

The deep features of the image are extracted with the help of DCNN. These are the state-of-the-art techniques used to analyze the patterns in images. DCNN receives input in the form of images and uses them to train the classifier. Along with input and output layers, the DCNN has four layers: convolution, pooling, activation, and fully connected layers. In the convolution layer, a convolution filter is applied to the image during which

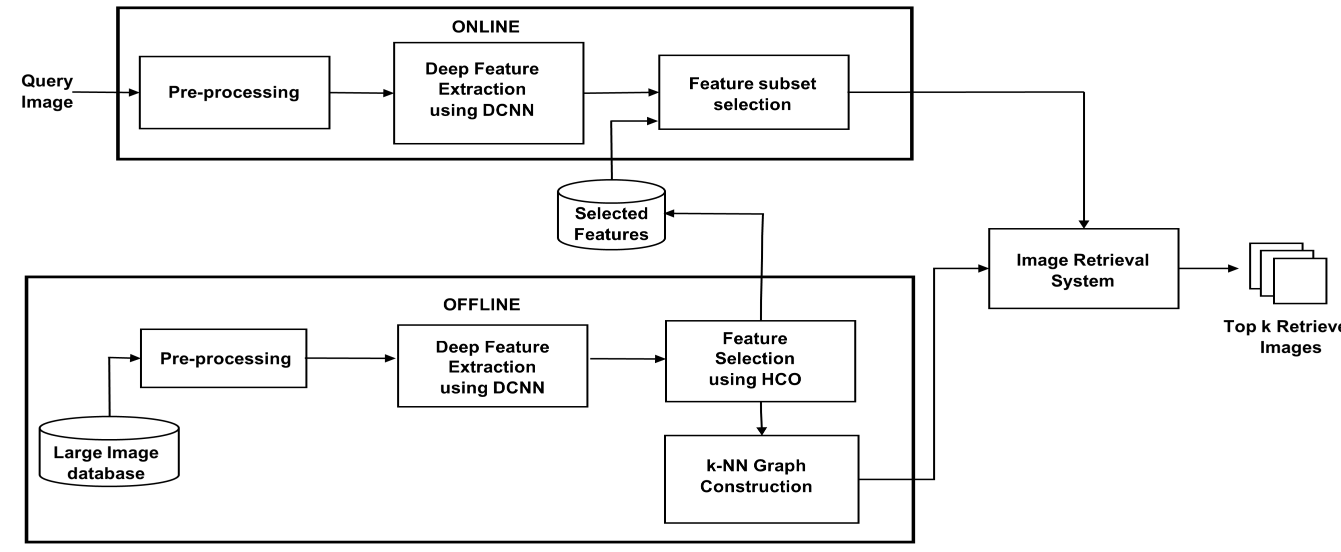

Fig. 1. Proposed system architecture

a kernel or filter is passed over the image multiple times and a scalar product operation is performed by multiplying the weight value. The pooling layers progressively scale down the image while retaining only the most important details. The image is divided into sub-regions. For each sub-region, the maximum value is considered as output. By reducing the amount of calculations and parameters in the network, pooling layers helps in controlling the problem of overfitting. In the activation layer, the rectified linear unit activation function $f(x) = \max(0, x)$ is used. This function will return 0 for negative input and returns the same value if the input is positive. In the fully connected layer, a linear transformation through a weight matrix is applied to the input vector. And then, a nonlinear transformation is applied using an activation function f. By applying a softmax function to the outputs of fully connected layers, we obtain a probability distribution.

The DCNN produces a feature vector with deep features. The detailed description of each layer in the proposed DCNN model is shown in Fig. 2. The proposed DCNN has convolution layers-8, max-pooling layers-2, and avg-pooling layer-1. Finally, there will be a softmax layer. The details of input, filter, and output in each layer are given below:

1. The convolution layer 1 (C1) has 5×3 with filter size 32. The output of C1 is passed on to convolution layer (C2).

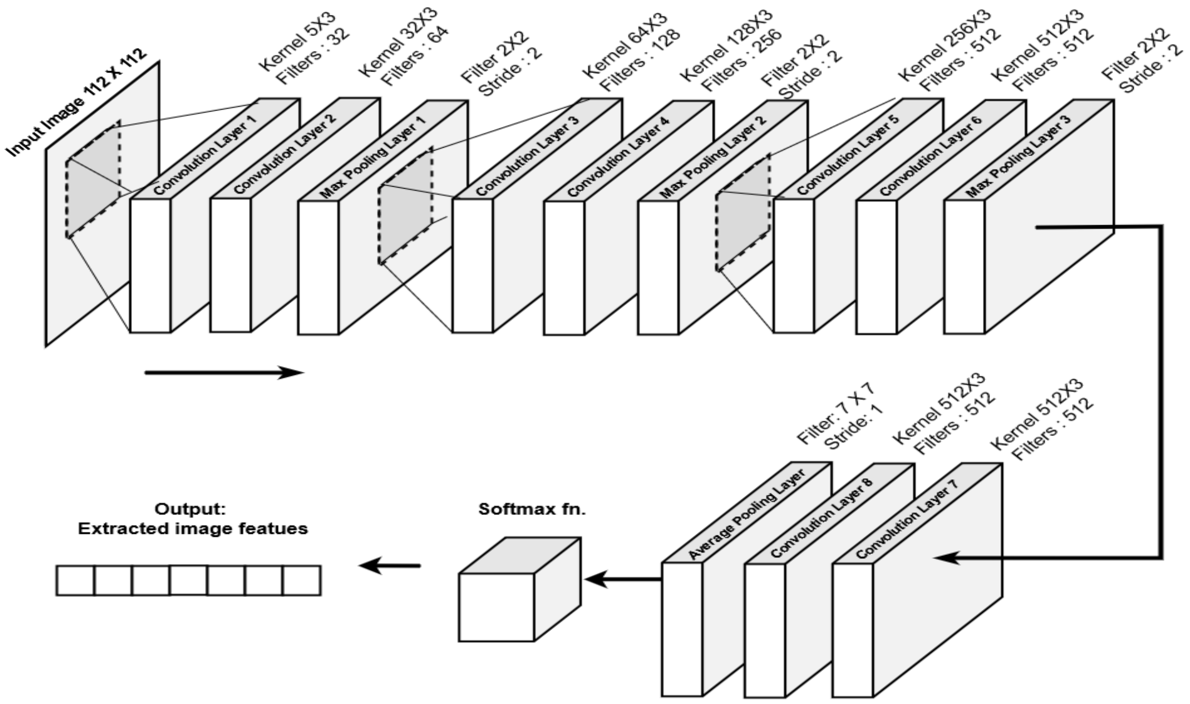

Fig. 2. Architecture of DCNN

2. C2 has 32×3 with filter size 64. The output of C2 is passed on to Max-pooling layer 1 (M1).
3. M1 has filters 2×2 with stride 2. The output of M2 is passed on to convolution layer 4 (C3).
4. C3 has 64×3 with filter size 128. The output of C3 is passed on to convolution layer 4 (C4).
5. C4 has 128×3 with filter size 256. The output of C4 is passed on to Max-pooling layer 2 (M2).
6. M2 has filters 2×2 with stride 2. The output of M2 is passed on to convolution layer 5 (C5).
7. C5 has 256×3 with filter size 512. The output of C5 is passed on to convolution layer 6 (C6).
8. C6 has 512×3 with filter size 512. The output of C6 is passed on to Max-pooling layer 3 (M3).
9. M3 has filters 2×2 with stride 2. The output of M3 is passed on to convolution layer 7 (C7).
10. C7 has 512×3 with filter size 512. The output of C7 is passed on to convolution layer 8 (C8).
11. C8 has 512×3 with filter size 512. The output of C8 is passed on to Average Pooling layer (AP).
12. AP has filters 7×7 with stride 1.
13. The output features of AP are then sent to softmax function to normalize the obtained features.

3.3. Feature selection

3.3.1. Cheetah optimization

Cheetahs often use the following strategies to hunt down their prey:

1. Searching.
2. Sitting and waiting.
3. Attacking.
4. Leaving the prey.

3.3.2. Searching

Like all the predators, cheetahs will sit or roam, searching for prey that might have entered their territory (search space). To mathematically model these search methods, let $\rho^t_{i,j}$ indicate the current position of cheetah i ($i = 1, 2, 3, \ldots, n$), in the organization j ($j = 1, 2, 3, \ldots, d$), where n is the cheetah population and d is the dimension of the optimization problem. Each prey is considered to be the position of a decision variable that corresponds with the optimal solution. The states of cheetahs in different arrangements construct the population. In order to update the position of cheetah i in each organization, equation (1) is proposed:

$$\rho^{t+1}_{i,j} = \rho^t_{i,j} + \hat{\gamma}^{-1}_{i,j} . \delta^t_{i,j} , \tag{1}$$

where $\rho^t_{i,j}$ and $\rho^{t+1}_{i,j}$ represent the present and next position of cheetah i in the organization j, respectively. t refers to time of hunting where T is maximum of hunting time. $\hat{\gamma}_{i,j}$ refers to random numbers which are normally distributed. $\delta_{i,j}$ is the step length and in most of the cases it is greater than zero and set at $\frac{t}{T} \times 0.001$ as cheetahs move slowly, and search for prey. When a cheetah comes across other enemies, it will quickly move and change its direction. To show this behavior, $\hat{\gamma}^{-1}_{i,j}$ is used differently for different cheetahs in different hunting times. $\delta_{i,j}$ in each cheetah organization is derived by the product of the distance between i-th cheetah and a randomly selected cheetah. At any point of time there will be a cheetah who is closer to the prey. This cheetah is considered to be the leader and this position will be updated based on the distance between a cheetah and the prey.

3.3.3. Sitting and waiting

There are situations where both prey and cheetah are present in their field of vision. In such situations, to reduce the chance of prey's escape, the cheetah may attempt to surprise its prey by hiding itself by lying in the bushes and starts waiting for the prey to move closer. So in this strategy, the cheetah's position will not change until the prey comes closer. This can be represented by equation (2):

$$\rho^{t+1}_{i,j} = \rho^t_{i,j} . \tag{2}$$

3.3.4. Attacking

After the prey has moved into the closer proximity, cheetahs start attacking. In group hunting mode, all the cheetahs adjust their positions based on both the leader and prey positions. This can be expressed as equation (3):

$$\rho^t_{i,j} = \rho^t_{B,j} + \check{\gamma}_{i,j} . \acute{\delta}^t_{i,j} , \tag{3}$$

where $\rho^t_{B,j}$ is the present position of the prey and the best position in the population. The turning factor of cheetah is represented by $\check{\gamma}_{i,j}$ and its interaction factor is represented as $\acute{\delta}^t_{i,j}$. This is because the next position of the leader is determined by the movement of the prey, and the position of any cheetah in the organization is based on the leader's position. The turning factor $\check{\gamma}_{i,j}$ is calculated by equation (4):

$$\check{\gamma}_{i,j} = |\gamma_{i,j}|^{\exp(\gamma_{i,j}/2)} \sin(2\pi\gamma_{i,j}), \tag{4}$$

where $\gamma_{i,j}$ is a random number normally distributed from standard normal distribution.

In order to select from searching or attacking strategy, a random value H is calculated. Initially, a random number R from $[0,1]$ is selected. This R is used to calculate H using equation (5):

$$H = e^{2(1-t/T)} \times (2R - 1). \tag{5}$$

Initially two random numbers r_1, and r_2 from $[0,1]$ are chosen. If $r_2 > r_1$ choose sit and wait strategy. Otherwise another random number r_3 from $[0,3]$ is chosen. If $H < r_3$ then search strategy is opted. Otherwise attack strategy is opted.

3.3.5. Leave the prey

Sometimes, after an unsuccessful pursuit, the cheetah might have to change positions to hunt down its prey or go home. The CO strategies are graphically represented in Fig. 3.

3.4. Hybrid cheetah optimization (HCO)

In order to achieve significantly higher optimization results, GA is combined with CO to propose a new method, HCO. The algorithm for HCO is presented in Algorithm 1. GA involves

Algorithm 1. Hybrid cheetah optimization algorithm

```
Input:
    $POP_{size}$:Population
    D:Image Dataset
    C:Classification algorithm
Output: Global best solution
1  initialize MaxIt, $T_{\max}$
2  t←0
3  it←1
4  $\Gamma_{IFM}$ ← extractFeature(D)        // $|\Gamma_{IFM}| \Longrightarrow POP_{size} \times f : f = No.offeatures$
5  foreach Cheetah $X_i$ : i=1 to $POP_{size}$ do
6      foreach j of Cheetah $X_i$: i=1 to f do
7          $X_{ij}$ ← rand(0,1)
8      end
9      $X_i^{fitness}$ ←computeFitness($X_i$, C, $\Gamma_{IFM}$)        // using Alg. 4
10 end
11 prey←select best solution from X
12 leader←select second best solution from X
13 while $it \le MaxIt$ do
14     m ←randInteger(2,$POP_{size}$)
15     $\Psi$ ← select m cheetahs from X
16     foreach cheetah $\Psi_i$ in $\Psi$ : i=1 to m do
17         foreach j of Cheetah $\Psi_i$: j=1 to f do
18             calculate $\hat{\gamma}, \check{\gamma}, \delta, \acute{\delta}, H$        // using equation (4) and (5)
19             $r_1$ ← random(0,1)
20             $r_2$ ← random(0,1)
21             if $r_1 \le r_2$ then
22                 $r_3$ ← random(0,3)
23                 if $H \ge r_3$ then
24                     perform Attack strategy using equation (3)
25                 else
26                     perform Search strategy using equation (1)
27                 end
28             else
29                 perform Sit and Wait strategy using equation (2)
30             end
31         end
32         update $\Psi_i$ and leader
33     end
34     t ← t+1
35     if $t > (rand() \times T_{\max})$ and leader unchanged then
36         $\Psi_i$ ← prey
37         leader ← Select new leader
38     end
39     update prey
       /* Adding mutation and crossover to improve feature selection performance */
40     $X_{C1}, X_{C2}$ ←Crossover(prey, leader)        // crossover first and second optimal solutions using Alg. 2
41     $X_{C1}^{fitness}$ ←computeFitness($X_{C1}$, C, $\Gamma_{IFM}$)        // using Alg. 4
42     $X_{C2}^{fitness}$ ←computeFitness($X_{C2}$, C, $\Gamma_{IFM}$)        // using Alg. 4
43     prey←select best solution from prey, $X_{C1}, X_{C2}$
44     $pr\acute{e}y \leftarrow Mutate(prey)$        // perform mutation using Alg. 3
45     $pr\acute{e}y^{fitness}$ ←computeFitness($pr\acute{e}y$, C, $\Gamma_{IFM}$)        // using Alg. 4
46     prey←select best solution from prey, $pr\acute{e}y$
47     it ← it+1
48 end
49 return prey(Global best solution)
```

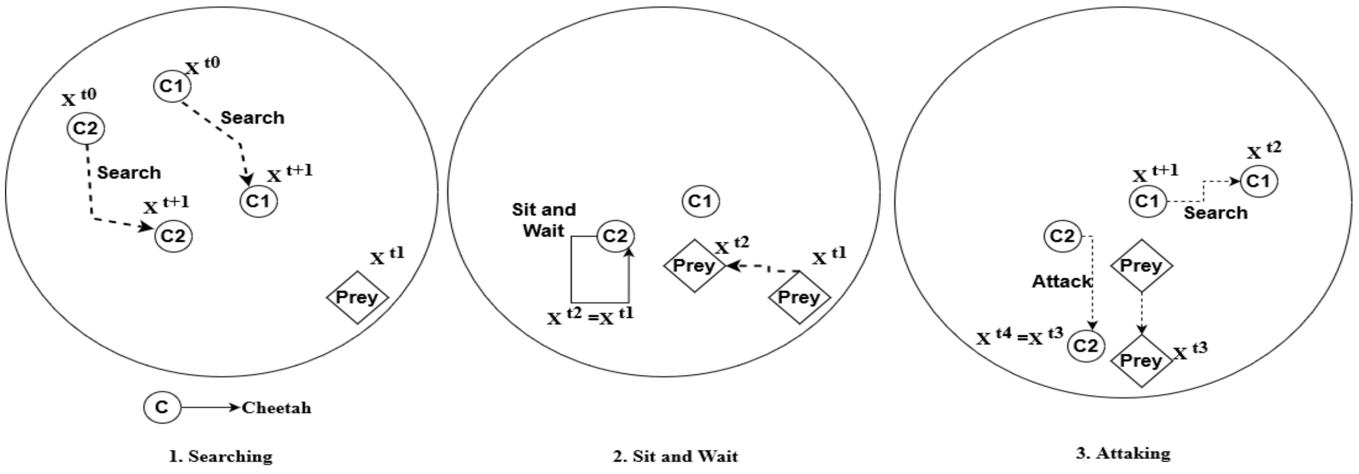

Fig. 3. Graphical representation of cheetah optimization strategies

initialization of the population, calculation of fitness, selection, mutation, and crossover. For HCO, crossover operation is applied for two optimal solutions namely $Parent_1$ – first best solution, $Parent_2$ – second best solution. The single-point crossover technique is followed with the selected parents, where a random crossover point (CP) is selected. The values of both parents are switched until CP, and the remaining values are retained. The crossover operation is illustrated in Fig. 4. The algorithm for crossover operation is given in Algorithm 2.

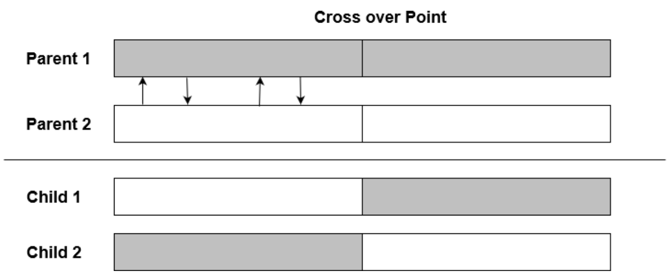

Fig. 4. Single-point crossover

Algorithm 2. Crossover

```
Input:
    $\Theta_1$:Parent1
    $\Theta_2$:Parent2
Output:
    $\Delta_1$:Child1
    $\Delta_2$:Child2
1 len ← length($\Theta$)
2 CP ← randInteger(1,len)
3 $\Delta_1 \leftarrow \Theta_1$
4 $\Delta_2 \leftarrow \Theta_2$
5 for i in range(1 to CP )                // crossover
6  do
7  |  $\Delta_1[i], \Delta_2[i] \leftarrow \Theta_2[i], \Theta_1[i]$
8 end
9 return $\Delta_1, \Delta_2$
```

During mutation, two different random mutation points, namely, the lower mutation point LMP and the upper mutation point UMP, are selected. The LMP will be generally from the top half of the chromosome and UMP will be generally from the lower half of the chromosome. The values between LMP and UMP are substituted with their corresponding 1's complement values. The mutation operation is illustrated in Fig. 5. Fitness of the solution is calculated by equation (6):

$$fitnessScore = W_a \times M_a + W_q(1 - FSR), \tag{6}$$

where W_a and W_q represent the weights of classification accuracy and feature selection ratio respectively. M_a represents the classification model accuracy which is calculated using equation (7):

$$M_a = \frac{\textit{number of correctly classified images}}{sizeOf(\textit{Testing set})}. \tag{7}$$

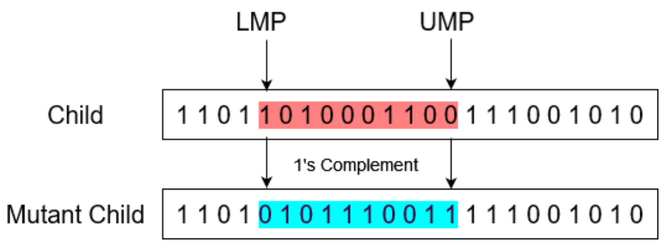

Fig. 5. Mutation of a chromosome

FSR represents feature selection ratio, which is calculated using equation (8):

$$FSR = \frac{F_{ls}}{F_n}. \tag{8}$$

F_{ls} and F_n represents length of the subset of features selected and total number of features in dataset respectively. The algorithm for mutation operation is given in Algorithm 3.

Algorithm 3. Mutate

```
Input: $\Delta$:Chromosome
Output: $\acute{\Delta}$:Mutant Child
1  $\acute{\Delta} \leftarrow \Delta$
2  len ← length($\Delta$)
3  LMP ← randInteger(1,len)
4  UMP ← randInteger(1,len)
5  if $LMP > UMP$ then
6      swap(LMP,UMP)
7  end
8  for i in range(LMP to UMP) do
9      $\acute{\Delta}$[i] ← 1's Complement of $\Delta[i]$
10 end
11 return $\acute{\Delta}$
```

Algorithm 4. ComputeFitness

```
Input:
    X:Solution
    C:Classification algorithm
    $\Gamma_{IFM}$:Image Feature Matrix
Output: fitnessScore
1  let $W_a \leftarrow 0.8$
2  let $W_q \leftarrow 0.2$
3  index ← getSelectedFeatures(X) // extract selected features' index
4  $\Gamma_{IFM} \leftarrow \Gamma_{IFM}[:, index]$ // extracting the selected features
5  $X_{train}, X_{test}, Y_{train}, Y_{test}$ ← train_test_split($\Gamma_{IFM}$, test_size=0.3)
6  model ← constructModel($X_{train}, Y_{train}$, C)
7  $M_a$ ← calculate accuracy(model,$X_{test}, Y_{test}$) // using equation (7)
8  $FSR$ ← calculate Feature Selection Ratio // using equation (8)
9  $fitnessScore \leftarrow W_a \times M_a + W_q(1 - FSR)$
10 return fitnessScore
```

3.5. Approximate nearest neighbor

The similarity measure is calculated for all the images in the database against the query image by using the selected features from the previous stages. This strategy, known as the linear search method, will always identify the precise nearest neighbor. This could be expensive when the dataset is larger. To address this, approximate nearest neighbor algorithms can be used. In our work, we have used a graph-based approach to perform ANN. A k-NN graph is constructed offline, and we do a greedy search on the graph to retrieve the nearest node for the given query.

A k-NN graph is a digraph $G = (N, E)$, where N is the set of nodes and E is the edge. Node N_i is connected to N_j if N_j is one of the k-NNs of N_i. If the value of k is smaller, then the graph will become sparse. On the other hand, choosing a larger k becomes costly. The approximate k-nearest neighbor search is performed on the k-NN graph from a randomly selected node N_t. N_t gets updated every time when a new nearest neighbor to the query is found. This is done using equation (9):

$$N_t = \underset{N \in C(N_{t-1}, n, G)}{\arg\min} \; d(N, q), \tag{9}$$

where $C(N, n, G)$ returns first $n \leq k$ neighbors of N from G. The distance measure d is the Euclidean distance. After a predetermined number of greedy steps, S, the algorithm stops. At this point, the most recent top K nodes are returned, and these K nodes are the K-nearest neighbors to the query.

3.6. Time complexity

The time complexity for initializing the population is $O(P \times N)$ where P represents the population size and N represents the total number of objectives. In order to evaluate the fitness, the complexity is $O(P \times \text{MaxIt} \times c)$, where MaxIt is the maximum number of iterations and c is the cost for evaluating the objective function. In order to update the position of cheetahs the cost is $O(P \times \text{MaxIt} \times u)$ where u is the cost for updating a cheetah's position. After incorporating mutation and crossover operations, the complexity of entire process is $O(N \times \text{MaxIt} \times P \times c \times u \times (M + C))$, where M and C indicate the cost of mutation and crossover operations, respectively which is very much similar to other methods like PSO, FA and GWO. For constructing k-NN graph, the time complexity is $O(n \times \log(n))$ where n is the number of nodes. The time complexity for finding the k nearest neighbors is $O(\sqrt{n} \times k)$.

3.7. Space complexity

The space complexity for HCO is $O(P \times N)$ where P represents the population size and N represents the total number of objectives. The space complexity for k-NN graph approach is $O(n)$ where n is the number of nodes.

4. IMPLEMENTING APPROACHES

The details of benchmark datasets, classifiers, and performance evaluation methodologies used in the paper are briefly discussed here.

4.1. Datasets used

Utilizing five publicly available benchmark datasets (LFW [32], MultiPie [33], color FERET [34], DigiFace-1M [35] and CelebA [36, 37]), the proposed face recognition and retrieval system performance is evaluated.

4.2. Supervised learning algorithms used

In this proposed study, we used the two most well-known supervised classification methods. To build the model, a 10-fold cross-validation approach is used.

1. Naive Bayes (NB) classifier: The NB classifier is a statistical classifier that is capable of predicting the probability of class membership for given data [38, 39].
2. Support vector machine (SVM) classifier: The SVM classifier [40] creates the best decision boundary, called a hyperplane, so that the data points can be divided into two classes.

The border of the hyperplane is defined by the support vectors.

4.3. Performance evaluation criteria

The performance of proposed HCO algorithm is evaluated using following evaluation criteria: accuracy (success rate), precision (μP), recall (R), and F-Score (F_1). These values are determined by the factors sorted out below.

1. True Positive(T_{pos}): Positive classes are classified as positive.
2. True Negative(T_{neg}): Negative classes are classified as negative.
3. False Positive(F_{pos}): Negative classes are classified as positive.
4. False Negative(F_{neg}): Positive classes are classified as negative.

The success rate is calculated by equation (10):

$$SR = \frac{T_{\text{pos}} + T_{\text{neg}}}{P + N}, \tag{10}$$

where P and N are the total number of positive and negative classes. The Precision (μP) is the percentage (%) of positive data labeled as positive is calculated by equation (11):

$$\mu P = \frac{T_{\text{pos}}}{T_{\text{pos}} + F_{\text{pos}}}. \tag{11}$$

The Recall (R) is the percentage of identified true positives which is calculated by equation (12):

$$RC = \frac{T_{\text{pos}}}{T_{\text{pos}} + F_{\text{neg}}}. \tag{12}$$

The F_1-Score(F_1) is a metric that evaluates the classification model efficacy by considering both precision (μP) and recall (R) is calculated by equation (13):

$$F_1 = \frac{2 \times \mu P \times RC}{\mu P + RC} = \frac{T_{\text{pos}}}{T_{\text{pos}} + \frac{1}{2}(F_{\text{pos}} + F_{\text{neg}})}. \tag{13}$$

To evaluate the retrieval performance of the proposed HCO feature selection algorithm combined with the k-NN algorithm, the result similarity value is calculated using equation (14). Result similarity is the average cosine similarity [41] between the query image and the retrieved top k images:

$$Result\ Similarity = \frac{1}{k} \times \sum_{j=0}^{k} S_c(q, Y_j), \tag{14}$$

where q is the query image, $Y_j \in Y$, which is the set of retrieved images and k is the size of Y. Cosine similarity $S_c(q,Y)$ is calculated using equation (15):

$$S_c(q,Y) = \frac{\sum_{i=0}^{n} q_i Y_i}{\sqrt{\sum_{i=0}^{n} q_i^2}\sqrt{\sum_{i=0}^{n} Y_i^2}}. \tag{15}$$

4.4. Environmental setup

The system is implemented on 2.3 GHz – Intel Core i5 system with 16 GB RAM memory. Python 3.7.3 is used for programming. TensorFlow library is employed for feature extraction along with matplotlib library for plotting the graphs.

4.5. Experimental results

The proposed HCO algorithm performance is compared with various feature selection techniques listed below:

1. Genetic algorithm (GA) [42].
2. Particle swarm optimization algorithm (PSO) [43].
3. Firefly algorithm (FA) [44].
4. Grey wolf algorithm (GWO) [22].

4.5.1. Comparison of accuracy and performance on LFW dataset

Table 1 shows the comparison of results from various aforementioned feature selection techniques on the LFW dataset using the NB and SVM classifiers. The proposed HCO feature subset selection methodology has shown 94.24% accuracy with the NB classifier. Similarly, the precision, recall, and F1 score are 0.94 each. With the SVM classifier, the accuracy is also 94.24%. The proposed method has shown 0.940 precision, 0.941 recall, and a 0.941 F1 score. This clearly indicates that HCO-based method shows better results comparing to alternate feature selection methodologies. The classification accuracy distribution of various methods along with the proposed HCO-based feature selection method in 20 runs on the dataset is depicted in Fig. 6

Table 1
Performance comparison on LFW dataset

Methods	NB					SVM				
	AC	ERR	P	RC	F1	AC	ERR	P	RC	F1
GA	84.26%	15.74%	0.8391	0.842	0.8397	84.26%	15.74%	0.8391	0.842	0.8399
PSO	85.2%	14.8%	0.8489	0.8513	0.8494	85.2%	14.8%	0.8493	0.8513	0.8497
FA	89.68%	10.32%	0.8938	0.8962	0.8945	89.68%	10.32%	0.8949	0.8962	0.8952
GWO	91.65%	8.35%	0.9143	0.9159	0.9148	91.65%	8.35%	0.9145	0.9159	0.9149
HCO	94.24%	5.76%	0.9406	0.9417	0.9409	94.24%	5.76%	0.9409	0.9417	0.9411

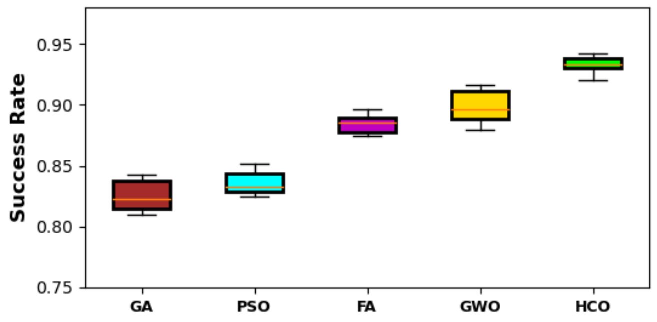

Fig. 6. Success rate using NB classifier on LFW dataset

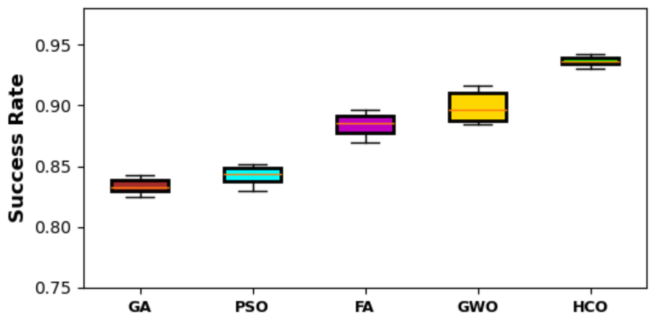

Fig. 7. Success rate using SVM classifier on LFW dataset

Table 2
Multi-run performance on LFW dataset

Methods	NB				SVM			
	Max	Min	Avg	SD	Max	Min	Avg	SD
GA	0.8426	0.8096	0.8266	0.0132	0.8426	0.8246	0.8353	0.0068
PSO	0.852	0.8247	0.8368	0.0103	0.852	0.8297	0.8438	0.0078
FA	0.8968	0.8745	0.8854	0.0082	0.8968	0.8696	0.8857	0.0104
GWO	0.9165	0.8795	0.8978	0.0134	0.9165	0.8845	0.8998	0.013
HCO	0.9424	0.9296	0.9355	0.0048	0.9424	0.9306	0.9385	0.0029

and 7. The variation and outliers are very low in the proposed method, and it is clearly evident that HCO is consistent and efficient when compared with the other methods. The repeatability test results are shown in Table 2 with values such as the best, worst, average, and standard deviation of the accuracy score for the feature selection methods using NB and SVM classifiers. The mean accuracy score for the HCO method is 93.55% and 93.85%, respectively, which is better than other methods.

4.5.2. Comparison of accuracy and performance on MultiPie dataset

The comparison results of various feature selection methods on MultiPie dataset are tabulated in Table 3. The HCO shows 93.96% accuracy, 0.9358 precision, 0.9396 recall, and 0.9376 F1 score with the NB classifier and 93.96% accuracy, 0.9357 precision, 0.9396 recall, and 0.9376 F1 score with the SVM classifier. With MultiPie dataset, our HCO feature selection method clearly outperforms the other feature selection methods. After 20 independent runs on the MultiPie dataset with NB and SVM classifiers using various feature selection methods, the repeatability test results are depicted in Table 4, and the classification accuracy distribution is depicted as boxplots in Fig. 8 and 9.

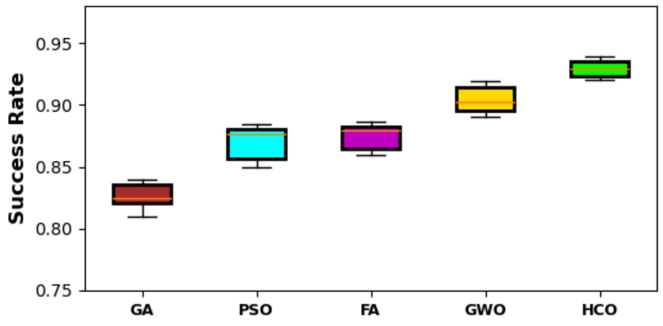

Fig. 8. Success rate using NB classifier on MultiPie dataset

Table 3
Performance comparison on MultiPie dataset

Methods	NB					SVM				
	AC	ERR	P	RC	F1	AC	ERR	P	RC	F1
GA	83.96%	16.04%	0.8317	0.8396	0.8352	84.96%	15.04%	0.8418	0.8496	0.8452
PSO	88.46%	11.54%	0.8782	0.8846	0.8811	88.46%	11.54%	0.8781	0.8846	0.881
FA	88.96%	11.04%	0.8835	0.8896	0.8863	88.96%	11.04%	0.8834	0.8896	0.8862
GWO	91.96%	8.04%	0.9148	0.9196	0.9171	91.96%	8.04%	0.9149	0.9196	0.9171
HCO	93.96%	6.04%	0.9358	0.9396	0.9376	93.96%	6.04%	0.9357	0.9396	0.9376

Table 4
Multi-run performance on MultiPie dataset

Methods	NB				SVM			
	Max	Min	Avg	SD	Max	Min	Avg	SD
GA	0.8396	0.81	0.827	0.0111	0.8496	0.815	0.8316	0.0133
PSO	0.8846	0.85	0.8704	0.0132	0.8846	0.855	0.8715	0.0116
FA	0.8896	0.86	0.876	0.0113	0.8896	0.855	0.8736	0.0136
GWO	0.9196	0.89	0.9057	0.0117	0.9196	0.87	0.8978	0.0201
HCO	0.9396	0.92	0.9306	0.0076	0.9396	0.92	0.9295	0.0076

The proposed HCO feature selection method, with average accuracy of 93.06% and 92.95% using NB and SVM classifiers, is the best and most consistent among the listed methods.

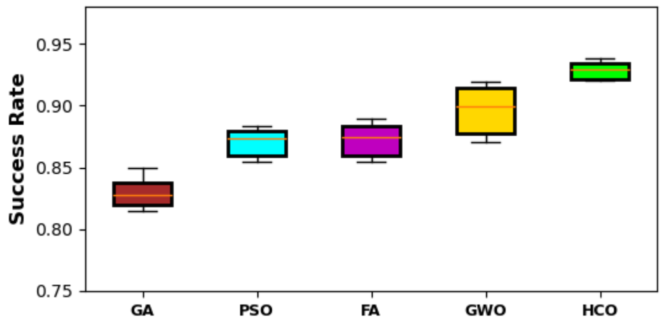

Fig. 9. Success rate using SVM classifier on MultiPie dataset

4.5.3. Comparison of accuracy and performance on color FERET dataset

When the proposed method is applied using the NB classifier, an accuracy of 93.72%, precision of 0.9369, a recall of 0.9371, and an F1 score of 0.9368 are obtained. With SVM classifiers, the values 94.23%, 0.9408, 0.9417, and 0.9410 are obtained as accuracy, precision, recall, and F1 score. The entire comparison results are displayed in Table 5. The multi-run statistics are represented in Table 6, after performing 20 independent runs on the dataset for each method using NB and SVM classifiers. With average accuracy of 93.05% and 93.38% using NB and SVM classifiers, the HCO-based method has shown superior performance. Also, the classification accuracy distribution is consistent with both classifiers for the proposed methods. The classification accuracy distribution is shown in Fig. 10 and 11.

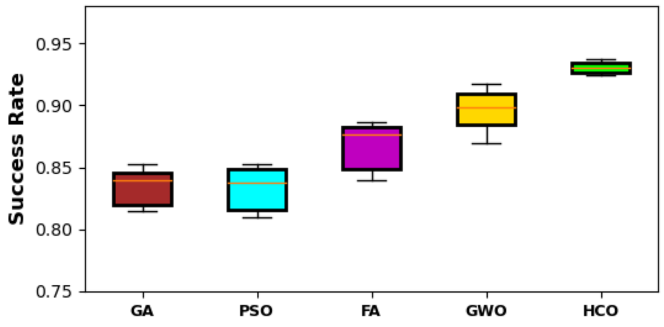

Fig. 10. Success rate using NB Classifier on color FERET Dataset

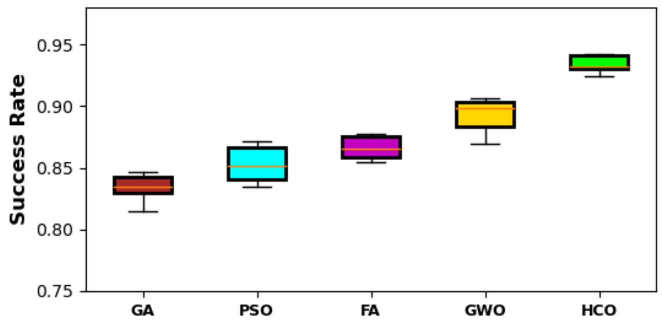

Fig. 11. Success rate using SVM Classifier on color FERET Dataset

Table 5
Performance comparison on color FERET dataset

Methods	NB					SVM				
	AC	ERR	P	RC	F1	AC	ERR	P	RC	F1
GA	85.24%	14.76%	0.8515	0.852	0.8513	84.62%	15.38%	0.843	0.8457	0.8437
PSO	85.24%	14.76%	0.8509	0.852	0.851	87.17%	12.83%	0.8695	0.8711	0.8698
FA	88.66%	11.34%	0.8859	0.8863	0.8857	87.71%	12.29%	0.8744	0.8766	0.875
GWO	91.75%	8.25%	0.9174	0.9173	0.9171	90.66%	9.34%	0.9044	0.9062	0.9049
HCO	93.72%	6.28%	0.9369	0.9371	0.9368	94.23%	5.77%	0.9408	0.9417	0.941

Table 6
Multi-run performance on color FERET dataset

Methods	NB				SVM			
	Max	Min	Avg	SD	Max	Min	Avg	SD
GA	0.8524	0.8145	0.8362	0.015	0.8462	0.8146	0.8339	0.0103
PSO	0.8524	0.8096	0.8339	0.0168	0.8717	0.8345	0.8532	0.0139
FA	0.8866	0.8395	0.8678	0.0192	0.8771	0.8546	0.8664	0.0085
GWO	0.9175	0.8695	0.8968	0.0176	0.9066	0.8695	0.8924	0.0128
HCO	0.9372	0.9245	0.9305	0.0045	0.9423	0.9246	0.9338	0.0062

4.5.4. Comparison of accuracy and performance on DigiFace-1M dataset

An accuracy of 93.06%, precision of 0.9314, a recall of 0.9306, and an F1 score of 0.9308 are obtained when the proposed method is applied using the NB classifier. With SVM classifiers, accuracy of 94.44%, precision of 0.9452, a recall of 0.9444, and an F1 score of 0.9446 are obtained. The complete comparison results are displayed in Table 7. The multi-run statistics are provided in Table 8, after performing 20 independent runs on the dataset for each method using NB and SVM classifiers. The performance of HCO-based method is superior with average accuracy of 92.61% and 93.53% using NB and SVM classifiers. Also, it has consistent distribution of the classification accuracy with both classifiers for the proposed methods. The classification accuracy distribution is shown in Fig. 12 and 13.

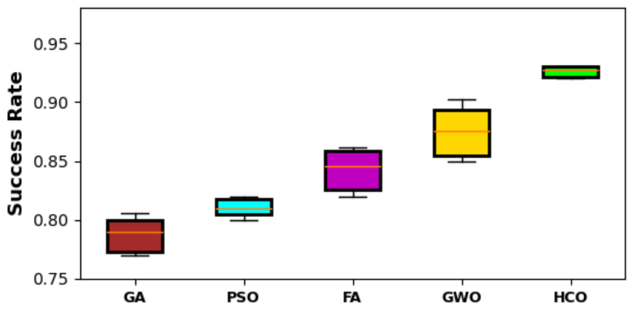

Fig. 12. Success rate using NB Classifier on DigiFace-1M Dataset

Table 7
Performance comparison on DigiFace-1M dataset

Methods	NB					SVM				
	AC	ERR	P	RC	F1	AC	ERR	P	RC	F1
GA	80.56%	19.44%	0.8077	0.8056	0.8061	80.56%	19.44%	0.8078	0.8056	0.8061
PSO	81.94%	18.06%	0.8215	0.8194	0.82	81.94%	18.06%	0.8215	0.8194	0.82
FA	86.11%	13.89%	0.8628	0.8611	0.8615	81.94%	18.06%	0.8215	0.8194	0.82
GWO	90.28%	9.72%	0.904	0.9028	0.9031	90.28%	9.72%	0.904	0.9028	0.9031
HCO	93.06%	6.94%	0.9314	0.9306	0.9308	94.44%	5.56%	0.9452	0.9444	0.9446

Table 8
Multi-run Performance on DigiFace-1M dataset

Methods	NB				SVM			
	Max	Min	Avg	SD	Max	Min	Avg	SD
GA	0.8056	0.77	0.7876	0.014	0.8056	0.77	0.7908	0.0146
PSO	0.8194	0.8	0.8105	0.0072	0.8194	0.8	0.8113	0.0073
FA	0.8611	0.82	0.8427	0.0162	0.8194	0.81	0.8153	0.0035
GWO	0.9028	0.85	0.8746	0.0205	0.9028	0.85	0.8803	0.0197
HCO	0.9306	0.92	0.9261	0.0041	0.9444	0.925	0.9353	0.0081

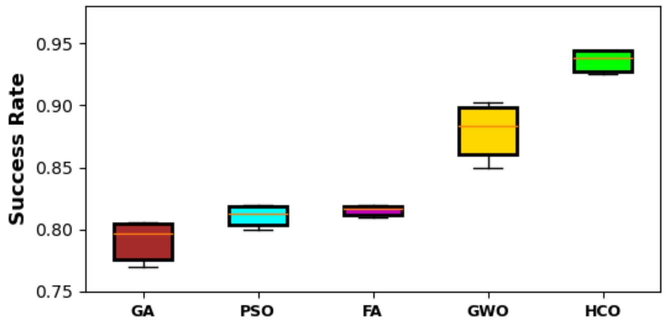

Fig. 13. Success rate using SVM Classifier on DigiFace-1M Dataset

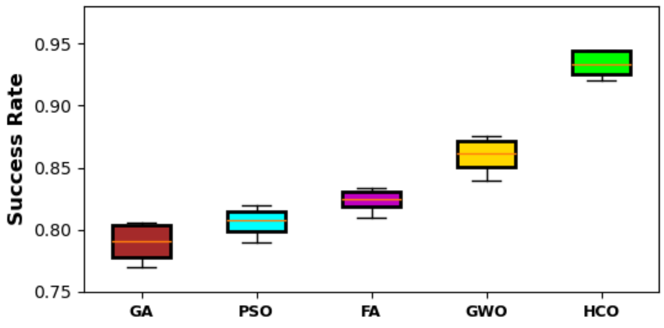

Fig. 14. Success rate using NB classifier on CelebA dataset

4.5.5. Comparison of accuracy and performance on CelebA dataset

The HCO-based method outperforms other methods with an accuracy of 94.44%, precision of 0.9451, a recall of 0.9444, and an F1 score of 0.9446 using the NB classifier. An accuracy of 94.44%, precision of 0.9452, a recall of 0.9444, and an F1 score of 0.9446 are obtained with SVM classifiers. The complete comparison results are displayed in Table 9. After performing 20 independent runs on the dataset for each method using NB and SVM classifiers, the statistics are provided in Table 10. HCO has better performance with average accuracy of 93.35% and 94.07% using NB and SVM classifiers provided, consistent distribution of the classification accuracy with both classifiers. The same is shown in Fig. 14 and 15.

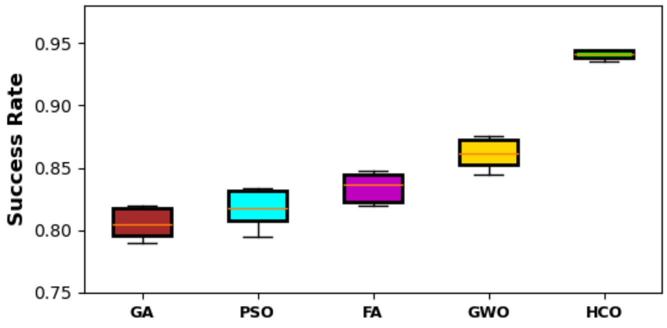

Fig. 15. Success rate using SVM classifier on CelebA dataset

Table 9
Performance comparison on CelebA dataset

Methods	NB					SVM				
	AC	ERR	P	RC	F1	AC	ERR	P	RC	F1
GA	80.56%	19.44%	0.8078	0.8056	0.8061	81.94%	18.06%	0.8215	0.8194	0.82
PSO	81.94%	18.06%	0.8215	0.8194	0.82	83.33%	16.67%	0.8352	0.8333	0.8338
FA	83.33%	16.67%	0.8353	0.8333	0.8338	84.72%	15.28%	0.849	0.8472	0.8477
GWO	87.5%	12.5%	0.8765	0.875	0.8754	87.5%	12.5%	0.8765	0.875	0.8754
HCO	94.44%	5.56%	0.9451	0.9444	0.9446	94.44%	5.56%	0.9452	0.9444	0.9446

Table 10
Multi-run performance on CelebA dataset

Methods	NB				SVM			
	Max	Min	Avg	SD	Max	Min	Avg	SD
GA	0.8056	0.77	0.7902	0.0136	0.8194	0.79	0.8054	0.0111
PSO	0.8194	0.79	0.8063	0.0106	0.8333	0.795	0.8165	0.0142
FA	0.8333	0.81	0.8238	0.008	0.8472	0.82	0.8343	0.0109
GWO	0.875	0.84	0.8599	0.0123	0.875	0.845	0.8612	0.0108
HCO	0.9444	0.92	0.9335	0.0093	0.9444	0.935	0.9407	0.0035

4.5.6. Comparison of retrieval performance

The retrieval performance is evaluated by comparing the Result Similarity Score for the proposed methods of feature selection and image retrieval. When the k-NN search algorithm is combined with the proposed HCO algorithm, the Result Similarity Score obtained is 0.963, 0.957, 0.971, 0.950 and 0.960 for LFW, MultiPie, Color FERET, DigiFace-1M and CelebA datasets, respectively. This shows that while using k-NN search algorithm for retrieving top k similar images, the results are better. The comparison results are illustrated in Fig. 16.

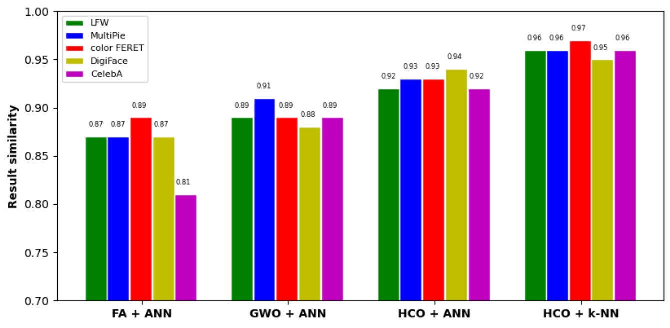

Fig. 16. Comparison of result similarity

5. CONCLUSIONS

This paper proposes a novel facial image retrieval system to retrieve human faces for a query image from a large database. The system employs DCNN for feature extraction, HCO for feature selection, and the k-NN algorithm for image retrieval. The performance of the proposed system is compared with four feature selection techniques on five benchmark datasets. The extensive result analysis demonstrates that the suggested method is exceedingly superior to other methods.

Table 11
List of Symbols

Symbol	Description
$\rho_{i,j}^{t}$	Position of cheetah i in organization j at time t
$\hat{\gamma}$	Random number
δ	Step length
ρ_B	Position of the prey
$\check{\gamma}$	Turning factor
$\acute{\delta}$	Interaction factor
POP_{size}	Population Size
D	Image Dataset
C	Classification Algorithm
$MaxIt$	Maximum iteration count
$T_{\max}$	Maximum of time t
Γ_{IFM}	Extracted Image features
X_i	i^{th} Cheetah (Solution)
$prey$	Global best solution
$leader$	Global second best solution
$X_i^{fitness}$	Fitness of i^{th} Cheetah
Ψ	Selected m Cheetahs (Solutions)
Θ	Parent chromosome
Δ	Child chromosome
$\acute{\Delta}$	Mutant Child chromosome
CP	Crossover point
LMP	Lower Mutation Point
UMP	Upper Mutation Point
G	k-NN graph
N	A node in k-NN graph
W_a	Weight for classification accuracy
W_q	Weight for feature selection ratio
M_a	Classification accuracy
FSR	Feature selection ratio
F_{ls}	Selected feature subset length
F_n	Total Feature length
X_t	Node in the k-NN graph at time t
T_{pos}	True Positive
T_{neg}	True Negative
F_{pos}	False Positive
F_{neg}	False Negative
SR	Success rate
μP	Micro precision
RC	Recall
F_1	$F_1 - score$
S_c	Cosine similarity

ACKNOWLEDGEMENTS

The authors are grateful to the Principal and the management of Mepco Schlenk Engineering College Sivakasi, for providing space and infrastructure for conducting this work.

REFERENCES

[1] R. Kapoor, D. Sharma, and T. Gulati, "State of the art content based image retrieval techniques using deep learning: a survey," *Multimed. Tools Appl.*, vol. 80, no. 19, pp. 29 561–29 583, 2021, doi: 10.1007/s11042-021-11045-1.

[2] J. Suresh Kumar and M. C. Vigila S., "A review on content based image retrieval techniques," in *2023 International Conference on Circuit Power and Computing Technologies (ICCPCT)*, 2023, pp. 1251–1256, doi: 10.1109/ICCPCT58313.2023.10245360.

[3] S. Singh and S. Prasad, "Techniques and challenges of face recognition: A critical review," *Procedia Comput. Sci.*, vol. 143, pp. 536–543, 2018, doi: 10.1016/j.procs.2018.10.427.

[4] B. Li, "The current situation and potential development of face recognition," *Appl. Comput. Eng.*, vol. 4, pp. 308–316, 2023, doi: 10.54254/2755-2721/4/20230478.

[5] R. Szmurło and S. Osowski, "Ensemble of classifiers based on cnn for increasing generalization ability in face image recognition," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 70, no. 3, p. e141004, 2022, doi: 10.24425/bpasts.2022.141004.

[6] J. Wang and D. Chen, "A few-shot fine-grained image recognition method," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 71, no. 1, p. e144584, 2023, doi: 10.24425/bpasts.2023.144584.

[7] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[8] M.A. Akbari, M. Zare, R. Azizipanah-abarghooee, S. Mirjalili, and M. Deriche, "The cheetah optimizer: a nature-inspired metaheuristic algorithm for large-scale optimization problems," *Sci. Rep.*, vol. 12, no. 1, Jun 2022, doi: 10.1038/s41598-022-14338-z.

[9] J. Naruniec, "A survey on facial features detection," *Int. J. Electron. Telecommun.*, vol. 56, no. 3, pp. 267–272, 2010, doi: 10.2478/v10177-010-0035-y.

[10] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997, doi: 10.1109/34.598228.

[11] V. Purandare and K.T. Talele, "Efficient heterogeneous face recognition using scale invariant feature transform," in *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*. IEEE, 2014, doi: 10.1109/cscita.2014.6839277.

[12] S. Gupta, K. Thakur, and M. Kumar, "2d-human face recognition using SIFT and SURF descriptors of face's feature regions," *Vis. Comput.*, vol. 37, no. 3, pp. 447–456, 2020, doi: 10.1007/s00371-020-01814-8.

[13] J. Bobulski, "Multimodal face recognition method with two-dimensional hidden markov model," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 65, no. 1, pp. 121–128, 2017, doi: 10.1515/bpasts-2017-0015.

[14] R. Hecht-Nielsen, "Theory of the backpropagation neural network," vol. 1, 1989, pp. 593–605, doi: 10.1109/IJCNN.1989.118638.

[15] H.B. Fredj, S. Bouguezzi, and C. Souani, "Face recognition in unconstrained environment with CNN," *Vis. Comput.*, vol. 37, no. 2, pp. 217–226, 2020, doi: 10.1007/s00371-020-01794-9.

[16] S. Khan, M.H. Javed, E. Ahmed, S.A.A. Shah, and S.U. Ali, "Facial recognition using convolutional neural networks and implementation on smart glasses," in *2019 International Conference on Information Science and Communication Technology (ICISCT)*. IEEE, 2019, doi: 10.1109/cisct.2019.8777442.

[17] A. Elmahmudi and H. Ugail, "Deep face recognition using imperfect facial data," *Futur. Gener. Comput. Syst.*, vol. 99, pp. 213–225, 2019, doi: 10.1016/j.future.2019.04.025.

[18] R. Pati, A.K. Pujari, and P. Gahan, "Face recognition using particle swarm optimization based block ICA," *Multimed. Tools Appl.*, vol. 80, no. 28-29, pp. 35 685–35 695, Apr. 2021, doi: 10.1007/s11042-021-10792-5.

[19] H. Zhi and S. Liu, "Face recognition based on genetic algorithm," *J. Vis. Commun. Image Represent.*, vol. 58, pp. 495–502, Jan. 2019, doi: 10.1016/j.jvcir.2018.12.012.

[20] P. Annamalai, "Automatic face recognition using enhanced firefly optimization algorithm and deep belief network," *Int. J. Intell. Eng. Syst*, vol. 13, no. 5, pp. 19–28, Oct 2020, doi: 10.22266/ijies2020.1031.03.

[21] T. Kumar, S. Bhushan, and S. Jangra, "An improved biometric fusion system of fingerprint and face using whale optimization," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 1, 2021, doi: 10.14569/ijacsa.2021.0120176.

[22] H. Sikkandar and R. Thiyagarajan, "Soft biometrics-based face image retrieval using improved grey wolf optimisation," *IET Image Process.*, vol. 14, no. 3, pp. 451–461, 2020, doi: 10.1049/iet-ipr.2019.0271.

[23] M. Tubishat, N. Idris, L. Shuib, M.A. Abushariah, and S. Mirjalili, "Improved salp swarm algorithm based on opposition based learning and novel local search algorithm for feature selection," *Expert Syst. Appl.*, vol. 145, p. 113122, 2020, doi: 10.1016/j.eswa.2019.113122.

[24] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A.H. Gandomi, "Marine predators algorithm: A nature-inspired metaheuristic," *Expert Syst. Appl.*, vol. 152, p. 113377, 2020, doi: 10.1016/j.eswa.2020.113377.

[25] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[26] M. Tubishat, M. Alswaitti, S. Mirjalili, M.A. Al-Garadi, M.T. Alrashdan, and T.A. Rana, "Dynamic butterfly optimization algorithm for feature selection," *IEEE Access*, vol. 8, pp. 194 303–194 314, 2020, doi: 10.1109/access.2020.3033757.

[27] M. Malkauthekar, "Analysis of euclidean distance and manhattan distance measure in face recognition," in *Third International Conference on Computational Intelligence and Information Technology (CIIT 2013)*. IET, 2013, pp. 503–507, doi: 10.1049/cp.2013.2636.

[28] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *J. ACM*, vol. 45, no. 6, pp. 891–923, 1998, doi: 10.1145/293347.293348.

[29] W. Li *et al.*, "Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1475–1488, 2020, doi: 10.1109/tkde.2019.2909204.

[30] X. Xu, C. Li, Y. Wang, and Y. Xia, "Multiattribute approximate nearest neighbor search based on navigable small world graph," *Concurr. Comput. Pract. Exp.*, vol. 32, no. 24, 2020, doi: 10.1002/cpe.5970.

[31] C. Fu, C. Xiang, C. Wang, and D. Cai, "Fast approximate nearest neighbor search with the navigating spreading-out graph," *arXiv:1707.00143*, 2017, doi: 10.48550/ARXIV.1707.00143.

[32] LFW Face Database : Main. Labeled-faces-in-the-wild-home. [Online]. Available: http://vis-www.cs.umass.edu/lfw/ (Accessed June-2022).

[33] The CMU Multi-PIE Face Database. The-multipie-dataset. [Online]. Available: https://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html (Accessed March-2022).

[34] The color FERET Database, "Colorferet-dataset." [Online]. Available: https://www.nist.gov/itl/products-and-services/color-feret-database (Accessed June-2022).

[35] G. Bae *et al.*, "Digiface-1m: 1 million digital face images for face recognition," in *2023 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2023, doi: 10.48550/arXiv.2210.02579.

[36] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," *CoRR-arXiv abs/1411.7766*, 2014, doi: 10.48550/arXiv.1411.7766.

[37] Y. Jiang, Z. Huang, X. Pan, C.C. Loy, and Z. Liu, “Talk-to-edit: Fine-grained facial editing via dialog,” *CoRR-arXiv abs/2109.04425*, 2021, doi: 10.48550/arXiv.2109.04425.

[38] I. Rish, “An empirical study of the naive bayes classifier,” in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM New York, 2001, pp. 41–46.

[39] R.F. Rahman and Suharjito, “Crowd face detection with naive bayes in attendance system using raspberry pi,” *E3S Web Conf.*, vol. 388, p. 02010, 2023, doi: 10.1051/e3sconf/202338802010.

[40] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, 1998, doi: 10.1109/5254.708428.

[41] H. V. Nguyen and L. Bai, “Cosine similarity metric learning for face verification,” in *Computer Vision – ACCV 2010*. Springer Berlin Heidelberg, 2011, pp. 709–720, doi: 10.1007/978-3-642-19309-5_55.

[42] H. Zhi and S. Liu, “Face recognition based on genetic algorithm,” *J. Vis. Commun. Image Represent.*, vol. 58, pp. 495–502, 2019, doi: 10.1016/j.jvcir.2018.12.012.

[43] Y. Zhang and L. Yan, “Face recognition algorithm based on particle swarm optimization and image feature compensation,” *SoftwareX*, vol. 22, p. 101305, 2023, doi: 10.1016/j.softx.2023.101305.

[44] W. Xie, L. Wang, K. Yu, T. Shi, and W. Li, “Improved multi-layer binary firefly algorithm for optimizing feature selection and classification of microarray data,” *Biomed. Signal Process. Control.*, vol. 79, p. 104080, 2023, doi: 10.1016/j.bspc.2022.104080.