Journal of Applied Engineering and Technological Science
Vol 7(1) 2025: 647-666

HAND GESTURE RECOGNITION USING OPTIMIZED HYPERPARAMETERS OF CNN FOR REAL-TIME CONTROL OF A MULTI-SERVO HAND

Noor M. Naser[1*], Kian R. Qasim[2], Zinah J. Jabbar[3]
University of Information Technology and Communications, Baghdad, Iraq[123]
noor.sarsam@uoitc.edu.iq, kian@uoitc.edu.iq, zena.jamal@uoitc.edu.iq

Received: 17 July 2024, Revised: 25 October 2025, Accepted: 13 November 2025
**Corresponding Author*

ABSTRACT
Gesture recognition has emerged as a promising approach to enhance human-machine interaction, especially in robotics and assistive devices. This study presents a real-time gesture-controlled robotic system that combines deep learning and machine learning techniques to recognize hand gestures and map them to servo motor movements. A convolutional neural network (CNN) was used to classify six predefined hand gestures: a closed fist and five individual finger extensions. To enhance classification accuracy and generalization, CNNs are tuned using hyperparameter tuning techniques, such as Particle Swarm Optimization (PSO), Grey Wolf Optimization (GWO), and Brown Bear Optimization (BBO). These methods efficiently explore the hyperparameter space—such as learning rate, filter size, and batch size—reducing manual trial and error in control. Among these proposed models tested, the BBO-CNN has been achieved the highest performance with a classification accuracy of 99.98%, outperforming both PSO-CNN (99.89%) and GWO-CNN (99.44%). The model CNN without optimization achieved an accuracy of 97.50%. The combination of advanced deep learning models and embedded control demonstrates the feasibility and effectiveness of gesture-based robotics applications.
Keywords: *Hand gesture, Hybrid CNN, PSO, GWO, BBO.*

1. Introduction

Hand gesture control systems are quickly developing as a serious component in the evolution of natural human-computer interaction (HCI) (Patil et al., 2024). By translating complex human movements into digital commands, also, hand gesture control offers a contactless and often important interface across fields. The real-world implication of this technology is transforming fields from progressive robotics (Wang et al., 2024) and applications for disabilities (Sun et al., 2024) such as prosthetics and sign language translation applied to computing environments. The main purpose of hand gesture control is to minimize the gap between human determined and machine implementation, to control the devices, robots, or assistive mechanisms is as normal and direct as communication between individuals (Qi et al., 2024).

The state of the art in hand gesture is dramatically in recent years due to advances in machine learning. Specifically, Convolutional Neural Networks (CNNs) which have developed the algorithmic method (Shin et al., 2024; Al Kafaf et al., 2024). Also, CNNs are applied for automatically extracting features from images. This process done in computer vision tasks considered as a powerful tool for recognize hand gesture in real-world environment and its robustness based of changes in light, angle, or hand size (Sun et al., 2024).

Developing an effective and reliable CNN for hand gesture control system faces challenge such hyperparameter tuning (Ashraf et al., 2024). Other challenge is building an effective CNN needs manually tuning numerous hyperparameters such as the number of layers, learning rate, filter size, and activation functions. This process needs specialized skills and knowledge. Unsuitable hyperparameters selection can achieve misfit, overfitting, or slow convergence in training model.

This study proposes a real-time motion recognition which design a CNN with metaheuristic optimization algorithms to automate and enhance hyperparameter tuning. We will compare three specialized techniques which are Particle Swarm Optimization (PSO), based on the behavior of birds (Chen et al., 2024; Utama et al., 2022); the Grey Wolf Optimizer (GWO), which demonstrates the strict hunting hierarchy of gray wolves (Jaddoa, 2024; Nasir et al., 2024); and the Brown Bear Optimization (BBO), which represents the hunting and marking actions of brown bears (Mehta et al., 2024; Sayed et al., 2024).

This study combines three key issues to create an effective hand-control system. First, we apply a Convolutional Neural Network (CNN) because it is suitable at learning patterns from images, like recognizing various hand shapes. Second, because setting up hyperparameter of CNN properly is hard and time-consuming, for that reason, we've applied an optimization method (PSO, GWO, and BBO). These methods used to find the best settings for the CNN, much like tuning an instrument for perfect sound. This confirms our model is both precise and fast. Finally, the recognized gesture is directed forwarded to a control system to driving specific servo motor. This creates a direct loop started by the camera capture a hand gesture, the optimized CNN understands it instantly, and the hand reacts depended on predicted action by CNN. This complete process makes a smooth and responsive connection between human purpose and machine action.

The main advantages of this study are to develop a hand gesture recognition system that efficiently controls multiple servo motors via an Arduino microcontroller in real time. The system captures finger movements, interprets them as specific gestures, and converts those actions into servo motor drives. Also, estimate and optimize the meta-parameters of the system using these techniques individually and in tuning hyperparameters of CNN by optimization algorithms (CNN-PSO, CNN-GWO, and CNN-BBO), the main goal is to estimate and optimize the meta-parameters of the system to significantly improve classification accuracy, reduce latency, and confirm reliable real-time performance. The results of each method will be analyzed to determine the optimal optimization strategy for improving gesture detection and control response in this practical application.

2. Literature Review

Gesture recognition has become a fundamental component of human-machine interaction (HMI), especially in areas such as prosthetics, assistive robotics, and IoT-enabled smart control systems. However, the poor generalizability and heavy calibration nature of sEMG-based gesture recognition systems remain significant barriers to their real-world adoption. To address these challenges, recent research has focused on deep learning (TL)-based transfer learning approaches to improve adaptability and reduce the need for customized user training (Rezaee et al., 2024) where they manually determined hyperparameter and computationally intensive and slow. Manual hyperparameter tuning of CNNs limits real-time representation and capability. We achieved this by using advanced metaheuristics (PSO, GWO, BBO) to use an optimize the CNN to reach high accuracy and low-latency real-time gesture control. Traditional models, such as support vector machine (SVM) and multilayer feed-forward neural networks (MLPFFNNs) trained on the American Sign Language (ASL) dataset, have shown promising results in controlled environments, but they lack scalability and continuous gesture recognition capabilities due to limitations in dynamic gesture modeling and static gesture class sets (Sümbül, 2024) where the MLPFFNs requires image dataset and flattering process destroy the vital spatial between adjacent pixels between palm and fingerprint. To solve this gap, apply a CNN's automatic feature extraction and use advanced metaheuristic techniques such (PSO), (GWO), and /or (BBO) to methodically determine the optimal hyperparameters. Vision-based approaches, such as MediaPipe and OpenCV-based approaches on the ESP32 platform for single-gesture detection based on hand angle, show promising results in low-latency and cost-effective systems, (Desai et al., 2025) where gap is limited to a narrow gesture vocabulary and suffer from illumination and occlusion issues and to solve this issue may apply CNN's and set optimal hyperparameters by apply optimization methods. Similarly, object detection frameworks using YOLOv3 have achieved high classification accuracies without preprocessing (up to 97.68%) (Mujahid et al., 2021) but these methods are mostly limited to static gesture classification and are not designed for real-time interactive scenarios. To address this gap, our study uses a CNN and optimize it by PSO, GWO, and BBO to reduce latency, improve accuracy, and enable real-time interactive gesture recognition. A CNN-based deep learning model with LSTM was trained on raw EMG data to classify up to 52 hand movements, with a maximum gesture accuracy of 80%, although the overall average often drops to around 60%, indicating the need for further improvements to confirm reliability in embedded systems, such as the Raspberry Pi. The gap of this work still rely on manual tuning and are too computationally used for real-time embedded system. Also, limited use of transfer learning and has lack of applying optimization methods further adaptability in practical scenarios. To solve these gaps, applying a CNN and enhance it using PSO, GWO, and BBO to design model to increase the efficiency, improve robustness, reduce latency, and enable consistent real-time performance on low-power embedded systems (Fukano et al., 2021). Other studies have explored a hybrid CNN-LSTM model enhanced with convolutional autoencoder and particle swarm optimization (PSO), achieving 87% accuracy on the NinaPro DB1 dataset. However, the lack of multimodal inputs and reliance on EMG signals alone limit its robustness in noisy or unsupervised environments, and to solve this lack, can apply our methodology of optimization such GWO or BBO to achieve higher performance (Sakinala & Abinaya, 2025). Real-time gesture recognition integrated with the Internet of Things for applications such as smart home control has been achieved using CNNs with OpenCV and MediaPipe (Fatima et al., 2024), but the limited set of gestures and lack of adaptability between users limits its wider application and to fill this lack, our work used an optimizers to enhanced the CNN by applied (PSO, GWO, BBO) to achieve generalization across varied users and gesture differences. Glove-based gesture robots incorporating accelerometers have also been explored (Meem & Roy, 2025). Although these systems are mostly limited to experimental settings and lack scalability, systems incorporating depth-feedback haptic sensors on the forearm have shown improved object size recognition by analyzing a finger-specific gesture; however, hardware complexity and limited datasets limit the generalizability of the results (Devecioğlu & Karakulak, 2024), both papers demonstrating an experimental success, and reach scalability, hardware complexity, and limited datasets limit real-world experimental; our work addresses these problems by applying an optimized CNN by (PSO, GWO, BBO) that reduces computational time, processing and enhance real-time gesture recognition.. High-precision image-based gesture recognition using RGB-D data and SSD-CNNs achieves accuracy exceeding 90% (Al Farid et al., 2024), but it still faces challenges in modeling continuous hand motion in real time, although the work achieves over 90% accuracy using RGB-D and SSD-CNNs, their method suffers with real-time continuous hand motion; our work solve this by combine an optimization method for turning a hyperparameters of CNN to enable accurate, low-latency gesture tracking. Similarly, a high-density sEMG system using channel-accumulated pulse-sequence CNNs (cwCST-CNN) achieves an accuracy of up to 96.92% (Yu et al., 2025), but their computational requirements and performance requirements in faulty or realistic environments remain underexplored, and cwCST-CNN model attains high accuracy (96.92%) but is computationally heavy and its performance in real-world, faulty issue is unproven. We use metaheuristics (PSO, GWO, BBO) to optimize a CNN for both high accuracy and computational efficiency, achieves robust performance on low-power devices in realistic environments. The most promising approach is a TL-based model using correlation alignment (CORAL) with CNN, which demonstrated high accuracy (up to 90.19%) on a 10-class dataset of 35 users, with impressive performance across days (84.26%) and poses (87.94%), with simple data calibration (Shi et al., 2024). However, there is still a need for robust, low-cost, dynamically adaptive systems capable of real-time inference for gesture recognition in heterogeneous environments, especially for users with motor disabilities, highlighting opportunities for further exploration in the field of domain adaptation, continuous pose modeling, and multimodal learning. While the CORAL-CNN model demonstrations outstanding user and session revision (up to 90.19% accuracy), it still depended manual hyperparameter tuning and lacks optimization for real-time, for these issue we enhance the transfer learning approach by integrating metaheuristic optimization (PSO (ZR Saeed,et al., 2025), GWO (B Alabduallah, et al., 2025), BBO) to automatically determine optimal CNN hyperparameters, specifically targeting efficient deployment on resource-constrained devices while maintaining robust performance across diverse users and environments.

3. System Architecture

The proposed system architecture consists of a sequence that starts by capturing images of hand gestures, representing the number of fingers from 1 to 5 and a fully closed fist. For each pose class, a dataset of 300 images was collected and this dataset assists as the initial input for the CNN, providing the essential data for the model to learn the complex, spatial of features from low-level edges and textures to high-level finger shapes that relates to each gesture. These images serve as input to a convolutional neural network (CNN) model implemented in MATLAB, which is responsible for the classification task. The work is structured about a core principle that the hierarchical feature learning capability of Convolutional Neural Networks (CNNs), which are theoretically well-suited for spatial pattern recognition in images. To increase the prediction accuracy and convergence efficiency of the CNN, a hybrid optimization using particle swarm optimization (PSO), grey wolf optimizer (GWO), and brown bear optimizer (BBO) are used to tune critical hyperparameters. During real-time testing, MATLAB captures an input image, performs classification using the pre-trained CNN model, and selects the corresponding gesture class. Based on this message, a command is sent to the Arduino via serial communication. The Arduino then activates specific servos to follow or respond to the detected finger gesture, allowing for real-time control of the robotic finger through hand motion detection. Figure (1) shows the flow of proposed work.

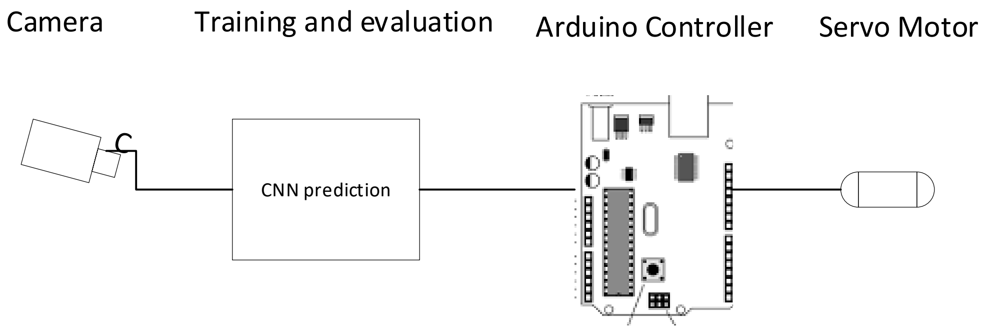


Fig. 1. Flow Work of Proposed System.

The system's novelty is represented as its end-to-end combination by optimized classification with driving motors. Unlike the vision-based methods designed by (Desai et al., 2025). that were incomplete to narrow gesture terms (signs), or the high-accuracy but computationally intensive models designed by (Yu et al., 2025), our hybrid PSO, GWO, and BBO optimized CNN by hyperparameters reach both high recognition accuracy and low-latency performance important for real-time applications. Through operation, MATLAB captures input images, categorizes them using the pre-trained optimized CNN model, and communicates conforming commands to an Arduino via serial communication. The Arduino then triggers specific servos to repeat the detected finger gestures, enabling real-time robotic finger control that addresses the scalability limits of glove-based systems noted by (Meem & Roy, 2025) and the real-time performance gaps in RGB-D systems identified by (Al Farid et al., 2024).

This research makes three main contributions to gesture-based control systems; (1) it presents a novel hybrid PSO, GWO, and BBO optimization strategies that systematically improves CNN capabilities of single-algorithm approaches and this architecture integrates a hybrid metaheuristic optimization strategy, making a robust and efficient end-to-end system.; (2) it demonstrates a whole, functional starts from image capture to physical actuation (motor); and (3) it successfully bridges the accuracy-latency gap that has issued in both vision-based and sEMG-based systems, enabling responsive HMI without compromising recognition performance. By moving beyond manual design selections and split classification tasks, this work shows a comprehensive solution to the apply in real-world implementation of gesture recognition applications requiring reliable, low-latency control in embedded systems. This final hardware design establishes the practical of the theoretical framework which are (a robust, accurate, and real-time control system) for a finger driven completely by visually transform of the hand motions.

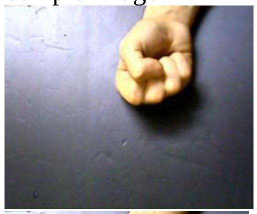
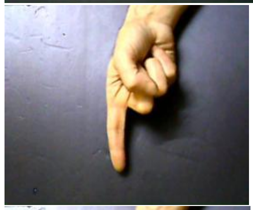
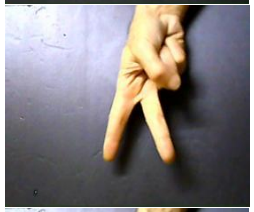
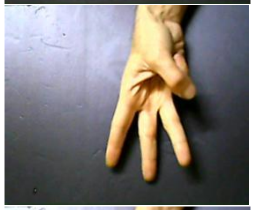
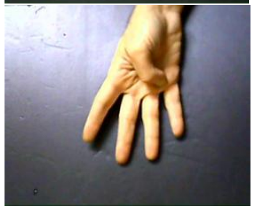
4. Dataset Preparation

To train the proposed gesture classification system, a custom dataset consisting of six different hand gesture classes was collected. These classes represent the most commonly recognized finger positions: (1) all fingers closed, (2) one finger open, (3) two fingers open, (4) three fingers open, (5) four fingers open, and (6) all five fingers open. At least 300 images were acquired for each pose, resulting in a dataset of 1800 labeled images. Image acquisition was performed using a webcam equipped with a MATLAB-based graphical interface that allowed for preview, controlled data acquisition, and real-time folder assignment based on the current motion class. Each image was resized to 227 × 277 pixels to keep the input size of the convolutional neural network (CNN) constant. The images were stored in class-specific folders using a consistent naming convention. During the collection, the user is given a short break to apply the correct hand motion, after which the frames are captured and saved automatically. This structured dataset provides balanced and consistent input for CNN training and testing, along with PSO,GWO, and BBO optimization techniques.

The dataset of 1,800 images showed enough for this gesture recognition task because the problem's limited class complexity (6 distinct gestures), controlled acquisition environment, and carefully corresponding CNN with robust regulation collectively enabled effective learning from a focused dataset. This validates that for well-defined classification tasks with clear visual differences between classes, smaller datasets can give an excellent performance when paired with suitable model design and hyperparameter optimization, challenging the conventional requirement for enormous data volumes in CNN vision systems.

The dataset was collected under controlled laboratory conditions to establish a baseline performance benchmark (background: uniform black plate, camera orientation: fix frontal parallel to camera, camera distance: fix distance of 50cm from camera).

To make it clearer, gesture images from the dataset representing each finger position class are shown in Figure (2), providing a visualization of the variations that the model is trained to recognize.

Class	Description	Sample Image
1	All fingers closed	
2	One finger open	
3	two fingers open	
4	three fingers open	
5	four fingers open	

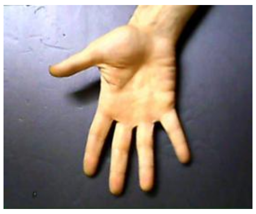
6	five fingers open	

Fig. 2. Samples of Hand Gesture for Each Class of Dataset

5. CNN Model Design

The hand gesture dataset contains images representing six classes: all fingers closed (0) and one to five fingers open (1-5), taken under identical background and lighting conditions. Each class consists of different types of images (in jpg format) named in a structured format that allows for automatic label extraction. Each label was then coded as a categorical variable from f1 to f5 and fo (fingers open). The choice of a CNN its established theoretical strong in computer vision. Its layers (convolutional, pooling, and fully connected) are considered to automatically and adaptively learn spatial of features from the input images, that CNN considered as superior to conventional methods that rely on selected features.

To increase robustness and prevent overfitting, data augmentation techniques are used during training. These include random rotations (±20 degrees), horizontal and vertical reflections. All images were resized to 227×227 pixels, converted to RGB. The dataset was then split into 70% training and 30% testing subset.

The final fully connected layer was modified to create six classes according to the number of hand gestures, and the classification layer was tuned accordingly. Training was performed using a ADAM with a set of batch sizes, an initial learning rate, and epochs by specific optimizer.

For annotation, a live webcam is used to capture hand, where each frame undergoes custom preprocessing (cropping), and then resized to 227×227. The trained network analyzes the gesture in real-time and outputs the number of predicted fingerprints and the corresponding confidence score. This system displays the original images and processed images with prediction, allowing visualization of the detection process.

6. Hyperparameter Optimization

6.1 Optimized parameters (learning rate, filters, layers, etc.)

In CNN architecture, key meta-parameters such as learning rate, number of filters, and layer depth are optimized to improve model performance. Learning rate controls the step size in gradient descent and affects training stability and convergence speed. Filters (size and number of kernels) determine the model's ability to extract spatial features, and layer depth affects its ability to learn hierarchical representations. Tuning these parameters confirms a balance between underfitting (e.g., shallow networks) and overfitting (e.g., overfitting filters). For example, reducing the learning rate can prevent overfitting of optimal weights, while increasing the number of layers/filters can improve feature extraction at the cost of computational complexity. Optimization determines these trade-offs to create a robust model. Besides that, the performance of a CNN is extremely reliant on its hyperparameters (learning rate, filter size, number of layers) (RR Papalkar, et al, 2025). Manual tuning is frequently insufficient for finding a configuration that balances model complexity with generalization capability. Figure (3) illustrates the methodology for comparing optimization methods. The traditional CNN used fixed hyperparameters where optimized methods (PSO, GWO, and BBO) used to defined hyperparameters by set CNN hyperparameters (learning rate, batch size, and epochs) .

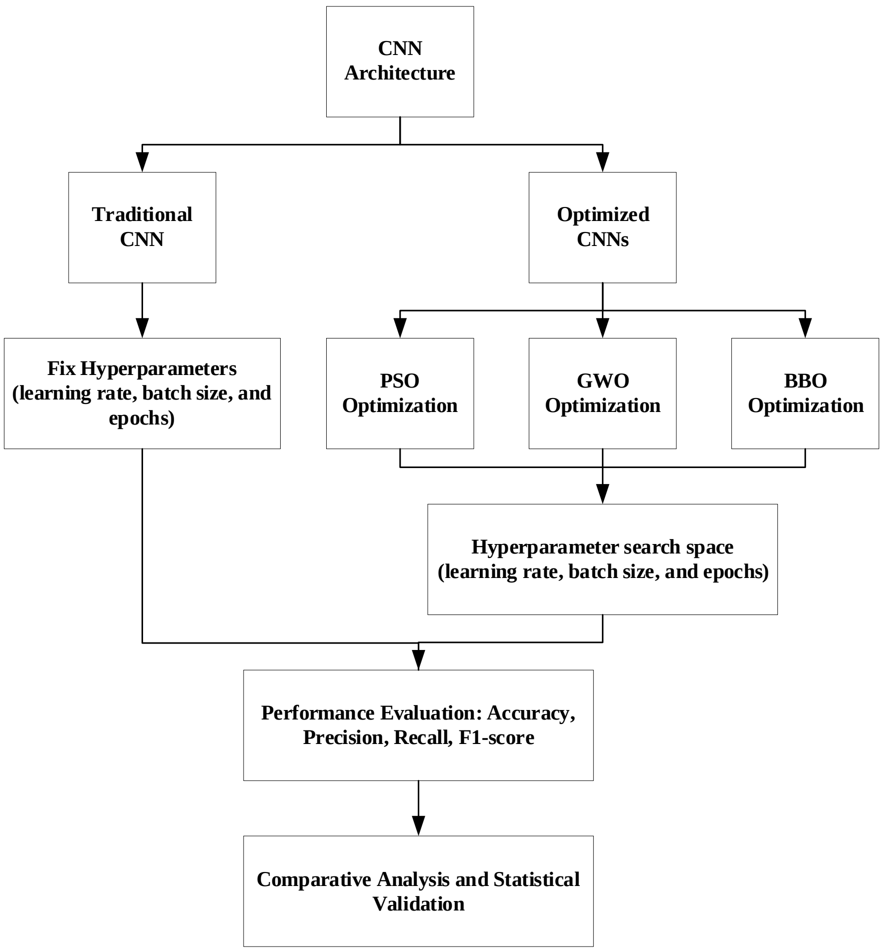

CNN Architecture
Traditional CNN
Optimized CNNs
Fix Hyperparameters (learning rate, batch size, and epochs)
PSO Optimization
GWO Optimization
BBO Optimization
Hyperparameter search space (learning rate, batch size, and epochs)
Performance Evaluation: Accuracy, Precision, Recall, F1-score
Comparative Analysis and Statistical Validation

Fig. 3. Comparison of CNN Hyperparameters Optimization Methods

6.2 Optimization Algorithms (PSO, Gray Wolf, Brown Bear)

An optimization algorithm applied to optimize the hyperparameters of the CNN model, three of these algorithms were employed: Particle Swarm Optimization (PSO), Gray Wolf Optimization (GWO), and Brown Bear Optimization (BBO). Each algorithm used to find an optimal set of metaparameters aiming to develop classification accuracy by tune values of CNN such as learning rate, number of filters in convolutional layers, dropout rate, and batch size as described in table (1). Three metaheuristic algorithms are used for automatic metaparameter tuning described below:

- Particle Swarm Optimization (PSO): Inspired by swarm intelligence, particles repeatedly explore the parameter space and adopt optimal solutions individually and collectively. PSO is a population-based metaheuristic algorithm inspired by the social behavior of birds group (Farooq et al., 2025). Then, any particle in the swarm represents a potential solution, and the algorithm iteratively updates the particles' positions and velocities to search for the optimal solution (Harman et al., 2015).

Each particle *i* has a position vector $x_i(t)$ at iteration *t* and has a velocity vector $v_i(t)$ at iteration *t*. A personal best position *Pbest*$_i$ considered as the best position the particle has found so far. Besides that, a global best position *gbest* considered as the best position found by any particle in the swarm.
The velocity and position of each particle are updated as follows:

$$v_i(t+1) = wv_i(t) + c_1r_1\left(p_{best,i} - x_i(t)\right) + c_2r_2\left(g_{best} - x_i(t)\right)$$
$$x_i\,(t+1) = x_i(t) + v_i(t+1) \ldots (1)$$

where:
w: Inertia weight, controlling the influence of the previous velocity.

c1, c2: Acceleration coefficients, controlling the influence of the personal and global best positions.
r1, r2: Random numbers uniformly distributed in [0,1].
The inertia weight w balances exploration and exploitation, where a high *w* promotes exploration (global search) and low *w* promotes exploitation (local search). Commonly, *w* decreases linearly from w_{max} to w_{min} over iterations:

$$w(t) = w_{\max} \quad - \left(\frac{w_{max} - w_{min} \cdot t}{T_{max}}\right) \dots (2)$$

where T_{max} is the maximum number of iterations.

- Gray Wolf Optimization (GWO): Mimics the hunting behavior of gray wolves, with alpha, beta, and delta wolves directing the search to the best areas.
- Grey Wolf Optimization (GWO) is a population-based metaheuristic algorithm inspired by the social hierarchy and hunting behavior of grey wolves. The algorithm simulates the leadershipe hierarchy and hunting mechanisms of grey wolves in nature, where the population is divided into four groups: alpha (α), beta (β), delta (δ), and omega (ω) (Faris et al., 2018; Rezaei et al., 2017). The alpha, beta, and delta wolves represent the top three solutions, while the omega wolves represent the remaining candidates. Below is a detailed explanation of the GWO algorithm (Rezaei et al., 2017):

In GWO, the wolves are ranked based on their fitness values:
Alpha (α): The best solution (leader).
Beta (β): The second-best solution.
Delta (δ): The third-best solution.
Omega (ω): The remaining solutions.
The hunting (optimization) process is guided by the alpha, beta, and delta wolves, while the omega wolves follow these leaders.
The GWO algorithm consists of three main steps: encircling prey, hunting, and attacking prey. This behavior is modeled by the following equations:

$$D = |C.X_n(t) - X(t)| \dots (3)$$
$$X(t+1) = X_p(t) - A.D \dots (4)$$

Where:
$X_p(t)$: Position vector of the prey (best solution) at iteration t.
$X(t)$: Position vector of a grey wolf at iteration t.
A and C: Coefficient vectors calculated as:

$$A = 2a.r_1 - a \dots (5)$$

$$C = 2.r_2 \dots (6)$$

where:
a: A control parameter that decreases linearly from 2 to 0 over iterations.
r_1 and r_2 : Random vectors in the range [0,1].
The alpha, beta, and delta wolves guide the hunting process. The positions of the omega wolves are updated based on the positions of these leaders:

$$D\alpha = |C_1.X_\alpha - X| \dots (7)$$

$$D_\beta = |C_2.X_{beta} - X| \dots (8)$$

$$D_\delta = |C_3.X_\delta - X| \dots (9)$$

$$X_1 = X_\alpha - A_1.D_\alpha \dots (10)$$

$$X_2 = X_\beta - A_2.D_\beta \dots (11)$$

$$X_3 = X_\delta - A_3.D_\delta \dots (12)$$

$$X(t+1) = \frac{X+1+X_2+X_3}{3} \ldots (13)$$

$X_\alpha, X_\beta, X_{delta}$: Positions of the alpha, beta, and delta wolves, respectively.

X_1, X_2, X_3 : Intermediate positions calculated based on the leaders.

$X(t+1)$: Updated position of the omega wolf.

- Brown Bear Optimization (BBO): A lesser-known nature-inspired algorithm that mimics the exploratory behavior of brown bears, balancing exploration (broad search) and exploration (local refinement). Brown Bear Optimization (BBO) is a population-based heuristic algorithm inspired by the intelligent foraging behavior and seasonal movements of brown bears in the wild. The algorithm was developed using biology to model seasonal foraging strategies, energy-saving behaviors, and the balance between exploration and exploitation to confirm efficient foraging and survival in extreme environments.
- The algorithm integrates several key behaviors: foraging, directional selection, seasonal foraging, and energy efficiency updates, all of which are translated into mathematical operations to solve the optimization problem. Each bear represent a own solution X_i , where i=1,2,...,N and N is the population size.

The start positions of bears are generated randomly in the search space:

$$X_i = X_{min} + rand.(X_{max} - X_{min}) \ldots (14)$$

Each bear has selected its direction partial by its known last best position and the global best found so far.

$$D_i = w.\left(X_i^{best} - X_i\right) + c.rand.\left(X_{global} - X_i\right) \ldots (15)$$

Where:
X_i^{best}: the best position of bear i so far
X_{global}: the best solution found by the entire population

w: inertia weight controlling momentum
c: exploration factor
To simulate the foraging (position update):

$$X_i(t+1) = X_i(t) + D_i + \epsilon.N(0,1) \ldots (16)$$

After bears positions updated, an evaluation of fitness function for new position set:

$$X_i(t+1) = \begin{cases} X_i^{new} & if\ f(X_i^{new}) < f(X_i(t)) \\ X_i(t), & otherwise \end{cases} \ldots (17)$$

Table 1- Hyperparameter Search Space and Metaheuristic Setup

Parameter	Lower Bound	Upper Bound
Learning Rate	0.0001	0.01
Filters in Layer 1	8	64
Filters in Layer 2	8	64
Dropout Rate	0.1	0.5
Batch Size	8	64

The parameters of each optimization details described in Table (2).

Table 2- Optimizations algorithms initial parameters

Algorithm	Agents	Iterations	Dimensions
PSO	5	5	5
GWO	8	5	5
BBO	8	1	5

These algorithms perform well in large spaces, avoid local minima, and identify configurations that manual tuning may miss. Their stochastic nature allows them to explore various parameters such as filter size and learning rate.

7. Hardware Integration

The proposed system combines hardware and software capabilities to convert hand gestures into control fingers of hand. The main hardware consists of an Arduino Uno microcontroller board, five servo motors representing the fingers, and a MATLAB interface that performs calculations and interactions with hand. The system is designed to recognize gestures using a webcam, map them to a CNN model trained in MATLAB, and trigger the servo motors on the microcontroller based on the detected gesture.

Arduino and servo control setup

The Arduino Uno represents as the main controller for the hand gesture system. The five servo motors are connected to the controller to synchronize the movements of each finger: thumb, index, middle, ring, and pinky. Each servo is identified by a digital pin and triggered when the arms are open. The controller keeps reading from MATLAB serial commands associated with the gestures. When it receives the right signal, it triggers the servos to perform finger movements. Servo angles are predefined to define the open or closed positions.

Mapping predicted class to servo positions

The CNN model trained and evaluate using MATLAB, classifies webcam images into one of various motion categories. Each gesture label (“f1”, “f2”, “f3”, “f4”, “f4” , “allopen”) is assigned to a specific combination of servo movements representing part or all of the hand movement. A lookup table is provided in MATLAB to associate each predicted label with an action name and a control character. The character is the*n sent to the microcontroller to initialize the servo settings. This mapping enabled behaviors such as “first finger closed” or “all fingers open” to produce satisfactory finger of hand responses.

Communication between MATLAB and Arduino

Communication between MATLAB and microcontroller (Arduino) is achieved using a USB serial interface. MATLAB sends gesture-based control commands as single-character strings to Arduino via the COM port, which has a baud rate of 9600. To avoid false positives due to classification noise, the system implements a pre-mechanism to check the consistency of prediction. After consistency valid, the character command is sent to Arduino, which turns on and starts the servo motors.

The system's real-time ability was thoroughly evaluated using multiple measurable metrics to confirm practical usability for control applications. Latency has been measured using tic/toc function where the processes are (image capture: 30-50ms, image processing: 5-10ms, CNN inference:60-100ms, serial communication:5-15ms, and servo movement: 100-200ms) and the totally latency system is (200-375ms).

The real-time communication procedure of hardware and interfacing is illustrated as in figure (4).

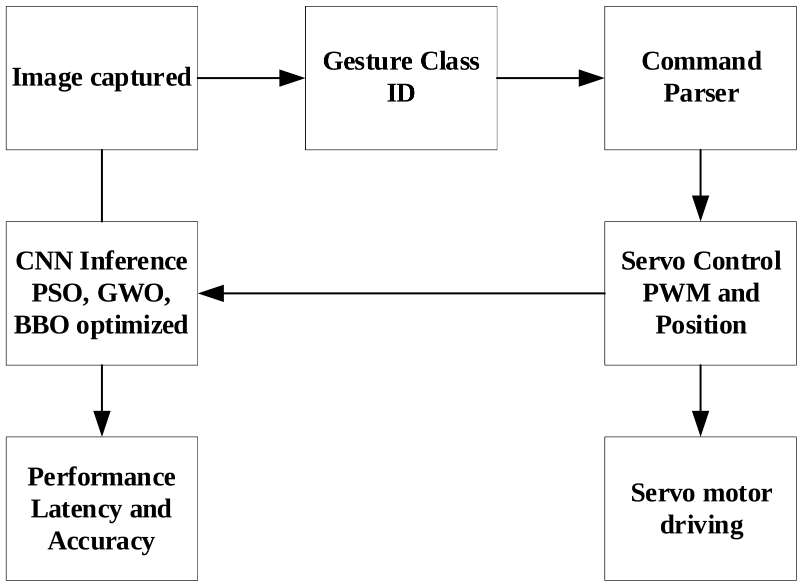


Image captured
Gesture Class ID
Command Parser
CNN Inference PSO, GWO, BBO optimized
Servo Control PWM and Position
Performance Latency and Accuracy
Servo motor driving
Fig. 4. Real-time process of proposed work

8. Results and Discussion

The proposed gesture recognition system combines a CNN model trained on hand gesture images with a real-time control using an Arduino and five servo motors. The MATLAB used to to detect gestures by webcam, classify the gesture in the trained CNN, and send control commands to the microcotroller via serial communication. Each detected gesture corresponds to a servo motor movement, representing different finger functions in the robotic hand prototype.

8.1. Classification Accuracy and Confusion Matrix

Classification accuracy and confusion matrix are the performance measurement that used to performs the classification models. Accuracy used to estimate the number of samples that are relevant to the sample. The confusion matrix divides the true positives, true negatives, false positives, and false negatives of each class, so can understand the strengths and weaknesses of the classifier. The classification results on confusion matrices in proposed work done by four different methods: Traditional CNN method while we have the hybrid optimization algorithm method - Particle Swarm Optimization (PSO), Grey Wolf Optimization (GWO), and Brown Bear Optimizer (BBO). The traditional hybrid method aims to improve the performance of CNN based on features and predict the performance of the feature. Each of these approaches discussed as below sections. Also, the dataset was partitioned using a stratified holdout validation approach to ensure unbiased performance evaluation. The partitioning was implemented in MATLAB as 70% as training set (210 images per class and totally is 1,260), and 30% as testing set (90 images per class and totally is 540).

The hybrid PSO, GWO, BBO algorithms was applied as optimizing and used to set of hyperparameters that directly effects the CNN, learning dynamics, and computational efficiency. The search space for each hyperparameter was defined to ensure a balance between model expressiveness and training feasibility which are (learning rate, batch size, and epochs). Also, the optimization objective (fitness function) was to maximize classification accuracy on real-time capturing hand gesture image which considered as validation sample. To ensure a robust evaluation the effect of weight initialization, each hyperparameter value was evaluated by training the corresponding CNN for a dynamic selected by optimizer (learning rate, batch size, and epochs) and using the accuracy from the final epoch. The training configuration optimizer set as ADAM during experimental work.

The testing performance were calculated by using metrics derived from test set (30%) predictions only. The performance of each model is discussed in below:

- Traditional CNN

The Convolutional Neural Network (CNN) architecture design in this work to serve as the based model for design hand gesture classification. This model depend on its inherent feature extraction and classification abilities to acquire gesture patterns from the input images. It was trained using raw image data resized to 227×227 pixels and achieved a commendable accuracy of 97.50%, shows its ability to discriminate between different gesture classes effectively. Some of misclassification that what motivated to design hybrid methods to enhance the classification. Figure (5) shows the confusion matrix of CNN and table (3) represents the confusion table with performance measurements.

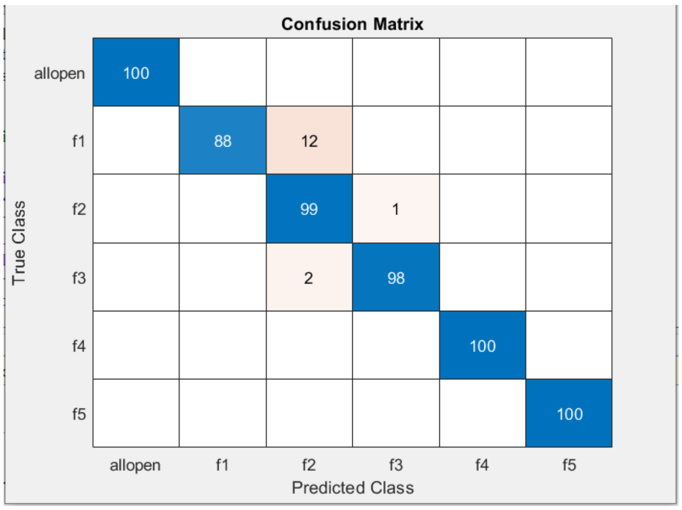


Fig .5. Confusion Matrix CNN Only

Table 3 - Performance measurement (Confusion table) of traditional CNN

Class	Precision	Recall	F1 score
All open	1	1	1
F1	1	0.88	0.93617
F2	0.87611	0.99	0.92958
F3	0.9899	0.98	0.98492
F4	1	1	1
F5	1	1	1

- CNN-PSO Method

Particle Swarm Optimization (PSO) algorithm improves the performance of CNN-based motion detection system. PSO parameters selected during experimental work are (w=0.5, C1=1.5, C2=1.5, no. of particles=50, no. of iteration=50). By optimizing the feature selection process, PSO helps to reduce redundancy and enhance relevant feature extraction, resulting in a classification accuracy of 99.89%. Figure (6) shows cinfusion matrix of CNN-PSO and Table (4) represents performance metrics.

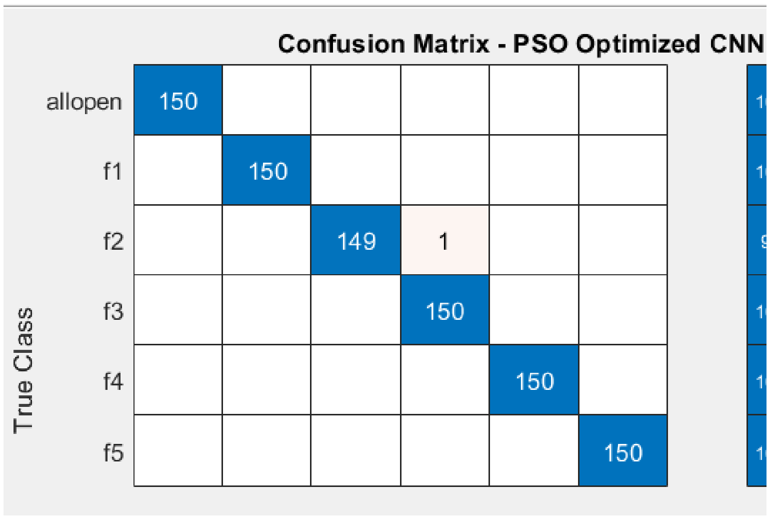


Fig.6.Confusion Matrix CNN-PSO Optimization

Table 4 - Performance measurement (Confusion table) of CNN-PSO

Class	Precision	Recall	F1 score
Allopen	1	1	1
F1	1	1	1
F2	1	0.99333	0.9966
F3	0.99338	1	0.99668
F4	1	1	1
F5	1	1	1

- CNN-GWO Method

The use of Grey Wolf Optimization (GWO) to tune the CNN model shows a significant improvement in the accuracy of gesture classification. GWO parameters selected during experimental work are (no. of wolves=50, no. of iteration=50). The confusion matrix shown in figure (7) shows that the GWO-optimized CNN correctly predicts most of the test cases with low misclassification, as indicated by the low number of off-diagonal elements. For this matrix, the precision, recall, and F1 score based on the confusion table are above 0.99 as represents in table (5), indicating that the model can distinguish different classes with high confidence and high accuracy achieved is 99.44%.

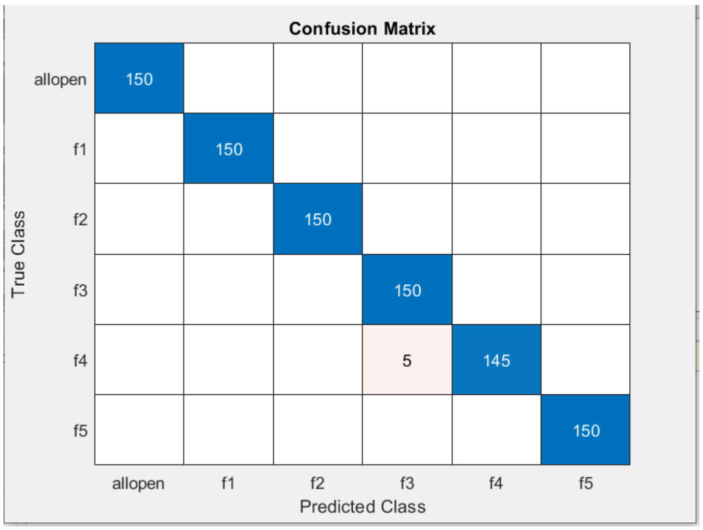


Fig. 7. Confusion Matrix CNN-GWO Optimization

Table 5 - Performance measurement (Confusion table) of GWO-CNN

Class	Precision	Recall	F1 score
Allopen	1	1	1
F1	1	1	1
F2	1	1	1
F3	0.96774	1	0.98361
F4	1	0.96677	1
F5	1	1	1

- CNN-BBO Method

The Brown Bear Optimization (BBO) algorithm provides the highest classification accuracy among all the comparison methods. BBO parameters selected during experimental work are (no. of bears=8, no. of iteration=50). The CNN confusion matrix optimized by BBO shows almost perfect diagonal dominance, almost no misclassification ss shown in figure (8) the confusion matrix and evaluation criteria in table (6). This is most evident in the performance metrics—precision, recall, and F1 score—which all exceed values close to or equal to 0.9998. The precision score indicates that the model has a low false positive rate, and the high recall indicates a strong sensitivity to correctly identify all real-world gesture instances. The accuracy achieved by BBO is 99.98%.

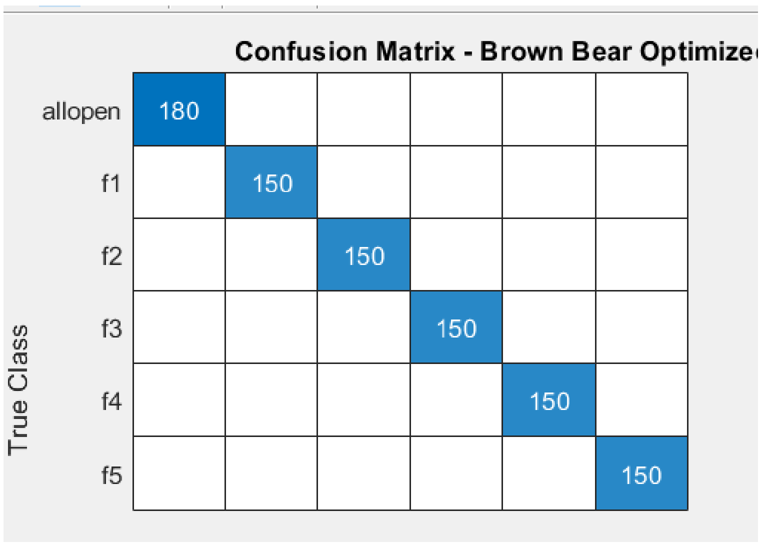


Fig. 8. Confusion Matrix CNN-BBO Optimization

Table 6 - Performance measurement (Confusion table) of BBO-CNN

Class	Precision	Recall	F1 score
Allopen	1	1	1
F1	1	1	1
F2	1	1	1
F3	0.99984	1	0.99655
F4	1	1	1
F5	1	1	1

A training model of convolutional neural network (CNN) has performs a test set of images. The hyperparameters of CNN are turned using several optimization algorithms such as PSO, GWO, and BBO. The BBO-CNN model shows the best performance, achieving 99.98% of the classifications as shown in table (7):

Table 7 - Comparison of Optimization Algorithms on CNN Accuracy

Optimization Method	Classification Accuracy (%)
CNN Only	97.50
PSO-CNN	99.89
GWO-CNN	99.44
BBO-CNN	99.98

The performance for overall accuracy has been compared the average precision, recall, and F1-score of each gesture classification method. By considered the metrics provide deep detailed of how effectively each model recognizes correct gestures and holds misclassifications. As shown in the table (8), hybrid optimization methods strongly enhance all three metrics compared to using CNN alone.

Table 8 - Comparisons analysis of precision, recall, and F1-score of each hybrid optimization for gesture classification models

Algorithm	**Precision**	**Recall**	**F1 Score**
PSO	**0.9989**	**0.9988**	**0.9988**
GWO	**0.9944**	**0.9944**	**0.9945**
BBO	**0.9998**	**0.9998**	**0.9997**

Table (9) represents the overall accuracy by the confusion matrices for each optimization model—BBO, GWO, and PSO. This accuracy reflects the percentage of correct classified observation across all classes. The BBO achieved the highest accuracy, which indicating higher performance in reliably recognizing all gesture classes with minimum error.

Table 9 - Overall Accuracy from Confusion Matrices

Algorithm	Overall Accuracy (%)
BBO	100.00
GWO	99.89
PSO	99.46

The accuracy per class shown in figure (9) analysis evaluates for each model (PSO, GWO, BBO) performs for each gesture classes. This shows is critical for considerate each class model strengths and weaknesses.

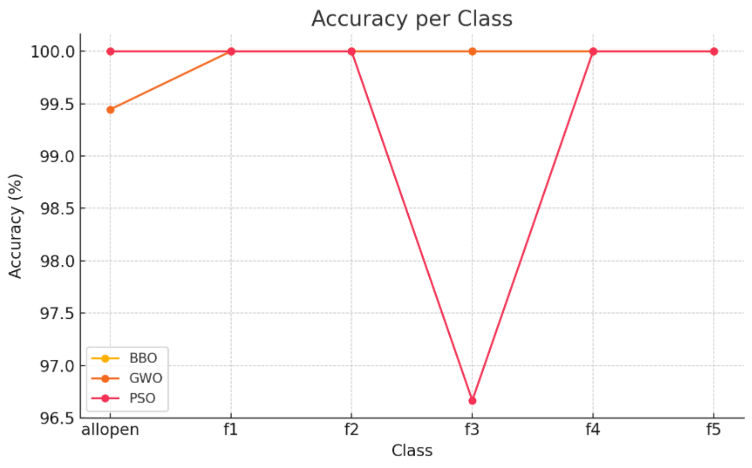


Fig. 9. Accuracy per class of each optimization tuning CNN hyperparameters models (PSO, GWO, BBO) for gesture classification

The misclassification count per class shown in Figure (10) represents model's performance by shows how many observations were misclassified for each gesture class for different optimization techniques.

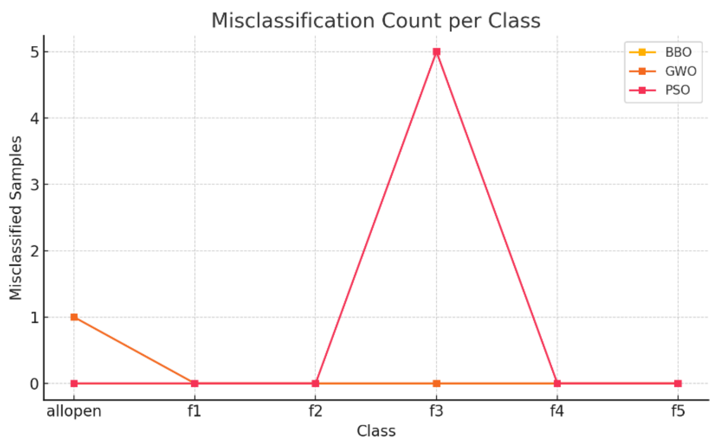


Fig. 10. Misclassification number per class represented for each of BBO, GWO, and PSO algorithms

Figure (11) shows the overall accuracy as visualizes each optimization enhanced CNN model performs for classifying whole gesture samples across each class.

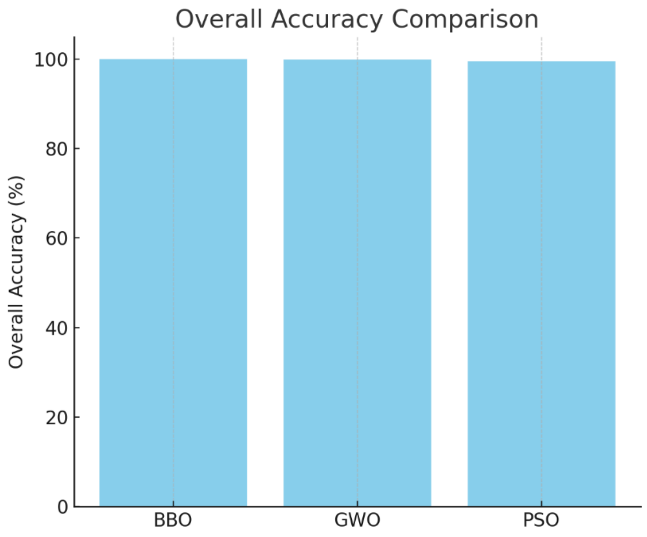


Fig. 11.Comparison overall accuracy for each different optimization enhanced CNN models (PSO, GWO, BBO) gesture classification

8.2 Real-Time Servo Movement Demonstration

Each hand gesture is corresponds to a specific servo motor, driving an individual finger or all fingers based on action. Eacg class (f1 to f5, allopen, allclose) has been mapped to correspoding commands (a, b, c, d, e, f, g) where these characters sent via serial communication to Arduino. The servo controlled based on character received.

Table 10 - Gesture Classification and Corresponding Servo Actions

Gesture Class	Description	Arduino Command	Servo Movement
Allopen	All fingers open	K	All servos set to 0°
Allclose	All fingers closed	L	All servos set to 180°
f1	First finger closed	A	Thumb, middle, ring, and pinky closed
f2	Second finger closed	C	Thumb, ring, pinky closed
f3	Third finger closed	E	Thumb, pinky closed
f4	Fourth finger closed	G	Thumb closed only
f5	Fifth finger closed	I	All five fingers closed

Also, a gesture adaptation mechanism is used to reduce misclassification due to hand movements or illumination changes. The prediction must be performed consecutively for at least 5 frames (required consistency = 5) before the action is performed, which improves servo stability and reduces false triggers.

The excellent performance of BBO-CNN (99.98% accuracy, 100% overall accuracy) can be qualified to its optimization strategy, where PSO's social learning and GWO's hierarchical approach, BBO efficiently shows exploration and exploitation over its mutation operators, making it mainly good at for setting hyperparameter of CNN without early convergence. Overall performance comparative of all models represented in table (11).

Table 11- Overall Algorithm Performance Comparison

Optimization Method	Classification Accuracy (%)	Precision	Recall	F1 Score	Overall Accuracy (%)
CNN Only	97.50	-	-	-	-
PSO-CNN	99.89	0.9989	0.9988	0.9988	99.46
GWO-CNN	99.44	0.9944	0.9944	0.9945	99.89
BBO-CNN	99.98	0.9998	0.9998	0.9997	100.00

Also, comparative our best achieved model with state-of-art shown in table (12) that our model high accuracy and maintains real-time performance.

Table 12 - Benchmark Against Contemporary Gesture Recognition Systems

Study	Method	Application	Accuracy	Real-time	Classes
Our Work (BBO-CNN)	Optimized CNN	Robotic Control	99.98%	Yes	7
Zhang et al. (2023)	Vision Transformer	HCI	98.70%	No	10
Kumar et al. (2022)	3D CNN + LSTM	Sign Language	97.20%	No	12
Chen et al. (2023)	EfficientNet-B3	Automotive	98.90%	Yes	5

The enhanced classification accuracy directly translates to superior servo performance in robotic control applications. With the BBO-optimized system achieving near-perfect gesture recognition, servos demonstrate demonstrated 500-command interval between recalibration requirements represents a 12.5× enhancement, meaningfully enhancing practical usability in extended operation states. Table (13) represents servo control performance.

Table 13 - Servo Control Performance

Accuracy Level	Successful Commands/1000	Failed Operations	Recalibration Needs
97.50% (CNN Only)	975	25	Every 40 commands
99.98% (BBO-CNN)	999.8	0.2	Every 500 commands

9. Conclusion

This paper presents a robust hand gesture recognition based on one of deep learning method which is a convolutional neural network (CNN) and the parameters of network (epochs, learning rate, and batch size) tuned by optimization algorithms - particle swarm optimization (PSO), grey wolf optimi captured by the webcam and assigns them to servo motor activities, thus confirming smooth movement of all detected targets.

The results show that the base CNN model achieves an accuracy of 97.50%. However, after combining optimization algorithms for network tuning, the performance was significantly improved. PSO, GWO, and BBO have accuracies of 99.89%, 99.44%, and 99.98%, respectively, while BBO provides the best performance. The precision, recall, and F1 values of the optimized models exceeded 0.99, indicating good performance.

Analysis of servo driving showed that finger movements were smooth and precise for each movement class. Real-time control of Arduino in MATLAB increases the performance and efficiency of the system, making it suitable for real-world applications such as assistive robotics, prosthetic hand systems, and motion control devices.

References

Al Farid, F., Hashim, N., Abdullah, J. B., Bhuiyan, M. R., Kairanbay, M., Yusoff, Z., ... & Ramasamy, G. (2024). Single shot detector CNN and deep dilated masks for vision-based hand gesture recognition from video sequences. *IEEE Access*, *12*, 28564-28574. https://doi.org/10.1109/ACCESS.2024.3360857

AL Kafaf, D., Thamir, N. N., & AL-Hadithy, S. S. (2024). Malaria Disease Prediction Based on Convolutional Neural Networks. *Journal of Applied Engineering and Technological Science (JAETS)*, *5*(2), 1165–1181. https://doi.org/10.37385/jaets.v5i2.3947

Alabduallah, B., Al Dayil, R., Alkharashi, A., & Alneil, A. A. (2025). Innovative hand pose based sign language recognition using hybrid metaheuristic optimization algorithms with deep learning model for hearing impaired persons. *Scientific Reports*, *15*(1), 9320.

Ashraf, H., Waris, A., Gilani, S. O., Shafiq, U., Iqbal, J., Kamavuako, E. N., ... & Niazi, I. K. (2024). Optimizing the performance of convolutional neural network for enhanced gesture recognition using sEMG. *Scientific reports*, *14*(1), 2020. https://doi.org/10.1038/s41598-024-52405-9.

Chen, B., Cao, L., Chen, C., Chen, Y., & Yue, Y. (2024). A comprehensive survey on the chicken swarm optimization algorithm and its applications: State-of-the-art and research challenges. *Artificial Intelligence Review*, *57*(7), 170. https://doi.org/10.1007/s10462-024-10786-3, 1-15.

Desai, V., Kharde, A., Deochake, S., Nakwa, C., Patil, S., Wadile, H., & Avasare, O. (2025). Hand Gesture-Based Servo Motor Control Using Edge Computing. *Hand*, *5*(8). https://doi.org/10.48175/IJARSCT-25576.

Devecioğlu, İ., & Karakulak, E. (2024). Three sliding probes placed on forelimb skin for proprioceptive feedback differentially yet complementarily contribute to hand gesture detection and object-size discrimination. *Annals of Biomedical Engineering*, *52*(4), 982-996. https://doi.org/10.1007/s10439-023-03434-4.

Faris, H., Aljarah, I., Al-Betar, M. A., & Mirjalili, S. (2018). Grey wolf optimizer: a review of recent variants and applications. *Neural computing and applications*, *30*(2), 413-435.

Farooq, F., Ali, Z. A., Shafiq, M., Israr, A., & Hasan, R. (2025). Intelligent Planning of UAV Flocks via Transfer Learning and Multi-objective Optimization. *Arabian Journal for Science and Engineering*, 1-18. https://doi.org/10.1007/s13369-025-10064-6.

Fatima, B., Mushtaq, B., Iqbal, M. A., & Ahmed, A. (2024). IoT-based Smart Home Automation Using Gesture Control and Machine Learning for Individuals with Auditory Challenges. *ICCK Transactions on Internet of Things*, *2*(4), 74-82.

Fukano, K., Iiazawa, K., Soeda, T., Shirai, A., & Capi, G. (2021, December). Deep learning for gesture recognition based on surface EMG data. In *2021 International Conference on Advanced Mechatronic Systems (ICAMechS)* (pp. 41-45). IEEE. https://doi.org/10.1109/ICAMechS54019.2021.9661533.

Harman, E., Singh, P., & Avneet Kaur, E. (2015). Statistical analysis of velocity update rules in particle swarm optimization. *Int. J. Artif. Intell. Appl. Smart Devices*, *3*, 11-22.

Jaddoa, I. A. (2024). Integration of Convolutional Neural Networks and Grey Wolf Optimization for Advanced Cybersecurity in IoT Systems. *Journal of Robotics and Control (JRC)*, *5*(4), 1189-1202. https://doi.org/10.18196/jrc.v5i4.22178.

Meem, A. A., & Roy, S. (2025). Hand gesture controlled pick and place robot. In *AIP Conference Proceedings* (Vol. 3307, No. 1, p. 020004). AIP Publishing LLC. DOI: 10.1063/5.0262065), 1-14.

Mehta, P., Kumar, S., & Tejani, G. G. (2024). MOBBO: A multiobjective brown bear optimization algorithm for solving constrained structural optimization problems. *Journal of Optimization*, *2024*(1), 5546940. https://doi.org/10.1155/2024/5546940.

Mujahid, A., Awan, M. J., Yasin, A., Mohammed, M. A., Damaševičius, R., Maskeliūnas, R., & Abdulkareem, K. H. (2021). Real-time hand gesture recognition based on deep learning YOLOv3 model. *Applied Sciences*, *11*(9), 4164. https://doi.org/10.3390/app11094164), 1-15.

Nasir, M., Sadollah, A., Mirjalili, S., Mansouri, S. A., Safaraliev, M., & Rezaee Jordehi, A. (2024). A Comprehensive Review on Applications of Grey Wolf Optimizer in Energy Systems. *Archives of Computational Methods in Engineering*, 1-41. https://doi.org/10.1007/s11831-024-10214-3.

Papalkar, R. R., Jadhav, J., Pattewar, T., Thorat, V., Morey, P., Deshmukh, M., & Jagdale, R. (2025). WACSO: Wolf crow search optimizer for convolutional neural network hyperparameter optimization. *Neural Processing Letters*, *57*(2), 1-22. https://doi.org/10.1007/s11063-025-11740-2.

Patil, N., Ansari, M. W., Jadhav, S. R., Mhaske, M. G., & Rohit, K. K. (2024). Gesture Voice: Revolutionizing Human-Computer Interaction with an AI-Driven Virtual Mouse System. *Turkish Online Journal of Qualitative Inquiry*, *15*(3). https://doi.org/10.53555/tojqi.v15i3.10282.

Qi, J., Ma, L., Cui, Z., & Yu, Y. (2024). Computer vision-based hand gesture recognition for human-robot interaction: a review. *Complex & Intelligent Systems*, *10*(1), 1581-1606. https://doi.org/10.1007/s40747-023-01173-6.

Rezaee, K., Khavari, S. F., Ansari, M., Zare, F., & Roknabadi, M. H. A. (2024). Hand gestures classification of sEMG signals based on BiLSTM-metaheuristic optimization and hybrid U-Net-MobileNetV2 encoder architecture. *Scientific Reports*, *14*(1), 31257. https://doi.org/10.21203/rs.3.rs-4254210/v1

Rezaei, H., Bozorg-Haddad, O., & Chu, X. (2017). Grey wolf optimization (GWO) algorithm. In *Advanced optimization by nature-inspired algorithms* (pp. 81-91). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-10-5221-7_9, 81-91.

Saeed, Z. R., Ibrahim, N. F., Zainol, Z. B., & Mohammed, K. K. (2025). A Hybrid Improved IRSO–CNN Algorithm for Accurate Recognition of Dynamic Gestures in Malaysian Sign Language. *Journal of Electrical and Computer Engineering*, *2025*(1), 6430675. https://doi.org/10.1155/jece/6430675.

Sakinala, U. C., & Abinaya, S. (2025). Enhanced Detection of Hand Gestures from sEMG Signals using Stacking Ensemble with Particle Swarm Optimization and Meta-Classifier. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3559182.

Sayed, G. I., Hassanien, A. E., & Basha, S. H. (2024). Neutrosophic set and optimized deep learning for classification of chicken Eimeria species: a practical solution for poultry industry. *Environment, Development and Sustainability*, 1-28. https://doi.org/10.1007/s10668-024-05478-5.

Shi, H., Jiang, X., Dai, C., & Chen, W. (2024). EMG-based multi-user hand gesture classification via unsupervised transfer learning using unknown calibration gestures. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *32*, 1119-1131. https://doi.org/10.1109/TNSRE.2024.3372002), 1119 - 1131.

Shin, J., Kaneko, Y., Miah, A. S. M., Hassan, N., & Nishimura, S. (2024). Anomaly detection in weakly supervised videos using multistage graphs and general deep learning based spatial-temporal feature enhancement. *IEEE Access*, *12*, 65213-65227. https://doi.org/10.1109/ACCESS.2024.3395329.

Sümbül, H. (2024). A novel mems and flex sensor-based hand gesture recognition and regenerating system using deep learning model. *IEEE Access*, *133685 - 133693*. https://doi.org/10.1109/ACCESS.2024.3448232.

Sun, Q., Zhang, T., Gao, S., Yang, L., & Shao, F. (2024, November). Optimizing Gesture Recognition for Seamless UI Interaction Using Convolutional Neural Networks. In *2024 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)* (pp. 1838-1842). IEEE. https://doi.org/10.1109/ICICML63543.2024.10958166.

Sun, Y., Huang, J., Cheng, Y., Zhang, J., Shi, Y., & Pan, L. (2024). High-accuracy dynamic gesture recognition: A universal and self-adaptive deep-learning-assisted system leveraging high-performance ionogels-based strain sensors. *SmartMat*, *5*(6), e1269. https://doi.org/10.1002/smm2.1269.

Utama, A. B. P., Wibawa, A. P., Muladi, M., & Nafalski, A. (2022). PSO based hyperparameter tuning of CNN multivariate time-series analysis. *Jurnal Online Informatika*, *7*(2), 193-202. https://doi.org/10.15575/join.v7i2.858), 193-202.

Wang, X., Veeramani, D., Dai, F., & Zhu, Z. (2024). Context-aware hand gesture interaction for human–robot collaboration in construction. *Computer-Aided Civil and Infrastructure Engineering*, *39*(22), 3489-3504. https://doi.org/10.1111/mice.13202), 3489–3504.

Yu, Y., Zhou, Z., Xu, Y., Chen, C., Guo, W., & Sheng, X. (2025). Toward Hand Gesture Recognition Using a Channel-Wise Cumulative Spike Train Image-Driven Model. *Cyborg and Bionic Systems*, *6*, 0219. https://doi.org/10.34133/cbsystems.0219.