Chapter 7

Improving sEMG-based dynamic Hand Gesture Recognition through Salp Swarm Algorithm-Driven Feature Selection and Fusion

In this chapter, we refine the recognition of dynamic hand gestures using surface electromyography (sEMG) by leveraging a modified Salp Swarm Algorithm (SSA) tailored for feature selection and fusion. This approach specifically addresses the issues associated with our problem of high-dimensional feature space and associated intensive computational demands.

The original SSA, inspired by the swarming behavior of salps in the ocean, is effective for global optimization by balancing exploration and exploitation. However, it faces challenges such as premature convergence and limited exploration capabilities.

To address these limitations, we have introduced modifications to create an enhanced version of SSA. This enhanced SSA algorithm includes modifications aimed at improving feature selection effectiveness and computational efficiency. Validated on benchmark UCI datasets, our enhanced SSA outperforms various baseline techniques. The improved accuracy highlights the potential of our integrated feature selection and fusion methodology for further advancements in sEMG-based gesture recognition.

Section 7.1 outlines the foundational aspects of our approach, including a discussion of the problem statement and an overview of the solution. The experimental setup is given in Section 7.2, which covers the materials and methods employed, including the collected dataset, data preprocessing strategies using various filters, and the feature extraction techniques employed. Section 7.3 discusses the preliminary concepts of SSA and the proposed modifications. Section 7.4 presents the experimental results and discussion. Finally, Section 7.5 summarizes the key findings and conclusions.

7.1 Introduction

Feature selection methods generally navigate the solution space to balance conflicting goals, such as increasing relevance to the target class while reducing redundancy among the selected features [280]. Based on their evaluation criteria, these methods can be classified into two main categories: feature ranking and subset selection [281]. Feature ranking methods assign a score S_i to each feature f_i based on a specified criterion. Features with scores below a certain threshold τ are removed:

$$S_i = \text{Criterion}(f_i), \quad \text{if } S_i < \tau,\ f_i \text{ is removed}$$

In contrast, subset selection methods search the space of all possible feature subsets to find the optimal subset. Given n features, the search space consists of 2^n

possible subsets. The goal is to find a subset F_{opt} that optimizes a given objective function J:

$$F_{\text{opt}} = \arg \max_{F \subseteq \{f_1, f_2, \ldots, f_n\}} J(F)$$

While feature ranking methods evaluate each feature independently without considering inter-feature relationships, subset selection methods take these dependencies into account. These methods reduce computational complexity but may compromise performance due to their simplistic assumptions.

Feature subset selection is an NP-Hard problem [282]. Due to the high cost of evaluating all possible subsets, it is essential to find a feature subset that balances computational complexity and relevance [283]. Although the optimal subset can theoretically be found by evaluating every possible combination, this approach is impractical for medium to large datasets because of its extreme computational cost. The computational cost C of evaluating all subsets is $C = O(2^n)$.

Meta-heuristic algorithms offer a viable solution to these challenges by providing approximate solutions within an acceptable timeframe rather than guaranteeing the optimal solution [284, 285]. These algorithms use strategies to escape local optima and are widely applicable to various optimization problems.

Meta-heuristic algorithms can be broadly categorized into Evolutionary Algorithms (EA) and Swarm Intelligence (SI) [284]. EA mechanisms are inspired by biological evolution, incorporating processes like reproduction, mutation, recombination, and selection. Candidate solutions represent individuals in a population, and their quality is assessed using a fitness function ϕ:

$$\phi(\mathbf{x}) = \text{Fitness Function}(\mathbf{x}), \quad \mathbf{x} \in \text{Population}$$

Through iterative application, the population evolves towards global optimization [286].

Swarm Intelligence-based optimization methods consist of simple agents interacting to produce complex, intelligent behavior. Inspired by natural phenomena, these methods involve agents performing simple tasks, with their interactions and somewhat random reactions leading to a collective intelligence that surpasses the capability of individual agents [281].

Other meta-heuristic approaches, such as Genetic Algorithms [287] and Simulated Annealing, also attempt to solve the feature selection problem. However, these methods often face limitations like slow convergence rates and high computational costs, particularly when dealing with large datasets or complex search spaces [281].

Given this context, Swarm Intelligence methods, such as the Salp Swarm Algorithm (SSA), stand out for their balance between exploration and exploitation [285]. The original SSA, inspired by the swarming behavior of salps, is effective for global optimization but faces challenges such as premature convergence and limited exploration capabilities.

To overcome these limitations and enhance sEMG-based dynamic hand gesture recognition, we propose an improved SSA. This enhanced algorithm incorporates several modifications to address these issues and improve feature selection effectiveness and computational efficiency. Validated on 14 UCI datasets, the enhanced SSA demonstrates superior performance, addressing the high-dimensional feature vectors and computational costs associated with handwriting gesture recognition.

7.1.1 Problem Statement and overview of solution

7.1.1.1 Problem Statement

Given a dataset D consisting of N recordings of handwritten characters, each recording includes both sEMG signals and IMU data. Each sEMG recording $T_{Si}(u) = \{u_1, u_2, u_3, \ldots, u_N\}$ for i in the range 1 to 8 represents signals collected from eight sensors, and each IMU recording $T_{Ii}(v) = \{v_1, v_2, v_3, \ldots, v_M\}$ for i in the range 1 to 3 represents data collected from three IMU sensors. How can we accurately predict

the class label of each handwritten character using early feature fusion of extracted features from both sEMG and IMU signals? The goal is to develop a classification model that ensures robust performance. The challenge is to utilize meta-heuristic feature selection techniques to identify and optimize the most significant features from these multi-sensor signals, thereby enabling precise classification of each character.

The proposed Handwritten Character Recognition (HCR) model aims to effectively decipher human handwritten traces by identifying letters and symbols from data collected from diverse sensors, particularly in dynamic environments such as whiteboards. Within this HCR framework, we describe a set of potential handwritten characters Z as:

$$Z = [Z_m]_{m=1}^{x} \tag{7.1}$$

Where x signifies the variety of symbols inscribed on a whiteboard. The data, embodying sequential patterns that illustrate unique handwriting characteristics, can be represented as:

$$\begin{aligned}
R_1 &= \{e_{11}, e_{12}, \ldots, e_{1u}, \ldots, e_{1p}\} \ \ (\text{Sensor}_1 \text{ readings}) \\
R_2 &= \{e_{21}, e_{22}, \ldots, e_{2u}, \ldots, e_{2p}\} \ \ (\text{Sensor}_2 \text{ readings}) \\
&\vdots \\
R_q &= \{e_{q1}, e_{q2}, \ldots, e_{qu}, \ldots, e_{qp}\} \ \ (\text{Sensor}_q \text{ readings})
\end{aligned}$$

where R_q, $1 \leq q \leq k$, represents the readings from different sensors, e_{qp} represents the q-th sensor's reading at time t, and p represents the sequence length.

In our setup, data is primarily sourced from two specific sensors: sEMG and IMU. The sEMG sensors gauge the activation patterns of the writer's hand muscles,

whereas the IMU sensors track the trajectory and stance of the hand during writing. Utilizing this collated data, the goal is to develop a model H that predicts the character label $\hat{Z}$, using the information obtained from the input sensors:

$$\hat{Z} = \left[\hat{Z}_l\right]_{l=1}^{y} = H[\psi(R_1, \ldots, R_q)], \text{where } \hat{Z}_l \in Z \text{ and } y \leq x \tag{7.2}$$

Rather than directly using sequences, model H processes the outcome of a transformation function, $\psi()$. The function $\psi()$ signifies feature extraction in our pipeline. By processing the sEMG and IMU raw sequences, we extract various pivotal features, converting these sequences into a set of d-dimensional vectors, $\psi(R_1, \ldots, R_q) \in \mathbb{R}^d$.

A new meta-heuristic feature selection approach is adopted to improve this extracted feature set. The meta-heuristic algorithm iteratively improves candidate solutions by focusing on a given measure of quality. The general steps of the meta-heuristic feature selection algorithm can be outlined as follows, and the optimized feature subset can be denoted as:

$$\text{Selected\_Features} = \phi_{\text{MetaHeuristic}}(\varphi(\mathcal{S}_1, \ldots, \mathcal{S}_j)) \tag{7.3}$$

Algorithm 14 Meta-Heuristic Feature Selection Algorithm

```
Initialize the population of solutions
Define the fitness function for feature selection
while stopping criterion not met do
    for each solution in the population do
        Evaluate the fitness of the solution
    end for
    Select the best-performing solutions
    Generate new solutions through variation operators (e.g., crossover, mutation)
    Replace the worst-performing solutions with new solutions
end while
return Selected_Features
```

Once armed with the optimal feature subset, these features are used to classify the handwriting character sequence to predict class labels. Let the true handwritten character sequence label (ground truth) be denoted as:

$$Z' = [Z'_l]_{l=1}^{y}, \text{where } Z'_l \in Z \tag{7.4}$$

The overarching goal of the HCR framework is to craft a classifier H that correlates the selected feature subset to the handwritten character class label predictions. The mission is to minimize the discrepancy between the predicted characters $\hat{Z}$ and the ground truth Z', typically illustrated by a loss function $L(H(\text{Selected\_Features}), Z')$. The proposed method is illustrated schematically in Figure 7.1. To validate the effectiveness of the proposed model, we formulated the following research questions(RQs):

1. RQ15:*How effective is MSSA in identifying the relevant feature subset for recognizing handwritten characters on whiteboards using sEMG and IMU data through feature fusion?* (Addressed in Section 7.4.1)

2. RQ16:*How does MSSA compare to other baseline Algorithms in terms of feature selection and classification accuracy when validated on benchmark datasets from the UCI repository?* (Addressed in Section 7.4.2)

3. RQ 17:*How effective is the Modified Salp Swarm Algorithm (MSSA) compared to the original other modified versions Salp Swarm Algorithm's capabilities in feature space exploration and exploitation ?*(Addressed in section 7.4.2.1)

7.1.2 Major Contributions of the Work

The key contributions of this research are summarized as follows:

- Developed an efficient pipeline for identifying and interpreting whiteboard-based handwritten characters.

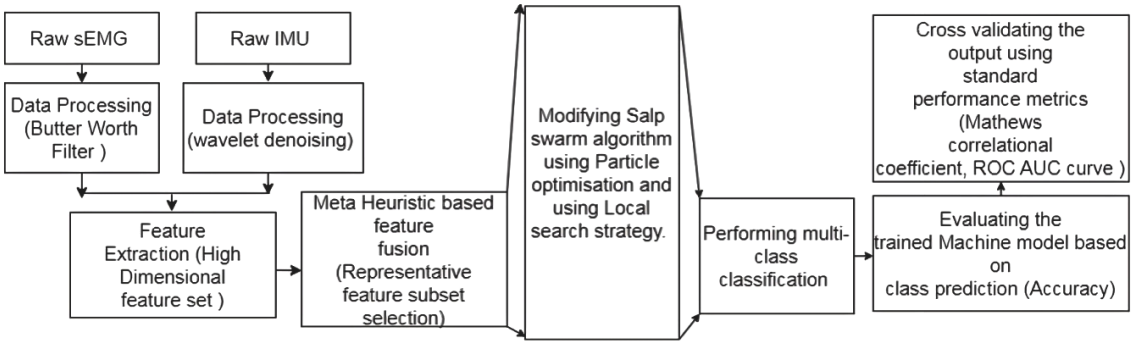

FIGURE 7.1: Schematic diagram of the pipeline used for handwritten characters recognition

- Introduced the Modified Salp Swarm Algorithm (MSSA) for enhanced feature selection, incorporating a weight factor to balance exploration and exploitation, Particle Swarm Optimization (PSO) for improved global search efficiency, and a Local Search Algorithm (LSA) for refined exploitation.
- Validated the MSSA's effectiveness using 14 benchmark datasets from the UCI repository, demonstrating superior performance in feature selection and classification accuracy compared to the original and other modified versions of the Salp Swarm Algorithm.
- Applied MSSA to identify the most relevant feature subset for recognizing handwritten characters on whiteboards, highlighting its practical utility in real-world applications.
- Performed multi-modal feature fusion using sEMG and IMU sensors to analyze the impact on handwritten character recognition.

7.2 Materials and methods

7.2.1 Data acquisition

In our research, we gathered sEMG (surface electromyography) and IMU (inertial measurement unit) signals corresponding to each of the 26 lowercase English

alphabets. The data was collected using the Myo armband by Thalamic Lab [164], an affordable wearable sensor. This device includes an eight-channel sEMG sensor with a 200 Hz sampling rate and an IMU that operates at 50 Hz [165]. The IMU is equipped with a triaxial accelerometer, a triaxial magnetometer, and a triaxial gyroscope, which are instrumental in collecting data related to hand movements.

The IMU component relies on the InvenSense MPU-9150 [164], a budget-friendly, low-energy module widely used for nine-axis motion tracking. The device's operations are controlled by the Freescale Kinetis ARM Cortex M4 120 MHz MK22FN1M MCU processor. Data transmission from the device is conducted via Bluetooth, using the BLE NRF51822 chip [164]

7.2.2 Dataset

For this analysis, we utilized the MM-HCR dataset, which comprises both EMG and IMU signals. Detailed descriptions of the dataset can be found in Section 2.3.9.

sEMG Data Representation: The sEMG signals were recorded using eight built-in sEMG sensors. The data points collected from these sensors over a fixed duration are treated as separate time series for each sensor. Let the time series for each sEMG sensor be denoted as $Ts_{emg_i}(v) = \{v_1, v_2, v_3, v_4, \ldots, v_p\}$, where p represents the number of samples collected from a particular sensor. Considering multiple r sEMG channels, the collection of raw sEMG signals can be represented as a multivariate time series:

$$MM - HCR_{semg} = \{Ts_{emg_1}(v), Ts_{emg_2}(v), Ts_{emg_3}(v), \ldots, Ts_{emg_r}(v)\}$$

IMU Data Representation: Similarly, the IMU sensors provide information about the orientation and motion of the subject's hands during writing. The

IMU data includes measurements such as accelerometer, gyroscope, and magnetometer readings. Let the time series for each IMU sensor be denoted as $Ts_{imu_i}(v) = \{v_1, v_2, v_3, \ldots, v_p\}$, where p represents the number of data points collected from a specific IMU sensor. Since the MyoArmband consists of $k = 13$ channels associated with IMU sensors, the collection of raw IMU signals can be represented as a multivariate time series:

$$MM - HCR_{imu} = \{Ts_{imu_1}(v), Ts_{imu_2}(v), Ts_{imu_3}(v), \ldots, Ts_{imu_k}(v)\}$$

7.3 Preliminaries concepts and proposed method

7.3.1 Salp Swarm Algorithm: An introduction

The Salp Swarm Algorithm (SSA) [285] is an optimization technique developed based on the behavior of Salps, a type of barrel-shaped planktonic tunicate that moves by contracting and pumping water through its jellied body. The SSA mimics the swarm behavior of Salps in nature called the salp chain, which is believed to aid the creatures in foraging and movement.

The SSA operates by dividing the population into a leader and follower group. The leader is the front salp of the chain, while the followers comprise the remaining salps. The algorithm determines the position of the salps in an n-dimensional search space, with n representing the problem's variables. The swarm's objective is to search for a food source, which serves as the optimization problem's target.

The SSA updates the leader's position using equation 7.5 that balances exploration and exploitation phases while taking into account the upper and lower bounds of the problem's search space.

$$x_1^j = \begin{cases} F_j + \frac{c_1(ub_j - lb_j)}{c_2 + lb_j}, & \text{if } c_3 \leq 0.5 \\ F_j - \frac{c_1(ub_j - lb_j)}{c_2 + lb_j}, & \text{otherwise} \end{cases} \tag{7.5}$$

In this algorithm, we define x_1^j as the position of the leader in the j-th dimension. The food source in this dimension is represented by F_j, and its range is defined by the upper bound ub_j and lower bound lb_j. Random values c_2 and c_3 are generated within the range of 0 to 1 to maintain the search space. Another crucial parameter is c_1, which plays a vital role in balancing the exploration and exploitation phases, and it is calculated as follows:

$$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2} \tag{7.6}$$

where, L is the maximum number of iterations and l is the number of current iteration.

The followers' positions are then updated using the given equation that considers the leader's position.

$$x_i^j = \frac{1}{2}\left(x_i^j + x_{i-1}^j\right) \tag{7.7}$$

x_i^j is the i-th follower position within j-th dimension and i is greater than 1.

7.3.2 Salp Swarm Algorithm for Feature Selection

7.3.2.1 Solution Representation

When working on feature selection problems, solutions are restricted to binary values. To apply the SSA algorithm to this problem, it is necessary to create a binary version. This study defines a solution as a one-dimensional vector, where the length of the vector corresponds to the number of features in the original dataset.

Each cell in the vector contains either a "1" or a "0". If the value is "1", it indicates that the corresponding feature is selected; otherwise, the value is "0" [288, 289].

The following equation is used to map continuous values to binary -

$$z_i^j = \begin{cases} 0, & \text{if } x_i^j \leq 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{7.8}$$

7.3.2.2 Fitness Function

The process of selecting features can be seen as a problem of optimization that involves two objectives with competing goals: achieving maximum classification accuracy while minimizing the number of features selected. The best solution would be one that has a high accuracy rate with a small number of features selected [288].

To balance the need for a minimum number of selected features and maximum classification accuracy, the following equation can be utilized -

$$F_{\text{itness}} = \rho \times \frac{\text{err}(d)}{d} + \Psi \times \frac{|F|}{|T|} \tag{7.9}$$

where,

- $\text{err}(d)$ is the classification error rate of the subset
- ρ and Ψ are constants with ρ in [0,1] and $\Psi = 1 - \rho$
- $|F|$ is the size of subset and $|T|$ is the total number of features

7.3.3 Modified Salp Swarm Algorithm

During our analysis, we identified a certain scope of improvement in the basic Salp Swarm Algorithm (SSA) that affects its exploration and exploitation capabilities. These limitations include less sufficient exploration and average exploitation

Algorithm 15 Salp Swarm Algorithm

Initialize the salp positions $\vec{x}_i$ $(i = 1, 2, ..., n)$ considering ub and lb
Initialize $l = 0$
while $l <$ max iterations **do**
    Calculate the fitness of each search agent (salp)
    $F =$ the global best search agent
    Update c_1 using Eq.7.6
    for each salp $(\vec{x}_i)$ **do**
        if $\vec{x}_i$ is the leading salp **then**
            Update the position of the leading salp using Eq.7.5
        else
            Update the position of the follower salp using Eq. 7.7
        end if
        Amend the salps based on the upper and lower bounds of variables
    end for
    $l = l + 1$
end while
return F

power.

One of the main factors contributing to this behavior is that the basic SSA relies solely on a dynamic parameter, c_1, to control exploration. This parameter primarily affects the position of the leader within the swarm. As a result, the exploration ability of the algorithm is limited, which may lead to less effective exploitability of the search space.

Furthermore, the position update mechanism for followers in the basic SSA is based solely on their current position and the position of their first neighbor. This approach narrows the exploitation power of the algorithm and may facilitate the likelihood of getting trapped in local optima.

Considering the above observations as a scope for improvement and to facilitate the balance between exploration and exploitation, the proposed modified algorithm introduced new parameters and hybridized it with the Particle Swarm Optimization (PSO) technique. This integration aimed to enhance exploration by incorporating PSO's global search capability while still leveraging SSA's local search capabilities.

By doing so, the modified algorithm seeks to overcome the limitations of the basic SSA and achieve a better balance between exploration and exploitation. The proposed modifications aim to provide better navigation in the search space, discover relevant features, and improve overall performance.

7.3.3.1 Including a Weight Factor

We introduce a weight factor w in the equation of leader position update to have a better balance between exploration and exploitation.

The updated equation for the leader salp is given as follows:

$$x_j^1 = \begin{cases} wF_j + c_1 \frac{ub_j - lb_j}{c_2 + lb_j}, & \text{if } c_3 \geq 0.5 \\ wF_j - c_1 \frac{ub_j - lb_j}{c_2 + lb_j}, & \text{otherwise.} \end{cases} \tag{7.10}$$

The following equations are used to update w:

$$w_1 = e^{-0.5} \left(1 - \frac{l}{L}\right)^2 \tag{7.11}$$

$$w_2 = 0.883 + 0.316 e^{-\left(\frac{l}{L}\right)^2} \tag{7.12}$$

$$w = \begin{cases} w_1, & \text{if rand}() \leq 0.5 \\ w_2, & \text{otherwise.} \end{cases} \tag{7.13}$$

From the graph in Figure 7.2, we can observe that as the number of iterations increases, w_1 increases from around 0.6 to 1 and w_2 decreases from around 1.2 to 1. We multiply w with the food source to increase the possibility of exploration. Choosing w_1 implies exploring towards a smaller feature set size, and choosing w_2 implies exploring towards a bigger feature set size.

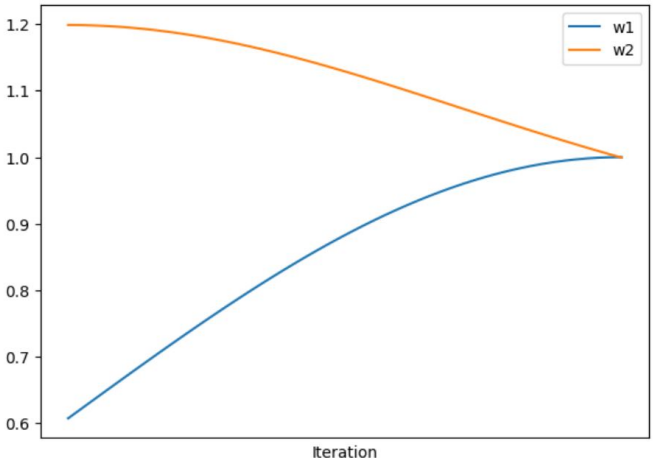

FIGURE 7.2: Plot of w1 and w2 with iterations.

As the number of iterations increases, w_1 and w_2 move towards 1. At $l = L$, $w_1 = w_2 = 1$. Thus, as the number of iterations increases, the salps search closer to the food source, thus increasing exploitation.

Therefore, it is assumed that the equation provides a balance between exploration and exploitation. In the initial iterations, the search space is explored more, and as the number of iterations increases, exploitation becomes more dominant.

7.3.3.2 Hybridizing with Particle Swarm Optimization

We include the velocity term of the Particle Swarm Optimization algorithm [290] in the position update equation of follower salps. We choose between the original and modified equations using a parameter c_4. The modified equations are given as:

$$x_j^i = \begin{cases} \frac{1}{2}(x_j^i + x_j^{i-1}), & \text{if rand}() \leq c_4 \\ \frac{1}{2}(x_j^i + x_j^{i-1} + v_j^i), & \text{if rand}() > c_4 \end{cases} \tag{7.14}$$

The original equation updates the position of the salp only based on its previous position and its neighbor's position; it does not depend on the food.

$$c_4 = e^{-\left(\frac{l}{L}\right)^2} \tag{7.15}$$

$$v_j^i = 0.9v_j^i + 2r_1(x_j^* - x_j^i) + 2r_2(F_j - x_j^i) \tag{7.16}$$

Since the velocity term depends on the food and the local optimum, the addition of the velocity term helps in searching towards the food and the local optimum, thereby exploiting the space. The c_4 parameter decreases from 1 to around 0.36. Thus, initially, there is a higher probability of choosing the original equation leading to exploration, and the probability of choosing the modified equation over the original one increases as the number of iterations increases, thereby increasing the possibility of exploitation.

7.3.3.3 Local Search Algorithm

To enhance the exploitation, we use the below approach at the end of each iteration of MSSA.

Firstly, we find the food/global optimum solution and then perform the following steps: It consists of the following 2 stages -

1. From the selected features, eliminate one feature at a time with a probability of Mp (= 5%) and calculate the fitness. If found better, replace it with the food.

2. From the unselected features, add one feature at a time with a probability of Mp (= 5%) and calculate the fitness. If found better, replace it with the food.

The first stage checks if removing one feature can increase fitness or not. While the second stage checks if adding one feature can increase the fitness or not. Thus, we check for possibilities near the global optimum to find if a better solution exists.

Figure 7.3 illustrates the operation of our proposed MSSA algorithm, incorporating the modifications discussed.

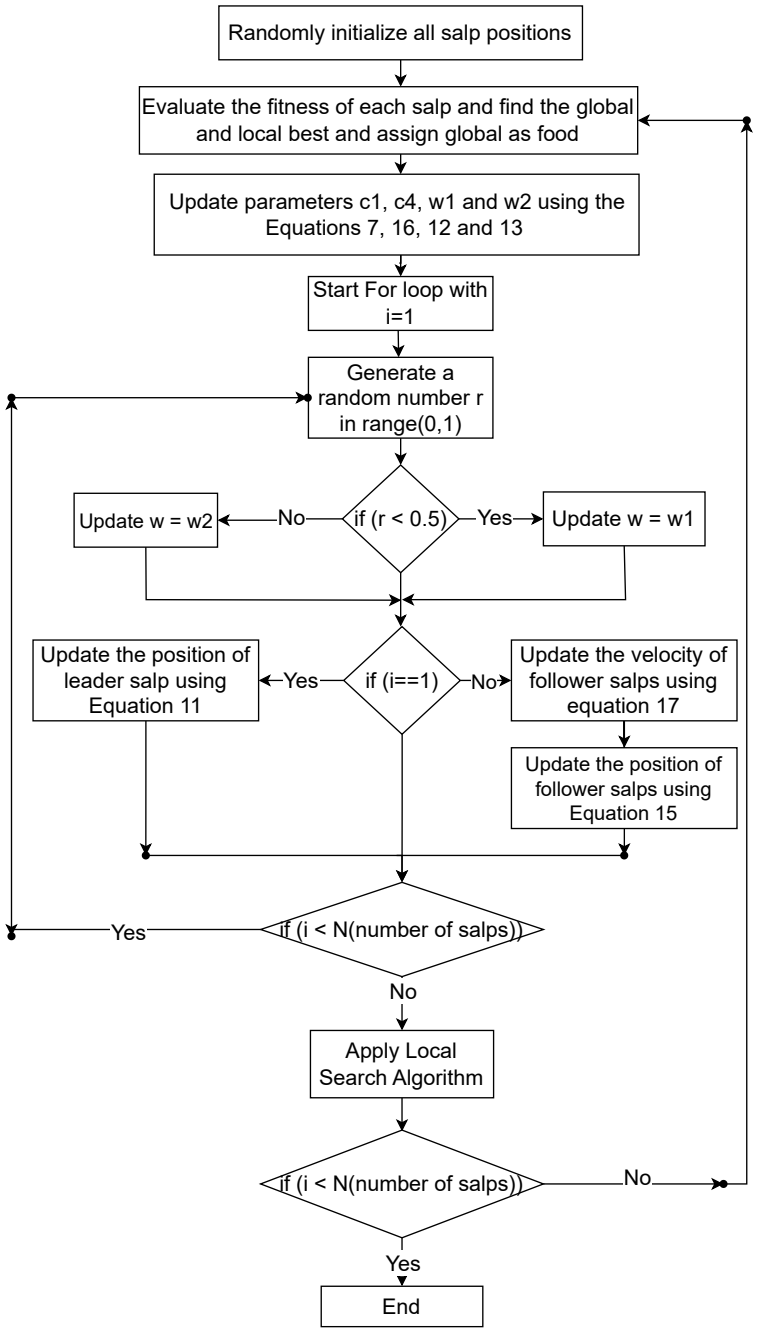

FIGURE 7.3: Workflow of the proposed MSSA feature selection algorithm

Algorithm 16 Local Search Algorithm

Calculate the fitness of each salp
F = the global best search agent
fitness = fitness of F
Define vector one_positions that store location of selected features in F
Define $X_{\text{mut1}} = F$
for each i in one_positions **do**
    Generate a random number r in range(0,1)
    if $r < M_p$ **then**
        $X_{\text{mut1}}[i] = 0$
        fitness_mut = fitness of X_{mut1}
        if fitness_mut < fitness **then**
            fitness = fitness_mut
            $F = X_{\text{mut1}}$
        end if
        $X_{\text{mut1}}[i] = 1$
    end if
end for
Define vector zero_positions that store location of unselected features in F
Define $X_{\text{mut2}} = F$
for each i in zero_positions **do**
    Generate a random number r in range(0,1)
    if $r < M_p$ **then**
        $X_{\text{mut2}}[i] = 1$
        fitness_mut = fitness of X_{mut2}
        if fitness_mut < fitness **then**
            fitness = fitness_mut
            $F = X_{\text{mut2}}$
        end if
        $X_{\text{mut2}}[i] = 0$
    end if
end for
return F

7.4 Results and Discussion

We assessed the effectiveness of our developed pipeline for various tasks using our collected and public UCI datasets. The primary goals of our experiments were as follows:

Algorithm 17 Modified Salp Swarm Algorithm (MSSA)

Initialize the salp positions $\vec{x_i}$ (i = 1, 2, ..., n) considering ub and lb
Initialize the salp velocity $\vec{v_i}$ (i = 1, 2, ..., n) considering bounds
Initialize $l = 0$
while $l <$ max iterations **do**
    Calculate the fitness of each search agent (salp)
    F = the global best search agent
    x^* = the local best search agent
    Update c_1 and c_4 using Eq.7.7 and Eq.7.6 respectively
    Update w_1 and w_2 using Eq.7.11 and Eq. 7.12 respectively
    for each salp $\vec{x_i}$ **do**
        Generate a random number r in range(0,1)
        if $r < 0.5$ **then**
            $w = w_1$
        else
            $w = w_2$
        end if
        if $\vec{x_i}$ is the leading salp **then**
            Update the position of the leading salp using Eq.7.10
        else
            Update the velocity of the follower salp using Eq. 7.16
            Amend the velocity based on the bounds of the variable
            Update the position of the follower salp using Eq. 7.14
        end if
        Amend the salps based on the upper and lower bounds of variables
    end for
    Apply LSA on salps
    $l = l + 1$
end while
return F

- To evaluate and analyze the performance of our metaheuristic-based feature selection specifically for Handwritten Character Recognition (HCR) tasks.
- To compare the performance of our proposed pipeline with state-of-the-art methods for Electromyography (EMG)-based Handwritten Character Recognition.
- To assess the effectiveness of our Modified Salp Swarm Algorithm (MSSA) as a feature selection tool on a benchmark dataset, namely the 14 UCI datasets.

- To evaluate the performance of our MSSA in feature selection in comparison with other proposed modifications of the Salp Swarm Algorithm (SSA).

  We approached the task of handwritten character recognition as a multi-class classification. To assess the performance of our experiments, we employed statistical measures, including Accuracy, Precision (P), Recall (R), and F1 Score (F1).

7.4.0.1 Experimental Setup

During experimentation, we employed a k-nearest neighbors classifier with k set to 5 as part of the fitness function. To evaluate the model's effectiveness and prevent overfitting, we used a 5-fold cross-validation approach. In each iteration, we split the data so that 20% was used for testing and the remaining 80% for training, rotating the test set in each iteration to ensure all instances were tested once. The datasets were normalized to the range of 0 to 1 using MinMaxScaler.

For the Modified Salp Swarm Algorithm (MSSA), we configured it with 10 search agents (salps), set the number of iterations to 100, and ran it 20 times. The ρ was set to 0.99, and the bounds were set between 0 and 1. We compared this modified feature selection approach with four other baseline metaheuristic algorithms: the Original Salp Swarm Algorithm [285], Genetic Algorithm [287], Particle Swarm Optimization (PSO) [290], and Bat Algorithm [291].

The settings for PSO included acceleration constants C1 and C2 both set at 2, and a weight (w) of 0.9. For the Genetic Algorithm, we used a crossover ratio of 0.5 and a mutation ratio of 0.02. The Bat Algorithm settings involved a maximum frequency of 2, a minimum frequency of 0, a maximum loudness of 2, a pulse rate, and constants α and γ both set at 0.9.

Each algorithm was executed 20 times on every dataset, and we used the average results of these runs for comparison purposes.

7.4.1 Multi-modal sensors feature fusion for handwritten character recognition

We employed an early feature fusion approach to investigate the combined impact of EMG and IMU sensors on handwritten character recognition tasks. The strategy begins by separately extracting features from each modality, i.e., EMG and IMU signals. Following the individual analysis of these EMG and IMU data streams, the extracted features from each are then amalgamated into a single, cohesive representation. This approach mirrors the process described in Snoek et al. [292], which suggests the integration of the distinct feature vectors from each modality through concatenation to form a comprehensive combined vector. Initially, the individual datasets of EMG and IMU, each comprising 750 features, were merged, creating a unified dataset with a total of 1,500 features. This fusion strategy aims to leverage the complementary information provided by both sensor types before any feature selection or classification process.

The analysis of the EMG dataset demonstrated that the feature selection process effectively reduced the dimensionality from 750 to an average of 139.1 features across 20 runs, achieving an average classification accuracy of 93.65%. In the best run, 62 features were selected, leading to an accuracy of 94.36%. This highlights the EMG data's effectiveness in capturing essential characteristics for handwriting recognition.

For the IMU dataset, the feature selection process similarly reduced the feature set to an average of 137.45 features, with an average accuracy of 87.7% across 20 runs. The best subset consisted of 42 features, yielding an accuracy of 90.48%. These results underscore the IMU's contribution to capturing dynamic motion information, although with slightly lower accuracy compared to the EMG dataset.

Applying the MSSA to this integrated feature vector (EMG+IMU), we aimed to identify the most effective subset of features. The early fusion of EMG and IMU datasets proved to be highly synergistic. The MSSA successfully condensed the

combined feature set from 1,500 to an average of 278.2 features, substantially increasing the average accuracy to 98.92%. More impressively, the best feature subset, comprising 196 features from the fused dataset, achieved a remarkable accuracy of 99.19% as compared to sEMG and IMU individual datasets.

These results underscore the effectiveness of early feature fusion in enhancing classification performance. By integrating the diverse characteristics of muscle activity and motion sensors at the outset, the model benefits from a richer, more informative feature space, leading to significantly improved accuracy in the classification task. Table 7.1 highlights the performance of the above scenario in detail.

TABLE 7.1: Comparison of Feature Selection and Accuracy

Dataset	Average features size (20 runs)	Average Accuracy (20 runs)	Features selected (best run)	Accuracy with selected feature subset (best run)
EMG	139.1	93.65%	62	94.36%
IMU	137.45	87.7%	42	90.48%
EMG+IMU	278.2	98.92%	196	99.19%

TABLE 7.2: Publicly available dataset used for the experiments

Dataset	**Attributes**	**Classes**	**Instances**
Chess [169]	36	2	3196
Tic-tac-toe [170]	9	2	958
Libras [171]	91	15	360
CNAE [172]	857	9	1080
Ionosphere [173]	34	2	351
Car [174]	6	4	1728
Wine [175]	13	3	178
German [176]	20	2	1000
Zoo [177]	16	7	101
Promoter [178]	58	2	106
Parkinsons [179]	22	2	197
Sonar [180]	60	2	208
Splice [181]	61	3	3190
Turkish [182]	50	3	400

The combination of EMG and IMU datasets, as evidenced by the experimental results, offers a compelling advantage in character recognition systems. The combined dataset not only outperforms the individual EMG and IMU datasets in terms of accuracy but also demonstrates the synergistic potential of integrating diverse data sources. By leveraging the strengths of both EMG and IMU sensors, the combined approach captures a more comprehensive range of motion and muscle activity,

resulting in enhanced recognition capabilities. This integration is crucial for applications requiring high precision and reliability, making the EMG+IMU combination a superior choice for whiteboard-based handwritten character recognition systems. Other insightful observations are discussed as follows.

7.4.1.1 Visualization of MSSA-Selected Features using t-distribution Stochastic Neighbor Embedding (t-SNE)

To better understand and visualize the features chosen by the Modified Salp Swarm Algorithm (MSSA), we employed t-distribution Stochastic Neighbor Embedding (t-SNE). t-SNE can effectively map high-dimensional data into a 2-dimensional space using nonlinear dimensionality reduction capabilities. By doing so, it attempts to preserve both local and global patterns from the original space, representing them in reduced dimensions. Moreover, It ensures proximity between points in the high-dimensional space is maintained in the lower-dimensional visualization. Figure 7.5 demonstrates the 42 IMU features subset selected by using MSSA for $MM-HCR_{IMU}$ dataset, while Figure 7.4 demonstrates the 62 features selected using MSSA for $MM-HCR_{sEMG}$ plotted in 2-D using t-SNE. Each of the 26 classes is differentiated using unique colors in the t-SNE plots, with the algorithm employing specific symbols to denote different class data points. The visualization was achieved with hyper-parameters 'max-iteration' and 'perplexity' set to 1000 and 30, respectively. Focusing on Figure 7.4, the presented plot reveals non-overlapping distributions for different classes, each representing unique handwritten characters. The t-SNE algorithm, while transitioning from N-D to 2-D feature space, maintains the proximity of data points that are close in high dimensions within the new lower-dimensional mapping. Therefore, the non-overlapping clusters in shared feature spaces indicate class separability.

Moreover, these minimally overlapping clusters denote the discriminative ability of the selected feature subset, assisting machine learning classifiers in improving their

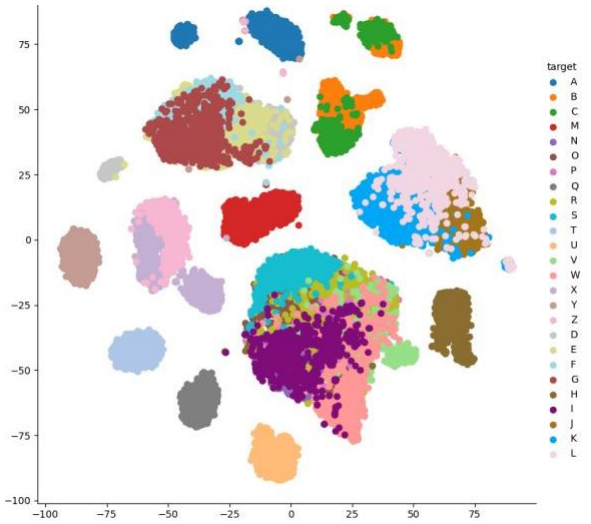

FIGURE 7.4: t-SNE plot showing the feature distribution learned for dataset $MM - HCR_{EMG}$

performance on classification tasks. A similar inference can be made for Figure 7.5 for selected feature subset by MSSA for HCR_{IMU} dataset.

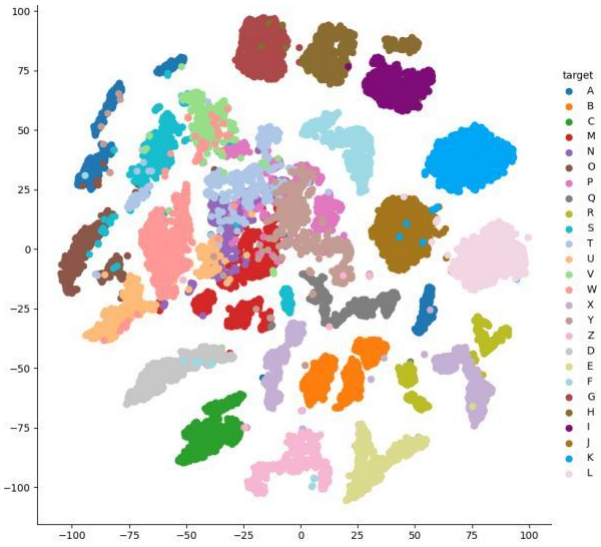

FIGURE 7.5: t-SNE plot showing the feature distribution learned for the dataset. $MM - HCR_{IMU}$

7.4.2 Evaluating the MSSA performance on publicly available UCI dataset

To evaluate the effectiveness of the proposed MSSA approach, we employed a set of 14 benchmark datasets sourced from the UCI Machine Learning Repository. To identify the optimal feature subset, we incorporated a k-nearest neighbors classifier with k=5 into the fitness function of the wrapper approach. The dataset was divided into training and test sets using a 5-fold cross-validation method, starting with 80% for training and 20% for testing in the initial iteration. The training set's purpose was to establish an optimal feature set, while the test set helped evaluate classification accuracy and the overall effectiveness of the method. The chosen datasets were diverse, with attribute numbers ranging from 9 to 857, making them particularly suitable for feature selection due to their extensive attribute range. These datasets also varied in terms of classes, from 2 to 15, and instances, from 101 to 3196. All experiments were conducted on a PC equipped with an Intel(R) Core(TM) i7-7700 CPU at 3.60 GHz and 16 GB of RAM.

Mainly, for assessing the Modified Salp Swarm Algorithm (MSSA), we used the following performance metrics

- Classification Accuracy: This is calculated as the average accuracy across 20 runs, using the features selected by MSSA on the test dataset.
- Fitness Value: These values are measurements indicating the performance in terms of classification accuracy and selected feature subset size as described in Equation 7.9.
- Feature Selection Size: This represents the average number of features MSSA selects in each run.
- Execution Time: This metric measures the average time in seconds for MSSA to complete a 20-run for a given optimization task.

The classification accuracy achieved using MSSA on 14 UCI datasets (details summarized in Table 7.2) was compared with the baseline metaheuristics optimization algorithm used for feature selection (Genetic Algorithm(GA) [287], Particle swarm optimization(PSO) [290], Bat algorithm(BA) [291], and original salp swarm algorithm(SSA) [285]. Table 7.3, presents the classification accuracy on 14 UCI data sets. In 12 out of 14 UCI datasets, our proposed MSSA algorithm outperforms the other baseline and original SSA algorithms except for the CAR [174] and Tic-tac-toe [170]. The average accuracy of all the algorithms turned out to be MSSA 84.60%, SSA 81.20%, PSO 83.31%, GA 81.10%, and BA 82.44%. The average accuracy rates show that MSSA performs well on overall datasets and maintains a high level of consistency across diverse datasets, underscoring its robustness as a feature selection method.

A similar observation was made while considering the Average Feature selection size. For instance, in the Chess dataset [169] with 36 features, MSSA impressively reduces the feature subset to 14.5, surpassing SSA 16.8 and GA 16.9. Similarly, in the Libras Movement [171] and Ionosphere dataset [173], with 91 and 34 features, respectively, MSSA significantly trims down the feature subsets to 37.45 and 8.3, outperforming other algorithms where the feature selection exceeds 40 in some cases. This trend of efficient feature selection by MSSA is also observed in the Splice dataset [181], where it selects only 9 features compared to the notably higher numbers by PSO and GA. In general, out of 14 datasets, MSSA results in 8 best feature subsets. This demonstrates MSSA's capability to select smaller feature subsets but also ensures their relevance and effectiveness for classification tasks. Consequently, MSSA stands out as a robust, potential tool for feature selection, balancing accuracy with computational efficiency. Table 7.4 presents the average feature subset size selected using MSSA on the benchmark 14 UCI dataset.

In another analysis, we consider the Execution time for the MSSA. The table 7.5 shows the average execution time across all datasets, where MSSA leads with an average of 71.53 seconds. This is compared to SSA (76.88 seconds), PSO (73.98

seconds), GA (81.77 seconds), and BA (81.93 seconds), indicating that MSSA, on average, is the most time-efficient algorithm among those compared.

Considering the Average Fitness value, MSSA consistently records the lowest fitness values in several datasets, indicating its superior performance. For instance, in the Chess dataset [169] with 36 features, MSSA's fitness value is 0.077, which is notably lower than SSA (0.116) and GA (0.116). Similarly, in the CNAE dataset [172] with 857 features, MSSA again leads with a fitness value of 0.132, significantly outperforming SSA (0.177) and GA (0.184). Overall, out of 14 benchmark datasets, MSSA holds better results in 12 datasets.

TABLE 7.3: Performance comparison(Classification Accuracy) of MSSA with other methods on benchmark datasets

Data	Features	MSSA	SSA	PSO	GA	BA
[169]	36	**92.57%**	88.65%	90.59%	88.73%	91.04%
[170]	9	76.18%	75.21%	76.31%	76.24%	77.03%
[171]	91	**73.26%**	71.12%	73.15%	70.87%	71.36%
[172]	857	**87.19%**	82.63%	86.44%	81.86%	83.42%
[173]	34	**90.61%**	88.26%	90.014%	88.16%	89.35%
[174]	6	81.12%	81.12%	80.85%	81.20%	81.25%
[175]	13	**98.67%**	97.86%	98.14%	98%	98.62%
[176]	20	**76.74%**	75.91%	76.59%	75.85%	76.65%
[177]	16	**97.03%**	95.99%	96.28%	96.14%	96.83%
[178]	58	**90.94%**	87.26%	90.71%	86.56%	87.64%
[179]	22	**87.89%**	86.30%	87.35%	86.87%	87.82%
[180]	60	**75.36%**	67.74%	74.27%	67.21%	71.05%
[181]	61	**82.38%**	68.48%	72.09%	68.71%	70.57%
[182]	50	**74.41%**	70.31%	73.5%	69.05%	71.56%

The comparison of MSSA with standard meta-heuristic algorithms offers a thorough insight into its effectiveness in areas such as Feature subset selection, Average accuracy, Execution time, and Fitness functions. Preliminary findings from Table 7.3, Table 7.5, Table 7.4, and Table 7.6 indicate enhanced performance compared to the original SSA. Furthermore, the results demonstrate that MSSA is adequately efficient to serve as a tool for feature selection.

TABLE 7.4: Performance comparison (Average Feature Selection Size) of MSSA with other methods on benchmark datasets

Dataset	Features	MSSA	SSA	PSO	GA	BA
[169]	36	**14.5**	16.8	15.45	16.9	16.45
[170]	9	5.4	5.35	5.6	5.45	5
[171]	91	**37.45**	43.5	40.55	44	40.9
[172]	857	504.2	428.55	423.8	431.55	436.55
[173]	34	**8.3**	13.65	10.7	13.75	11.25
[174]	6	**4**	4.0	4.2	4.0	4.0
[175]	13	6.7	7.05	7.2	7.75	6.5
[176]	20	11.8	12.5	11.8	12	11.3
[177]	16	9.35	9.25	8.9	9.75	8.5
[178]	58	**25.6**	27.95	27.1	27.8	29.5
[179]	22	**7.6**	8.85	8.3	8.55	8.5
[180]	60	**18.35**	25.35	24.25	26.85	25.7
[181]	61	**9**	26.2	22.15	27.75	22.35
[182]	50	21.45	23.7	23.4	24.1	24.3

TABLE 7.5: Performance comparison (Average Execution Time) of MSSA with other methods on benchmark datasets

Dataset	Features	MSSA	SSA	PSO	GA	BA
[169]	36	190.01	195.82	184.3	214.36	213.08
[170]	9	42.37	40.61	42.18	46.09	44.24
[171]	91	22.74	22.49	22.38	22.74	23.16
[172]	857	189.64	160.51	159.52	191.68	192.22
[173]	34	21.64	23.4	20.98	30.71	31.22
[174]	6	75.32	72.68	70.59	66.82	66.54
[175]	13	11.84	11.58	11.74	11.84	11.85
[176]	20	58.44	57.32	61.27	56.41	56.48
[177]	16	13.16	13.01	13.07	13.03	13.19
[178]	58	27.16	26.44	27.18	26.48	28.45
[179]	22	17.79	18.16	17.34	18.94	18.29
[180]	60	37.33	37.84	37.73	37.67	38.57
[181]	61	307.56	416.45	384.76	433.89	432.45
[182]	50	43.14	41.96	41.92	41.29	44.38
Average	–	**71.53**	76.88	73.98	81.77	81.93

7.4.2.1 Comparison with proposed MSSA with others published SSA modifications

In recent years, SSA modifications have been proposed for optimization and feature selection. As our MMSSA is curated for feature selection, we have compared our work with similar approaches tuned for feature selection. In such a work, Aljarah et al. [293] proposed a modified SSA with asynchronous updating rules and a new leadership structure inspired by termite colony behaviors. The SSA is divided into sub-chains, each with unique updating strategies. The findings demonstrate that having half of the salps act as leaders significantly boosts SSA's accuracy. Similarly, Hegazy et al. [288] proposed a binary version of SSA for feature selection where the authors used a control parameter that adjusts the best solution during exploration

TABLE 7.6: Performance comparison (Fitness Values on 20 runs) of MSSA with other methods on benchmark datasets

Dataset	Features	MSSA	SSA	PSO	GA	BA
[169]	36	**0.077**	0.116	0.097	0.116	0.093
[170]	9	0.24	0.251	0.24	0.24	0.233
[171]	91	**0.269**	0.291	0.270	0.293	0.288
[172]	857	**0.132**	0.177	0.139	0.184	0.169
[173]	34	**0.095**	0.120	0.102	0.121	0.108
[174]	6	0.194	0.194	0.197	0.192	0.192
[175]	13	**0.018**	0.026	0.024	0.025	0.019
[176]	20	**0.236**	0.245	0.238	0.245	0.237
[177]	16	**0.035**	0.045	0.042	0.043	0.036
[178]	58	**0.094**	0.131	0.097	0.138	0.128
[179]	22	**0.123**	0.139	0.129	0.133	0.124
[180]	60	**0.247**	0.323	0.258	0.329	0.290
[181]	61	**0.176**	0.316	0.280	0.314	0.295
[182]	50	**0.257**	0.298	0.267	0.311	0.286

and exploitation. They compared the results of modification with the basic SSA and four other baseline swarm methods and claimed to achieve better results. In another work, Ahmed et al. [294] proposed a modified SSA that uses a Chatoic theory concept to deal with random numbers. The authors suggest that using chaotic maps instead of random numbers can improve the efficiency of SSA algorithmic terms for feature selection. Faris et al. [295] propose another SSA modification where they employed eight transfer functions and converted continuous SSA to a binary version. Moreover, they use the crossover operator to enhance exploration.

Ibrahi et al. [296] merge the Salp Swarm Algorithm (SSA) with Particle Swarm Optimization, enhancing the efficiency of both exploration and exploitation steps. The authors validated their modification for optimization and feature selection. The performance of SSAPSO is validated through two experimental series: firstly, by comparing it with similar methods using benchmark functions, and secondly, by applying it to various UCI datasets to identify the most effective feature sets.

Sayed et al. [297] introduce a hybrid solution, the Chaotic Salp Swarm Algorithm (CSSA), which combines SSA with chaos theory to overcome stagnation in local optima and low convergence rates issues. Their results indicate CSSA's effectiveness in optimally selecting features for maximum classification accuracy with minimal feature count, highlighting the logistic chaotic map as particularly enhancing SSA's

performance.

Sawm-based algorithms are likely to get stagnant in local optima and have a scope of improvement in convergence rates [297]. SSA, being a swarm-based algorithm, is also likely to have a similar issue. To deal with limitation, Hegazy et al. [288] introduced an inertia weight parameter that affects the food source variable of the SSA algorithm in order to enhance algorithm exploration and exploitation. However, they used a constant factor of value 0.7. Inspired by their approach, we introduce an adaptive controlled parameter that facilitates the convergence speed in the search process. Additionally, it tries to strike a balance between exploiting and exploring capabilities, enabling it to effectively escape many local solutions in feature selection tasks and gain a more precise understanding of the optimal solution. Moreover, we addressed the exploration capability by hybridizing it PSO. The motive was to exploit PSO's global search ability while still leveraging SSA's local search capabilities. The velocity term depends on the food and the local optimum, thus enhancing the exploitation of the search space.

At the end of each iteration, the MSSA applies a local search algorithm to further enhance exploitation. This involves evaluating the possibility of increasing fitness by either removing or adding features one at a time, thereby checking for potential improvements near the global optimum. The local search was inspired by the modification of grey wolf optimization [298]. This structured approach ensures a more efficient exploration and exploitation balance, aiming for optimal feature selection.

To compare our work with other proposed SSA modifications, we extracted the results from the common dataset. Table 7.7 highlights the accuracy achieved by MSSA and other SSA modifications on the common UCI datasets. The MSSA demonstrated comparable or superior performance across a range of common UCI datasets. However, the analysis is limited by the lack of information about data preprocessing, parameter tuning, and the experimental setup. These factors are

crucial in influencing the overall effectiveness of a model. This absence of detailed procedural information might limit the depth of our comparative analysis.

TABLE 7.7: The accuracy achieved on UCI data-set through different SSA modifications

Dataset	MSSA	[288]	[293]	[294]	[295]	[297]
Chess [169]	**97.69%**	-	-	-	-	84.63%
Tic-tac-toe [170]	**83.59%**	-	-	-	78.71%	-
CNAE [172]	**90.02%**	89.12%	93.77%	-	-	-
Ionosphere [173]	**95.84%**	85.30%	99.77%	-	90.28%	92.78%
Wine [175]	**99.86%**	97.8%	-	54%	99.18%	-
German [176]	**79.23%**	-	-	-	-	76.39%
Zoo [177]	**99.52%**	87.20%	99.28%	-	93.40%	88.15%
Parkinson [179]	**98.84%**	85.29%	-	-	-	89.82%
Sonar [180]	96.9%	73.4%	94.8%	-	86.54%	**98.94%**

7.4.3 State-of-the-art comparison

Prior research in this field has shown the potential of using sEMG signals for classifying handwritten characters, yet there remains scope for further enhancement. Linderman et al. [14] demonstrated that sEMG signals generated during writing could be effectively converted into font characters, attaining a 97% classification accuracy. However, their research was limited to a small set of characters (digits 0–9). In another study, Huang et al. [84] employed Dynamic Time Wrapping (DTW) to classify ten digits, 10 Chinese characters, and 26 English alphabets. The data collected for English alphabets showed an 84.29% accuracy rate. A related DTW-based study [85] introduced a two-phase template matching method with an adjusted Mahalanobis distance, enhancing recognition accuracy to 92.42%. Moreover, Hernandez et al. [86] utilized a deep learning framework to recognize English alphabets, achieving a 94.85% accuracy rate. The deep learning approach automatically extracts features, resulting in models that offer less interpretability regarding the features utilized. Singh et al. [74] leveraged an unsupervised technique to extract the most representative feature subset for sEMG-based handwritten character classification. Using a stacked sparse denoising autoencoder, they reported a classification accuracy of 98.62% while classifying 26 lowercase English alphabets. Later et al. [299] in the field of Airwriting Recognition, Specifically the use of Surface Electromyography

(sEMG) signals for recognizing dynamic gestures like writing letters in the air. The work focuses on a multi-loss minimization framework for sEMG-based air writing recognition, aiming to learn a feature embedding vector that minimizes triplet loss while also learning the parameters of a classifier to recognize alphabets. The authors achieved an accuracy of 81.26% for English uppercase alphabets. However, writing characters in the air differs from writing on a whiteboard as no friction resistance is faced while writing in the air. Also, on the whiteboard, a fixed 2D plane has to be maintained, which is not a compulsion in Air writing. Similarly, Tripathi et al. [118] study explored multiple time-domain features to create EMG envelopes and two time-frequency image representations (short-time Fourier transform and continuous wavelet transform) as inputs for a deep learning model designed for air writing recognition. Using a 2-D convolutional neural network (CNN), the model achieved the highest accuracy of 78.50% in user-dependent scenarios. Most of the state-of-the-art methods, as discussed, focused on handwritten characters using pen-and-paper or pen with a digital screen. Less work has been performed to address the challenges of whiteboard writing. Our proposed work aims to achieve sufficient recognition accuracy for handwritten characters on whiteboard. Our methodology combines meta-heuristic feature selection with a baseline machine learning classifier, targeting English lowercase alphabets. It achieved a promising accuracy of 98.92%. Our approach not only addresses a less-explored area in the field but also offers comparable or superior performance to related existing state-of-the-art methods. Table 7.8 summarized the comparison of state-of-the-art approaches with our proposed method.

Our proposed work aims to address this gap by achieving high recognition accuracy for characters written on a whiteboard.

TABLE 7.8: Comparison with state-of-the-art approaches

Paper	Methodology	Classes	Sensor	Writing Approaches	Accuracy
Linderman et al. [14]	Template matching	(0–9 digits)	sEMG	Pen-Digitizing tablet	97.00%
Huang et al. [84]	Dynamic time warping	English uppercase alphabets	sEMG	Pen Paper	84.29%
Li et al. [300]	Dynamic time warping	English lowercase alphabets	sEMG	Pen Paper	92.42%
Beltran et al. [86]	Deep learning (LSTM+RNN)	English lowercase alphabets and (0-9) decimal digits	sEMG	Pen-Digital screen	94.85%
Singh et al [74]	SSDAE + SVM	English lowercase alphabets	sEMG	Pen-Paper	98.62%
Tripathi et al [299]	Multi-loss minimization framework	English uppercase alphabets	sEMG	Gestures in Air	81.26%
Tripathi et al [118]	2-D CNN	English uppercase alphabets	sEMG	Gestures in Air	78.50%
Proposed approach	Meta-heuristic feature selection + ML classifier	English lowercase alphabets	sEMG	Whiteboard	98.92%

7.4.4 Threats to Validity

While the proposed MSSA algorithm demonstrates strong performance across multiple datasets, several threats to validity should be considered. These are outlined below.

Internal Validity

MSSA's performance depends heavily on careful parameter tuning. Inconsistent or suboptimal configurations can lead to varied results within the same dataset, affecting internal reliability. This sensitivity also introduces computational overhead and may limit reproducibility in environments lacking domain expertise or automated tuning methods.

External Validity

Although MSSA performed well across diverse datasets, its effectiveness may not generalize to datasets with high noise levels, atypical distributions, or irregular structures. In addition, scalability to very large datasets remains uncertain. Further

evaluation on broader, more complex datasets is necessary to fully establish its generalizability.

Construct Validity

Swarm-based feature selection algorithms are designed to identify subsets of features that optimize a given objective, often based on statistical relevance or performance metrics. However, in many domains, true feature importance is influenced by contextual, semantic, or task-specific factors that such algorithms may overlook. As a result, the selected features may not fully represent the underlying concept being modeled, which can limit the interpretability and domain alignment of the results.

Conclusion Validity

The current study does not analyze key theoretical properties of MSSA, such as convergence guarantees or formal complexity bounds. Without these insights, it is difficult to assess the algorithm's stability and efficiency across varying conditions. This limits the strength of conclusions regarding long-term reliability and broader applicability.

Future Work: To address these limitations, future efforts should focus on enhancing MSSA's scalability through parallelization or integration with distributed computing frameworks. Additionally, developing adaptive or automated parameter tuning strategies could significantly improve the algorithm's robustness and user-friendliness. A comprehensive theoretical investigation into MSSA's behavior and computational efficiency will also be essential for advancing its development and broader adoption.

7.5 Summary

In this chapter, we enhanced sEMG-based dynamic hand gesture recognition by employing a Modified Salp Swarm Algorithm (MSSA) for feature selection and fusion. This approach specifically addressed the challenges associated with high-dimensional feature spaces and intensive computational demands inherent in sEMG-based gesture recognition.

The original Salp Swarm Algorithm (SSA) was adapted to improve exploration and exploitation capabilities, which were previously limited by issues like premature convergence. Our modifications, including the introduction of a weight factor, hybridization with Particle Swarm Optimization (PSO), and a Local Search Algorithm (LSA), significantly improved feature selection effectiveness and computational efficiency.

Our experimental validation on 14 UCI benchmark datasets demonstrated that the enhanced MSSA outperforms baseline techniques, achieving superior classification accuracy and reducing feature subset sizes effectively. The integration of IMU data with sEMG signals through multi-modal sensor fusion provided richer, more informative feature spaces, leading to improved accuracy in handwritten character recognition tasks.

The chapter concludes with a demonstration of the MSSA's robust performance in addressing the high-dimensional feature vectors and computational costs associated with handwriting gesture recognition. Moreover , Swarm-based or metaheuristic feature selection methods, such as SSA, are well-suited for resource- constrained environments due to their ability to identify compact and high-quality feature subsets. It also suggest that hybridizing these algorithms can enhance the balance between exploration and exploitation, leading to more efficient feature selection.

Future work should focus on refining the algorithm's scalability to other benchmark datasets, further enhancing its computational efficiency, and exploring adaptive parameter tuning to maximize performance across diverse applications. The integration of multi-modal data also contributed to our performance, making it a promising solution for applications in smart classrooms and clinical handwriting analysis.