Hyperparameter Optimization for Convolutional Neural Networks using the Salp Swarm Algorithm

Entesar H. Abdulsaed[1], Maytham Alabbas[1*], Raidah S. Khudeyer[2]
[1] Department of Computer Science, College of Computer Science and Information Technology, University of Basrah, Basrah, Iraq.
[2] Department of Computer Information Systems, College of Computer Science and Information Technology, University of Basrah, Basrah, Iraq.
Email: hopidhadha@gmail.com, ma@uobasrah.edu.iq, raidah.khudayer@uobasrah.edu.iq
[*] Correspondence

Keywords: deep learning, convolutional neural networks, salp swarm algorithm, hyperparameters optimization

Received: November 22, 2023

Convolutional neural networks (CNNs) have exceptionally performed across various computer vision tasks. However, their effectiveness depends heavily on the careful selection of hyperparameters. Optimizing these hyperparameters can be challenging and time-consuming, especially when working with large datasets and complex network architectures. In response, we propose a novel approach for hyperparameter optimization in CNNs using the Salp Swarm Algorithm (SSA). Based on the natural behavior of mollusks, SSA mimics the collective intelligence that governs feeding and navigation. Taking advantage of SSA's unique properties, our research thoroughly explores the hyperparameter space. This exploration aims to identify the algorithm that maximizes CNNs performance. This paper presents the architecture of the SSA-based framework for hyperparameter optimization and compares it to other established optimization techniques, such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). We also present experimental results using the MNIST and fashion MNIST datasets, achieving an impressive classification accuracy of 99.46% for MNIST and 94.53% for fashion-MNIST. This case study not only contributes to the fields of deep learning and hyperparameter optimization by demonstrating the effectiveness of SSA in optimizing CNNs, but it also provides benefits to researchers and practitioners who are looking for optimal hyperparameter configurations for CNNs in a variety of computer vision applications. We also evaluate the scalability and robustness of our proposed method in the context of different CNNs structures. The insights we gained highlight SSA's potential for addressing challenges related to hyperparameter optimization.

Povzetek: Članek predstavlja optimizacijo hiperparametrov v konvolucijskih nevronskih mrežah s pomočjo algoritma Salp Swarm, ki izboljša učinkovitost in natančnost.

1 Introduction

Deep learning has emerged as a powerful and versatile field within the broader domain of machine learning [1]. It has revolutionized various domains, such as computer vision, natural language processing, and speech recognition [2, 3]. One of the fundamental techniques used in deep learning is convolutional neural networks (CNNs), which have demonstrated exceptional performance in image recognition, object detection, and classification tasks.

CNNs are designed to process grid-like data, such as images, by capturing spatial and hierarchical relationships between different features. Their architecture consists of multiple layers, including convolutional (Conv.), pooling, and fully connected layers (FC layers), allowing them to automatically extract meaningful features from input data [4]. This inherent capability makes CNNs highly effective in analyzing visual data and extracting intricate patterns that may not be discernible to the human eye [5].

While CNNs offer numerous advantages, including their ability to handle large amounts of data, learn complex representations, and achieve state-of-the-art performance in various tasks, they also possess specific weaknesses [6]. One of the critical challenges in utilizing CNNs effectively is selecting appropriate hyperparameters [7]. Hyperparameters are the configuration settings that control the behavior and performance of a CNNs model, such as learning rate, batch size, dropout rate, and kernel size [8].

The optimal selection of hyperparameters significantly impacts CNN models' performance and convergence speed. However, choosing the right combination of hyperparameters is a challenging and time-consuming task. Traditional methods, such as grid search, random search, and Bayesian optimization, suffer from computational inefficiency and may not explore the entire hyperparameter space effectively. Therefore, there is a need for advanced techniques that can efficiently and effectively optimize hyperparameters for CNNs [9].

In recent years, metaheuristic optimization algorithms have gained significant attention in deep learning for hyperparameter optimization. One such algorithm is the salp swarm algorithm (SSA), inspired by the collective behavior of natural salp swarms. SSA is a population-based metaheuristic algorithm that mimics the social behavior of salps to search for the global optimum in a given search space. It has shown promising results in solving various optimization problems since its proposal, including feature selection, neural network training, clustering analysis, image processing, engineering optimization, and financial portfolio optimization.

The current work is a step forward in this regard. It focuses on investigating the effectiveness of SSA in optimizing hyperparameters for CNNs. We will conduct experiments using the MNIST dataset [10] and fashion MNIST [11] and compare the performance of SSA with other methods. SSA is chosen for its ability to approximate optimal solutions with satisfactory convergence rates and solution space coverage. Its straightforward mathematical framework makes it simpler to comprehend and implement [12]. The results obtained will provide valuable insights into the efficiency and effectiveness of SSA for hyperparameter optimization in CNNs and contribute to the ongoing research efforts in improving the performance of deep learning models.

This paper is structured as follows: Section 2 delves into prior research focused on enhancing CNNs. Moving to Section 3, we explore the components and tools integral to CNNs. Our proposed work is introduced in Section 4, while Section 5 is dedicated to presenting our experimental evaluation. The outcomes and analysis of our experiments are detailed in Section 6, and in Section 7, we conclude the paper while also addressing potential avenues for future research.

2 Related work

Many researchers have used different techniques to automatically set hyperparameters and choose the structure for CNNs. This is done to avoid the additional voltage and time required for manual network construction and to improve the performance of CNNs. In this section, we preview the state-of-the-art studies in this field and summarize them in Table 1.

Some researchers use particle swarm optimization (PSO) to improve the accuracy of CNNs. In [13], PSO was recommended for use in CNNs. To increase accuracy, PSO is used in the training phase to optimize the results of the solution vectors on CNNs.

In [14], CNNs were optimized using microcanonical annealing. As suggested by the authors, the performance of the original CNNs may be significantly improved using this proposed method.

In [15], a genetic algorithm (GA) was used to optimize multiple parameters of CNNs at once. Various types and ranges of GA parameters were also used. After a lengthy process, an approximation to the global optimal solution emerged. Training a large amount of data at once did not result in high precision.

In [16], the authors suggest using distributed particle swarm optimization (DPSO) to improve the hyperparameters of CNNs for image classification tasks. The DPSO method employs a mixed-variable encoding strategy and associated update operations for each particle to encode CNNs, enabling automatic and global search for the best CNNs model. The method also employs a distributed framework to reduce execution time and accelerate optimization. However, one possible drawback is the requirement for many particles to achieve good results, which can increase computational complexity.

In [17], the authors proposed an automatic method incorporating enhanced metaheuristic algorithms (the tree growth and firefly algorithms) for optimizing hyperparameters and designing structures. However, the proposed methods had a higher computational cost, restricting the inclusion of more datasets in the study.

In [18], the researchers proposed optimizing CNNs hyperparameters using linearly decreasing weight particle swarm optimization (LDWPSO). The architecture of this model is LeNet-5. They mentioned the need for additional research and testing to validate the method's effectiveness. Additionally, they did not discuss potential limitations or challenges associated with the proposed method, such as computational complexity or convergence issues.

In [19], this work discusses a method for image classification that utilizes histograms of oriented gradient (HOG) features, gray-level co-occurrence matrix (GLCM) features, and support vector machine (SVM) for classification. The authors used the fashion-MNIST dataset to examine the accuracy of this method, and they obtained an accuracy of 91.59%.

The work in [20] focuses on the swarm intelligence component of the OpenNAS system for neural architecture search (NAS). The authors use PSO and ACO swarm algorithms with transfer learning for feature extraction (VGG16).

In [21], the study introduces the competitive activation function (CAF) concept and derives the parameter-free rectified exponential unit (PFREU) as a particular kind of CAF. The authors use two architectures for classification: LeNet-5 on fashion-MNIST and ResNet-110 on CIFAR-10.

In [22], the authors presented IntelliSwAS, a method for optimizing deep neural network architectures for classification and regression tasks. They used DAGRNN [23] to improve the search technique. IntelliSwAS effectively located high-quality CNNs cells, but these cells had to be manually incorporated into larger CNNs architectures.

In [24], this study proposed a hybrid particle swarm optimization and grey wolf optimization (HPSGW) algorithm to optimize these hyperparameters and enhance the accuracy of the CNNs model.

In [25], the authors proposed a simple deterministic selection genetic algorithm (SDSGA) to optimize the hyperparameters of two well-known machine learning models: CNNs and the random forest (RF) algorithm.

In [26], the authors use a multiple convolutional neural network (MCNN15) with 15 convolutional layers.

In [27], the proposed work compared the performance of CNNs and ANNs for image classification on the Fashion-MNIST apparel dataset using various optimizers.

In [28], the study employed large-scale deep learning networks such as VGG16 and ResNet to enhance classification accuracy and an approximate dynamic learning rate update algorithm to ensure rapid convergence and reduced training time.

In [29], the authors utilized the local autonomous competitive harmony search (LACHS) algorithm to achieve the highest classification accuracy on the Fashion-MNIST and CIFAR-10 datasets. The VGGNet was the main network used for experimental research in this work.

Table 1: Summarization of the related works.

Ref.	year	Method	Parameters for optimization	Limitations	Dataset	Accuracy %
[13]	2016	CNN-PSO	kernel size, pool size, learning rate	• CNNPSO consumes a longer time than CNN. • CNNPSO accuracy is slightly lower than CNN optimized with simulated annealing.	MNIST	95.08
[14]	2017	CNN-MAA	kernel size, pool size	• MA needs more processing time	MNIST	98.75
[15]	2019	GA	Learning rate, dropout, batch size, no. of layers	• The experiment took a long because of the large dataset.	MNIST	99.4
[16]	2020	PSO (DPSO)	kernel size, type of pooling, Activ.Fun. in FC, dropout, Learning rate	• Many particles required for good results can increase computational complexity.	MNIST, Fashion-MNIST	99.3, 92.92
[17]	2020	SI (tree growth & firefly) algorithms	no. of convolution, no. of FC, kernel size, no. of kernels per conv. layer, FC-layer size	• Use a single dataset to evaluate the accuracy of the method.	MNIST	99.18
[18]	2020	PSO (LDWPSO)	no. of kernels, kernel size, activation fun., no. of neurons, batch size, optimizer	• The technique uses a simple and basic CNN architecture (LeNet-5). • There is a lack of comparison with other optimization methods or CNN architectures.	MNIST	98.95
[20]	2020	PSO & ACO	no. of kernels, kernel size, dropout rate	• There is no comparison with other neural architecture search methods and no performance-efficiency trade-off analysis.	Fashion-MNIST	94.5
[21]	2021	CAF	activation fun.	• There is no comparison to other state-of-the-art activation functions. • There is no theoretical analysis of CAF. • There is no investigation of hyperparameter impact on performance.	Fashion-MNIST	91.21
[22]	2022	PSO (IntelliSwAS)	convolution, depthwise-separable convolution, dilated convolution	• IntelliSwAS found high-quality CNN cells but required manual incorporation into larger CNN architectures.	MNIST	95
[24]	2022	PSO&GWO (HPSGW)	No. of Kernel, Kernel Size, Batch size No. of Epochs	• It can optimize a few CNN hyperparameters. • High computational cost.	MNIST	99.4
[25]	2022	GA SDSGA	learning rate, batch size	• Selection may reduce diversity, and a fixed mutation rate may not work for all problems.	MNIST	99.2
[26]	2022	MCNN15	no. of the kernel, kernel size, batch size, no. of neurons	• There is no evaluation of model performance or comparison to state-of-the-art.	Fashion-MNIST	94.04

[27]	2022	ANN and CNN	optimizer	• Difficulty handling complex or novel images. • Sensitivity to hyperparameter choices. • High computational cost and time consumption.	Fashion-MNIST	91
[28]	2023	VGG16, ResNet, and approximate dynamic learning rate update algorithm	learning rate	• Deep network hierarchies and complex parameters can overfit, limiting training time, especially with small samples.	Fashion-MNIST	93
[29]	2023	LACHS	no. of the kernel, kernel size, Activ.Fun. in conv., no. of neurons, learning rate, batch size, Momentum	• No comparison to other hyperparameter optimization techniques makes it hard to evaluate the usefulness and superiority of LACHS.	Fashion-MNIST	93.34

3 Tools

In this section, we look at the necessary background for the two main techniques used in our proposed approach: CNNs and SSA.

3.1 Convolution neural networks (CNNs)

CNNs represent a category of deep neural networks that have gained significant prominence in computer vision applications. These networks have revolutionized the field by achieving cutting-edge results across various tasks. They have proven their mettle in diverse domains, such as handwriting recognition [30], automotive safety [31], video surveillance [32], face detection [33], semantic segmentation [34], and speech recognition [35]. This versatility has rendered them indispensable in modern computing systems.

Explicitly designed for data with a grid-like structure, such as images, CNNs have surged in importance due to their capacity to automate the once manual and time-intensive feature extraction process. At the heart of CNNs lies a pivotal feature: weight sharing. By sharing weights, these networks reduce the number of trainable parameters, a feat that enhances generalization capabilities and curbs overfitting issues.

Unlike traditional neural networks, CNNs capitalize on the intrinsic spatial organization present in images. This enables them to capture local relationships and learn hierarchical representations. The architecture of CNNs encompasses a multi-stage structure, integrating both linear and nonlinear operations to undertake feature extraction and classification.

The initial feature extraction stage encompasses a sequence of primary layers, including the Conv. layer housing an activation function (Act. Fun.) and a subsequent pooling layer. Conversely, the classification stage contains numerous FC layers [36]. It is important to note that this architecture necessitates substantial data for training, demanding a significant time investment and considerable expertise for manual construction. To address this, many optimization techniques have been deployed to fine-tune hyperparameters and structures [37]. Researchers have engineered several CNNs models, training them on specific problem areas using varied datasets and achieving impressive results within these domains. Leveraging a pre-trained network involves tailoring it to a particular task. Commonly referred to as "transfer learning," this approach enables the classification of images across a vast array of 1,000 distinct categories, avoiding the need to build CNNs from scratch. The fine-tuning of hyperparameters through transfer learning can involve freezing or unfreezing layers [38].

An array of pre-trained models, including DenseNet [25], EfficientNet [26], MobileNetV3 [27], and more, offer additional options for developers and researchers in this realm.

The overall structure of the network is built in the form of layers stacked on top of each other to process the data that enters it, extract features from it, and classify it according to the problem. These layers are:

3.2 Input layer

The input layer, which is the leftmost layer, represents the input image to the CNNs.

3.3 Convolutional layers

The convolutional layers are the foundation of CNNs. They contain the learned kernels (filters, weights) that are used to extract features from images. This is done by convolving the input image with a stack of kernels. Each kernel extracts a specific feature, such as edges, textures, or object parts [39]. By using multiple kernels, CNNs can capture a variety of spatial patterns. This allows them to automatically learn relevant visual features without requiring manual feature engineering.

Each convolutional layer has a set of hyperparameters that are initialized before the layer is used. These hyperparameters determine the number of connections and output size for the feature maps. The hyperparameters are:

- Number of filters: This determines the depth of the resulting feature maps.
- Filter size: This gives the spatial dimensions of the filters.
- Padding: This determines the amount of zero-padding applied to the input data.
- Stride: This specifies the step size for shifting the filters over the input.

After all layers with weights (also known as trainable layers, such as Conv. layers and FC layers) in CNNs architecture, nonlinear activation layers are used. The non-linearity of the activation layers means that the mapping from input to output is nonlinear. This allows the CNNs to learn complex things [9]. CNNs commonly use the rectified linear unit (ReLU), sigmoid, and hyperbolic tangent (tanh) activation functions. ReLU is the most popular Act. Fun. because of its simplicity and ability to mitigate the vanishing gradient problem.

3.4 Pooling layers

Pooling layers are commonly used in CNNs to reduce the spatial dimensions of feature maps while preserving their most essential features. This helps to reduce the amount of computing power required to process data and makes the model more robust to small spatial variations.

There are two main types of pooling layers: max pooling and average pooling. In max pooling, the maximum value in each region of the feature map is selected. In average pooling, the average value in each region is selected.

The pooling size and stride are two hyperparameters that need to be specified for each pooling layer. The pooling size determines the size of the pooled region, and the stride determines the step size used to move the pooling region over the feature map.

The features extracted by the pooling layers are then passed to the next layer in the CNN, which is typically an FC layer. The FC layer takes the features from the pooling layers and combines them to predict the input image.

Prior to the entire FC layer, the previous layer's output must be transformed into a one-dimensional vector. This flattening process converts multidimensional feature maps into a format compatible with completely connected layers.

3.5 Fully connected layers

The FC layers are positioned just before a CNNs output layer. They are responsible for converting the learned features into class probabilities or regression values.

FC layers connect every neuron in the previous layer to every neuron in the next layer. This allows them to combine high-level features and make accurate predictions. However, they also introduce many parameters, which can lead to overfitting if not carefully regularized.

The number of neurons in an FC layer determines the layer's output size. The number of neurons required varies depending on the specific task being performed. For example, in image classification, the output layer may contain neurons representing different classes, while in object detection, it may contain neurons for bounding box coordinates and class probabilities.

Here are some of the hyperparameters that need to be tuned for FC layers to give the best results in CNNs:

- Number of hidden layers: The more hidden layers, the better the network performs, which can also lead to overfitting. A good starting point is to use two or three hidden layers.
- Number of neurons: The number of neurons in each hidden layer should be related to the complexity of the task. The task requiring a higher level of prediction requires more neurons.
- Activation function: The activation function determines how the output of each neuron is transformed. A commonly used Act. Fun. is the rectified linear unit (ReLU).
- Weight initialization: The weights of the FC layers are initialized randomly. A suitable initialization method can help to prevent the network from converging to a suboptimal solution.
- Regularization: Regularization techniques can help to prevent overfitting. A commonly used regularization technique is dropout.

3.6 Output layer

The output layer is the final layer of neurons in CNNs. It generates the network output, typically a class prediction or a regression value.

The output layer is typically an FC layer, which means that each neuron in the output layer is connected to all of the neurons in the previous layer. This allows the output layer to combine the features extracted by the earlier layers and make a prediction about the input data.

The choice of activation function in the output layer is determined by the specific task being performed by the CNN. For classification tasks, a commonly used activation function is the softmax function. The softmax function takes a vector of outputs from the previous layer and transforms it into a vector of probabilities, where each probability represents the likelihood that the input data belongs to a particular class. Figure 1 shows the general architecture of CNNs.

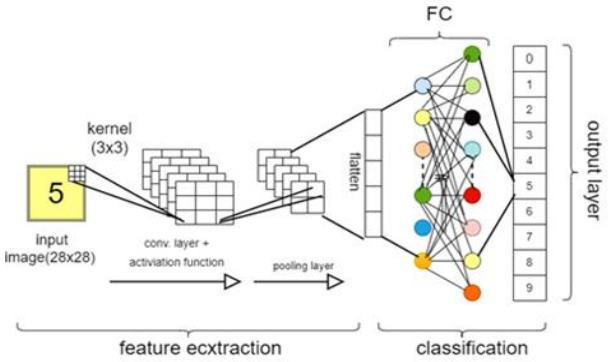

Figure 1: Simple structure of CNNs

CNNs are trained using backpropagation, which involves iteratively passing data through the network (forward propagation) and then adjusting the network weights based on the error (backpropagation). The number of Conv., pooling, and FC layers can vary depending on the task and the available computational resources.

CNNs have several strengths, including understanding spatial hierarchies of features, handling large amounts of data, and generalizing well to new data. However, they also have limitations, such as the need for extensive training data and the computational expense.

3.7 Salp Swarm Algorithm (SSA)

SSA is a population-based metaheuristic algorithm inspired by the chain formation behavior of salps, which are gelatinous marine organisms [40]. The SSA algorithm maintains a population of solutions, each representing a salp. The salps move towards better solutions by adjusting their positions and velocities. The movement of a salp is influenced by three main factors: the current location of the best solution, the position of the best solution in its neighborhood, and a randomization factor [41].

The SSA algorithm has been shown to be effective for various optimization problems, including function optimization, engineering design, and parameter estimation.

The SSA algorithm has several advantages over other metaheuristic algorithms, such as PSO, GA, and DE. These advantages include [42]:

- Simple to implement: The SSA algorithm is relatively simple to implement, making it easy to understand and use.
- Fewer parameters: The SSA algorithm has fewer parameters than other metaheuristic algorithms, making tuning easier.
- Robust: The SSA algorithm is robust to noise and outliers, making it a good choice for problems with noisy data.
- Efficient: The SSA algorithm is efficient, making it a good choice for large-scale optimization problems.

The pseudocode for the SSA algorithm is clearly illustrated in Figure 2.

Initialize the salp population x_i (i= 1, 2, ..., n) considering ub and lb
While (end condition is not satisfied)
  Calculate the fitness of each search agent (salp)
  F=the best search agent
  Update c1 by use: $c^1 = 2e^{-\left(\frac{4t}{L}\right)^2}$
  For each salp (x_i)
    if ($\dot{\iota}$ ==1)
      Update the position of the leading salp using:

$$x_j^1 = \begin{cases} F_j + c_1((ub_j - lb_j)c_2 + lb_j) & c_3 \geq 0 \\ F_j - c_1((ub_j - lb_j)c_2 + lb_j) & c_3 < 0 \end{cases}$$

    else
      Update the position of the follower salp using:

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1})$$

    endif
  endfor
  Amend the salps based on the upper and lower bounds of variables
endwhile
return F

Figure 2: Pseudocode of the SSA algorithm

4 Current work

In the present work, we use the SSA to identify the optimal hyperparameter settings for enhancing the performance of CNNs. We strategically select six key hyperparameters: the number of kernels, kernel size, pool size, dropout rate, hidden units, and learning rate. The learning rate is particularly influential, as it profoundly impacts the network's operation. The number of available network weights limits the scale of the candidate solution population. We can address this challenge by creating multiple iterations of the CNNs architecture. We create distinct versions of the CNNs by assigning various values to each of the six identified hyperparameters. We then use the SSA to individually train each version, using a diverse set of candidate solutions for training. This diversified approach to hyperparameters primarily aims to maximize the classification accuracy of CNNs. By systematically exploring and optimizing the hyperparameters, we aim to extract the best possible performance from CNNs, pushing the boundaries of their classification capabilities.

4.1 Individual representation

To scrutinize the scalability and robustness of the SSA-CNNs method, we undertook extensive training and evaluation across multiple CNN architectures, datasets, and a range of hyperparameter configurations. This iterative process led us to the current set of hyperparameters and settings.

The current individual is characterized by a 6-dimensional vector corresponding to the following hyperparameters for CNNs. Each of them is defined within specific ranges, with lower and upper boundaries as follows:

- Number of kernels: 32, 64, 128, 256, 512, or 1024
- Kernel size: 3, 5, or 7
- Pool size: 3, 5, or 7
- Dropout rate: 0.2, 0.3, or 0.4
- Learning rate: 0.001, 0.0001, or 0.00001
- Hidden units: 64, 128, 256, 512, or 1024

Figure 3 shows a visual representation of an individual in the context of this work.

Number of kernels	Kernel size	Pool size	Dropout rate	Learning rate	Hidden units

Figure 3: Presentation of SSA individual

4.2 Fitness evaluation

Our method's goal, consistent with the principles of all metaheuristic algorithms, is to quickly identify an individual with superior accuracy (or minimized errors). In our current approach, fitness evaluation involves assessing the accuracy of individual CNNs, each of which is represented as an independent entity.

The architectural details of the salp are stored in the population. These details are then transferred to CNNs with the corresponding architecture. The CNNs model is

then trained on the supplied training data using multiple epochs to evaluate the "salp."
Figure 4 depicts the flowchart of the present method.

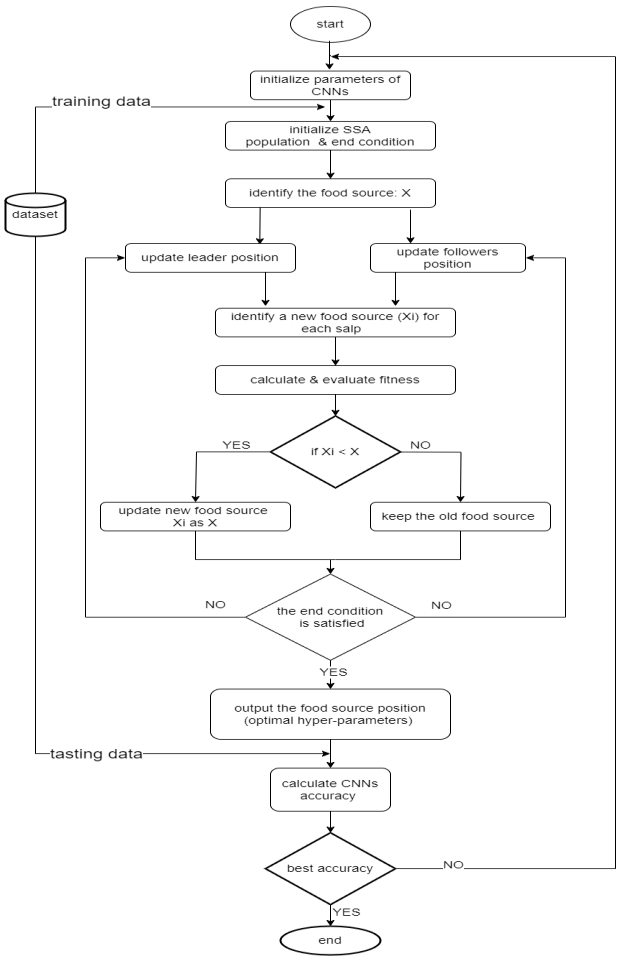

Figure 4: Flowchart of the current work

5 Experimental evaluation

- *Dataset*

To assess the effectiveness of our approach, we selected the MNIST and Fashion-MNIST datasets for their widespread use in the deep learning community, manageable dataset sizes, and diverse content, enabling us to effectively assess the generalizability of the current work across different domains.

- The MNIST dataset. This dataset is a widely used benchmark for image classification tasks, and it consists of 70,000 grayscale images of handwritten digits. The images are 28 × 28 pixels in size [17], and they have been preprocessed, normalized, and formatted to improve their consistency [43]. The MNIST dataset is divided into two sets: 60,000 images for training and 10,000 images for testing. The training set is used to train the image recognition model, and the test set is used to evaluate the model's performance. The MNIST dataset is valuable for developing and evaluating image recognition models. It is a standardized benchmark that allows researchers to compare their results with other researchers.
- The Fashion-MNIST dataset comprises 70,000 grayscale images of fashion items from 10 categories, each having 7,000 images. The images are of size 28 × 28 pixels. The dataset has a training set of 60,000 images and a test set of 10,000 images. This dataset is intended to replace the original MNIST dataset as it has the exact image dimensions, data format, and training/testing split structure [11].

- *Parameters Setting*

To understand the parameters used in this study, please refer to Table 2. The table has two categories of parameters: CNNs Training and SSA.
The first category of parameters defines the basic CNNs architecture. These include the number of convolutional layers, the number of pooling layers, the activation function for convolutional layers, the stride, the padding configuration, the specifications for hidden layers, the activation function for FC layers, the activation function for the output layer, the chosen loss function, the optimizer selection, the metrics used, the designated epochs, and the batch size.
The second category of parameters controls the behavior of the SSA. This category has two variables: the number of salps and the maximum number of generations.

Table 2: Parameters used for evaluating the current work.

	CNNs parameters	***values***
1	No. of convolutional layers	2
2	No. of pooling layers	2
3	Activation function for convolutional layers	Relu
4	Stride	1
5	padding	same
6	Hidden layers	3
7	Activation function for FC layers	Relu
8	Activation function for the output layer	softmax
9	Loss function	Categorical-crossentropy
10	Optimizer	Adam
11	Metrics	accuracy
12	epochs	10
13	Batch-size	128
	SSA parameters	***values***
1	Number of salps	15
2	Max-generations	100

We utilized Google Colab, which offers a cost-free environment and hardware acceleration for Python 3 programming, equipped with the GPU T4.

6 Results and analysis

We present the results of our approach and compare it to other methods in terms of their accuracies on the MNIST and fashion-MNIST datasets. We also show the optimal architectures found in this work.

Tables 3-4 show the experimental results of our approach compared to other methods.

Table 3: Comparison of accuracy of the current work and different models on the MNIST dataset.

Ref.	Method	Acc.%
[13]	CNN-PSO	95.08
[14]	CNN-MAA	98.75
[15]	GA	99.4
[16]	PSO (DPSO)	99.3
[17]	SI (tree growth & firefly) algorithms	99.18
[18]	PSO(LDWPSO)	98.95
[22]	PSO (IntelliSwAS)	95
[24]	PSO&GWO (HPSGW)	99.4
[25]	GA(SDSGA)	99.2
	Our method (SSA-CNNs)	**99.46**

Table 4: Comparison of current work accuracy and different models on the fashion-MNIST dataset.

Ref.	Method	Acc.%
[16]	PSO (DPSO)	92.92
[20]	PSO & ACO	94.5
[21]	CAF	91.21
[26]	MCNN15	94.04
[27]	Multioptimizers	91
[28]	approximate dynamic learning rate update algorithm	93
[29]	LACHS	93.34
	Our method (SSA-CNNs)	**94.53**

Table 3 illustrates that the SSA-CNNs method (99.46%) outperforms most other techniques on the MNIST dataset. Specifically, SSA-CNNs achieved higher accuracy than CNN-PSO (95.08%), CNN-MAA (98.75%), PSO (DPSO) (99.3), PSO (LDWPSO) (98.95), PSO (IntelliSwAS) (95), and SI (tree growth & firefly) algorithms (99.18). It also performed on par with the best-performing techniques, GA (99.4), PSO&GWO (HPSGW) (99.4), and GA (SDSGA) (99.2). These results suggest that the SSA-CNNs method is highly competitive and may offer superior accuracy compared to other optimization techniques when applied to the MNIST dataset. It showcases the effectiveness of SSA in enhancing CNNs for image classification tasks.

Table 4 shows that the SSA-CNNs method achieves the best accuracy on the Fashion-MNIST dataset, with an accuracy of 94.53%. This is higher than the accuracy of any of the other techniques listed, including PSO (DPSO) (92.92%), PSO & ACO (94.5), CAF (91.21), MCNN15 (94.04), Multioptimizers (91), approximate dynamic learning rate update algorithm (93), and LACHS (93.34). This suggests that the SSA-CNNs method is a promising approach for image classification tasks and may be particularly well-suited for datasets such as Fashion-MNIST, which contain many classes.

Overall, the SSA-CNNs technique has proven to be a highly effective method for image classification on both the MNIST and Fashion-MNIST datasets. With a remarkable 99.46% accuracy on MNIST and a competitive 94.53% accuracy on Fashion-MNIST, SSA-CNNs showcases its versatility and robustness. This approach, which integrates SSA with CNNs, offers a promising path for optimizing image classification tasks, consistently delivering outstanding results.

The top-performing individuals achieving the highest accuracy on MNIST and Fashion-MNIST are depicted in Figures 5 and 6, respectively.

Number of kernels	Kernel size	Pool size	Dropout rate	Learning rate	Hidden units
512	3×3	3×3	0.3	0.001	512

Figure 5: Best individual for MNIST dataset.

Figure 5 presents the best individual layered architecture for MNIST.

Number of kernels	Kernel size	Pool size	Dropout rate	Learning rate	Hidden units
512	5×5	5×5	0.3	0.0001	128

Figure 6: The best individual for the fashion-MNIST dataset.

```
Test loss: 0.01803884468972683
Test accuracy: 0.9946000078201294
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 512)       5120

 max_pooling2d (MaxPooling2D  (None, 9, 9, 512)        0
 )

 conv2d_1 (Conv2D)           (None, 9, 9, 512)         2359808

 max_pooling2d_1 (MaxPooling  (None, 3, 3, 512)        0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 512)         2359808

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 512)        0
 2D)

 dropout (Dropout)           (None, 1, 1, 512)         0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 4,992,522
Trainable params: 4,992,522
Non-trainable params: 0
_________________________________________________________________
```

Figure 7: The present method of layered architecture for MNIST dataset.

7 Conclusion

We present a new approach to optimizing CNNs using the SSA method. This approach has several advantages. It balances accuracy, computational efficiency, and training time well. It also achieves exceptional classification accuracy on the MNIST and fashion-MNIST datasets. This SSA-based optimization method outperforms other

algorithms that require significant computational resources and time, making it a promising candidate for practical applications.

The proposed method allows seamlessly integrating CNNs into real-world scenarios, particularly in resource-constrained and time-sensitive settings. Future research could explore the adaptability of the SSA-based optimization technique to other deep-learning architectures and tasks beyond computer vision. Additionally, delving into the theoretical underpinnings of the SSA algorithm and refining parameter tuning strategies could help broaden its adoption in optimization and machine learning.

Our plan to improve the SSA-based hyperparameter optimization framework involves four main goals. Firstly, we will test the effectiveness of the SSA-based framework on different CNN architectures and datasets. Secondly, we intend to create new or improved SSA variants for hyperparameter optimization. Thirdly, we will integrate the SSA-based framework with other hyperparameter optimization techniques to develop a hybrid approach. Lastly, we will apply the SSA-based framework to other machine learning tasks, like natural language processing and computer vision. By pursuing these goals, we aim to make essential contributions to hyperparameter optimization.

References

[1] Gadri, S., Developing an efficient predictive model based on ml and dl approaches to detect diabetes. Informatica, 2021. 45(3). http://dx.doi.org/10.31449/inf.v45i3.3041

[2] Abdulla, M. and A. Marhoon, Agriculture based on Internet of Things and Deep Learning. Iraqi Journal for Electrical and Electronic Engineering, 2022. 18(2): p. 1-8. http://dx.doi.org/10.37917/ijeee.18.2.1

[3] Xu, Y., et al., Batch normalization with enhanced linear transformation. arXiv preprint arXiv:2011.14150, 2020. https://doi.org/10.48550/arXiv.2011.14150

[4] Shrestha, A. and A. Mahmood, Review of deep learning algorithms and architectures. IEEE access, 2019. 7: p. 53040-53065. http://dx.doi.org/10.1109/access.2019.2912200

[5] Hassan, N.F.A., A.A. Abed, and T.Y. Abdalla, Face mask detection using deep learning on NVIDIA Jetson Nano. International Journal of Electrical & Computer Engineering (2088-8708), 2022. 12(5). http://dx.doi.org/10.11591/ijece.v12i5.pp5427-5434

[6] Gaafar, A.S., J.M. Dahr, and A.K. Hamoud, Comparative Analysis of Performance of Deep Learning Classification Approach based on LSTM-RNN for Textual and Image Datasets. Informatica, 2022. 46(5). http://dx.doi.org/10.31449/inf.v46i5.3872

[7] Wang, Y., H. Zhang, and G. Zhang, cPSO-CNN: An efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks. Swarm and Evolutionary Computation, 2019. 49: p. 114-123. http://dx.doi.org/10.1016/j.swevo.2019.06.002

[8] Darwish, A., D. Ezzat, and A.E. Hassanien, An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis. Swarm and evolutionary computation, 2020. 52: p. 100616. http://dx.doi.org/10.1016/j.swevo.2019.100616

[9] Alzubaidi, L., et al., Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data, 2021. 8(1): p. 53 DOI: 10.1186/s40537-021-00444-8.

[10] LeCun, Y., The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist/, 1998.

[11] Xiao, H., K. Rasul, and R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017. https://doi.org/10.48550/arXiv.1708.07747

[12] Zhang, H., et al., Differential evolution-assisted salp swarm algorithm with chaotic structure for real-world problems. Eng Comput, 2022. 39(3): p. 1735-1769 DOI: 10.1007/s00366-021-01545-x.

[13] Syulistyo, A.R., et al., Particle swarm optimization (PSO) for training optimization on convolutional neural network (CNN). Jurnal Ilmu Komputer dan Informasi, 2016. 9(1): p. 52-58. http://dx.doi.org/10.21609/jiki.v9i1.366

[14] Ayumi, V., et al. Optimization of convolutional neural network using microcanonical annealing algorithm. in 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS). 2016. IEEE. http://dx.doi.org/10.1109/icacsis.2016.7872787

[15] Yoo, J.-H., et al. Optimization of hyper-parameter for CNN model using genetic algorithm. in 2019 1st International conference on electrical, control and instrumentation engineering (ICECIE). 2019. IEEE. http://dx.doi.org/10.1109/icecie47765.2019.8974762

[16] Guo, Y., J.-Y. Li, and Z.-H. Zhan, Efficient hyperparameter optimization for convolution neural networks in deep learning: A distributed particle swarm optimization approach. Cybernetics and Systems, 2020. 52(1): p. 36-57. http://dx.doi.org/10.1080/01969722.2020.1827797

[17] Bacanin, N., et al., Optimizing Convolutional Neural Network Hyperparameters by Enhanced Swarm Intelligence Metaheuristics. Algorithms, 2020. 13(3).DOI: 10.3390/a13030067.

[18] Serizawa, T. and H. Fujita, Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization. arXiv preprint arXiv:2001.05670, 2020. https://doi.org/10.48550/arXiv.2001.05670

[19] Greeshma, K. and J.V. Gripsy, Image classification using HOG and LBP feature descriptors with SVM and CNN. Int J Eng Res Technol, 2020. 8(4): p. 1-4. DOI : 10.17577/IJERTCONV8IS04021

[20] Lankford, S. and D. Grimes, Neural architecture search using particle swarm and ant colony optimization. 2020.

[21] Ying, Y., et al., Improving convolutional neural networks with competitive activation function.

Security and Communication Networks, 2021. 2021: p. 1-9.
[22]Nistor, S.C. and G. Czibula, IntelliSwAS: Optimizing deep neural network architectures using a particle swarm-based approach. Expert Systems with Applications, 2022. 187: p. 115945. http://dx.doi.org/10.1016/j.eswa.2021.115945
[23]Moodie, E.E. and D.A. Stephens, Comment: Clarifying endogeneous data structures and consequent modelling choices using causal graphs. 2020. http://dx.doi.org/10.1214/20-sts777
[24]Challapalli, J.R. and N. Devarakonda, A novel approach for optimization of convolution neural network with hybrid particle swarm and grey wolf algorithm for classification of Indian classical dances. Knowledge and Information Systems, 2022. 64(9): p. 2411-2434. http://dx.doi.org/10.1007/s10115-022-01707-3
[25]Raji, I.D., et al., Simple deterministic selection-based genetic algorithm for hyperparameter tuning of machine learning models. Applied Sciences, 2022. 12(3): p. 1186. http://dx.doi.org/10.3390/app12031186
[26]Nocentini, O., et al., Image classification using multiple convolutional neural networks on the fashion-MNIST dataset. Sensors, 2022. 22(23): p. 9544. http://dx.doi.org/10.3390/s22239544
[27]Sumera, S.R., N. Anjum, and K. Vaidehi, Implementation of CNN and ANN for Fashion-MNIST-Dataset using Different Optimizers. Indian Journal of Science and Technology, 2022. 15(47): p. 2639-2645. http://dx.doi.org/10.17485/ijst/v15i47.1821
[28]Shin, S.-Y., G. Jo, and G. Wang, A Novel Method for Fashion Clothing Image Classification Based on Deep Learning. Journal of Information and Communication Technology, 2023. 22(1): p. 127-148. http://dx.doi.org/10.32890/jict2023.22.1.6
[29]Liu, D., et al., Hyperparameters Optimization of Convolutional Neural Network Based on Local Autonomous Competition Harmony Search Algorithm. Journal of Computational Design and Engineering, 2023: p. qwad050. http://dx.doi.org/10.1093/jcde/qwad050
[30]Altwaijry, N. and I. Al-Turaiki, Arabic handwriting recognition system using convolutional neural network. Neural Computing and Applications, 2021. 33(7): p. 2249-2261. http://dx.doi.org/10.1007/s00521-020-05070-8
[31]Ren, L., et al., A data-driven auto-CNN-LSTM prediction model for lithium-ion battery remaining useful life. IEEE Transactions on Industrial Informatics, 2020. 17(5): p. 3478-3487. http://dx.doi.org/10.1109/tii.2020.3008223
[32]Ashraf, A.H., et al., Weapons detection for security and video surveillance using cnn and YOLO-v5s. CMC-Comput. Mater. Contin, 2022. 70: p. 2761-2775. http://dx.doi.org/10.32604/cmc.2022.018785
[33]Zamir, M., et al., Face Detection & Recognition from Images & Videos Based on CNN & Raspberry Pi. Computation, 2022. 10(9): p. 148. http://dx.doi.org/10.3390/computation10090148
[34]Li, C., et al., Segmenting objects in day and night: Edge-conditioned CNN for thermal image semantic segmentation. IEEE Transactions on Neural Networks and Learning Systems, 2020. 32(7): p. 3069-3082. http://dx.doi.org/10.1109/tnnls.2020.3009373
[35]Haque, M.A., et al. Experimental evaluation of CNN architecture for speech recognition. in First International Conference on Sustainable Technologies for Computational Intelligence: Proceedings of ICTSCI 2019. 2020. Springer. http://dx.doi.org/10.1007/978-981-15-0029-9_40
[36]Khudeyer, R.S. and N.M. Almoosawi, Combination of machine learning algorithms and Resnet50 for Arabic Handwritten Classification. Informatica, 2023. 46(9). http://dx.doi.org/10.31449/inf.v46i9.4375
[37]Fregoso, J., C.I. Gonzalez, and G.E. Martinez, Optimization of convolutional neural networks architectures using PSO for sign language recognition. Axioms, 2021. 10(3): p. 139. http://dx.doi.org/10.3390/axioms10030139
[38]Alhijaj, J.A. and R.S. Khudeyer, Integration of EfficientNetB0 and Machine Learning for Fingerprint Classification. Informatica, 2023. 47(5). http://dx.doi.org/10.31449/inf.v47i5.4724
[39]Al, N.M.A.-M.M. and R.S. Khudeyer, ResNet-34/DR: a residual convolutional neural network for the diagnosis of diabetic retinopathy. Informatica, 2021. 45(7). http://dx.doi.org/10.31449/inf.v45i7.3774
[40]Mirjalili, S., et al., Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in engineering software, 2017. 114: p. 163-191. http://dx.doi.org/10.1016/j.advengsoft.2017.07.002
[41]Duan, Q., et al., Improved salp swarm algorithm with simulated annealing for solving engineering optimization problems. Symmetry, 2021. 13(6): p. 1092. http://dx.doi.org/10.3390/sym13061092
[42]Faris, H., et al., Salp swarm algorithm: theory, literature review, and application in extreme learning machines. Nature-inspired optimizers: theories, literature reviews and applications, 2020: p. 185-199. http://dx.doi.org/10.1007/978-3-030-12127-3_11
[43]Wu, H., CNN-Based Recognition of Handwritten Digits in MNIST Database. Research School of Computer Science. The Australia National University, Canberra, 2018.