Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Optimizing the Architecture of Convolutional Neural Networks Using Modified 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.
Email: hopidhadha@gmail.com, ma@uobasrah.edu.iq
[2]Department of Computer Information Systems, College of Computer Science and Information Technology, University of Basrah, Basrah, Iraq.
Email: raidah.khudayer@uobasrah.edu.iq

ARTICLE INFO
Article history:
Received: 3 /3/2024
Rrevised form: 21 /3/2024
Accepted : 28 /3/2024
Available online: 30 /3/2024

Keywords:
Deep learning
Convolutional neural networks
Hyperparameters optimization
Salp swarm algorithm

ABSTRACT

Deep learning is highly effective in dealing with complex tasks such as image classification and recognition. However, finding the optimal architecture's hyperparameters for Convolutional Neural Networks (CNNs) to achieve the best performance and parameter regularization can be challenging. Metaheuristic optimization algorithms can be utilized to find solutions in this context. In this research, a computerized CNN was adjusted using an improved Salp Swarm Algorithm (SSA) to enhance crucial CNN settings, like dropout rate, hidden units, learning rate, and batch size. The refined design was tested on two standard datasets. MNIST and Fashion MNIST. The outcomes displayed model performance achieving accuracy levels of 99.6% for MNIST and 94.08% for Fashion MNIST. This tuned system outperformed the existing practices by 0.2% and 0.04% for each dataset while also cutting down on computational expenses. The fusion of SSA with CNNs displayed adaptability and resilience opening up possibilities, in image classification and consistently delivering outstanding outcomes.

MSC.

https://doi.org/10.29304/jqcsm.2024.16.11450

1. introduction

Deep Learning (DL) falls under the umbrella of machine learning. Relies, on Artificial Neural Networks (ANNs) to learn from data and enhance performance on specific tasks [1]. These networks imitate the structure and functionality of the brain enabling them to process information and make decisions in a manner. DL can be viewed as a group of interconnected classifiers each tasked with recognizing features or patterns within the input data [2]. These classifiers use regression in conjunction with activation functions to extract more advanced features ultimately producing the desired output. It should be emphasized that achieving optimal performance from CNNs relies greatly on fine-tuning their hyperparameters [3, 4]. These settings, including the learning rate, batch size, and kernel size, dictate how the network functions and directly affects its overall performance [5, 6]. Consequently, finding the ideal

*Corresponding author *Entesar H. Abdulsaed Almuhsen*

Email addresses: hopidhadha@gmail.com

Communicated by 'sub etitor'

combination of these hyperparameters necessitates the use of efficient techniques that can effectively explore the extensive hyperparameters space to uncover this optimal configuration. Metaheuristic optimization algorithms like Gray Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC) have proven to be valuable tools in this regard [7-9].

In this research, we delve into how effective SSA, is in tuning the hyperparameters of CNN architectures. Opting for SSA is a choice because of its established advantages; rapid convergence, comprehensive exploration of solution spaces, and an algorithm that facilitates the identification of optimal configurations. Additionally, its simple mathematical framework makes it easier to implement and comprehend [10]. To validate SSA's performance, we conducted experiments on the MNIST and fashion MNIST datasets, pitting them against other optimization methods [11, 12]. The results will not only reveal SSA's effectiveness and efficiency in optimizing CNN hyperparameters but also offer valuable insights for further research on deep learning model optimization.

This paper is structured as follows: Section 2 delves into existing research relevant to our work. Section 3 lays the theoretical groundwork, introducing CNNs and SSA. The proposed approach is then meticulously described in Section 4. Section 5 presents comprehensive evaluation experiments, followed by a thorough analysis and visualization of results in Section 6. Finally, Section 7 draws insightful conclusions and wraps up the paper.

2. Related Works

Recent trends in evolutionary algorithms and reinforcement learning approaches have shown promise in developing and applying diverse techniques for autonomously setting hyperparameters and selecting the structure of CNNs. These methods aim to eliminate the need for tedious grid search and manual parameter tweaking, thereby reducing computational costs and saving valuable time during network construction. In this section, we delve into some of the latest research within this field, exploring the innovative methods that are shaping the future of automated CNN design.

In [13] proposes using a Genetic Algorithm (GA) to optimize several CNN parameters. They investigated a variety of GA parameters and ranges, and their exhaustive search culminated in an approximation of the global optimum. Notably, even training with a large dataset did not achieve optimal accuracy, suggesting potential limitations in the chosen parameter space. In [14], it explores using distributed Particle Swarm Optimization (DPSO) to optimize CNNs hyperparameters. Their method utilizes an encoding technique, for CNNs customizing update operations for each element. This allows an automated search to find the CNNs model. Additionally, they use a distributed framework to speed up optimization and reduce processing time. However, obtaining results might involve using several elements potentially raising computational expenses. In another study, an automated strategy is introduced that combines algorithms (tree growth and firefly) to optimize hyperparameters and create CNN structures [15]. While effective the computational requirements of these methods restricted the analysis to a dataset than intended. Another proposal suggests employing Decreasing Weight Particle Swarm Optimization (LDWPSO) to fine-tune CNNs hyperparameters [16]. The goal is to enhance the LeNet 5 architecture. Yet the authors acknowledge the necessity for research and validation to confirm its effectiveness. Moreover, it is crucial to investigate the drawbacks of LDWPSO, in CNNs optimization related to convergence speed and computational expenses.

Furthermore, another approach is presented for categorizing the Fashion MNIST dataset by combining features extracted from level co-matrices “GLCMs” Histograms of Oriented Gradients (HOGs) and Support Vector Machines (SVMs) resulting in an impressive accuracy rate of 91.59% [17]. In [18] introduces the idea of Activation Functions (CAFs). Suggests the Parameter Free Rectified Exponential Unit (PFREU), as a specific CAF example. Their paper then demonstrates the effectiveness of CAFs for data classification by applying them to two architectures: ResNet-110 on CIFAR-10 and LeNet-5 on Fashion-MNIST. In [19] IntelliSwAS is a technique for optimizing Deep Neural Network (DNN) architectures for both classification and regression tasks. To improve the search process, the authors leverage Directed Acyclic Graphs Recurrent Neural Networks DAGRNN [20], a machine learning model specifically designed to predict network architecture quality. While IntelliSwAS successfully identified high-performing CNNs cells (building blocks of the network), manually integrating them into larger CNNs architectures remained a bottleneck. In [21] proposes a Hybrid Particle Swarm Optimization and Grey Wolf Optimization (HPSGW) algorithm to enhance the performance of CNNs models by optimizing their hyperparameters.

The researcher in [22] introduces a Simple Deterministic Selection Genetic Algorithm (SDSGA) for optimizing the hyperparameters of two popular machine learning models: CNNs and the Random Forest (RF) algorithm. In [23] proposes a 15-convolutional-layer Multiple CNN (MCNN15). The authors in [24] investigate the performance of CNNs and ANNs for image classification on the Fashion-MNIST apparel dataset, evaluating the effect of different optimizers.

In [25] advantages established deep learning architectures like VGG16 and ResNet to achieve high classification accuracy. The authors further propose an approximate dynamic learning rate update algorithm to promote faster convergence and shorten training time. In [26] employs the Local Autonomous Competitive Harmony Search (LACHS) algorithm to optimize the classification accuracy of a VGGNet on both Fashion-MNIST and CIFAR-10 datasets. The

researchers in [27] introduce a novel Q-learning Reinforcement Learning (RL)-based Optimization Algorithm (ROA) for optimizing CNN hyperparameters. The paper tests the proposed ROA on two datasets: MNIST and CIFAR-10. In [11] focuses on implementing a deep CNNs model to improve recognition accuracy for the MNIST handwritten digit dataset.

Table 1. outlines the limitations and effectiveness of different approaches for optimizing CNN hyperparameters.

Table 1: Related Works.

Ref	Method(s)	Parameters for optimization	Limitations	Datasets	Acc. %
[13] 2019	GA	Learning rate, dropout, batch size, no. of layers	• The investigation was time-consuming due to the extensive dataset	MNIST	99.4
[14] 2020	PSO (DPSO)	Kernel size, type of pooling, Activ. Fun. in FC, dropout, Learning rate	• Increasing computational complexity may occur when numerous particles are necessary for achieving satisfactory results.	MNIST, Fashion-MNIST	99.3, 92.92
[15] 2020	Tree growth & firefly algorithms	No. of Conv, no. of FC, kernel size, no. of kernel, FC-layer size	• Employ a solitary dataset to assess the precision of the approach.	MNIST	99.18
[16] 2020	PSO (LDWPSO)	No. of kernels, kernel size, Activ. Fun., no. of neurons, batch size, optimizer	• The method employs a straightforward CNN architecture (LeNet-5). • Comparisons with alternative optimization methods or CNN architectures are missing.	MNIST	98.95
[18] 2021	CAF	Activ. Fun.	• In comparison to other cutting-edge Activ. Fun. , there is none. • No theoretical analysis exists regarding CAF. • The efficacy impact of hyperparameters has not been investigated.	Fashion-MNIST	91.21
[19] 2022	PSO (IntelliSwAS)	Conv., depthwise-separable Conv., dilated Conv.	• Identified CNN cells of superior quality; however, their integration into more extensive CNN architectures necessitated manual effort	MNIST	95.0
[21] 2022	PSO&GWO (HPSGW)	No. of kernel, kernel size, batch size, no. of epochs	• A limited number of CNN hyperparameters can be optimized. • High cost of computation	MNIST	99.4
[22] 2022	GA SDSGA	Learning rate, batch size	• Diversity may be diminished by selection, and a fixed mutation rate might not apply to all issues	MNIST	99.2
[23] 2022	MCNN15	No. of the kernel, kernel size, batch size, no. of neurons	• No assessment of model efficacy or comparison to the current state of the art is provided	Fashion-MNIST	94.04
[24] 2022	ANN and CNN	Optimizer	• Challenges in managing intricate or innovative visuals • Varies in response to hyperparameter selections.	Fashion-MNIST	91.0

			• Extra effort and financial investment in computation.		
[25] 2023	approximate dynamic learning rate update algorithm, ResNet, and VGG16	Learning rate	• Particularly with small samples, deep network hierarchies, and intricate parameters can overfit, thereby limiting training time.	Fashion-MNIST	93
[26] 2023	LACHS	No. of kernel, Kernel size, Activ. Fun. in Conv., no. of Neurons, learning rate. batch size, Momentum	• The absence of a comparative analysis with other hyperparameters optimization techniques complicates the assessment of LACHS's utility and superiority.	Fashion-MNIST	93.34
[27] 2023	Reinforcement Learning (RL) algorithm	no. of kernel, Kernel sizes, **A**ctiv**. F**un**.**, no. of neurons, learning rate	• It lacks comparison with other state-of-the-art studies.	MNIST	98.97
[11] 2023	deep CNN	Batch sizes, kernel sizes, batch normalization, Activ. Fun., and learning rate	• Model efficacy or comparison did not try another benchmark.	MNIST	99.4

3. Tools

This section introduces the main tools: SSA and CNNs, which are employed in our proposed methodology.

3.1. Convolution neural networks (CNNs)

CNNs have emerged as a leading class of deep neural networks in computer vision, achieving cutting-edge results in various tasks and revolutionizing the field, such as automotive safety, handwriting recognition, face detection, video surveillance, semantic segmentation, and speech recognition [28-33].

CNNs leverage the inherent spatial structure in images for superior performance compared to traditional neural networks. Their multi-stage architecture combines linear and non-linear operations. The initial feature extraction stage utilizes a series of convolutional (Conv.) layers, each followed by a pooling layer and activation function, enabling both local feature extraction and hierarchical representation building. In contrast, the classification stage employs several FC layers [34] to map these features to output categories.

Weight sharing lies at the heart of CNNs' success, significantly reducing the number of learnable parameters and enhancing their ability to learn generalizable patterns while avoiding overfitting. However, this efficiency comes at a cost: training CNNs demands substantial data, requiring considerable time, expertise, and manual construction. To address this data requirement, researchers have developed various optimization techniques to fine-tune hyperparameters and structures [35].

To avoid building CNNs from scratch, some researchers leverage transfer learning (TL), an approach that repurposes information from a pre-trained model to achieve superior intrusion detection performance compared to other models [36]. Within TL for deep learning models, fine-tuning can further enhance effectiveness. This technique involves retraining a subset of the pre-trained model's top layers on the new dataset while keeping the majority of the frozen (weight-preserved) layers intact [37]. Popular models suitable for TL include DenseNet, MobileNetV3, EfficientNet, VGG, GoogleNet, and Inception-ResNet.

The network's architecture comprises a carefully orchestrated stack of layers, each designed to process incoming data, extract meaningful features, and ultimately classify it according to the problem at hand. These layers include:

3.1.1. Input layer

The input layer, positioned at the leftmost edge of the network's architecture, serves as the entry point for raw image data. It encodes and presents this visual information to the subsequent layers for further processing.

3.1.2. Convolutional layers (Conv. layers)

Convolutional layers form the heart of CNNs, housing a collection of learned kernels (filters, weights) that meticulously extract diverse features from images. This extraction is achieved through a process known as convolution, in which each kernel meticulously scans the input image to uncover specific patterns. Each kernel specializes in detecting distinct features, such as edges, textures, or object parts, resulting in a rich tapestry of feature maps that illuminate the image's key characteristics [38].

Multiple convolutional kernels within CNNs allow them to detect a range of spatial patterns in images. This automatic feature extraction eliminates the need for manual engineering, a significant advantage. Its hyperparameters are set before using a convolutional layer, defining the output dimensions and number of connections in the resulting feature maps. These key hyperparameters include:

- Number of filters: determines the depth of the output feature maps.
- Filter size: defines the spatial dimensions of the kernels.
- Padding: specifies the amount of zero padding added to the input data.
- Stride: controls the step size of the kernel's movement across the input.

Following any trainable layer (i.e., convolutional or fully connected) in a CNN architecture comes a non-linear activation layer. These layers introduce non-linearity into the network, enabling it to learn complex concepts, unlike a purely linear model [8]. Sigmoid, hyperbolic tangent (tanh) and Rectified Linear Unit (ReLU) are common activation functions in CNNs. ReLU is the most popular due to its computational efficiency and ability to mitigate vanishing gradients.

3.1.3. Pooling layers

Pooling layers play a crucial role in CNNs by reducing the spatial dimensions of feature maps while preserving their essential characteristics. This significantly decreases computational demands and boosts the model's robustness to minor spatial variations. There are two types of pooling: max pooling (keeps the highest value) and average pooling (calculates mean). Each layer requires two hyperparameters adjustments before training:

- Pooling size: Defines the spatial dimensions of the pooling region.
- Stride: Determines the step size by which the pooling window moves across the feature map.

Following feature extraction, the extracted features feed into the next layer, usually an FC layer. This layer integrates information from the pooling layers to generate predictions for the input image. Before entering the FC layer, the previous layer's output must be transformed into a one-dimensional vector. This flattening process converts multi-dimensional feature maps into a format compatible with fully connected layers.

3.1.4. Fully connected layers (FC)

Connected (FC) layers, positioned close, to the end of a CNNs play a role in converting extracted features into either regression values or probabilities for different classes. They achieve this by connecting every neuron in the preceding layer with every neuron in the following layer allowing them to combine high-level features for predictions. Nevertheless, FC layers also bring about hyperparameters that need initialization and regularization to prevent overfitting. These include:

- Number of layers: While having hidden layers may enhance performance, it also raises the risk of overfitting. It's practice to start with two or three layers and adjust as necessary.
- Number of neurons: The complexity of the task determines the number of neurons assigned to each hidden layer. Tasks that are more intricate and require predictions will need several neurons. For instance, in image classification, the output layer may have neurons representing classes. In contrast, in object detection, it might contain neurons, for both calculating bounding box coordinates and determining class probabilities.
- Activation function: This function dictates how each neuron output is transformed. ReLU is a choice because of its efficiency and its ability to address vanishing gradients.
- Regularization: Methods such, as dropout, which randomly deactivates a portion of neurons during training can aid in averting overfitting by diminishing model intricacy and promoting generalization.

3.1.5. Output layer

The last layer outputs class predictions or regression values (e.g., probabilities). The choice of activation function (e.g., softmax for classification) depends on the task.

Layer count varies based on resources and task complexity. Training uses backpropagation (iteratively adjusting weights based on errors).

While powerful in learning spatial features, handling big data, and generalizing well, CNNs face challenges in their high computational cost and need for large datasets [39].

3.2. Salp Swarm Algorithm (SSA)

Inspired by the chain formation behavior of marine salps, the SSA is a population-based metaheuristic algorithm for optimization. Salps adjust their velocities and positions based on a combination of randomness, the current best solution, and its local neighbors, guiding the entire population towards better solutions [9].

SSA demonstrates effectiveness in diverse optimization tasks like parameter estimation, engineering design, and function optimization. Compared to other metaheuristics like PSO, GA, and Differential Evolution (DE), SSA offers several advantages [40]:

- Less parameter tuning: Requires fewer parameters to adjust, making it easier to optimize.
- Simple implementation: Straightforward implementation facilitates understanding and application.
- Robustness: Resists noise and outliers, making it suitable for unpredictable data.
- Efficiency: Scales well to large-scale optimization problems.

Fig. 1 visually illustrates the SSA pseudocode.

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: $c^1 = 2e^{-\left(\frac{4l}{L}\right)^2}$
  For each salp (x_i)
    if (i ==1)
      Update the position of the leading salp using:

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

    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

Fig. 1. Pseudocode of SSA [9].

3.3. Good-Point Set (GPS)

Good-point set are a method for selecting well-distributed points as initial samples for population-based optimization algorithms [41]. Unlike random initialization, GPS aim to provide uniform coverage that spans the entire objective function landscape. Popular GPS sampling strategies include Sobol sequences, Halton sequences, Hammersley sequences, and Latin Hypercube sampling. These low-discrepancy sequences cover the parameter space more efficiently by reducing gaps and clustering. Initializing the population with GPS promotes diversity, allowing for broader exploration and quicker convergence of stochastic optimization algorithms. This enhances the likelihood of discovering the global optimum. In summary, GPS initialize optimization algorithms more intelligently than randomization does, improving coverage through strategic, uniform sampling of the search space.

Fig. 2 outlines the mathematical procedure for introducing the GPS.

1: *Create a point* $= (r_1, r_2, \cdots, r_D)$, $r_i = \left\{2\ cos\frac{2\pi i}{p}\right\}$, *where* $1 \le i \le D$, *p represents the minimum prime number content with* $p \ge 2*D+3$.

2: *Let* $P_n(k) =$, *where* $\{r_i * k\}$ *is the decimal fraction of* $r_1 * k$, *k=1, 2, ..., SN, then set the points {P1, P2, ..., PSN} is referred to as a good point set.*

3: *The map is defined as follows:*

$X_{ij} = Lb_j + \{r_i * k\} * (Ub_j - Lb_j)$,

which means the good point is mapped to search space.

Fig. 2. Pseudocode of the GPS [42].

4. The proposed approach

To optimize CNN classification performance, we propose a new enhancement to the SSA for the architecture's hyperparameters selection. Our approach identifies the best settings for key parameters influencing the training process and model accuracy. We strategically focus on four crucial hyperparameters:
Dropout rate, hidden units, learning rate (which has a significant and wide-ranging impact on network behavior), and batch size.

To overcome the challenge of constraining the candidate solution population by the number of network weights, we leverage multiple iterations of the CNN architectures. Each iteration represents a different combination of values assigned to the chosen hyperparameters. Subsequently, we utilize the SSA to individually train each variant with a diverse set of candidate solutions.

4.1. Representation of individuals

To gauge the scalability and robustness of our CNNs-SSA approach, we implemented extensive training and evaluation across diverse CNN architecture, datasets, and hyperparameters settings. This iterative process yielded the optimal hyperparameters and configurations with defined lower and upper bounds presented in Table 2 Within our framework, a four-dimensional vector corresponding to the chosen CNN hyperparameters represents each individual (candidate solution).

Table 2: CNNs structure hyperparameters.

Hyperparameters	Range
Dropout rate	0.2, 0.3, 0.4
hidden units	64, 128, 256, 512
learning rate	0.01, 0.001, 0.0001, 1e-05
batch size	32, 64, 128

4.2. Initial population based on GPS

While replacing a random search space distribution with a uniform one could improve SSA effectiveness, it risks neglecting essential areas. Therefore, for wider solution space coverage in most scenarios, the initial distribution should be further refined. This study introduces a hybrid approach: the initial population is generated half-randomly and the remaining half leverages the GPS method for a more targeted exploration.

4.3. Fitness assessment

To measure the quality of an individual, we utilize the following objective function:

$$F = a \times MSE + b \times ep/epmx \quad (1)$$

Where ep and epmx are current and maximum epochs respectively. a and b $\in$ [0,1] and a+b =1.

This objective function, F, strikes a balance between model accuracy and speed. It achieves this by weighing the quality of the result with factor **a** and the epoch status with factor **b**. These factors, constrained to sum to 1, allow fine-tuning the priority between minimizing training error and encouraging training progression, thereby preventing both overfitting and stagnation.

5. Experimental results

5.1. Parameters Setting

Table 3 provides an overview of the parameters employed in this research, categorizing them into two sections: CNNs Training and SSA. The first category defines the core CNN architecture, including parameters like the number of convolutional layers, number of kernels per layer, kernel size, activation function for convolutional layers, stride and padding values, pooling size, number of pooling layers, specifications for hidden layers (number of neurons, etc.), activation function for fully-connected layers, activation function for the output layer, loss function, optimizer selection, evaluation metrics, and specified epoch (training iterations).

The second category governs the behavior of the SSA algorithm, encompassing the number of salps (population size), and the maximum number of iterations.

Table 3: Hyperparameters for current work.

CNNs parameters	Values
No. of Conv. layers	3
No. of kernels	1,2,3
Kernel size	512,256,256
Act. Fun. for Conv. layers	Relu
Batch Normalization	2
No. of pooling layers	2
Max Pooling size	3
Padding	Same
Stride	1
No. Hidden layers	3
Activ. Fun. for FC layers	Relu
Activ. Fun. for the out. layer	softmax
Loss function	Categorical-cross-entropy
Optimizer	Adam
Metrics	accuracy
Epochs	50
SSA parameters	**values**
No. of salps	10
Max-generations	100

5.2. Datasets

To benchmark our methodology's effectiveness, we selected the widely used MNIST and Fashion-MNIST datasets. Their popularity in the deep learning community, manageable sizes, and diverse content ensured thorough evaluation across different domains.

5.2.1. The MNIST dataset

The MNIST dataset, a cornerstone benchmark for image classification, features 70,000 greyscale images of handwritten digits, each pixelated canvas measuring 28×28 [43]. Divided into 60,000 training and 10,000 testing samples, MNIST boasts pre-centered and normalized data, further bolstered by a validation set split from the training data (55,000/5,000) [11]. While the training set fuels model development, the test set rigorously evaluates its

performance. This standardization allows researchers to directly compare their findings, fostering collaboration and progress in the field.

5.2.2. The Fashion-MNIST dataset

Fashion-MNIST mirrors its predecessor, offering 70,000 grayscale images of fashion items, neatly categorized into ten classes with 7,000 samples each. Each image, like its MNIST counterpart, occupies a 28×28 pixel canvas. The dataset divides neatly into a 60,000-image training set and a 10,000-image test set, making it a perfect substitute for the original MNIST due to its identical dimensions, data format, and split structure [12].

6. Results and analyses

Leveraging Google Colab Pro+ with its generous 500 monthly compute units, our GPU T4-powered environment, and Python 3 hardware acceleration facilitated the smooth execution of our method within 12-15 hours. To display its effectiveness, we compare our approach's accuracy against established methods on MNIST and Fashion-MNIST datasets, further highlighting the optimal architectures uncovered through this work. Detailed comparisons can be found in Figures 3 and 4.

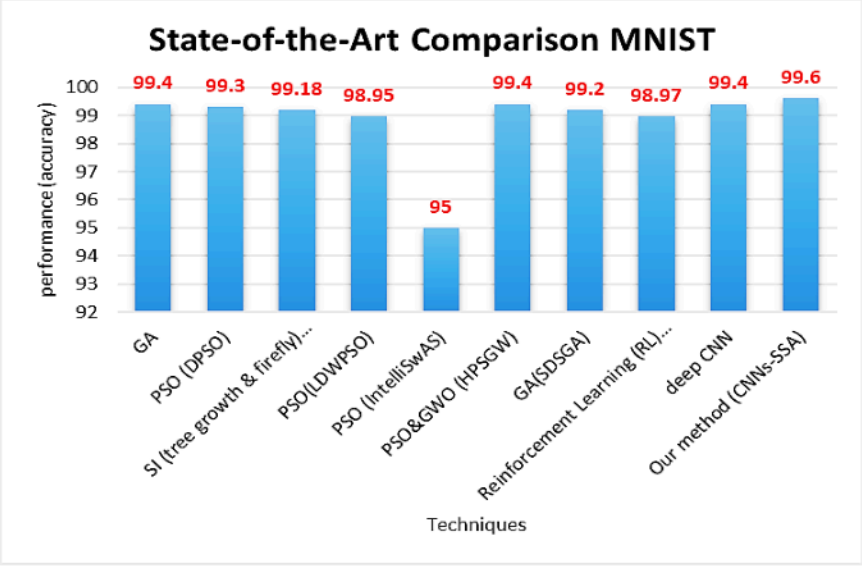

Fig. 3. A comparison of the accuracy of the MNIST dataset of the current work and various models.

As can be seen from Fig. 3, the CNNs-SSA technique shines on MNIST, achieving a remarkable 99.60% accuracy, edging past well-established rivals like PSO and deep CNNs. This slight, yet statistically significant advantage (further testing recommended) marks a promising advance in image classification. While other methods like GA and PSO variants come close, CNNs-SSA stands out, demonstrating its potential for wider application beyond MNIST. Exploring generalizability to more complex datasets and delving deeper into its internal workings will refine and solidify its claim as a top contender in the increasingly competitive field of image classification.

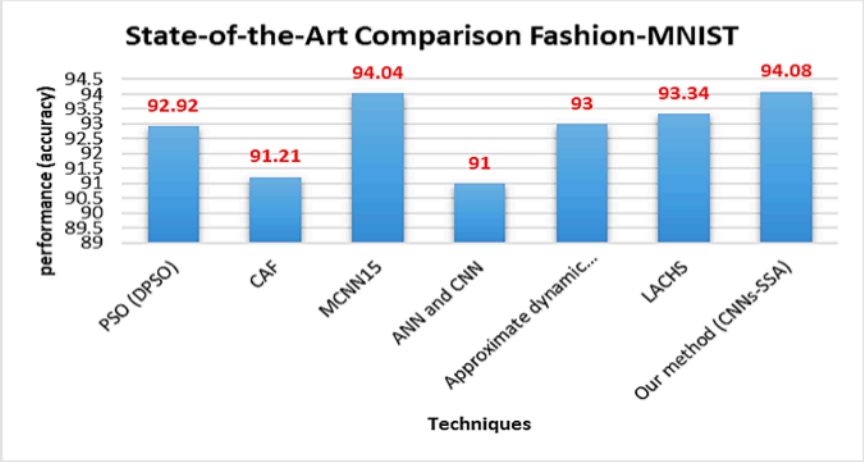

Fig. 4. A comparison of the accuracy of the Fashion-MNIST dataset of the current work and various models.

Fig. 4 convincingly demonstrates the superior performance of the CNNs-SSA method on the Fashion-MNIST dataset. It achieves the highest recorded accuracy (94.08%), surpassing well-established techniques like PSO (DPSO), CAF, multi-optimizers, and LACHS by margins ranging from 0.74% to 3.08%. Notably, even though its efficiency is comparable to MCNN15 (94.04%), CNNs-SSA's distinctive accuracy advantage highlights its potential for diverse image classification tasks, especially those with intricate categories like Fashion-MNIST.

We conducted a statistical analysis to compare the performance of CNNs with SSA and other optimization techniques. This analysis aimed to determine which architecture is superior. To do this, we used the Mann-Whitney U test as a nonparametric method [44, 45]. The test analyzed the accuracy values of both architectures.

We set the null hypothesis (H_0) that both architectures yield the same accuracy ($\mu_0 = \mu_1$). The alternative hypothesis (H_1) stated that the CNN with SSA achieves higher accuracy than those with other techniques ($\mu_1 > \mu_0$). We set a significance level of 99% ($\alpha = 0.01$).

Table 4: Statistical analyses (MNIST dataset)

Ref.	Methods	P-Value
[13]	GA	0.002
[14]	PSO (DPSO)	0.001
[44]	SI (tree growth & firefly) algorithms	0.001
[16]	PSO(LDWPSO)	< 0.000
[19]	PSO (IntelliSwAS)	< 0.000
[21]	PSO&GWO (HPSGW)	< 0.000
[22]	GA (SDSGA)	< 0.000
[27]	Reinforcement Learning (RL) algorithm	< 0.000
[11]	Deep CNN	< 0.000

Table 5: Statistical analyses (Fashion-MNIST dataset)

Ref.	Methods	P-Value
[14]	PSO (DPSO)	0.008
[18]	CAF	0.005
[23]	MCNN15	0.003
[24]	ANN and CNN	0.002
[25]	Approximate dynamic learning rate Update algorithm, ResNet, and VGG16	0.001
[26]	LACHS	0.001

The results of the Mann-Whitney U test are presented in Tables 4 and 5 for both the MNIST and Fashion-MNIST datasets. For both datasets, the p-values are less than the chosen significance level ($p < 0.01$). Based on these findings, we reject the H_0 and conclude that for these specific datasets, auto-tuned CNNs with SSA outperform those with other optimization techniques in terms of accuracy.

Table 6 respectively displays the individuals (hyperparameters configurations) that achieved the highest accuracies on the MNIST and Fashion-MNIST evaluations.

Table 6: Optimal selection for the MNIST and Fashion-MNIST datasets.

Dataset	Dropout rate	hidden units	learning rate	batch size
MNIST	0.4	256	0.0001	64
Fashion-MNIST	0.2	128	0.0001	32

7. Conclusion

This work introduces CNNs-SSA, a new strategy for optimizing CNNs by leveraging an enhanced SSA. CNNs-SSA excels in balancing accuracy, computational speed, and training duration, as demonstrated by its outstanding performance on MNIST and Fashion-MNIST datasets. Notably, it outperforms computationally demanding algorithms, making it well suited for practical settings with limited resources and time constraints. This approach paves the way for seamless integration of CNNs into real-world applications, particularly those facing resource limitations. Future research could explore the applicability of SSA-based optimization to diverse deep learning architectures and tasks beyond computer vision. Additionally, delving deeper into the theoretical underpinnings of SSA and refining hyperparameters optimization strategies could unlock its broader potential in machine learning applications.

References

[1] Y. Bengio, I. Goodfellow, and A. Courville, "Deep learning (Vol. 1)," MIT Press Cambridge, MA, USA, vol. 22, pp. 23-24, 2017. DOI: 10.1007/s10710-017-9314-z

[2] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review,* vol. 40, p. 100379, 2021. doi https://doi.org/10.1016/j.cosrev.2021.100379.

[3] M. Abdulla and A. Marhoon, "Agriculture based on Internet of Things and Deep Learning," *Iraqi Journal for Electrical and Electronic Engineering,* vol. 18, no. 2, pp. 1-8, 2022. http://dx.doi.org/10.37917/ijeee.18.2.1.

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

[5] N. F. A. Hassan, 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),* vol. 12, no. 5, 2022. doi: http://dx.doi.org/10.11591/ijece.v12i5.pp5427-5434.

[6] Y. Wang, 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,* vol. 49, pp. 114-123, 2019. doi: http://dx.doi.org/10.1016/j.swevo.2019.06.002.

[7] A. Darwish, 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,* vol. 52, p. 100616, 2020, doi: http://dx.doi.org/10.1016/j.swevo.2019.100616.

[8] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J Big Data,* vol. 8, no. 1, p. 53, 2021, doi: 10.1186/s40537-021-00444-8.

[9] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," *Advances in engineering software,* vol. 114, pp. 163-191, 2017, doi: https://doi.org/10.1016/j.advengsoft.2017.07.002.

[10] H. Zhang *et al.*, "Differential evolution-assisted salp swarm algorithm with chaotic structure for real-world problems," *Engineering with Computers,* vol. 39, no. 3, pp. 1735-1769, 2023, doi: https://doi.org/10.1007/s00366-021-01545-x.

[11] H. Shao, E. Ma, M. Zhu, X. Deng, and S. Zhai, "MNIST Handwritten Digit Classification Based on Convolutional Neural Network with Hyperparameter Optimization," *Intelligent Automation & Soft Computing,* vol. 36, no. 3, 2023. https://doi.org/10.32604/iasc.2023.036323

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

[13] J.-H. Yoo, H.-i. Yoon, H.-G. Kim, H.-S. Yoon, and S.-S. Han, "Optimization of hyper-parameter for CNN model using genetic algorithm," in *2019 1st International conference on electrical, control and instrumentation engineering (ICECIE)*, Kuala Lumpur, Malaysia, 2019: IEEE, 2019, pp. 1-6, doi: 10.1109/ICECIE47765.2019.8974762.

[14] Y. Guo, 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,* vol. 52, no. 1, pp. 36-57, 2020, doi: 10.1080/01969722.2020.1827797.

[15] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, and M. Tuba, "Optimizing Convolutional Neural Network Hyperparameters by Enhanced Swarm Intelligence Metaheuristics," *Algorithms,* vol. 13, no. 3, 2020, doi: 10.3390/a13030067.

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

[17] K. Greeshma and J. V. Gripsy, "Image classification using HOG and LBP feature descriptors with SVM and CNN," *Int J Eng Res Technol,* vol. 8, no. 4, pp. 1-4, 2020.

[18] Y. Ying, N. Zhang, P. He, and S. Peng, "Improving convolutional neural networks with competitive activation function," *Security and Communication Networks,* vol. 2021, pp. 1-9, 13 May 2021 2021, Art no. 1933490 doi: https://doi.org/10.1155/2021/1933490.

[19] S. C. Nistor and G. Czibula, "IntelliSwAS: Optimizing deep neural network architectures using a particle swarm-based approach," *Expert Systems with Applications,* vol. 187, p. 115945, 2022. https://doi.org/10.1016/j.eswa.2021.115945

[20] E. E. Moodie and D. A. Stephens, "Comment: Clarifying endogeneous data structures and consequent modelling choices using causal graphs," 2020. https://doi.org/10.1214/20-sts777.

[21] J. R. Challapalli 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,* vol. 64, no. 9, pp. 2411-2434, 2022. https://doi.org/10.1007/s10115-022-01707-3

[22] I. D. Raji, H. Bello-Salau, I. J. Umoh, A. J. Onumanyi, M. A. Adegboye, and A. T. Salawudeen, "Simple deterministic selection-based genetic algorithm for hyperparameter tuning of machine learning models," *Applied Sciences,* vol. 12, no. 3, p. 1186, 2022. https://doi.org/10.3390/app12031186

[23] O. Nocentini, J. Kim, M. Z. Bashir, and F. Cavallo, "Image classification using multiple convolutional neural networks on the fashion-MNIST dataset," *Sensors,* vol. 22, no. 23, p. 9544, 2022. https://doi.org/10.3390/s22239544.

[24] S. R. Sumera, N. Anjum, and K. Vaidehi, "Implementation of CNN and ANN for Fashion-MNIST-Dataset using Different Optimizers," *Indian Journal of Science and Technology,* vol. 15, no. 47, pp. 2639-2645, 2022. https://doi.org/10.17485/ijst/v15i47.1821.

[25] S.-Y. Shin, G. Jo, and G. Wang, "A Novel Method for Fashion Clothing Image Classification Based on Deep Learning," *Journal of Information and Communication Technology,* vol. 22, no. 1, pp. 127-148, 2023, doi: https://doi.org/10.32890/jict2023.22.1.6.

[26] D. Liu, H. Ouyang, S. Li, C. Zhang, and Z.-H. Zhan, "Hyperparameters Optimization of Convolutional Neural Network Based on Local Autonomous Competition Harmony Search Algorithm," *Journal of Computational Design and Engineering,* p. qwad050, 2023. https://doi.org/10.1093/jcde/qwad050

[27] F. M. Talaat and S. A. Gamel, "RL based hyper-parameters optimization algorithm (ROA) for convolutional neural network," *Journal of Ambient Intelligence and Humanized Computing,* vol. 14, no. 10, pp. 13349-13359, 2023. https://doi.org/10.1007/s12652-022-03788-y

[28] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Computing and Applications,* vol. 33, no. 7, pp. 2249-2261, 2021. https://doi.org/10.1007/s00521-020-05070-8

[29] L. Ren, J. Dong, X. Wang, Z. Meng, L. Zhao, and M. J. Deen, "A data-driven auto-CNN-LSTM prediction model for lithium-ion battery remaining useful life," *IEEE Transactions on Industrial Informatics,* vol. 17, no. 5, pp. 3478-3487, 2020. https://doi.org/10.1109/tii.2020.3008223

[30] A. H. Ashraf *et al.*, "Weapons detection for security and video surveillance using cnn and YOLO-v5s," *CMC-Comput. Mater. Contin,* vol. 70, pp. 2761-2775, 2022. https://doi.org/10.32604/cmc.2022.018785

[31] M. Zamir *et al.*, "Face Detection & Recognition from Images & Videos Based on CNN & Raspberry Pi," *Computation,* vol. 10, no. 9, p. 148, 2022. https://doi.org/10.3390/computation10090148

[32] C. Li, W. Xia, Y. Yan, B. Luo, and J. Tang, "Segmenting objects in day and night: Edge-conditioned CNN for thermal image semantic segmentation," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 32, no. 7, pp. 3069-3082, 2020. https://doi.org/10.1109/tnnls.2020.3009373

[33] M. A. Haque, A. Verma, J. S. R. Alex, and N. Venkatesan, "Experimental evaluation of CNN architecture for speech recognition," in *First International Conference on Sustainable Technologies for Computational Intelligence: Proceedings of ICTSCI 2019*, 2020: Springer, pp. 507-514. https://doi.org/10.1007/978-981-15-0029-9_40

[34] R. S. Khudeyer and N. M. Almoosawi, "Combination of machine learning algorithms and Resnet50 for Arabic Handwritten Classification," *Informatica,* vol. 46, no. 9, 2023 https://doi.org/10.31449/inf.v46i9.4375.

[35] J. Fregoso, C. I. Gonzalez, and G. E. Martinez, "Optimization of convolutional neural networks architectures using PSO for sign language recognition," *Axioms,* vol. 10, no. 3, p. 139, 2021.

[36] X. Li, Z. Hu, M. Xu, Y. Wang, and J. Ma, "Transfer learning based intrusion detection scheme for Internet of vehicles," *Information Sciences,* vol. 547, pp. 119-135, 2021. https://doi.org/10.1016/j.ins.2020.05.130

[37] O. D. Okey, D. C. Melgarejo, M. Saadi, R. L. Rosa, J. H. Kleinschmidt, and D. Z. Rodríguez, "Transfer learning approach to IDS on cloud IoT devices using optimized CNN," *IEEE Access,* vol. 11, pp. 1023-1038, 2023. https://doi.org/10.1109/access.2022.3233775

[38] N. M. Almoosawi and R. S. Khudeyer, "ResNet-34/DR: a residual convolutional neural network for the diagnosis of diabetic retinopathy," *Informatica,* vol. 45, no. 7, 2021. https://doi.org/10.31449/inf.v45i7.3774

[39] E. Abdulsaed, M. Alabbas, and R. Khudeyer, "Hyperparameter Optimization for Convolutional Neural Networks using the Salp Swarm Algorithm," *Informatica,* vol. 47, no. 9, 2023. https://doi.org/10.31449/inf.v47i9.5148

[40] H. Faris, S. Mirjalili, I. Aljarah, M. Mafarja, and A. A. Heidari, "Salp swarm algorithm: theory, literature review, and application in extreme learning machines," *Nature-inspired optimizers: theories, literature reviews and applications,* pp. 185-199, 2020. https://doi.org/10.1007/978-3-030-12127-3_11

[41] A. Q. Obaid and M. Alabbas, "Hybrid Variable-Length Spider Monkey Optimization with Good-Point Set Initialization for Data Clustering," *Informatica,* vol. 47, no. 8, 2023, doi: https://doi.org/10.31449/inf.v47i8.4872.

[42] D. Liu, S. Zhang, B. Wang, and Z. Li, "Seagull algorithm based on good point set and dual hybrid strategy," in *International Conference on Cloud Computing, Performance Computing, and Deep Learning (CCPCDL 2023)*, 2023, vol. 12712: SPIE, pp. 79-84. https://doi.org/10.1117/12.2678849

[43] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, and M. Tuba, "Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics," *Algorithms,* vol. 13, no. 3, p. 67, 2020. https://doi.org/10.3390/a13030067

[44] B. Rosner and D. Grove, "Use of the Mann–Whitney U-test for clustered data," *Statistics in medicine,* vol. 18, no. 11, pp. 1387-1400, 1999. https://doi.org/10.1002/(sici)1097-0258(19990615)18:11<1387::aid-sim126>3.0.co;2-v

[45] S. Ioannou, H. Chockler, A. Hammers, A. P. King, and A. s. D. N. Initiative, "A study of demographic bias in CNN-based brain MR segmentation," in *International Workshop on Machine Learning in Clinical Neuroimaging*, 2022: Springer, pp. 13-22, doi: https://doi.org/10.48550/arXiv.2208.06613.