International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 10-No.4 (2024) pp. 718-730
http://www.ijcesen.com

Research Article

ISSN: 2149-9144

Enhancing Food Image Classification with Particle Swarm Optimization on NutriFoodNet and Data Augmentation Parameters

Sreetha E. S.[1]*, G. Naveen Sundar[2], D. Narmadha[3]

[1]Department of Computer Science and Engineering, Karunya Institute of Science and Technology, Tamilnadu, India
Christ College of Engineering. Kerala, India
* **Corresponding Author Email:** sreethasreedharan@gmail.com - **ORCID:** 0000-0002-7070-2687

[2]Department of Computer Science and Engineering, Karunya Institute of Science and Technology, Tamilnadu, India
Email: naveensundar@karunya.edu - **ORCID:** 0000-0002-4638-7867

[3]Department of Computer Science and Engineering, Karunya Institute of Science and Technology, Tamilnadu, India
Email: narmadha@karunya.edu - **ORCID:** 0000-0002-3384-4096

Article Info:

DOI: 10.22399/ijcesen.493
Received: 08 October 2024
Accepted: 14 October 2024

Keywords:

Food recognition
Data augmentation
Convolutional Neural Network
Particle Swarm Optimization
NutriFoodNet

Abstract:

A convolutional neural network (CNN) architecture, NutriFoodNet, enhanced through Particle Swarm Optimization (PSO) is suggested in this paper to optimize data augmentation parameters and key hyperparameters, specifically designed for food image recognition. Accurate food image classification plays a vital function in various applications, including nutrition management, dietary assessment, and healthcare, as it aids in the automated recognition and analysis of food items from images. The implementation aimed to improve classification accuracy on the Food101 dataset. Initially, the NutriFoodNet model achieved an accuracy of 97.3%. The model's performance was further refined by applying PSO, resulting in an increased accuracy of 98.5%. This optimized system was benchmarked against state-of-the-art architectures, including ResNet-18, ResNet-50, and Inception V3, showcasing its exceptional performance. The proposed system highlights the efficiency of PSO in fine-tuning augmentation parameters and CNN hyperparameters, leading to significant improvements in model accuracy for food image classification tasks. This advancement underscores the potential of enhanced food image classification systems in contributing to better dietary monitoring and healthcare outcomes.

1. Introduction

The food sector has also undergone a revolution in research yield, especially in the area of food image classification. Previously, distinguishing between different types of food items was relatively straightforward due to the limited variety. However, with the abundance of food options available today, accurately classifying them has become a daunting task. This challenge is a fundamental problem in image analysis and necessitates the development of intelligent food classification systems. Food image classification holds great importance in applications ranging from dietary analysis to food recommendation systems and nutritional monitoring. A significant obstacle in the categorization of images of food is the wide variety of food appearances, encompassing changes in color, shape, and texture. This diversity makes it challenging for humans to accurately classify food items based solely on visual inspection, let alone for algorithms running on computers. Still, new developments in deep learning have opened the door to more potent remedies for this issue. The NutriFoodNet model, a modification of the Inception V3 architecture, stands out as a high-accuracy model specifically designed for food image classification tasks. By leveraging deep learning techniques, NutriFoodNet addresses the complexities of food classification and offers improved accuracy and efficiency compared to traditional handcrafted methods. Currently, machine learning (ML) methods are used for image processing, which includes support vector machines (SVM), Gaussian

Naïve Bayes, Radial Basis Function Networks and Deep Neural Networks(DNN). CNN is the most recent and has demonstrated an impressive [1]. The architecture of CNN was modelled after the biological organization of the visual cortexes of mammals. CNNs are designed with various layers, including fully connected, pooling, and convolution. By stacking many layers, CNN automatically extracts features from input data [2]. Lenet and Alex-Net are regarded as the foundational CNN architectures [3]. CNN experienced significant growth in the field of computer vision as a result of these architectures' outstanding performance. Strong regularization techniques, GPU-accelerated computation, and the use of big datasets are further factors contributing to its effectiveness [4].

Data augmentation is a crucial step in CNN training because it increases the variety and volume of training data, which enhances the model's capacity for generalization and minimizes overfitting. However, manually tuning the augmentation parameters can be time-consuming and may not always lead to optimal results [5]. When constructing the CNN, certain structural considerations have to be made. Some of the factors include the total count of convolution layers, count of pooling layers, filter size, count of filters, stride rate, and location of the pooling layer. Finding the ideal set of criteria for outstanding performances is challenging since there are too many of them. Researchers are using a method of trial-and-error for designing architecture. To find the ideal set of parameters, some researchers employ grid search or random search strategies. While these methods aid in creating a good mixture, they are laborious and need a lot of processing power. The proper hyperparameter combination is now being studied by researchers as an optimization problem. The accuracy, precision, and recall of automated approaches are higher than those of manually designed structures. These automated designs lower the misclassification rate by combining hyperparameters based on past knowledge.

The proposed system for food image classification has two main objectives:

1. Optimization of parameters such as rotation angle, shear, flipping, etc., in data augmentation to bolster the robustness and generalization of CNN models.

2. Application of optimization techniques for selecting hyperparameters within the CNN architecture, including learning rate, dropout rate, and architecture of the network, to improve the classification model's effectiveness and efficiency.

The remainder of the document is structured in this manner. A thorough analysis of optimization techniques for convolutional neural networks is provided in Section 2. It highlights the principles and mechanisms of PSO as a powerful metaheuristic algorithm. Section 3 offers background information on the system, including details about the dataset and the application of PSO in data augmentation and model parameter optimization. The proposed system architecture and algorithm are detailed in Section 4, elucidating the integration of PSO into the training process to optimize data augmentation parameters and CNN hyperparameters. Section 5 conducts a comprehensive performance analysis, comparing the proposed system with existing methods and benchmarks. The work is finally concluded in Section 6, which summarizes the main conclusions and outlines future research directions for improving food picture categorization technologies.

2. Literature Review

The work of M.A.K. Raiaan et al. focuses on CNN optimization for a variety of ML problems utilizing PSO approaches [6]. It may highlight the importance of hyperparameter optimization in enhancing CNN performance, discuss the effectiveness of PSO algorithms in fine-tuning hyperparameters, and emphasize the potential impact on model accuracy and efficiency. The abstract likely serves as a brief overview of the study's objectives, methodologies, and key findings related to utilizing PSO techniques for optimizing CNN architectures. Various hyperparameter optimization algorithms are Tree-structured Parzen Estimator(TPE) [7], Gray Wolf Optimization(GWO) [8], Harmony Search Algorithm(HSA) [9], Differential Evolution(DE) [10], Genetic Algorithm Optimization(GAO) [11], Ant Colony Optimization(ACO) [12], Particle Swarm Optimization(PSO) [13], Firefly Algorithm(FA) [14], Nelder–Mead Method(NMM) [15] and Bayesian Optimization(BO) [16]. The table 1 and graphic show the accuracy of optimization techniques on the benchmark dataset. The PSO optimization techniques outperform well among these. A nature-inspired technique for finding the best neural network architecture is PSO. PSO can be made use to evolve neural network designs and weights similarly to genetic algorithms. Gudise and Venayagamoorthy developed a work utilizing PSO to train an Artificial Neural Network(ANN) in 2003 [17-30]. They demonstrated how ANNs PSO training is faster than regular backpropagation training. In a similar vein, Carvalho and Ludermir [31,32] created two distinct PSO algorithms in 2007 in order

Table 1. *Accuracy of Optimization algorithms using reference dataset*

Dataset	HSA	ACO	PSO	FA	NM
MNIST	99.25% [17]		99.89% [18]	99.16% [19]	
Fashion MNIST	97.96% [20]	93.40% [21]	92.67% [22]		
CIFAR 10	74.76% [17]	84.80% [22]	89.50% [23]	96.70% [24]	83.63% [25]
CIFAR 100		89.79% [26]	71.55% [18]	77.75%[24]	
Caltech 101	92.88% [27]		79.80% [28]		55.70% [29]

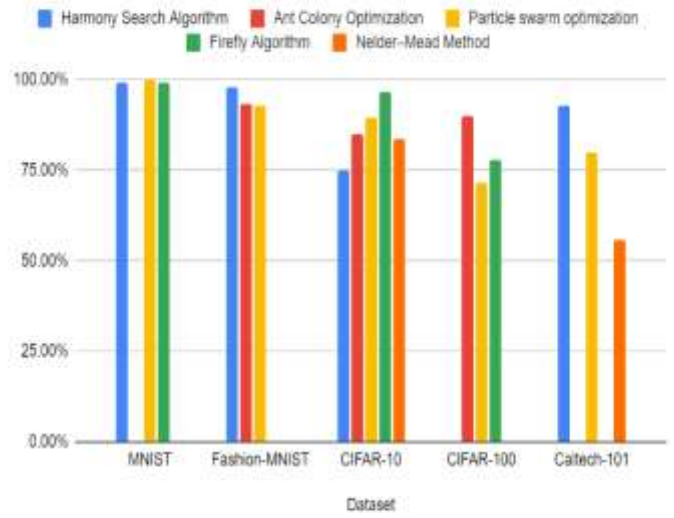


Figure 1. *Accuracy of Optimization algorithms using reference dataset*

to train ANNs and look for better architectures. When compared to alternative techniques, they demonstrated that PSO could be utilized to enhance ANN structures and produce results that were competitive. Nevertheless, the PSO method created there and in subsequent publications [33-36] can only look for fully connected neural network optimum topologies, which are unsuitable for image classification applications.

Fernandes Junior et.al. explores the practical application of PSO in automating the design of DNN architectures for the tasks involving classification of images. The proposed PSO algorithm, named psoCNN, aims to optimize the architecture of CNNs to improve classification performance. They introduce the concept of using meta-heuristic algorithms like PSO to automatically search for optimal network configurations, which can lead to improved performance and efficiency [37]. Aguerchi et.al. focuses on hyperparameter optimization for breast cancer classification using CNN in mammography images. They provide a number of hyperparameters that are optimised when CNN models are employed for classification. Specifically, PSO was used to search for suitable values for the parameters like: Kernel size, Stride and Filter number [38]. A PSO-based deep learning model for vehicle categorization is discussed by A. Alhudhaif et al. The research utilizes the GoogleNet architecture for feature extraction and implements PSO for feature selection to optimize computational burden and improve accuracy. Various CNNs are tested, with ResNet50 achieving the highest accuracy of 91.28%. With the used dataset, the accuracy of the model suggested is 96%. The Cubic SVM (CSVM) classifier is chosen for the model due to its balance of accuracy and time efficiency. The research contributes to advancements in image processing, computer vision, and intelligent transport systems, with potential applications in traffic analysis, surveillance, and security management [39]. Muhammad Asif Saleem et.al., discusses the development of a CNN architecture optimized for the early detection of paddy leaf diseases. Researchers have proposed optimization algorithms, such as firefly algorithms, Genetic Algorithm, Bayesian Optimization, and Evolutionary Algorithms, to improve the performance of architectures. To provide the best CNN architecture for a given dataset, the system presents an efficient CNN design based on MUT-PSO [40].

Liu, X et.al., focuses on enhancing hyperspectral image classification through a novel approach called Continuous PSO based Deep Learning Architecture Search. The paper investigates how applying this technique can improve deep learning models' performance and accuracy in picture categorisation tasks. By utilizing Continuous Particle Swarm Optimization, the researchers aim to optimize neural network architectures for hyperspectral image analysis [41]. Zhou C. et al. offer a unique method for super-resolution imaging by modifying the PSO algorithm. The research aims to enhance image super-resolution performance on lightweight architectures by introducing a joint training method that improves efficiency and accuracy. The proposed mutation strategy prevents premature optimization, and experiments on chest X-ray image classification demonstrate the model's ability to reconstruct useful information for pneumonia recognition [42]. Nistor et.al., proposed in the IntelliSwAS (Intelligent

Swarm Architecture Search) approach involves the integration of PSO with a machine learning model called DAGRNN (Directed Acyclic Graph Recurrent Neural Network) to optimize deep CNN architectures for image classification tasks [43].

3. Background of the System

3.1. Dataset: FOOD101

To achieve optimal classification accuracy, having a robust dataset of images is crucial. The dataset is built by considering the variety of food classes and types. An overview of the available food-related datasets, detailing the number of food classes and images are given in [44]. A large volume of food images is essential for training a food classification model, as deep learning techniques require substantial amounts of data. Food101, UEC-100, UEC-256, Food 85, and PFID are some of the widely used datasets for food images. Among these, Food101 is regarded as a benchmark dataset for food classification tasks.Th e Food101 dataset is a pioneering resource in the area of computer vision for food recognition tasks [45]. Curated from foodspotting.com, it features 101 distinct food categories with1000 images per category, intentionally retaining noise like intense colors and occasional mislabels to simulate authentic scenarios. The dataset's diversity encompasses visually and semantically similar dishes challenging algorithms to differentiate between subtle variations. The dataset is publicly accessible for developing and evaluating robust food recognition models capable of handling the complexities and nuances present in real-world food imagery. Sample images of the Food101 dataset are shown in the figure 2.

Figure 2 *Sample images from the Food101 dataset*

3.2 Data augmentation

One important method in deep learning-based systems, particularly for computer vision applications, is described as data augmentation. In scenarios when training data is scarce, of low quality, or non-existent, it entails expanding training data to support machine learning models in producing satisfactory results. Improving the amount, quality, and diversity of data to be taught is the main goal of data augmentation, which aims to increase the efficacy and reliability of machine learning (ML) models [46].

3.3 CNN architecture

One particular kind of deep learning model is a CNN, designed to analyze and interpret visual data. CNNs employ a series of convolutional layers, which use filters to scan input images and extract essential features like edges and textures. Activation functions, sometimes known as ReLUs (Rectified Linear Units), come after these layers and add non-linearity to the model. Next, pooling layers—like maxpooling—are applied to reduce the spatial dimensions of the data. This increases computing efficiency and lowers the likelihood of overfitting in the model. This hierarchical feature extraction allows CNNs to recognize increasingly complex patterns as the data progresses through the layers.

After the convolutional and pooling layers, the data is flattened into a single vector and passed through fully connected (dense) layers. These layers combine the extracted features to make the final prediction, often using a softmax function for classification tasks. The structure of CNNs, with shared parameters and local connectivity, makes them particularly effective for applications such as image classification, object detection, and segmentation. Their ability to maintain translation invariance ensures robust recognition of objects regardless of their position within an image. CNNs are widely used in various applications, from facial recognition and autonomous driving to medical image analysis and beyond, demonstrating their versatility and powerful performance in handling visual data [47-51]. A typical CNN architecture is shown in the figure 3.

3.4 Particle Swarm Optimization

In 1995, PSO—a naturalistic optimization technique—was introduced. Particles in PSO navigate the search space repeatedly to identify the optimal solution to a problem. The main components of a PSO algorithm are a fitness-function to evaluate the quality of a solution, a population of particles

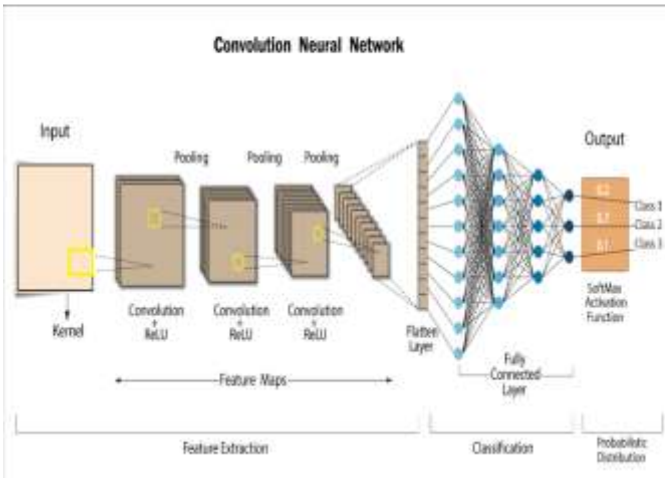


Figure 3. *CNN architecture*

with position and velocity in the search space, and techniques to modify the particle positions based on their own best-known-position (pbest) and the best-known-position of the swarm (gbest).

Particle positions and velocities in PSO are determined by equations that control particle movement in the search space. The standard equations in the PSO algorithm include:

Velocity Update Equation:

$$v_i(t+1)=w.v_i.(t)+c_1.r_1.(p_i-x_i)+c_2.r_2.(p_g-x_i) \quad (1)$$

where, $v_i(t+1)$ means the particle's velocity at the subsequent iteration, w is the weight of inertia, the coefficients of acceleration are c_1 and c_2, r_1 and r_2 are the random values ranging from 0 to 1. p_i is the best-known-position of particle, p_g is the best-known swarm position and Particle's current position is indicated by x_i.

Position Update Equation:

$$x_i(t+1)=x_i.(t)+v_i(t+1) \quad (2)$$

where $x_i(t+1)$ represents the particle's position i at the next iteration.

By updating their velocities and positions according to their individual experience (pbest) and the collective knowledge of the swarm (gbest), these equations control the movement of particles toward optimal solutions in the search space.

3.5 PSO in data augmentation and hyperparameters selection in CNN

In the context of CNNs, PSO can be made use of optimizing data augmentation parameters and hyperparameters. PSO explores various transformation parameters such as rotation, shift, shear, zoom, and flip to increase the training dataset's diversity for data augmentation artificially. This process helps improve CNN's robustness and generalization. By defining a fitness function based on the performance of the model, PSO iteratively adjusts the augmentation parameters, guiding the swarm of potential solutions toward the optimal set that yields the highest validation accuracy.

In addition to data augmentation, PSO is effective for hyperparameter optimization in CNNs. Epochs, learning rate, and the number of dense layers are the hyperparameters that significantly influence the training process and final model performance. PSO is used to initialise a swarm of particles, each of which indicates a possible solution. This automates the process of finding the optimal hyperparameter set. Iteratively convergent to the optimal hyperparameter configuration, these particles modify their locations and velocities in accordance with both their personal and the global best solutions. This approach ensures an efficient and comprehensive exploration of the hyperparameter space, often outperforming traditional methods like grid search or random search due to its ability to avoid local optima and efficiently handle the high-dimensional search space.

4. Proposed System

The figure 4 represents the overview of the proposed system.

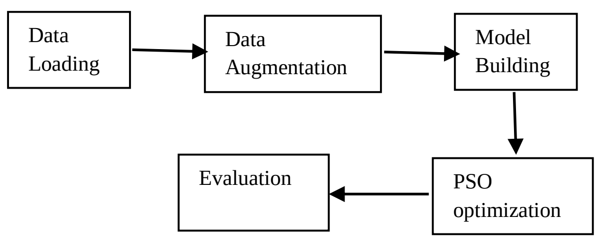


Figure 4. *Block diagram of Proposed System food recognition*

4.1 Load and Pre-process Data

The function load_data is designed to efficiently load an image dataset Food101 from a specified directory, pre-process the images, and divide the data into train and test sets. Initially, it initializes two empty lists, X and Y, to store image data and corresponding labels. It reads the names of subdirectories within the specified dataset, with each subdirectory representing a different class of food images. Once the images have loaded, resize them to 150 by 150 pixels and convert them to a numpy array. List X receives the appended picture array, while list Y receives the appended class label. After processing all images, the X and Y lists are converted to numpy arrays for efficient numerical computations and model training. Subsequently, the function divides the data into training and testing sets, designating 80% of the data for training and 20% for testing. To guarantee reproducibility, a random state is supplied. Finally, the function returns the training and testing data arrays. This procedure sets up the image data for subsequent steps in the image classification pipeline, such as data augmentation and model training.

4.2 Define Data Augmentation Function

The function `augment_data` is designed to apply data augmentation to an image dataset based on specified parameters. An `ImageDataGenerator` is initialized with augmentation settings including width, zoom range, shear range, height shift range, horizontal flip, rotation range and probability. The `horizontal_flip` parameter is converted from a continuous [0, 1] value to a boolean, determining whether horizontal flips should be applied. The generator is then fitted to the input images, `X`, and an augmented data generator is returned. This generator produces augmented image batches, `X`, and their corresponding labels, `Y`, in batches of 32. This augmented data generator can be used to train ML models with on-the-fly augmented data. These transformations can be represented by the following mathematical equations:

Width and Height Shift:

$$shifted\_width(x, shift\_range) = x + \Delta x \quad (3)$$

$$shifted\_height(y, shift\_range) = y + \Delta y \quad (4)$$

$$\Delta x = shift\_range.W \quad (5)$$

$$\Delta y = shift\_range.H \quad (6)$$

Where W represents the Width and H represents the Height of the image.

Rotation:

$$rotated_{image(x^{|}, y^{|}, \theta)} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (7)$$

where $(x^{|}, y^{|})$ are the coordinates of the rotated image and θ is the angle of rotation.

Zoom:

$$zoomed\_image(x^{|}, y^{|}, zoom\_factor) = \begin{bmatrix} zoom\_factor & 0 \\ 0 & zoom\_factor \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (8)$$

where $zoom\_factor$ is the scaling factor for the zoom.

Shear:

$$sheared\_image(x^{|}, y^{|}, \lambda) = \begin{bmatrix} 1 & \lambda \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (9)$$

where λ is the shear factor.

Horizontal Flip:

The $horizontal\_flip$ parameter is converted from a continuous [0,1] value to a boolean.

$$horizontal\_flip = \begin{cases} True & if\ flip\_probability > 0.5 \\ False & otherwise \end{cases} \quad (10)$$

where $flip\_probability$ is the value generated randomly within the range [0,1]

4.3 Build and Compile the Model

A function is designed to construct a fine-tuned image classification model using the pre-trained NutriFoodNet architecture [52]. Necessary modules are imported, and the NutriFoodNet model is loaded excluding its top layers. The input shape is set according to the provided dataset. NutriFoodNet is a CNN model for the categorization of images with 101 output classes; several hyperparameters have been optimized to enhance performance. 30% of the neurones are randomly deactivated during training at a dropout rate of 0.3 to avoid overfitting. Because of the ReLU' effectiveness in creating non-linearity and minimizing the vanishing gradient problem, it is utilized. By normalizing layer inputs, batch normalization is used for regularisation, which stabilizes and speeds up the training process. Using 0.001 as the learning rate, the Adam optimizer is selected due to its ability to learn adaptively and its effectiveness in managing sparse gradients.

4.4 Define Objective Function

An objective function is designed for PSO, aiming to optimize both data augmentation parameters and model hyperparameters for enhanced performance in an image classification task. A list of parameters is received, which is then split into augmentation parameters and model hyperparameters. The training dataset is subjected to data augmentation using the specified augmentation parameters, and a model is built with the provided hyperparameters. After the model is compiled, it is trained on the augmented data, its accuracy is calculated on a separate test set, and the negative accuracy is returned. By returning the negative accuracy, alignment with PSO's goal of minimizing the objective function is ensured, effectively maximizing the model's accuracy through the optimization process.

4.5 PSO to find the optimal parameters:

Finally, PSO is utilized to find the best data augmentation parameters for enhancing image classification performance. The `objective_function', parameter bounds, a swarm size of 20, and a maximum of 20 iterations are used when using the PSO function. Through this process, the best combination of augmentation settings is iteratively searched for. The `objective_function` evaluates the model accuracy for each set of parameters, and PSO aims to maximize this accuracy by minimizing the negative accuracy returned by the function. After the optimization process, the best parameters and the corresponding highest accuracy achieved are obtained. The best parameters indicate the optimal augmentation settings, while the best accuracy reflects the improved model performance due to these settings.

4.6 PSO algorithm

Initialization

1. Initialize particles randomly within lower_bounds and upper bounds
2. Randomly initialize velocities
3. Initialize personal_best_positions to particles
4. Evaluate the objective_function for each particle and store in personal_best_scores
5. Determine global_best_position and global_best_score from personal_best_scores

PSO Main Loop

6. for i ←1 to maxiteration
7.   for j ← 1 to swarmsize
8.   Generate random vectors r1 and r2
9. Update velocities[j] using PSO velocity update formula
10. Update particles[j] using updated velocities[j]
11. Clip particles[j] to be within lower_bounds and upper bounds
12. Evaluate objective_function at particles[j] and store in score
13. if score < personal_best_scores[j]:
14.   personal_best_scores[j] = score
15.   personal_best_positions[j] = particles[j]
16. if score < global_best_score:
17.   global_best_score = score
18.   global_best_position = particles[j]
19. Record accuracy and loss for global_best_position
20. return the values of global_best_score and global_best_position

4.7 Components of a Particle in PSO

The bounds represent the range of values that the data augmentation and model hyperparameter can take during the optimization process and are given for the proposed system in the following way:
Data augmentation parameters

1. rotation_range: Specifies the range for random rotations applied to images during data augmentation. It is bounded between 0 and 40 degrees, allowing for a maximum rotation of 40 degrees.
2. width_shift_range: Controls the range of horizontal translation of images during augmentation. It is bounded between 0 and 0.2, indicating a maximum shift of 20% of the image width.
3. height_shift_range: Similar to width_shift_range, but controls vertical translation. It also has a range of 0 to 0.2, allowing for a maximum shift of 20% of the image height.
4. shear_range: Determines the shear intensity, which distorts the shape of the image along the horizontal or vertical axis. It ranges from 0 to 0.2, representing a maximum shear intensity of 20%.
5. zoom_range: Specifies the range for random zooming applied to images. It ranges from 0 to 0.2, allowing for a maximum zoom of 20%.
6. horizontal_flip: Controls whether random horizontal flips are applied to images. It is a binary parameter, with a range of 0 to 1, where 0 represents no flip and 1 represents flipping.

Model hyperparameters

1. epochs: Denotes the quantity of CNN model training epochs. It ranges from 1 to 10, indicating a minimum of 1 epoch and a maximum of 10 epochs.
2. learning_rate: Sets the learning rate for training the model. It ranges from 10e-5 to 10e-2, representing values between 0.00001 and 0.01.

3. dense_units: Indicates how many units there are in the CNN model's dense (completely linked) layer. It allows for a variable number of units in the dense layer and runs from 32 to 256.

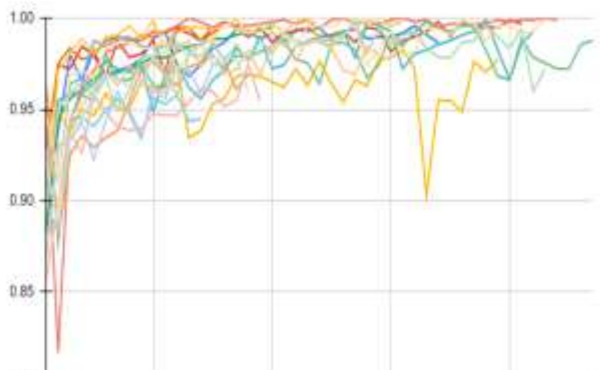
Initially, particles representing different combinations of these parameters were randomly initialized within their specified bounds. Velocities for these particles were also randomly assigned, enabling dynamic exploration of the parameter space. Once initialization was complete, the personal best positions and scores for each particle were recorded by evaluating the objective function. This included the CNN's performance on the training dataset following the use of the appropriate data augmentation techniques. The best-performing particle's position was set as the global best position. To track the progression of the optimization process, two lists—the accuracy list and the loss list—were initialized to record the model's performance metrics at each iteration. In each iteration of the PSO loop, particles were updated by adjusting their velocities and positions. Three factors affected the velocity updates: the particle's inertia, the cognitive factor (represents the particle's propensity to return to its optimal location), and the social component (represents the particle's propensity to advance toward the optimal position for everyone). Random factors were introduced in these updates to ensure diverse exploration of the parameter space. After updating velocities, particle positions were adjusted and clipped within the specified bounds to ensure they remained valid. The updated positions of the particles were then used to re-evaluate the objective function. This evaluation involved applying the updated data augmentation parameters to the training dataset, building and compiling a CNN model with the specified hyperparameters, and training the proposed model for specified number of epochs. After training, the model was assessed using a different test dataset to obtain the accuracy and loss, which were appended to the respective lists for tracking purposes. If a particle's new position yielded a better performance score (i.e., higher accuracy and lower loss), its personal best position and score were updated. Likewise, the score of the global best were adjusted in the event that any particle outperformed the global best. Finding the ideal combination of augmentation and model parameters was the final objective of this iterative process, which was done for a predetermined number of times. The best parameters found through this optimization process were then returned, representing the combination that maximized the CNN's performance on the image classification task.

4.8 Experimental setup

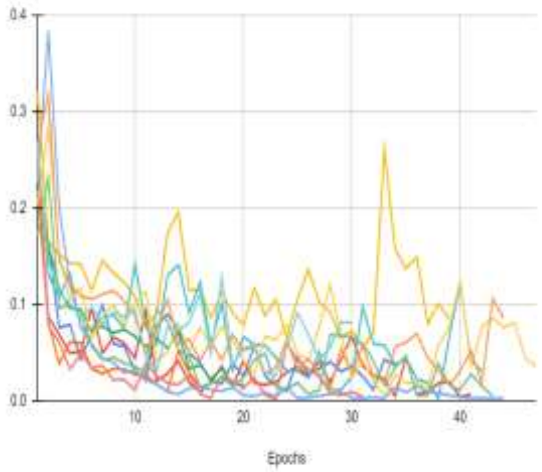
The system is powered by an NVIDIA GPU, specifically a Tesla T4, running driver version 525.85.12 and CUDA version 12.0. CUDA, a parallel computing platform and programming model, enables NVIDIA GPUs to be utilized for general-purpose computing. At the time of observation, the GPU temperature was 42°C, with a performance level of P0. The GPU's power consumption was 25W, with a power limit set at 70W, and its PCI bus ID was 00000000:00:04.0.

5. Results and Discussion

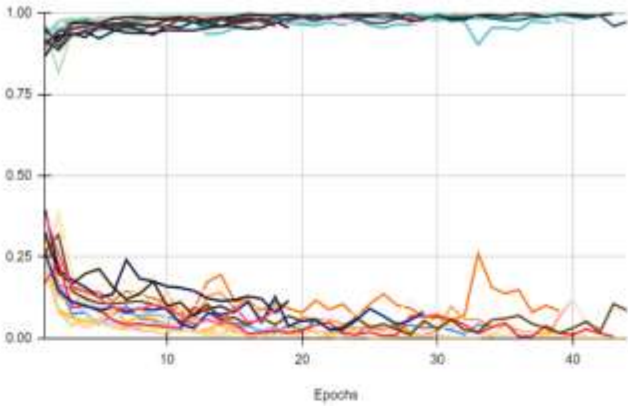
The NutriFoodNet CNN architecture, a model created for nutritional food categorization tasks, was optimized in this study using PSO. In order to maximize accuracy and minimize loss metrics, the CNN's hyperparameters were adjusted throughout the course of 20 iterations of optimization. Throughout the 20 iterations, both accuracy and loss were measured and recorded. The accuracy is an important metric for assessing the effectiveness of the model since it shows the percentage of correctly identified samples among all samples. Conversely, loss measures the discrepancy between the targets and the expected outputs, indicating how well the model is assimilating the data. Figure 5, which plots the results on a graph, shows the accuracy and loss values on the y-axis and the iteration number on the x-axis. The graph illustrates the progression of the optimization process, highlighting improvements in accuracy and reductions in loss as the PSO algorithm fine-tuned the CNN's parameters. This graphic aids in comprehending how well PSO works to improve NutriFoodNet's performance, demonstrating the convergence behaviour of the optimization process and the overall improvement achieved through the iterations.



(a) Epochs vs Accuracy



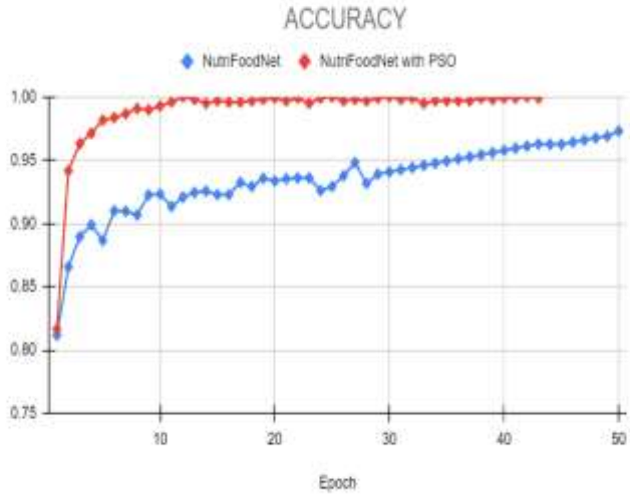
(b) Epochs vs Loss



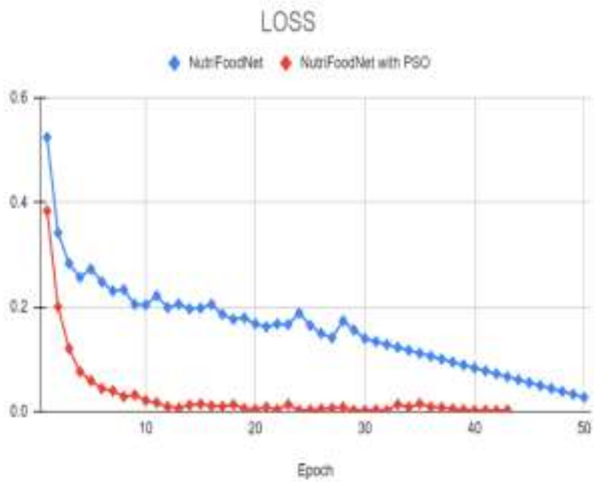
(c) Epochs vs Accuracy and Loss

Figure 5. *Accuracy and Loss over 20 iterations*

The results graph in figure 6 demonstrates a notable enhancement in the performance of the NutriFoodNet architecture when optimized with PSO. The graph shows that while the original NutriFoodNet achieved an accuracy of 97.3%, the PSO-optimized version achieved a higher accuracy of 98.5%. This improvement is also reflected in the loss metrics, where the PSO-optimized NutriFoodNet exhibits a lower loss, indicating better learning and generalization capabilities. The iterative optimization process facilitated by PSO effectively fine-tuned the hyperparameters, leading to enhanced model performance as evidenced by the upward trend in accuracy and the downward trend in loss over the 20 iterations. We evaluated the effectiveness of various cutting-edge CNN models for identifying foods, including ResNet-18, ResNet-50, and Inception v3. The following performance criteria were used to assess



(a) Epochs vs Accuracy



(b) Epochs vs Loss

Figure 6*. Analysis of performance using NutriFoodNet with and without PSO*

these models: F1-score, recall, accuracy, and precision. While precision indicates the percentage of accurate positive forecasts among all positive forecasts, recall displays the percentage of real positives among all actual positives. The model's overall soundness is reflected in accuracy, while the F1-score offers a trade-off between recall and precision.

$$Accuracy = \frac{(Tp+Tn)}{(Tp+Tn+Fp+Fn)} \tag{11}$$

$$Precision = \frac{Tp}{(Tp+Fp)} \tag{12}$$

$$Recall = \frac{Tp}{(Tp+Fn)} \tag{13}$$

$$F1\ Score = \frac{2*(Precision*Recall)}{(Precision+Recall)} \quad (14)$$

where, Fp denotes false positive, Fn stands for false negative, Tp represents true positive, and Tn for true negative. The results were plotted on a graph in figure 7 to visually compare the models' performance across these metrics.

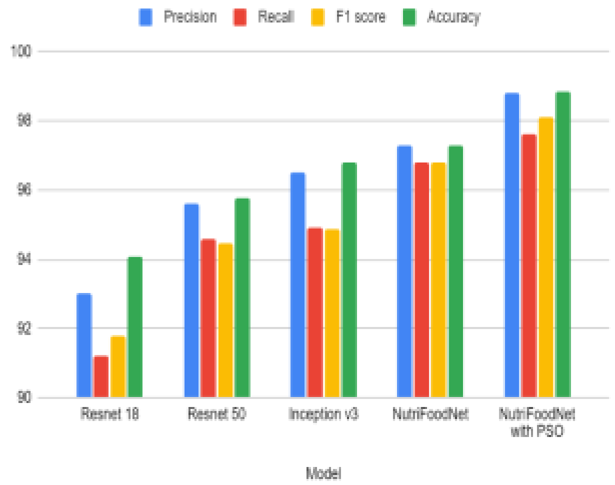


Figure 7. *Comparison of performance with different CNNs*

T-tests were conducted to compare the performance of different CNN models using accuracy as the evaluation metric. A p-value below 0.05 indicates a statistically significant difference in performance. As shown in Table 2, the p-value for the comparisons is less than 0.05, allowing the rejection of the null hypothesis

Table 2. *Comparison using T-Test*

Comparison	**NutriFoodNetPSO vs. ResNet18:**	**NutriFoodNetPSO vs. ResNet50**	**NutriFoodNetPSO vs. InceptV3**	**NutriFoodNetPSO vs. NutriF_Nt**
T-statistic	30.9466355	11.2934315	3.95889877	10.2111891
P-value:	3.43E-78	3.38E-23	1.05E-04	5.71E-20

6. Conclusion and Future Work

To sum up, our research presents a new approach to food image classification using the NutriFoodNet CNN architecture. Through the integration of PSO techniques, our model achieves significant advancements in accuracy and efficiency. By optimizing both data augmentation parameters and model hyperparameters, we ensure robust performance across various food categories. The NutriFoodNet CNN effectively learns and extracts features from food images, while PSO fine-tunes these features for enhanced classification accuracy. Our experiments demonstrate the superiority of the PSO-optimized NutriFoodNet model over traditional methods, positioning it as a promising solution for real-world food recognition applications. Future work will focus on several directions to further improve and expand the capabilities of our food image classification system. First, we plan to explore the integration of additional optimization algorithms, such as Genetic Algorithms and Bayesian Optimization, to potentially achieve even higher accuracy and efficiency. Second, adding real-world photos and broadening the dataset to encompass a wider variety of food items from various sources could improve the model's robustness and generalizability. Additionally, implementing transfer learning from larger pre-trained models and fine-tuning them on the Food101 dataset may further enhance performance. Finally, deploying the system in real-time applications, such as mobile apps for dietary tracking and integrating it with health monitoring systems, will be explored to assess its practical utility and impact on user health and nutrition management.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016, June). Deep residual learning for image recognition. *IEEE conference on computer vision and pattern recognition (CVPR)** (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

[2] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

(pp. 2261–2269). IEEE. https://doi.org/10.1109/CVPR.2017.243.

[3]Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., … Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data,* 8(1). doi:10.1186/s40537-021-00444-8

[4] Schoenauer, M., & Ronald, E. (1994). Neuro-genetic truck backer-upper controller. *In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence* (pp. 720–723). IEEE.https://doi.org/10.1109/ICEC.1994.349969

[5]Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1). doi:10.1186/s40537-019-0197-0

[6] Raiaan, M. A. K., Sakib, S., Fahad, N. M., Mamun, A. A., Rahman, M. A., Shatabda, S., & Mukta, M. S. H. (2024). A systematic review of hyperparameter optimization techniques in convolutional neural networks. *Decision Analytics Journal*, 11, 100470. https://doi.org/10.1016/j.dajour.2024.100470

[7]Nguyen, H.-P., Liu, J., & Zio, E. (2020). A long-term prediction approach based on long short-term memory neural networks with automatic parameter optimization by Tree-structured Parzen Estimator and applied to time-series data of NPP steam generators. *Applied Soft Computing*, 89, 106116. doi:10.1016/j.asoc.2020.106116

[8] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

[9] Yoon, J. H., & Geem, Z. W. (2021). Empirical convergence theory of harmony search algorithm for box-constrained discrete optimization of convex function. *Mathematics,* 9(545). https://doi.org/10.3390/math9050545

[10] Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11, 341–359.

[11] Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80, 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

[12] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine,* 1(4), 28–39. https://doi.org/10.1109/MCI.2006.190679

[13] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *In Proceedings of ICNN'95 - International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE. https://doi.org/10.1109/ICNN.1995.488968

[14] Yang, X.-S. (2010). Nature-inspired metaheuristic algorithms. Luniver Press.

[15] Ozaki, Y., Yano, M., & Onishi, M. (2017). Effective hyperparameter optimization using Nelder-Mead method in deep learning. *IPSJ Transactions on Computer Vision and Applications*, 9, 1–12. https://doi.org/10.118 6/s41045-017-0042-1

[16] Frazier, P. I. (2018). A tutorial on Bayesian optimization. arXiv preprint arXiv:1807.02811. Retrieved from https://arxiv.org/abs/1807.02811

[17] Lee, W.-Y., Park, S.-M., & Sim, K.-B. (2018). Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm. *Optik*, 172, 359–367. https://doi.org/10.1016/j.ijleo.2018.07.044

[18] Singh, P., Chaudhury, S., & Panigrahi, B. K. (2021). Hybrid MPSO-CNN: Multi-level particle swarm optimized hyperparameters of convolutional neural network. *Swarm and Evolutionary Computation*, 63, 100863. https://doi.org/10.1016/j.swevo.2021.100863

[19] Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I., & Tuba, M. (2020). Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics. *Algorithms*, 13(3), 67. https://doi.org/10.3390/a13030067

[20] Liu, D., Ouyang, H., Li, S., Zhang, C., & Zhan, Z.-H. (2023). Hyperparameters optimization of convolutional neural network based on local autonomous competition harmony search algorithm. *Journal of Computational Design and Engineering*. https://doi.org/10.1093/jcde/qwad050

[21] Lankford, S., & Grimes, D. (2020). Neural architecture search using particle swarm and ant colony optimization. *In Proceedings of the AICS 2020* (pp. 229–240).

[22] Yeh, W.-C., Lin, Y.-P., Liang, Y.-C., Lai, C.-M., & Huang, C.-L. (2023). Simplified swarm optimization for hyperparameters of convolutional neural networks. *Computers and Industrial Engineering*, 177, 109076. https://doi.org/10.1016/j.cie.2023.109076

[23] Serizawa, T., & Fujita, H. (2020). Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization. arXiv preprint arXiv:2001.05670. Retrieved from https://arxiv.org/abs/2001.05670

[24] Sharaf, A. I., & Radwan, E. F. (2019). An automated approach for developing a convolutional neural network using a modified firefly algorithm for image classification. In Applications of Firefly Algorithm and Its Variants: Case Studies and New Developments (pp. 99–118). Springer.

[25] Albelwi, S., & Mahmood, A. (2016). Automated optimal architecture of deep convolutional neural networks for image recognition. *In 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 53–60). IEEE.

[26] Rosa, G., Papa, J., Marana, A., Scheirer, W., & Cox, D. (2015). Finetuning convolutional neural networks using harmony search. *In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 20th Iberoamerican Congress, CIARP* 2015 (pp. 683–690). Springer.

[27] Huang, Y.-F., & Liu, J.-S. (2019). Optimizing convolutional neural network architecture using a self-adaptive harmony search algorithm. *In International*

Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (pp. 3–12). Springer.

[28] Sun, Y., Xue, B., Zhang, M., & Yen, G. G. (2018). A particle swarm optimization-based flexible convolutional autoencoder for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 30(8), 2295–2309. https://doi.org/10.1109/TNNLS.2018.2803384

[29] Albelwi, S., & Mahmood, A. (2016). Automated optimal architecture of deep convolutional neural networks for image recognition. *In 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 53–60). IEEE.

[30] Gudise, V., & Venayagamoorthy, G. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. *In Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS'03)* (Vol. 2, pp. 110–117). IEEE. https://doi.org/10.1109/SIS.2003.1202255

[31] Carvalho, M., & Ludermir, T. (2006). Particle swarm optimization of feed-forward neural networks with weight decay. *In 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06)* (pp. 5–5). IEEE. https://doi.org/10.1109/HIS.2006.264888

[32] Carvalho, M., & Ludermir, T. B. (2007). Particle swarm optimization of neural network architectures and weights. *In 7th International Conference on Hybrid Intelligent Systems (HIS 2007)* (pp. 336–339). IEEE. https://doi.org/10.1109/HIS.2007.45

[33] Kiranyaz, S., Ince, T., Yildirim, A., & Gabbouj, M. (2009). Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. *Neural Networks*, 22(10), 1448–1462. https://doi.org/10.1016/j.neunet.2009.05.013

[34]Zhang, J.-R., Zhang, J., Lok, T.-M., & Lyu, M. R. (2007). A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Applied Mathematics and Computation*, 185(2), 1026–1037. https://doi.org/10.1016/j.amc.2006.07.025

[35] Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A., & Benhaddou, D. (2017). Parameters optimization of deep learning models using particle swarm optimization. *In 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)* (pp. 1285–1290). IEEE. https://doi.org/10.1109/IWCMC.2017.7986470

[36] Kenny, A., & Li, X. (2017). A study on pre-training deep neural networks using particle swarm optimization. *In Simulated Evolution and Learning: 11th International Conference, SEAL 2017* (pp. 361–372). https://doi.org/10.1007/978-3-319-68759-9_30

[37] Fernandes Junior, F. E., & Yen, G. (2019). Particle swarm optimization of deep neural networks architectures for image classification. *Swarm and Evolutionary Computation*, 49, 62–74. https://doi.org/10.1016/j.swevo.2019.05.010

[38] Aguerchi, K., Jabrane, Y., Habba, M., & El Hassani, A. H. (2024). A CNN hyperparameters optimization based on particle swarm optimization for mammography breast cancer classification. *Journal of Imaging*, 10(2), 30. https://doi.org/10.3390/jimaging10020030

[39] Alhudhaif, A., Saeed, A., Imran, T., Kamran, M., & Alghamdi, A. S. (2022). A particle swarm optimization based deep learning model for vehicle classification. *Computer Systems Science and Engineering*, 40(1), 223–235.

[40]Saleem, M. A., Aamir, M., Ibrahim, R., Senan, N., & Alyas, T. (2022). An optimized convolutional neural network architecture for paddy disease classification. *Computers, Materials & Continua*, 71(3), 6053–6067. https://doi.org/10.32604/cmc.2022.022215

[41] Liu, X., Zhang, C., Cai, Z., Yang, J., Zhou, Z., & Gong, X. (2021). Continuous particle swarm optimization-based deep learning architecture search for hyperspectral image classification. *Remote Sensing*, 13(6), 1082. https://doi.org/10.3390/rs13061082

[42] Zhou, C., & Xiong, A. (2023). Fast image super-resolution using particle swarm optimization-based convolutional neural networks. *Sensors*, 23(4), 1923. https://doi.org/10.3390/s23041923

[43]Nistor, Sergiu & Czibula, Gabriela. (2021). IntelliSwAS: Optimizing deep neural network architectures using a particle swarm-based approach. *Expert Systems with Applications*. 187. 115945. 10.1016/j.eswa.2021.115945.

[44]Sreetha, E. S., Sundar, G. N., Narmadha, D., Sagayam, K. M., & Elngar, A. A. (2023). Technologies for Healthcare 4.0: From AI and IoT to blockchain.

[45] Bossard, L., Guillaumin, M., Van Gool, L. (2014). Food-101 – Mining Discriminative Components with Random Forests. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8694. Springer, Cham. https://doi.org/10.1007/978-3-319-10599-4_29

[46] Mumuni, A., & Mumuni, F. (2022). Data augmentation: A comprehensive survey of modern approaches. *Array*, 16, 100258. https://doi.org/10.1016/j.array.2022.100258

[47] BACAK, A., ŞENEL, M., & GÜNAY, O. (2023). Convolutional Neural Network (CNN) Prediction on Meningioma, Glioma with Tensorflow. *International Journal of Computational and Experimental Science and Engineering*, 9(2), 197–204. Retrieved from https://ijcesen.com/index.php/ijcesen/article/view/210

[48] Priti Parag Gaikwad, & Mithra Venkatesan. (2024). KWHO-CNN: A Hybrid Metaheuristic Algorithm Based Optimzed Attention-Driven CNN for Automatic Clinical Depression Recognition . *International Journal of Computational and Experimental Science and Engineering*, 10(3).491-506 https://doi.org/10.22399/ijcesen.359

[49] Agnihotri, A., & Kohli, N. (2024). A novel lightweight deep learning model based on SqueezeNet architecture for viral lung disease classification in X-ray and CT images. *International Journal of Computational and Experimental Science and*

Engineering, 10(4).592-613 https://doi.org/10.22399/ijcesen.425

[50] Jha, K., Sumit Srivastava, & Aruna Jain. (2024). A Novel Texture based Approach for Facial Liveness Detection and Authentication using Deep Learning Classifier. *International Journal of Computational and Experimental Science and Engineering*, 10(3). 323-331 https://doi.org/10.22399/ijcesen.369

[51] Radhi, M., & Tahseen, I. (2024). An Enhancement for Wireless Body Area Network Using Adaptive Algorithms. *International Journal of Computational and Experimental Science and Engineering*, 10(3).388-396 https://doi.org/10.22399/ijcesen.409

[52] Sreedharan, S.E., Sundar, G.N., Narmadha, D. (2024). NutriFoodNet: A high-accuracy convolutional neural network for automated food image recognition and nutrient estimation. *Traitement du Signal*, 41(4);1953-1965. https://doi.org/10.18280/ts.410425