International Journal of Basic and Applied Sciences, 14 (4) (2025) 763-772

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/zrxyex74
Research paper

Automating GAN Hyperparameter Selection: Insights from The PSO and ABC Techniques

Azlee Zabidi [1] *, Hasliza Abu Hassan [2], Mazlina Abdul Majid [1], Syed Umar Armaghan [1], Siti Salwani Yaacob [1], Zairi Ismael Rizman [3]

[1] *Faculty of Computing, Universiti Malaysia Pahang, Al-Sultan Abdullah, Pekan, 26600, Malaysia*
[2] *Faculty of Engineering and Life Sciences, Universiti Selangor, Bestari Jaya, 45500, Malaysia*
[3] *Faculty of Electrical Engineering, Universiti Teknologi MARA, 23000 Dungun, Terengganu, Malaysia*
**Corresponding author E-mail: azlee@umpsa.edu.my*

Received: July 3, 2025, Accepted: August 8, 2025, Published: August 27, 2025

Abstract

Generative Adversarial Networks (GANs) are powerful deep learning models that generate high-quality synthetic data for picture synthesis, data augmentation, and video production. However, training GANs is difficult and resource-intensive because of non-convergence, mode collapse, instability, and hyperparameter sensitivity. This research optimizes hyperparameters utilizing optimization methods to improve GAN, with an experiment applying to digit dataset generation. Hybrid Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) algorithms were presented to automate hyperparameter tweaking, focusing on Training Epochs, Batch Size, and Label Smoothing. This work introduced binarizing feature candidate selection instead of discrete values for optimization. This method represents parameters in binary form, where 1 signifies selection and 0 indicates non-selection, for more efficient parameter space search. Through experimental findings, the proposed technique significantly improves GAN training stability and output quality. Training for 25 epochs, with a batch size of 128 and a label smoothing of 0.8, yields the most consistent and high-quality outcomes, with ABC surpassing PSO. This study underscores the effect of hyperparameter optimization for enhancing GAN performance and introduces an innovative way for adjusting critical parameters. Optimization methods like PSO and ABC can improve GAN training and support their use in digit generation and other applications.

Keywords: *Artificial Bee Colony; Generative Adversarial Networks; Hyperparameter Optimization; Metaheuristic Algorithms; Particle Swarm Optimization.*

1. Introduction

Generative Adversarial Networks (GANs) represent a flexible deep learning structure that uses their adversarial training mechanism to generate high-quality synthetic information for several applications, which include image creation and data enhancement, as well as video generation [1], [2]. GAN training requires a significant amount of time as well as abundant system resources. Due to their adversarial nature, the generator and discriminator often come with issues such as non-convergence, mode collapse, and training instability [3]. The situation is aggravated by the fact that GANs are very sensitive to hyperparameter configurations as learning rates, batch size, latent space dimensions, and loss function parameters, and any small perturbations could destabilize training, leading to wasted time and computer resources for no increment to all the experiments and much repetition of tries for no fruitful yield [4], [5]. Setting up suitable initial parameters is crucial in ensuring stable GAN training and good-quality output generation.

The procedure is quite heuristic and requires considerable human tweaking and trial-and-error, thus making it impractical for large or high-resolution datasets [1]. The convergence rate depends heavily on the initialization phase, where proper value selection enables good results according to systematic and effective hyperparameter optimization methods. Several present answers to very hard optimization problems, such as Particle Swarm Optimization and Artificial Bee Colony, are used in a wide array of fields [6], [7]. These are nature-inspired algorithms capable of such a powerful search in extremely high-dimensional, non-convex problem spaces that they become suitable for optimizing GAN designs [8].

Particle Swarm Optimization (PSO) is an algorithm derived from the social behaviors of animals, such as bird flocks [9]. The actual working of this optimization algorithm is based on the image of a group of particles in search of a solution in the search area, thereby representing where each particle is, related to where the best position so far is, according to its personal knowledge. Therefore, particles swarm in space according to their individual best-known locations and those of their neighboring particles, thus pulling the swarm towards the optimal solution. PSO has proven useful in continuous optimization problems-especially in cases where the objective function is described with difficulty, such as when finding a good level of the hyperparameters of the GANs to assure stable training and good results [8], [10].

The Artificial Bee Colony (ABC) algorithm is based on the foraging behavior exhibited by honeybees. The population comprises employed bees, observation bees, and scout bees, each performing distinct roles in the exploration and utilization of the search region [11]. Some employed bees are responsible for carrying out the current solutions, observer bees are to probabilistically select the solutions based on their quality, and the scout bees explore the new routes within the search space, so that the algorithm does not prematurely converge. The ABC algorithm is probably one of the most well-balanced mechanisms between exploration and exploitation, which makes it practical and powerful for hyperparameter optimization in GANs, with huge search spaces and many local maxima.

This research analyses the application of PSO and ABC algorithms to systematically determine appropriate initial parameters for GAN training. The purpose of these algorithms is to make this hyperparameter optimization semiautomatic, making it less manual and onerous as well as increasing the GAN training stability and performance. In this work, we analyze the sensitivity of such key measures as training convergence, loss stabilization, data quality, and computational efficiency to the values of several enhanced parameters. This comparative analysis elaborates on the advantages and disadvantages of PSO and ABC in the function of GAN parameter optimization, which is very handy for the generative modelling expert [12].

This study improves the accessibility and use of GANs for practical applications by addressing the critical problem of hyperparameter selection with optimization techniques. The proposed technology reduces the training of GANs in time and resources while setting the ground for future advancements for the integration of optimization techniques within deep learning techniques.

2. Literature review

GANs are a very important new machine learning framework for image crafting, video creation, or text-to-image problems. In 2014, Goodfellow et al. introduced GANs which are based on the min-max optimization scheme using two neural networks: a generator that generates synthetic data and a discriminator that tests the validity of the given data. The adversarial counterpart provides motivation to the generator for producing data-like results. Despite their potential application, GANs face tons of obstacles that impede speed and cloud scale. Mode collapse is the predicament wherein the generator fails to encapsulate diversity in the underlying data distribution and generates repetitive or constrained outputs [3]. This may affect applications that are relying on variability among samples. GAN training is complicated by vanishing gradients, instability of convergence, and hyperparameter tuning [4], [5] dependencies. To ensure that GANs are adopted safely and effectively, these obstacles must be overcome.

Mode collapse [13] happens when a GAN learns to produce only one or a few different types of output and fails to capture the full range of variance in the data. Due to the discriminator's increased ability to spot bogus outputs, the generator may only produce a small variety of outputs. Another obstacle is training stability, which is the task of getting a GAN to converge. The GAN fails to converge when the discriminator and generator are engaged in a game of "cat and mouse," with one player constantly in the lead. Self-attention, spectral normalization [14], and Wasserstein GANs [15] There are a few methods that can assist in stabilizing the training process.

One essential problem in the field of Generative Adversarial Networks (GANs) has been mode collapse, which is a key problem, as its impact is substantial on samples that are generated [16]. While other GAN problems, such as training instability and vanishing gradients, can potentially impair the quality of the generated samples, they are not necessarily related to the generated samples' diversity. In contrast, mode collapse can result in repeated, low-quality, and non-diverse generated samples, which can be problematic for applications like image and video synthesis. The significance of addressing mode collapse in GANs has resulted in the development of numerous strategies to reduce this issue, such as adding regularization terms to the generator loss function or altering the architecture of the generator and discriminator networks. These strategies try to encourage the generator to capture a greater variety of modes in the target distribution, hence improving the diversity of samples generated.

In conclusion, mode collapse is an important issue in GANs that can drastically affect the quality and diversity of generated samples. While other difficulties in GANs can also impair the quality of the generated samples, they are not necessarily related to the diversity of the generated samples, making mode collapse a unique and essential challenge for GAN research to address. In [17], Researchers propose progressive growing during GAN training. The idea is to grow the generator and the discriminator at the same time: starting in low resolution, we progressively add layers that represent increasingly fine details with training progresses. This significantly accelerates and stabilises training, enabling the production of photos of exceptional quality. Researchers also propose a straightforward method for increasing the variety of generated images, achieving a record inception score of 8.80 in unsupervised CIFAR10 testing. Like [18]This study focuses on enhancing mode collapse in GAN. They present a unique training technique that generates additional discriminators in an adaptable manner to remember past modes of creation. Using many datasets, they demonstrate that a training strategy can be inserted into existing GAN frameworks to prevent mode collapse and improve common GAN assessment measures. Several researchers concentrate on fine-tuning the hyperparameters within GAN. There are at least 15 testable parameters in this study [19] that could have a major impact on how well GANs work. With many parameters, optimization tasks are crucial for determining the best solution for GAN performance.

The investigation of an adaptive hyperparameter in the GAN learning approach is presented in [20]. It is proposed to dynamically modify the number of training steps of the G and D for each set of training iterations based on the learning curves from relatively simple and limited datasets, such as MNIST, from which GANs have been demonstrated to deliver extremely significant results. Comprehensive experimental results demonstrate that the suggested method can greatly enhance the stability of GAN training and produce significantly more recognised objects in the Anime and CelebA datasets [21], [22]. However, one of the limitations of this study was that the proposed strategy was accomplished by controlling the training process using well-trained learning curves (prototypes) on relatively small datasets. There is a need for more effective training direction criteria that may encourage the convergence of GANs.

In other words [23], researcher combined the back propagation algorithm and the evolutionary algorithm to optimise deep generative models using E-GAN. During the evolutionary process, the parameters modified by various learning objectives are considered variation outcomes. So, the whole training process can be made more efficient and consistent if the proper mechanisms of evaluation and selection are introduced. E-GAN experiments showed that it helps stabilize the training of GAN models and performs very well in image generation tasks on various databases. During the GAN training process, the training stability is easily influenced by "bad" updating, which may cause the generated samples to be of poor quality or lack diversity. However, the proposed evolutionary mechanism largely avoids undesirable updating and guides the training in the desired direction. However, the training of the E-GAN model requires more time, despite the enhanced generative performance.

In conclusion, mode collapse is a major issue in GANs that needs to be addressed to ensure that generated samples are of high quality and diverse. To prevent mode collapse, various methods have been devised to modify the generator loss function with regularization terms, or to change the generator and discriminator networks' architecture. In addition, training stability and diversity of the generated samples are

improved using progressive growing and evolutionary algorithms. However, further research needs to be done to develop more effective training direction criteria that can encourage the convergence of GANs.

The optimization technique has been broadly applied to search for optimal system parameters. Stochastic optimization algorithms are generally divided into swarm-based and evolutionary-based algorithms. Examples of evolutionary algorithms are Genetic Algorithm (GA), Genetic Programming, Evolution Strategy (ES), and Evolutionary Programming (EP) [24]. Among the evolutionary algorithms, GA is the most used algorithm. Later, researchers found that GA has a disadvantage in terms of poor local search ability and premature convergence [25] leading to the development of other algorithms to resolve this problem.

In recent times, researchers have shown a preference for swarm-based optimization compared to evolutionary algorithms. A very popular swarm-based optimization, called PSO, is inspired by the social behaviour of bird flocks [26]. Additionally, there are other types of swarm-based optimization algorithms, such as ABC [27] and Ant Colony [28].

Comparing the two algorithms, GA and PSO have both been proven as effective techniques for optimization [29]. ABC was introduced in 2005 as a new optimization technique based on the swarm behaviour of bees [27]. ABC has played a major role in medical, bioinformatics, and many real-parameter optimization problems. The technique is widely accepted and applied to many applications, as it is well-suited for general assignment problems, cluster analysis, constrained problems optimization, structural optimization, and advisory systems [30]. Since then, much research has been focused on comparing several optimization techniques relative to ABC, such as PSO, Genetic Algorithm (GA), Differential Evolution (DE), Firefly Algorithm (FFA), and Ant Colony Optimization algorithm (ACO). The research suggests that ABC can escape a local minimum due to its 'food limit feature' [31]where, when there is any reliable solution or improvement, the searching position will be reset to a new one.

In conclusion, optimization strategies, for instance,e Evolutionary algorithms, such as Genetic Algorithms, and swarm-based methods, for instance, ABC and PSO, are suitable in tackling optimization challenges of different natures. These methods provide some interesting ways of investigation in GANs, particularly given that parameter optimization is instrumental in their performance and mode collapse-related issues. The adaptive nature of this kind of algorithm, particularly including the capacity of ABC to avoid getting trapped in a local minimum, aligns well with the challenges in GAN training, thereby making them a potent tool to enhance GAN optimization and ensure stable and diverse outputs.

3. Methodology

In this research, we are focusing on improving GAN performance by proposing optimization of GAN hyperparameters using an optimization algorithm. To achieve this, we are proposing to use the optimization algorithm, namely Particle Swarm Optimization and Artificial Bee Colony. Instead of using a discrete value for optimization, we propose a binarized selection of feature candidates by arranging a set of parameters into binary form, 1 for selected and 0 for not selected.

The selection of parameters to optimize in the context of optimizing GANs for enhanced performance is important for the efficiency and effectiveness of the model. For the present research, Training Epochs, Batch Size, and Label Smoothing were chosen for optimization, while other parameters such as activation functions, kernel initializers, and optimizers were excluded. The reasons behind the focused selection of these three parameters are grounded in both theoretical and empirical considerations that directly impact the convergence and stability of the GAN [32] training process.

Experiments were conducted using an Intel Core i7 processor with 8 physical cores and 16 threads. The system was equipped with 32 GB of DDR4 RAM, providing sufficient memory for data-intensive processing. For GPU acceleration, the system utilized an NVIDIA® GeForce RTX™ 3070 graphics card to enable modest execution of parallel computing and deep learning tasks.

3.1. GAN architecture

The generator and discriminator architectures are detailed in Tables 1 and 2, respectively. Both models follow the DCGAN design, using convolutional layers and activation functions suited for stable training and high-quality image generation.

Table 1: Generator Parameter

Layer Type	Output Shape	Kernel / Units	Stride	Activation / Notes
Input	(100,)	-	-	Latent noise vector
Dense	(10, 10, 256)	256 × 10 × 10	-	-
Reshape	(10, 10, 256)	-	-	Reshape dense output
BatchNormalization	(10, 10, 256)	-	-	-
LeakyReLU	(10, 10, 256)	-	-	$\alpha = 0.2$
Conv2DTranspose	(20, 20, 128)	4×4 filters	(2, 2)	BatchNorm + LeakyReLU
Conv2DTranspose	(40, 40, 64)	4×4 filters	(2, 2)	BatchNorm + LeakyReLU
Conv2DTranspose	(80, 80, 1)	4×4 filters	(2, 2)	Tanh

Table 2: Discriminator Parameter

Layer Type	Output Shape	Kernel / Units	Stride	Activation / Notes
Input	(80, 80, 1)	-	-	Grayscale image
Conv2D	(40, 40, 64)	4×4 filters	(2, 2)	LeakyReLU ($\alpha = 0.2$)
Conv2D	(20, 20, 128)	4×4 filters	(2, 2)	LeakyReLU ($\alpha = 0.2$)
Flatten	(51200,)	-	-	-
Dense	(1,)	1 unit	-	Sigmoid(real/fake probability)

3.2. Training epochs

Training Epochs require systematic, effective, and hyperparameter optimization methods, especially in the phase of initialization, when any good choices can have drastic positive effects regarding improving convergence and mitigating training instabilities. Particle Swarm Optimization, Artificial Bee Colony optimization, and so forth, have been seen to be techniques that have shown promise in solving such complex optimization problems in a variety of domains. Due to their robust searching capabilities in high-dimensional and nonconvex solution spaces, these bios-inspired algorithms can be used to tune the design of GANs. PSO is a nature-inspired algorithm inspired by the

social behavior of a flock of birds and schools of fish. It is a simulation of a swarm of particles where each particle represents a possible solution to the problem in the space of search.

3.3. Batch size

The batch size is the number of training instances that are trained on a single forward/backward pass during the training of a model. The choice of batch size can greatly influence the behavior of convergence in the GAN. Smaller batch sizes add high noise and can help the generator explore a more diverse space of possible outputs, which can, however, slow down the convergence rate. Steadier gradients are given by larger batches, but at the cost of promoting some premature convergence and thus having less diversity in the output. Optimizing the batch size gives a good balance in terms of training stability and explorative nature, and it contributes towards GAN's ability to learn good representations of the data while converging quickly.

3.4. Label smoothing

Label smoothing helps reduce the confidence of the discriminator in its predictions, making the training of the generator more stable. Label smoothing corresponds to marginally softening the target labels (e.g., replacing a definite label of 1 for real and 0 for fake with 0.9 and 0.1, respectively) so that the discriminator does not get too attached to drawing very strict decision boundaries. This method alleviates the possibility of mode collapse, wherein the generator yields a restricted array of outputs. Label smoothing enhances the stability of the GAN by reducing the likelihood of the discriminator dominating the generator, hence fostering more diverse and significant outputs. Enhancing this parameter can improve generator performance by preventing situations where the discriminator's supremacy obstructs the generator's learning efficacy. The candidate parameter selected for this task is shown in Table 3.

Table 3: Candidate Parameter

Hyperparameter	Tested Setting	Binary String
Training Epochs	5, 10, 15, 20, 25	[b1 b2 b3 b4 b5]
Batch Size	32, 64, 128, 256	[b1 b2 b3 b4]
Label Smoothing	0.8, 0.85, 0.9, 0.95	[b1 b2 b3 b4]

For selection purposes, bn is 1 when it is selected and vice versa. The number of candidates for each parameter is suggested so that it will at least cover at most typical parameter value. In GANs, the generator tries to minimize the following function (0) while the discriminator tries to maximize it (1):

$$E_x[\log (D(x) + E_z[\log \left(1 - D\big(G(z)\big)\right) \quad (1)$$

Where:
D(x): discriminator's estimate of the probability that real data instance x is real.
E_x: expected value over all real data instances.
G(z): generator's output when given noise z.
D(G(z)): discriminator's estimate of the probability that a fake instance is real.
E_z: expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).
1) Strategy for GANs: Balancing the generator and discriminator
In this research, we propose a novel optimization strategy of Generative Adversarial Networks (GAN), which makes the objective function:

$$\text{Fitness} = \min \left(D\big(G(z)\big)\right) + \max (1 - D(x)) \quad (2)$$

Here, $D\big(G(z)\big)$ represents the discriminator's estimate of the probability that the generated instance is real, and(x) represents the discriminator's estimate of the probability that a real instance (x) is real. The goal of this objective function is to achieve a balanced training dynamic between the Generator(G) and the Discriminator(D). The following sections will provide a detailed explanation of why this formula can be an effective approach in optimizing GAN performance.
2) Goal of Balancing Generator and Discriminator
In the conventional GAN framework, the generator tries to minimize the discrepancy between real and synthetic data, and the discriminator strives to make the best of distinguishing between the two. Nonetheless, this antagonistic training frequently results in disparities between the producer and the discriminator. If the discriminator is too powerful, it can overtake the generator and prevent the generator from improving its performance. If the generator gets excessively powerful, it may mislead the discriminator without truly enhancing the quality of the generated data.
To address this challenge, the proposed objective function directly influences both networks to encourage balanced learning. We keep the networks at an equilibrium where generators can deceive the discriminator, and the discriminator can distinguish between real and fake instances.
3) The first term: Minimizing $D\big(G(z)\big)$
The first part of the objective function, $\min \left(D\big(G(z)\big)\right)$, encourages the generator to produce outputs that are increasingly likely to be classified as real by the discriminator. As the generator improves, the discriminator's estimate of the probability that the generated instance is real should increase. Minimizing this term will push the generator to refine its outputs to be more indistinguishable from real data.
In standard GAN training, the generator typically aims to minimize the term$1 - D\big(G(z)\big)$, which is equivalent to trying to maximize $D\big(G(z)\big)$. However, this can easily cause gradient instability, especially when $D\big(G(z)\big)$ is close to 0 (meaning the discriminator is very confident in rejecting fake data). We can avoid the sharp gradients that arise in the conventional formulation and have smoother and more stable updates for the generator by directly minimizing $D\big(G(z)\big)$.
4) The second term: Maximizing $(1 - D(x))$
The second part of the objective function, $\max (1 - D(x))$, encourages the discriminator to be less confident about classifying real data as real. This is essentially regularization that prevents the discriminator from learning to become overconfident in its predictions. However,

mode collapse occurs when the generator becomes overconfident and learns to deceive the discriminator with a small number of data points.

This is because we want to maximize $(1 - \mathrm{D}(\mathrm{x}))$ to ensure that the discriminator is not too confident in distinguishing real data, thus reducing the chances of overfitting to the real data distribution. This term keeps the discriminator at a reasonable level of difficulty to avoid the discriminator achieving perfect accuracy too early. This allows the generator to learn to produce more diverse and realistic output over time and adapt.

5) Balanced training dynamic

The key advantage of combining these two terms is the balanced interaction between the generator and the discriminator. We optimize for both $\min\left(\mathrm{D}\left(\mathrm{G}(\mathrm{z})\right)\right)$ and $\max\,(1 - \mathrm{D}(\mathrm{x}))$ to force the generator to output data that confuses the discriminator, without letting the discriminator overconfidently guess on the data. By taking this balanced approach, a more stable and productive training process is promoted where both networks are required to adapt to the other's improvements continuously.

In traditional GAN training, if the discriminator becomes too strong, it can easily reject the generated data, preventing the generator from learning effectively. On the other hand, if the generator overpowers the discriminator, the discriminator will fail to provide meaningful feedback to the generator. This strategy incorporates both minimization and maximization components in the objective function such that the two networks evolve together with better performance in a cooperative manner.

6) Optimization strategy for GANs: Artificial Bee Colony and Particle Swarm Optimization Algorithm

Error! Reference source not found.4 shows the parameters used for both optimization algorithms during the optimization process. Swarm size is the number of agents searching the solution space simultaneously. Each agent in the swarm carries a role in the search process.

Table 4: Optimization Parameters Setting

Swarm Size (SS)	5
Max Iterations	5,10,15,20,25,30,35,40,45,50

Iterations represent the number of attempts allocated to the swarm to optimize the solution. Agents try to increase the fitness value of the solution in each iteration. Thus, it is important to set an appropriate maximum iteration count to allow the swarm enough time to converge effectively. In this study, the optimization process is focused on varying the iteration count while keeping the swarm size fixed. Multiple swarm sizes were not tested to avoid increasing computational time and load. It acts as a first test to set up the baseline for the GAN parameters optimization. For feature selection, suppose that the feature selection problem is defined as:

$$\mathrm{F}_{\mathrm{reduced}} \subseteq \mathrm{bn} \tag{3}$$

where Fc consists of n columns representing each candidate feature in the GAN parameter. To select a feature subset, $\mathrm{F}_{\mathrm{reduced}}$, a binary string of length $1 \times \mathrm{n}$ is defined, so that each column has a bit assigned to it. The initial value of the binary string can be randomly defined during initialization. A value of 1 indicates that the column will be considered in the construction of $\mathrm{F}_{\mathrm{reduced}}$, while the value of 0 indicates that the column is ignored.

In the swarm, each particle carries a $1 \times \mathrm{m}$ vector in solutions x_{id}. This vector contains the "probabilities of change". During optimization, the vector elements will be used as a reference to alter the binary string from its initial state (zero). If the particle vector element is more than 0.5, the binary bit will be changed to 1; otherwise, the bit value is maintained. Based on Table 4, the candidate potential feature size is 13, [b1 b2 b3 b4 b5] [b1 b2 b3 b4] [b1 b2 b3 b4].

4. Results and analysis

Three critical hyperparameters, epoch, batch size, and label smoothing were assessed in the experiments to improve GAN performance. Epoch was set to be [5, 10, 15, 20, 25], batch size [64, 128, 256, 512], and label smoothing [0.8, 0.85, 0.9, 0.95]. Two methods were used for optimization: Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). Iterations were varied from 10 to 50 in increments of 5, for each method with 5 particles. The goal was to minimize the fitness value, which is the sum of the discriminator error (abs (0.5 - d_loss[0])) and the generator error (abs (g_loss - 0.5)), to keep the balance of GAN's components.

As seen in Table 5, for all tested configurations, training for 25 epochs always results in the best fitness values, and both PSO and ABC achieve 0.2357 minimum fitness values under these parameters. The extended training periods are used to allow the GAN to converge well, so that the generator can produce more realistic outputs, and the discriminator can distinguish real and generated data better. The fitness values were higher when lower epoch values were used, i.e., 5, 10, or 15, since the training duration was insufficient for the GAN to reach equilibrium. An observation made across all the assessed batch sizes (64, 128, 256, and 512) was a constant observation that 128 was the smallest batch size that gave the lowest fitness value, which is the most optimal batch size. This finding shows that a moderate batch size optimally balances the trade-off between gradient accuracy and computation efficiency.

Table 5: Optimization Results

Cases	Iteration	Epoch	Batch	Label Smoothing	Result
PSO	30	25	128	0.8	0.235695
PSO	25	25	128	0.8	0.235695
ABC	25	25	128	0.8	0.235695
ABC	5	25	128	0.8	0.235695
ABC	45	25	128	0.8	0.235695
ABC	40	25	128	0.8	0.235695
PSO	45	25	64	0.95	0.318867
ABC	15	25	64	0.95	0.318867
ABC	50	25	64	0.95	0.318867
ABC	35	25	64	0.95	0.318867
PSO	20	25	64	0.95	0.318867
ABC	20	25	64	0.95	0.318867
PSO	5	25	256	0.9	0.407775
PSO	50	25	256	0.9	0.407775
PSO	15	25	256	0.9	0.407775

PSO	35	25	256	0.9	0.407775
ABC	30	25	256	0.9	0.407775
ABC	10	25	256	0.9	0.407775
PSO	10	20	128	0.95	0.505385
PSO	40	20	256	0.95	0.5906694

With smaller batch sizes, like 64, GAN training was found to be more sensitive to noise in GAN updates, leading to higher fitness values. For larger batch sizes (256, 512), suboptimal performance was observed, which may be attributed to reduced gradient variability, which could impede convergence and violate the equilibrium between the generator and discriminator. When label smoothing was set to 0.8, the lowest fitness values were found. The discriminator was able to provide effective feedback to the generator while maintaining a controlled level of confidence in its predictions, and this value served to obtain such results. Excessive smoothing also decreased the discriminator's ability to differentiate real and generated data, resulting in higher fitness values at higher label smoothing values, such as 0.9 or 0.95. On the contrary, small smoothing values in the analysed range improved the model's performance by keeping the level of uncertainty sufficient. In ideal configurations, 25 epochs, batch size of 128, and label smoothing of 0.8, both PSO and ABC were able to converge to an optimal fitness value of 0.2357. Results also show that ABC exhibited better stability with respect to the evaluated iterations, as it always returned identical fitness values between 25 and 50 iterations. However, PSO showed a higher variability of fitness outcome with the number of iterations, which is more sensitive to the hyperparameters. Optimization algorithms are influenced by the number of iterations. For both PSO and ABC, iteration values from 25 to 50 were found to be sufficient to reach optimal results.

Iteration counts of 10 or 15 limited the optimization process to being able to fully span the hyperparameter space and thus produce suboptimal fitness values. However, iterations above 50 were not analysed, but they could provide marginal gains in terms of computation time. The optimal fitness value of 0.2357 was attained using the subsequent hyperparameter configuration:

Epochs: 25; Batch Size: 128; Label Smoothing: 0.8; Optimization Algorithms: PSO and ABC; Iterations: 25 to 50

This finding implies that hyperparameter adjustment is a key factor for improving GAN performance. An adequate number of epochs that will allow learning enough time and a moderate batch size, and a label smoothing value that helps the GAN to reach a balanced state. This optimization challenge is addressed by both PSO and ABC, and ABC is shown to be more stable in the iterations. The results are important for future GAN use: hyperparameter choosing and optimization strategies require more care.

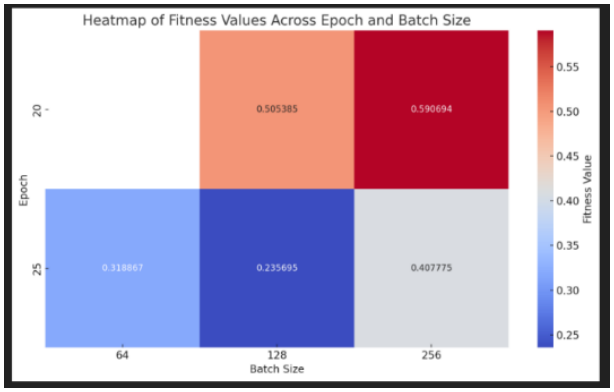

Fig. 1: Fitness Value Heatmap.

Figure 1 shows the heatmap of the fitness values obtained in the process of optimization, given different epochs and batch sizes. The research reveals important patterns of data that provide valuable information about how these hyperparameters affect the performance of GANs. Results show that the lowest fitness scores are obtained with 25 epochs, and especially with a batch size of 128. This combination yielded a minimal fitness value of 0.2357, meaning that it has an ideal trade-off between the generator and the discriminator. Since the GAN components were extended through 25 epochs of training, it may have been sufficient time for the GAN components to learn and become stable. On the other hand, with a reduced number of epochs (20), the fitness values were much higher, for example, 0.5054 with batch size 128 and 0.5907 with batch size 256. The results imply that the GAN is not able to converge well, and both the generator and discriminator do not receive enough training when the training duration is too short.

GANs are optimized by the batch size. Both techniques performed well, and a batch size of 128 demonstrated the highest efficacy in terms of producing the lowest fitness values for both techniques and iteration counts. The finding indicates the important trade-off between gradient variability and computing efficiency. Slightly better fitness values using reduced batch sizes, such as 64, may have been a result of increased noise in gradient computations, which can interfere with the training process. Specifically, batch sizes of 256 generated modest fitness values of 0.4078, which can be interpreted as decreasing gradient variability and therefore the learning process would suffer from inefficiency and yield suboptimal results. Both PSO and ABC achieved the same minimum fitness value with optimal parameters of 25 epochs, batch size of 128, and label smoothing factor of 0.8 in the assessment of optimization techniques. It turns out that ABC was more stable than other methods when different iteration counts are used, as it always reached the same fitness value for iterations of 25, 40, and 45. Increased variability in PSO indicated that it was more sensitive to the hyperparameters.

The number of iterations also influenced the process of optimization. Some configurations decreased the fitness value down to the 25 to 50 iterations, but lower iteration counts, like 10 or 15, typically result in a higher fitness value. This observation highlights the necessity for thorough investigation in the hyperparameter space to achieve the best performance of GANs. No count greater than 50 iterations was analysed, indicating opportunities for future research to find if additional improvements are possible with additional computational resources.

The results of the heatmap study emphasize the importance of hyperparameter tuning for GANs. With such a configuration, with 25 epochs, a batch size of 128, and a label smoothing parameter equal to 0.8, we obtained optimal results, indicating that it is crucial to find good values of the parameters used in GAN training, as we want to stick to the stable and balanced phase. In this environment, both PSO and ABC proved effective, and ABC was more consistent iteration by iteration. The results show that proper calibration of training configuration is required for good performance in future GAN applications.

Further analyses were performed using the optimal results derived from PSO and ABC. Both optimizers achieve the same minimum fitness of 0.235695, with PSO using 5 particles across 25 iterations, while ABC employs 5 particles over only 5 iterations. The number of

evaluations is different for both, as shown in Table 6 and **Error! Reference source not found.**7. The difference in evaluation counts between ABC (Table 6) and PSO (Table 7) is fundamentally associated with the inherent characteristics and operational mechanisms of these two optimization algorithms.

Table 6: PSO Evaluation

Epoch	Batch Size	Label Smoothing	Fitness
25	128	0.8	0.235695
25	256	0.9	0.407775
25	128	0.9	0.604992
25	128	0.95	0.759843
25	64	0.8	0.797482
15	128	0.95	0.833629
20	128	0.9	1.005759
25	32	0.9	1.080032
25	64	0.9	1.099532
25	256	0.8	1.530396
25	32	0.8	1.769262
25	0	0.9	10.00000
25	128	0	10.00000
25	32	0	10.00000
25	64	0	10.00000

PSO is an optimization algorithm based on population, making a compromise between exploration and exploitation. The position of each particle in the swarm is adjusted according to its individual best position and the best position found in the swarm. The quantity of particles and the quantity of iterations determine the number of evaluations in PSO. Here, PSO has 5 particles and 25 iterations, and therefore 125 assessments. PSO is rapid to converge because of its directed search mechanism that leads to refining positions toward the global optimal solution. In the search space used in this experiment, the way with fewer redundant evaluations will minimize explorations when the approach to convergence is achieved and select less the exploration that is unnecessary.

ABC is, however, affected by the foraging behaviour of honeybees, namely the employed bees, the spectator bees, and the scout bees. Foraging bees assess the area around the food sources to determine viability. Observer bees choose food sources based on quality and look at alternative options, scout bees seek new, random alternatives when a source is abandoned. The dynamic reassignment of bees increases the number of evaluations influenced by several aspects: the ongoing assessment for solutions near potential areas, the identification of new solutions by scout bees, and the distribution of responsibilities. The evaluation frequency of ABC is increased by repeated recruitment by observer bees and random searches of scout bees.

In the specified configuration, ABC utilizes five particles while evaluating a larger set of solutions (Figure 2) due to the iterative approach and the continuous integration of new evaluations by scout bees, even as optimal solutions are approached.

It has the effect of decreasing stagnation while increasing computational cost. The reason why ABC conducts more evaluations is that ABC has the iterative re-evaluation and exploration strategies, while PSO has the deterministic focused search strategy. In response to the ongoing reassessment by both employed and observer bees and their proactive inquiries from scout bees, ABC also carried out additional evaluations.

Table 7: ABC Evaluation

Epoch	Batch Size	Label Smoothing	Fitness
25	128	0.8	0.235695
25	64	0.95	0.318867
10	64	0.95	0.332928
5	256	0.9	0.338596
5	256	0.95	0.482846
20	256	0.95	0.590694
20	256	0.9	0.601941
25	128	0.9	0.604992
25	256	0.95	0.608086
15	256	0.95	0.818201
25	256	0.85	0.821025
5	256	0.85	0.840442
25	128	0.85	0.869569
20	64	0.9	1.211218
10	64	0.85	1.325192
20	256	0.85	1.350126
20	32	0.9	1.422504
25	64	0	10.0000
25	256	0	10.0000
0	256	0.9	10.0000
25	0	0.8	10.0000
25	0	0.95	10.0000
15	256	0	10.0000
15	128	0	10.00000
0	256	0	10.00000
0	256	0.95	10.00000
25	128	0	10.00000
5	256	0	10.00000

In contrast to PSO, ABC assigns dynamic roles and provides more evaluations as more bees evaluate solutions at the same time. Although both algorithms reached the same minimal fitness value of 0.235695, the ABC algorithm's iterative refinement and active exploration method of exploration was able to complete a larger number of evaluations. This implies that ABC is more appropriate for searching domains with much complexity and multiple modes, since ABC puts greater weight on resilience and low stagnation in comparison to

computational efficiency. On the other hand, the PSO method targeted at an optimization problem possesses computational efficiency and is more appropriate to be used in less complex optimization problems for which the required convergence is rapidly achieved.

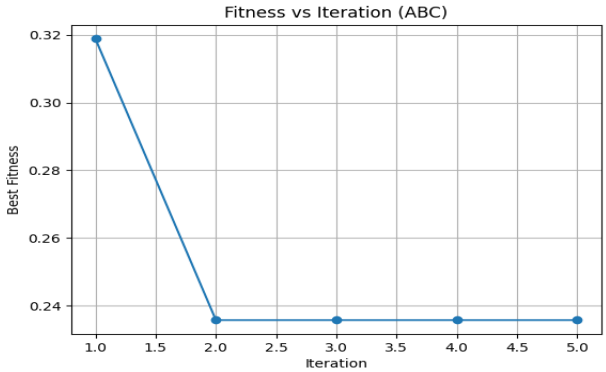

Fig. 2: ABC Optimization Process.

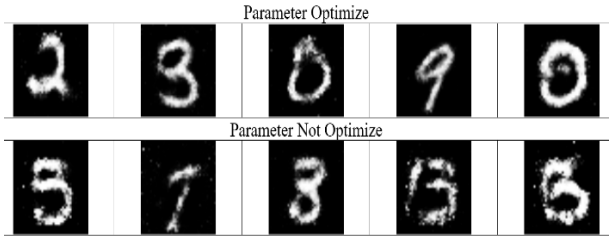

Fig. 3: Generated Image.

As shown in Figure 3, two groups of images are generated by a Generative Adversarial Network (GAN). Outputs from the upper row are obtained under optimized parameters (batch size, epoch, and label smoothing), and outputs from the lower row are generated without these optimizations. It is found that the generated images in the two cases are very different in quality. The optimized set consists of images that are characterized by clear and well-defined structures so that digits such as “2,” “3,” “0,” “9,” and “0” can be recognized. With batch size, the updates to the generator and discriminator become stabilized, and the train can oscillate and converge to the data distribution. Epoch adjustment helps balance the underfitting and overfitting and improves generalization. Regularization technique label smoothing helps to reduce overconfidence in the discriminator’s predictions, making GAN training more stable and producing better outputs.

Most digits in the non-optimized set of images are fragmented or unrecognizable due to the significant noise and structural distortions. The "5" and "3" are incoherent, and the "7" and "6" are notably distorted and contaminated with noise. Unstable training dynamics were likely caused by insufficient optimization, leaving the generator unable to represent the underlying data distribution adequately. Training instability may have been caused due to an unoptimized batch size, or the learning may not have been adequate because there were not enough epochs. It is possible that overfitting occurred in the discriminator due to the lack of label smoothing and thus destabilized the learning process of the generator.

The results highlight the importance of parameter optimization, particularly for batch size, epoch, and label smoothing, for achieving good performance in GANs. These parameters are properly tuned to stabilize the training process, learn the data distribution better, and produce high-quality, recognizable images. There are options when determining how to choose your keys and enhance security in GAN applications. This analysis demonstrates the need to carefully consider parameter optimization to ensure that any achieved results are reliable and meaningful.

5. Discussion

PSO and ABC algorithms were applied to the experiments to optimise the training parameters of GANs. It systematically tested epochs, batch sizes, and label smoothing values to uncover the impacts of those hyperparameters on the performance of GANs. This finding emphasizes the need for meticulous parameter selection to achieve stable training dynamics, the stability between the generator and discriminator, and high-quality images.

All evaluated configurations were trained for 25 epochs, with the highest fitness values obtained being from training for 25 epochs. The implication is that the GAN converges well when training for an extended duration, which facilitates the generator to produce more realistic outputs and the discriminator to tell the difference between real and generated data. Reduced epochs, such as 5 or 10, produced high fitness values, suggesting that reduced training epochs could result in under-training of the GAN. One should note that the chosen maximum epoch of 25 in this experiment may not be enough to adequately represent the data distribution, especially in complex GAN applications. This range of epochs is useful to recognize trends in parameter optimization.

A batch size was a critical factor, and 128 was found to be the optimal value. This computational efficiency and gradient variability, however, lead to the lowest fitness values among configurations, and this size optimised computational efficiency. Introducing noise into gradient calculations destabilised training even for smaller batch sizes such as 64. The gradient variability was decreased by increasing the batch size from 16 to 256 and 512; however, convergence was slowed, and the balance between generator and discriminator was disrupted. Optimal efficacy was observed at label smoothing set at 0.8, where the discriminator can provide good feedback to the generator without overconfidence. When the smoothing values are increased, like 0.9 or 0.95, the discriminator’s ability to differentiate is reduced, which is suboptimal.

Under optimal conditions of 25 epochs, batch size of 128, and label smoothing of 0.8, the PSO and ABC optimisation algorithms were able to reduce the fitness value to 0.2357. The stability of ABC was seen to be much better than PSO, since ABC resulted in consistent results over a larger dynamic range of iterations, compared to PSO, where it varied. It implies that ABC is more efficient in discovering and seeking to optimize a solution when computational resources and stability are critical.

A comparison between both ABC and PSO was made in terms of their computational efficiency and stability. ABC performed more evaluations compared to PSO, as it relies on the behavior of multiple bee roles—employed, onlooker, and scout bees—to continuously explore and re-evaluate potential solutions. The strategy increases the chance of escaping from local minima, thus giving a chance to find a more optimal solution ahead. Different from the PSO strategy, it updates particles using a simpler, more directed strategy based on both individual and global best positions. Thus, it converges quickly and with fewer iterations, making it more efficient in terms of computation. In summary, ABC offers a more thorough and resilient search at the cost of extra evaluations, while PSO favors speed and efficiency, particularly in less complex problem spaces. Selecting between these methods should depend on whether consistency or computational speed is the priority in each application.

Although PSO and ABC are nature-inspired, population-based optimization techniques, their performance is best understood when compared with conventional approaches such as Bayesian Optimization and Grid Search. Grid Search offers exhaustive exploration, but it can become computationally impractical in high-dimensional settings. Bayesian Optimization addresses this by leveraging probabilistic models to improve the search process, often achieving greater efficiency, especially when evaluation costs are high. Nevertheless, both methods can face limitations in complex, non-convex landscapes, where swarm-based approaches like PSO and ABC tend to perform better by effectively exploring a wider range of potential solutions.

6. Conclusion

This research shows that optimizing GAN performance depends on hyperparameter improvements. Using 25 epochs alongside 128 batch size, together with 0.8 label smoothing values, produced an optimal framework for controlling the GAN training process. Studies demonstrated that the application of PSO and ABC algorithms functioned well as optimization techniques, where ABC maintained better stability throughout each iteration.

This research uses a maximum epoch of 25 due to its unclear ability to fully unleash GAN capabilities for producing detailed and accurate images. GAN training requires extensive periods to achieve convergence targets because complicated data distributions need additional time. The experimental results can function as a base for guiding GAN training procedures, although the study's maximum training period is considered insufficient. The identified effective initial hyperparameter configurations provide researchers with foundations to increase GAN training efficiency and accuracy.

Training a GAN is time-consuming, meaning that a well-optimized starting point is crucial to greatly enhance training efficaciously, thus reducing the redesign operation for tuning through numerous extreme trial-and-error, which is costly in computation. This direction is a snapshot guide on how to improve the quality of outputs in later runs of the training with respect to convergence. Further work might take this study into the comparison of extending the number of training epochs beyond 25 while experimenting with more diverse hyperparameters, like different learning rates or momentum, to give a better explanation of GAN optimization. Additional research could verify the generalizability and robustness of these parameters when applied to other, more complex datasets. That would allow for more thorough application of GANs in other fields, such as the generation of images or data synthesis. Upcoming studies should investigate real-world uses, including the creation of synthetic medical images or data for diagnostic assistance, which would evaluate the model's effectiveness in critical areas. These developments could significantly enhance the wider acceptance of GANs in sectors where high-quality synthetic data is crucial. In addition, we also intend to explore the generation of traditional Songket patterns using GANs. This effort has the potential to support and revitalize Malaysia’s traditional textile industry by preserving cultural heritage through digital innovation

Acknowledgment

The authors would like to thank the Ministry of Higher Education for providing financial support under the Fundamental Research Grant Scheme (FRGS) No. FRGS/1/2024/ICT02/UMP/02/11 (University reference RDU240128)

References

[1] I. J. Goodfellow, J. P.-Abadie, M. Mirza, B. Xu, D. W.-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” Advances in Neural Information Processing Systems, vol. 27, pp. 1-9, 2014.

[2] T. Karras, S. Laine, and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 43, no. 12, pp. 4217–4228, Dec. 2018. https://doi.org/10.1109/TPAMI.2020.2970919.

[3] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved Techniques for Training GANs,” Advances in Neural Information Processing Systems, vol. 29, pp. 1-9, 2016.

[4] M. Lucic, K. Kurach, M. M. Google, B. O. Bousquet, and S. Gelly, “Are GANs Created Equal? A Large-Scale Study,” Advances in Neural Information Processing Systems, vol. 31, pp. 1-10, 2018.

[5] L. Mescheder, A. Geiger, and S. Nowozin, “Which Training Methods for GANs do actually Converge?,” International Conference on Machine Learning, pp. 3481-3490.

[6] J. Kennedy and R. Eberhart, “Particle swarm optimization,” International Conference on Neural Networks, vol. 4, pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.

[7] D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm,” Journal of Global Optimization, vol. 39, no. 3, pp. 459–471, Nov. 2007. https://doi.org/10.1007/s10898-007-9149-x.

[8] C. F. Chen, A. Mohd Zain, L. P. Mo, and K. Q. Zhou, “A New Hybrid Algorithm Based on ABC and PSO for Function Optimization,” IOP Conference Series: Materials Science and Engineering, vol. 864, no. 1, p. 012065. Jul. 2020. https://doi.org/10.1088/1757-899X/864/1/012065.

[9] D. Freitas, L. G. Lopes, and F. Morgado-Dias, “Particle Swarm Optimisation: A Historical Review Up to the Current Developments,” Entropy, vol. 22, no. 3, p. 362, Mar. 2020. https://doi.org/10.3390/e22030362.

[10] L. Tian, Z. Wang, W. Liu, Y. Cheng, F. E. Alsaadi, and X. Liu, “Empower parameterized generative adversarial networks using a novel particle swarm optimizer: algorithms and applications,” International Journal of Machine Learning and Cybernetics, vol. 13, no. 4, pp. 1145–1155, Apr. 2022. https://doi.org/10.1007/s13042-021-01440-3.

[11] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, “A comprehensive survey: Artificial bee colony (ABC) algorithm and applications,” Artificial Intelligence Review, vol. 42, no. 1, pp. 21–57, Mar. 2014. https://doi.org/10.1007/s10462-012-9328-0.

[12] K. G. Shreeharsha, C. G. Korde, M. H. Vasantha, and Y. B. Nithin Kumar, “Training of Generative Adversarial Networks using Particle Swarm Optimization Algorithm,” IEEE International Symposium on Smart Electronic Systems, pp. 127–130, 2021. https://doi.org/10.1109/iSES52644.2021.00038.

[13] Y. Kossale, M. Airaj, and A. Darouichi, "Mode Collapse in Generative Adversarial Networks: An Overview," IEEE 8th International Conference on Optimization and Applications, pp. 1-6 2022. https://doi.org/10.1109/ICOA55659.2022.9934291.
[14] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," 6th International Conference on Learning Representations, pp. 1-26, Feb. 2018.
[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," International Conference on Machine Learning, pp. 214-223, 2017.
[16] P. K. Saluja and D. Vathana, "Rectifying Mode Collapse in GANs," International Conference on Applied Artificial Intelligence and Computing, pp. 1718–1722, 2022. https://doi.org/10.1109/ICAAIC53929.2022.9792861.
[17] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," 6th International Conference on Learning Representations, pp. 1-26, Oct. 2017.
[18] K. Mangalam and R. Garg, "Overcoming Mode Collapse with Adaptive Multi Adversarial Training," 32nd British Machine Vision Conference, pp. 1-19, Apr. 2025.
[19] L. A. Courtenay and D. González-Aguilera, "Geometric Morphometric Data Augmentation Using Generative Computational Learning Algorithms," Applied Sciences, vol. 10, no. 24, p. 9133, Dec. 2020. https://doi.org/10.3390/app10249133.
[20] X. Ma, R. Jin, K. A. Sohn, J. Y. Paik, and T. S. Chung, "An Adaptive Control Algorithm for Stable Training of Generative Adversarial Networks," IEEE Access, vol. 7, pp. 184103–184114, 2019. https://doi.org/10.1109/ACCESS.2019.2960461.
[21] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," International Conference on Machine Learning, pp. 7354-7363. 2019.
[22] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep Learning Face Attributes in the Wild," IEEE International Conference on Computer Vision, pp. 3730-3738. 2015. https://doi.org/10.1109/ICCV.2015.425.
[23] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," IEEE Transactions on Evolutionary Computation, vol. 23, no. 6, pp. 921–934, Dec. 2019. https://doi.org/10.1109/TEVC.2019.2895748.
[24] X. S. Yang, Nature-inspired optimization algorithms. Academic Press, 2020. https://doi.org/10.1016/B978-0-12-821986-7.00013-5.
[25] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," Multimedia Tools and Applications, vol. 80, no. 5, pp. 8091–8126, Feb. 2021. https://doi.org/10.1007/s11042-020-10139-6.
[26] S. Mohammadi and S. R. Hejazi, "Using particle swarm optimization and genetic algorithms for optimal control of non-linear fractional-order chaotic system of cancer cells," Mathematics and Computers in Simulation, vol. 206, pp. 538–560, Apr. 2023. https://doi.org/10.1016/j.matcom.2022.11.023.
[27] H. Aoyang, Z. Shengqi, J. Xuehui, and Z. Zhisheng, "Short-term Load Forecasting Model Based on RBF Neural Network Optimized by Artificial Bee Colony Algorithm," IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering, pp. 486–489, Mar. 2021. https://doi.org/10.1109/ICBAIE52039.2021.9390043.
[28] Z. Li, K. Chen, and J. Wang, "An adaptive environmental sampling path planning technique for autonomous underwater vehicles based on ant colony optimization algorithm," IEEE 4th International Conference on Civil Aviation Safety and Information Technology, pp. 727–730, 2022. https://doi.org/10.1109/ICCASIT55263.2022.9987183.
[29] T. Kurban, P. Civicioglu, R. Kurban, and E. Besdok, "Comparison of evolutionary and swarm based computational techniques for multilevel color image thresholding," Applied Soft Computing, vol. 23, pp. 128–143, Oct. 2014. https://doi.org/10.1016/j.asoc.2014.05.037.
[30] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm," Applied Soft Computing, vol. 11, no. 1, pp. 652–657, Jan. 2011. https://doi.org/10.1016/j.asoc.2009.12.025.
[31] J. Tang, G. Liu, and Q. Pan, "A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends," IEEE/CAA Journal of Automatica Sinica, vol. 8, no. 10, pp. 1627–1643, Oct. 2021. https://doi.org/10.1109/JAS.2021.1004129.
[32] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," 4th International Conference on Learning Representations, pp. 1-16, Nov. 2015.