Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Bare-Bones Particle Swarm Optimization-based Quantization for Fast and Energy Efficient Convolutional neural networks

Jihene Tmamna[1] | Emna Ben Ayed[1] | Rahma Fourati*[1,2] | Amir Hussain[3] | Mounir Ben Ayed[1,4]

[1]Research Groups in Intelligent Machines, National Engineering School of Sfax (ENIS), University of Sfax, BP 1173, 3038, Sfax, Tunisia
[2]Faculty of Law, Economics and Management Sciences of Jendouba (FSJEGJ), University of Jendouba, Jendouba, Tunisia
[3]School of Computing, Edinburgh Napier University, Edinburgh, United Kingdom
[4]Computer Sciences and Communication Department, Faculty of Science of Sfax, University of Sfax, Sfax, Tunisia

Correspondence
*Rahma Fourati
Email: rahma.fourati@ieee.org

Abstract

Neural network quantization is a critical method for reducing memory usage and computational complexity in deep learning models, making them more suitable for deployment on resource-constrained devices. In this paper, we propose a method called BBPSO-Quantizer, which utilizes an enhanced Bare-Bones Particle Swarm Optimization algorithm, to address the challenging problem of mixed precision quantization of Convolutional Neural Networks (CNNs). Our proposed algorithm leverages a new population initialization, a robust checking process, and a local search strategy to improve the search performance and guide the population towards a feasible region. Additionally, Deb's constraint handling method is incorporated to ensure that the optimized solutions satisfy the functional constraints. The effectiveness of our BBPSO-Quantizer is evaluated on various state-of-the-art CNN architectures, including VGG, DenseNet, ResNet, and MobileNetV2, using CIFAR-10, CIFAR-100, and Tiny ImageNet datasets. Comparative results demonstrate that our method delivers an excellent tradeoff between accuracy and computational efficiency.

KEYWORDS:
Barebone PSO; Model compression; mixed precision quantization; Energy efficient model inference

1 | INTRODUCTION

The popularity of solving problems using deep neural networks (DNNs) has rapidly increased in recent years. DNNs have demonstrated remarkable success across various applications including [1,2,3,4,5,6,7,8,9,10,11,12,13,14]. However, it is important to note that this success often comes at the expense of increased complexity, and higher memory and computational requirements. These factors can make it challenging to deploy DNNs on devices with limited computational resources, such as IoT devices that often have constrained memory, processing power, and battery life [15,16].

To address these challenges, researchers and engineers have been exploring various methods for DNNs compression, including pruning [17,18], quantization [19,20], and knowledge distillation [21,22]. Pruning involves removing unnecessary connections or parameters in the network, resulting in a smaller model size. Quantization reduces the precision of the model's weights/activations, thereby decreasing memory requirements and computational complexity. On the other hand, knowledge distillation involves training a smaller model to mimic the behavior and performance of a larger model. These methods aim to reduce computational complexity while maintaining acceptable performance levels. Consequently, they enable the development of energy-efficient DNN models that can effectively operate in resource-constrained environments, thereby minimizing overall energy consumption. This not only enhances the performance of DNN applications but also contributes to a greener and more sustainable environment. In our work, we focus on Neural Network Quantization (NNQ) as a valuable compression method aimed at reducing the computational requirements and memory footprint of DNNs. NNQ achieves this by representing the model's weights and activations using reduced bit precision, such as 8-bit or even lower bit-widths, as opposed to the standard 32-bit floating-point numbers.

Binary precision quantization is one method employed in NNQ [23,24,25]. It involves representing weights and activations with only two values: -1 and 1. This method provides significant advantages in memory and computational efficiency, as the storage requirements are greatly reduced. **Ternary precision quantization** [26,27], on the other hand, aims to represent weights and activations using three possible values: -1, 0, and 1. This method strikes a balance between the information representation of full-precision floating-point numbers and the efficiency of binary quantization. **Low-bit precision quantization** [28,29] refers to the quantization of weights/activations using a small number of bits, typically less than 8 bits. This method reduces memory requirements and computational complexity, thereby improving the overall computing performance of DNNs.

While binary and ternary precision quantization methods enhance computing performance, they can present some issues, including increased sensitivity to quantization errors and significant accuracy degradation compared to higher precision representations [30,31]. To strike a balance between computing performance and accuracy, many studies have explored the adoption of low-bit precision quantization. This method effectively reduces memory requirements and computational complexity while still maintaining reasonable inference accuracy. Methods for low-bit precision quantization can be divided into two categories: fixed and mixed-precision quantization.

Fixed-precision or homogeneous quantization involves assigning the same bit-width to all or most layers in a neural network. Several works, such as [30,32,33,28,29], have focused on quantizing the weights and activations using the same bit-width. While fixed-precision quantization simplifies the implementation, it may not always yield optimal solutions and can result in significant accuracy degradation. This is because fixed precision quantization assigns a uniform bit-width to all layers, without considering the sensitivity of individual layers to quantization errors. As a result, the quantization levels may not be tailored to the specific needs of each layer, potentially leading to sub-optimal performance.

Mixed precision or layer-wise quantization methods have been developed to overcome the limitations of fixed precision quantization. These methods consider the varying sensitivities of layers to quantization errors and allow for different bit widths to be assigned to different layers. Critical layers, which are more sensitive to quantization errors, may require higher bit precision to maintain accuracy, while less critical layers could potentially benefit from lower bit precision [34,35,36]. By tailoring the quantization levels to the specific needs of each layer, it becomes possible to optimize the average bit-width and minimize accuracy degradation.

Indeed, the evolution-based search [37] and reinforcement learning [36,38] methods have been explored for mixed precision quantization. However, these methods can encounter certain complexities and limitations. One challenge is the exponential search space of different quantization bit widths. As the number of possible bit-width configurations grows exponentially with the number of layers in a neural network, exhaustive exploration becomes computationally expensive. Another limitation is the potential for these methods to get trapped in locally optimal solutions. Without sufficient theoretical guidance, there is a risk of sub-optimal solutions. The development of efficient mixed-precision search algorithms remains an important task in the field of neural network quantization. By addressing the challenges of exploration and computational cost, we can strive to find more effective and optimal solutions for mixed precision quantization, leading to improved efficiency and performance of the quantized model.

In this paper, we present a method for mixed precision quantization. The core idea of our method is to formulate mixed precision quantization as a constrained single-objective optimization problem. To solve this problem, an improved version of the Bare-Bones Particle Swarm Optimization (BBPSO-Quantizer) is introduced as the optimizer. This algorithm incorporates four mechanisms that aim to enhance the search performance.

- The first mechanism is the improved initialization of the particle positions. We carefully design the initialization process to ensure a good balance between exploitation and exploration, enabling the particles to effectively explore the solution space from the beginning.

- The second mechanism is the use of Deb's constraint handling method, which plays a crucial role in directing the population towards the feasible region. This method helps maintain the feasibility of the solutions and ensures that the optimization process adheres to the given constraints.

- The third mechanism is the inclusion of a similarity-checking process to avoid redundant evaluations of the same position vectors. By comparing the similarity of newly generated positions with those already evaluated, we can avoid redundant evaluations, saving computational resources and improving efficiency.

- The fourth mechanism is the integration of a local search strategy to enhance the exploitation ability of the optimizer. This strategy is designed to prevent the algorithm from getting stuck in local optima and promote the exploration of the search space. The local search strategy is employed to enhance the personal best position (Pb) of each particle in the BBPSO-Quantizer algorithm. By updating the Pb using a local search process, we aim to escape from potential local solutions and explore new regions that could lead to better solutions.

The paper's subsequent sections are organized as follows: Section 2 overviews the related work on quantization. Section 3 details the proposed method. Section 4 outlines the experimental setup and discusses the results. Section 5 presents the conclusions.

2 | RELATED WORK ON MIXED PRECISION QUANTIZATION

In general, the weights/activations of layers are commonly represented as 32-bit float values. However, one method to reduce the model's computational complexity is to decrease the bit-width representation. Mixed precision quantization involves utilizing different bit widths for different layers. This method takes the varying sensitivities of layers to quantization errors and aims to determine the optimal layer's bit-width. This process is depicted in Figure 1.

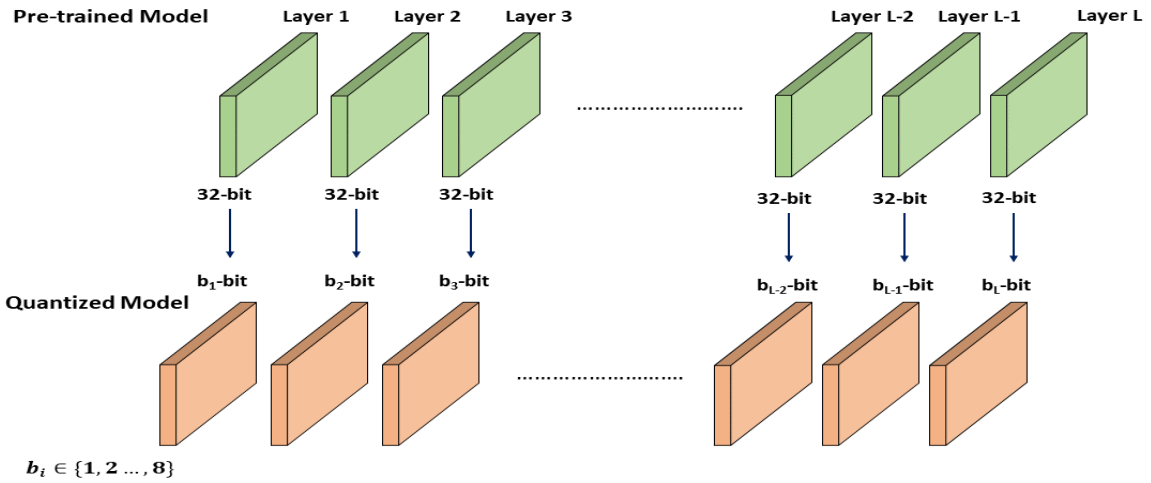


Figure 1 Illustration of mixed-precision quantization process

In the field of mixed-precision quantization, there have been several methods developed to determine the optimal layer's bit-width. These methods can be broadly classified into two categories: non-search and search-based mixed precision quantization methods.

2.1 | Non-search-based mixed-precision quantization

Non-search-based mixed-precision quantization refers to methods used to determine the optimal quantization configuration without conducting an explicit search process. These methods [35,34,39,40,19,41,20] rely on heuristics, rules, or predefined guidelines to assign different precision to various layers of the network, taking into consideration the quantization sensitivity of each layer. The non-search-based mixed precision quantization methods can be classified into three main types: Hessian-aware, accuracy-aware, and indicator-based.

Hessian-aware methods, such as HAWQ [35], HAWQ-v2 [34], and HAWQ-v3 [39], utilize the Hessian matrix to guide the quantization process and mitigate potential accuracy degradation that can occur with quantization. By considering the Hessian matrix, these methods can make informed decisions regarding the quantization levels for different network parameters, thus minimizing the negative impact on accuracy.

Accuracy-aware methods typically focus on estimating accuracy gains for individual layers or parts of the network. Liu et al. [40] and Bablani et al. [19] attempt to estimate the accuracy gain for each layer through a technique known as one-shot imprinting, which involves one epoch of fine-tuning per layer. Once the accuracy gain is estimated for each layer, the method leverages this information to guide the quantization process. Layers with a higher estimated accuracy gain are prioritized for higher precision quantization. Layers with lower estimated accuracy gain may be quantized with lower precision to achieve resource efficiency.

The **indicator-based method**, as presented in [41], utilizes specific parameters within quantization, specifically the scale factors in the quantizer, as indicators of importance. These indicators help differentiate whether one layer is more sensitive to quantization than others. However, the indicator-based method heavily relies on scale factors as indicators of importance, and these scale factors may not always accurately reflect the importance or significance of a layer. In some cases, certain layers with small-scale factors may still contain crucial information for the network's overall performance. Thus, relying solely on scale factors may lead to sub-optimal bit-width allocations.

While non-search-based mixed-precision quantization is less computationally expensive compared to search-based methods, it may not guarantee optimal quantization for every network. It relies on heuristics and assumptions, which may not always capture the intricacies of a specific model. Consequently, there is a possibility of sub-optimal quantization levels being assigned, leading to a potential loss of accuracy or performance.

It's worth noting that recent advancements in quantization research have focused on search-based methods, such as reinforcement learning or evolutionary algorithms, which can automatically explore the quantization space and find more optimal configurations.

2.2 | Search-based mixed-precision quantization

Search-based quantization is a methodology focused on automating the process of finding the optimal bit-width of the network's layers. Some search-based quantization methods, such as HAQ (Hardware-Aware Automated Quantization)[36] and AutoQ[38], incorporate reinforcement learning (RL) to find the optimal precision configuration. These methods use RL algorithms to explore and learn the best bit-width settings for different layers or operations in the network. However, these RL-based methods can incur significant computational costs due to the large search space and the need for extensive training and evaluation. Moreover, some methods search the bit-width precision of individual layers through stochastic gradient descent (SGD) while employing a modified loss function. For example, Nikolic et al.[42] proposed introducing a regularization term to the loss function, encouraging the network to select lower bit widths during training. Yang et al.[43] also proposed a method that optimizes the bit-widths of each layer using SGD and a carefully designed quantization-aware loss function.

Evolutionary algorithms[44] have also been applied to the quantization with mixed-precision settings. With the growth of the search space, searching for all possible quantization configurations incurs a significant computational cost. This limitation hinders their application in finding the optimal bit widths within acceptable time and energy constraints. As a result, these methods are not widely adopted in deep CNN quantization.

In line with these considerations, we propose the BBPSO-Quantizer method for mixed precision quantization. The novelty of our BBPSO-Quantizer lies in incorporating population initialization, screening, and local search strategies, which simplifies the search for optimal bit widths with reduced complexity and rapid convergence.

3 | PRELIMINARIES

This section provides an introduction to the fundamental concepts of neural network quantization, particle swarm optimization (PSO), and the constrained handling method.

3.1 | Neural network quantization

In quantization, two commonly used operations are clipping and projection. These operations are typically applied to the weights or activation of a convolutional layer to obtain the quantized values as illustrated in Figure 2. The quantization function is usually defined as[45]:

$$\overline{Z} = \prod\nolimits_{Q(a,b)} clip(Z, a) \tag{1}$$

where $\overline{Z}$ is the quantized value, and Z stands for the input value. $Q(a, b)$ denotes the quantization levels. $clip(Z, a)$ is the clipping function that truncates whether it's 32-bit weight or activation, to the range $[-a, a]$ and $[0, a]$, respectively. $\prod(.)$ plays the role of a projection function, facilitating the mapping of each clipped element X into the space defined by $Q(a, b)$.

3.2 | Particle Swarm Optimization

Particle Swarm Optimization (PSO)[46] is an optimization algorithm inspired by the social behavior of bird flocking or fish schooling. In PSO, a collection of particles moves through a search space to find the best solution to a given problem. Each particle represents a potential solution. The algorithm begins by randomly initializing a population of particles. Each particle's position and velocity are then iteratively updated based on its personal best position (Pb) and the global best position (Gb).

In 2005, Kennedy[47] proposed bare-bones PSO (BBPSO). While retaining the fundamental optimization idea of PSO, BBPSO differs from traditional PSO. On the one hand, it discards the update of particle velocity compared to PSO to overcome the disadvantage that conventional PSO relies too much on parameters. It replaces it with Gaussian distribution based on Pb and Gb.

For a swarm of size $\mathcal{N}$, each particle's position X, Pb, and Gb vectors at the t^th generation, are denoted as $X_i^t = \{x_{i1}, x_{i2}, ...x_{iD}\}$, $Pb_i^t = \{Pb_{i1}^t, Pb_{i2}^t, ..., Pb_{iD}^t\}$, $Gb^t = (Gb_1^t, Gb_2^t, ..., Gb_D^t)$, where i=1,2.. $\mathcal{N}$, j=1,2... $\mathcal{D}$, with $\mathcal{D}$ representing the problem's dimension. Each bit x_{ij} in the position vector X_i is updated as follows:

$$x_{ij}^{t+1} = \begin{cases} G(0.5(Pb_{ij}^t + Gb^t), |Pb_{ij}^t - Gb_j^t|), & \text{if } \mathrm{R} < 0.5 \\ Pb_{ij}^t, & \text{otherwise} \end{cases} \tag{2}$$

where R is a random value, $G(\beta, \gamma)$ represents a Gaussian distribution with mean β and variance γ.

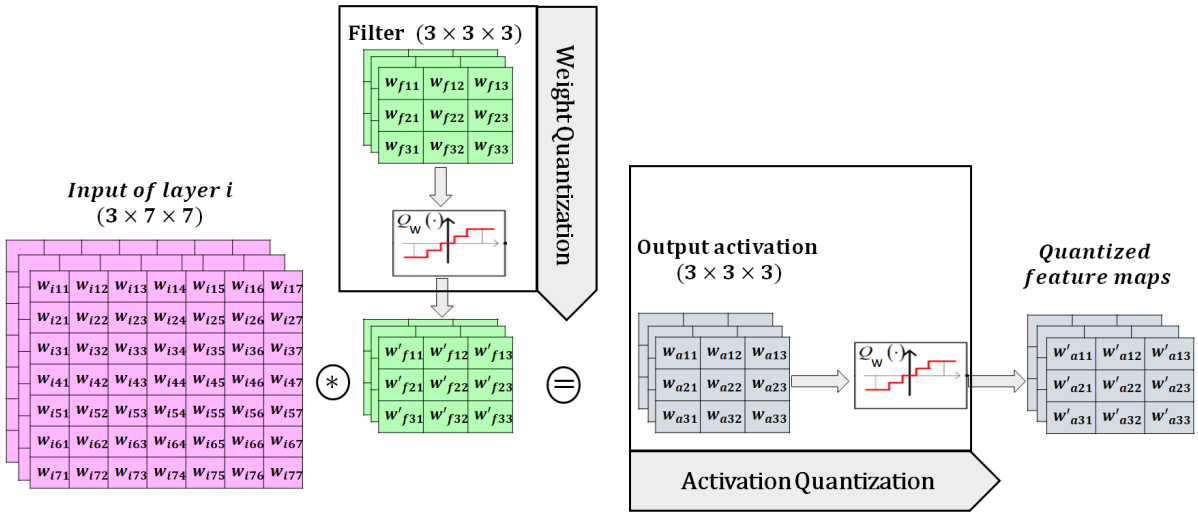


Figure 2 Illustration of convolutional layer quantization

3.3 | Constrained handling method

In our proposed method, ensuring that the optimal solution not only optimizes the objective function but also meets the functional constraint is of utmost importance. To achieve this, we incorporate Deb's constraint handling method (also known as Deb's rule) into our optimization algorithm [48]. Deb's rule is a widely used method in constrained optimization to guide the search toward feasible regions. Its main principles are:

- When comparing two feasible solutions, the algorithm selects the one with the better objective function value, which means selecting the solution that achieves the highest objective function value.
- When comparing one feasible solution with one infeasible solution, the feasible solution is selected. This principle ensures that if a feasible solution exists, it is always preferred over an infeasible one, even if the infeasible solution has a better objective function value.
- When comparing two infeasible solutions, the algorithm selects the solution with a smaller constraint violation. This means that if no feasible solutions are available, the algorithm selects the infeasible solution that violates the constraints to a lesser extent.

Algorithm 1 gives the algorithm of Deb's rule. By incorporating Deb's rule into our optimization algorithm, we ensure that the search process is guided from unfeasible regions toward feasible ones. This helps the algorithm explore and focus on regions where both the objective function and the functional constraint can be simultaneously optimized.

4 | CONSTRAINED BBPSO-BASED QUANTIZATION METHOD (BBPSO-QUANTIZER)

In this section, we introduce our BBPSO-Quantizer for the mixed precision quantization problem. Our method addresses significant challenges when developing a BBPSO-based quantization method, aiming to provide a more robust and effective method for generating high-quality quantized networks. It involves reducing complexity, maintaining population diversity, and achieving effective local searchability to enhance convergence. In this section, various mathematical symbols are used to represent different mathematical concepts. Table 1 summarizes these symbols, providing brief descriptions of their meanings.

4.1 | Problem formulation

The optimization of the mixed precision quantization problem revolves around two key factors: minimizing the computational cost of the model while maximizing accuracy. In our work, the quantization process aims to maximize accuracy under the constraint of the average operation bit, resulting in a constrained optimization problem. The fitness function is expressed as follows:

$$\begin{aligned} \max \quad & accuracy(\mathcal{M}_{quantized}) \\ \text{s.t.} \quad & C_{quantized}(\mathcal{M}_{quantized}) \leq C_{target} \end{aligned} \tag{3}$$

Algorithm 1 Deb's rule

Input: X_1, obj(X_1), constraint(X_1), X_2, obj(X_2), constraint(X_2), λ: predefined constraint

if $constraint(X_1) \geq \lambda$ and $constraint(X_2) \geq \lambda$ **then** ▷ check if the both solutions are feasible
  if obj(X_2) $\geq$ obj(X_1) **then** ▷ compare the objective function values
    $X_1 \leftarrow X_2$
    obj(X_1)$\leftarrow$ obj(X_2)
    constraint(X_1)$\leftarrow$ constraint(X_2)
  end if
else ▷ compare the constraint if any solution is feasible
  if $constraint(X_2) \geq constraint(X_1)$ **then**
    $X_1 \leftarrow X_2$
    obj(X_1)$\leftarrow$ obj(X_2)
    constraint(X_1)$\leftarrow$ constraint(X_2)
  end if
end if

Output: X_1, obj(X_1), constraint(X_1) ▷ Best solution

Table 1 Mathematical symbols Description

Symbol	Description
$\mathcal{M}$	Pre-trained model
bit_i	the i^{th} layer's bit-width
$\mathcal{L}$	Number of layer in the original model
s	bit-width candidate vector
$\mathcal{L}_{importance}$	Layer importance
X	particle
Gb	Global best solution
Pb	Personal best solution
MeM	Memory matrix
N	Swarm size
γ	LSP condition
$\mathcal{M}'$	Optimal quantized model

where $\mathcal{M}_{quantized}$ is the quantized model. The function $C_{quantized}()$ is responsible for evaluating the computational resources required and C_{target} is the target constraint adopted as average operation-bit-width used to assess the computational complexity in terms of bit-wise operations.

4.2 | Solution representation

One of the most important components of any population-based method is the representation of solutions. In the context of the quantization problem, the solution is encoded in integer form, with 'bit_i' representing the i^{th} layer's bit-width. By applying this encoding, the algorithm can identify the optimal quantization policy.

Given a CNN model $\mathcal{M}$ with $\mathcal{L}$ layers, we present the structure of the quantized model as $\mathcal{M}_{quantized} = (bit_1, bit_2, ..., bit_{\mathcal{L}})$, where bit_j is the j^{th} layer's bit-width. Therefore, the position of each particle P_i is defined by a vector X_i with dimension $\mathcal{L}$, encoded as:

$$X_i = \{x_{i1}, x_{i2}, ..., x_{i\mathcal{L}}\}, x_{ij} \in s \tag{4}$$

where i=1,2.. $\mathcal{N}$ (with $\mathcal{N}$ the swarm size), j=1,2...$\mathcal{L}$. The bit-width candidate vector, denoted as $s = 1, 2, 3, 4, 5, 6, 7, 8, 16, 32$, corresponds to the possible values of x_{ij}, where $x_{ij} = bit_j$ signifies the reduction of the j^{th} layer's bit-width to bit_j."

4.3 | BBPSO-Quantizer algorithm

In this section, we introduce BBPSO-Quantizer, an enhanced variant of BBPSO tailored for solving the integer optimization challenges in CNN quantization. It builds upon the standard BBPSO algorithm and incorporates several key improvements tailored for quantization tasks. The main proposed improvements in BBPSO-Quantizer will be detailed in the next section.

4.3.1 | Layer importance-guided population initialization (LIGPI)

Population initialization plays an important role in an optimization algorithm's convergence speed and effectiveness. Random initialization (RI), a widely employed method, involves creating the initial population by randomly assigning values within a certain range. However, RI alone may not be ideal as it lacks consideration of the intrinsic characteristics of the problem and can lead to sub-optimal solutions.

To overcome the limitations of random initialization, a novel population initialization strategy is proposed to generate a higher-quality population. This strategy is the use of layer weighting, where the importance of each layer is taken into account during population initialization. This initialization strategy helps the optimization algorithm to start with a population that is more likely to converge towards better solutions, reducing the risk of getting trapped in local optima. To estimate the layer importance in a pre-trained model $\mathcal{M}$ based on its pre-trained weights, we adopt the following equation:

$$\mathcal{L}_{importance} = \frac{\beta}{(1 + (exp(-\frac{(2\alpha(W - W_{min})}{(W_{max} - W_{min})} - \alpha)))} \tag{5}$$

where α is normalization range, $\beta \in [0.5, 1]$ and W is the L_1 norm value of layer weight matrix measured as follows:

$$W = \frac{1}{n_l} \sum_{i=1}^{n_l} ||w_i^l||_2 \tag{6}$$

where n_l denotes the number of filters, w_i^l denotes the filter's weight matrix.

Based on the importance measure, layers with higher importance values are considered more critical for preserving accuracy during the quantization process. As a result, these important layers are more likely to be quantized with higher bit precision.

4.3.2 | Pb and Gb updating strategy

To enhance the handling of constraints and optimize the search process, we propose integrating Deb's constraint-handling method into the BBPSO algorithm.

In our method, we adopt Deb's rule for updating Pb and Gb. Based on the dominance relation defined by Deb's rule, the algorithm determines whether the current position X_i can replace its corresponding Pb (Pb_i) based on the fitness value and resource constraint. If X_i dominates Pb_i, it becomes the new Pb for that particle.

The algorithm employs the same procedure to update the Gb position. It compares the feasibility of Pb_i and Gb and decides whether Gb should be updated or not. This strategy helps guide the population towards feasible regions, ensuring that the best solutions found are not only of high fitness but also satisfy the specified constraints. This allows for a more effective exploration of the solution space, leading to improved solutions for the mixed precision quantization problem.

4.3.3 | Updating each particle's position

In our work, the conventional BBPSO's continuous updating rules are not directly applicable due to the particle's position being an integer vector. To address this requirement, we adopt the updating rule introduced in a reference work[49]. Each bit x_{ij} in the position vector X_i at t^th iteration is updated as follows:

$$x_{ij}^t = \begin{cases} \lceil \frac{Pb_{ij}^t + Gb_j^t}{2} \rceil + \lceil G(0,1) | Pb_{ij}^t + Gb_j^t | \rceil, & \text{if rand} > 0.5 \\ Pb_{ij}, & \text{otherwise} \end{cases} \tag{7}$$

where i =1,2...N with N is the swarm size, j=1,2...$\mathcal{L}$, with $\mathcal{L}$ is the problem's dimension.

4.3.4 | Similarity screening process

During the search process, reevaluating the same position vector is possible since it's challenging to guarantee that new positions will always differ from those previously evaluated. To prevent redundant evaluations, we introduce a similarity screening process to assess the similarity between new positions and those already evaluated.

The main concept of the similarity screening process is as follows:

1. Firstly, we maintain a matrix called 'memory matrix' (MeM) that stores the positions of each evaluated particle.
2. When a new position is generated, we compare it with the positions stored in matrix MeM. If a similarity is detected, it suggests that the new position has already been evaluated. In this case, we skip evaluating this particle's position to avoid duplication.
3. On the other hand, if no similarity is found, indicating that the new position is unique and has not been evaluated before, the particle is considered for evaluation as part of the search process.

By implementing this similarity screening process, we can improve the efficiency of the search algorithm by avoiding redundant evaluations, focusing on exploring new and distinct positions during the search, and saving computational resources.

4.3.5 | Local search process

During the search process, we update the positions of particles by incorporating information from both Gb and their respective Pb. Nevertheless, if the Pb value significantly diverges from the optimal solution, there's a risk of the algorithm converging towards a local solution. To mitigate this problem, we introduce a Local Search Process (LSP) aimed at improving the Pb values and thereby enhancing the overall search performance.

The core idea of the LSP can be summarized as follows:

1. At the beginning, a certain number of bits, denoted as R, are randomly selected for modification to another value. This introduces exploration into the search process.
2. Following this, Deb's rule is employed to find the superior solution between Pb and the newly proposed solution, denoted as $NewPb$.
3. Therefore, if the $NewPb$ is better than the current Pb, indicating an improvement in the solution, Pb will be updated to $NewPb$. This update helps in refining the personal best solution and potentially avoiding local optima.

However, to handle additional computational requirements, the LSP is not used at every BBPSO-Quantizer iteration. We introduce the γ parameter, which controls the frequency of using the LSP. Specifically, the LSP has applied every γ iteration.

4.3.6 | Overall approach

The overall framework of our BBPSO-Quantizer is given in Algorithm 2. In the beginning, the position of each particle is initialized. Deb's rule algorithm is employed at each iteration to find the Pb and Gb values. The similarity screening process is utilized to assess the similarity of new particles. If the condition for improving Pb is satisfied, the local search process is adopted to update Pb.

5 | EXPERIMENTAL SETTINGS

This section offers a comprehensive overview of datasets, benchmark models, and parameter settings adopted to test the effectiveness of our BBPSO-Quantizer for mixed precision quantization.

5.1 | Datasets

To evaluate our BBPSO-Quantizer, we utilized three representative datasets: CIFAR-10, CIFAR-100[2], and Tiny-ImageNet[50]. Both CIFAR datasets comprise 50,000 training images and 10,000 testing images. The images in CIFAR are all 32×32 color images. These images are classified into 10 classes in CIFAR-10, whereas, they are classified into 100 classes in CIFAR-100. Tiny-ImageNet is a subset of the ImageNet dataset and comprises 100,000 training images and 10,000 validation images. The size of each image is 64×64. These images are classified into 200 classes, with 500 training images and 50 validation/test images per class. Table 2 presents a summary of the used datasets.

Algorithm 2 Overall framework of BBPSO-Quantizer

Input: pre-trained model $\mathcal{M}$, swarm size: $\mathcal{N}$, $\mathcal{T}$: max iteration, $\mathcal{L}$: number of layers, s:bit-width candidate vector, R: number of bits to be updated; γ: Local search process condition, λ: predefined resource constraint

$MeM \leftarrow \{\}$
for $i = 1$ to N **do** ▷ Initialization phase
  Initialize the particle positionX_i using the initialization method (section 4.3.1)
  $X_{i,fitness} \leftarrow fitness(X_i)$ ▷ Evaluate the particle fitness
  $X_{i,resource} \leftarrow resource(X_i)$ ▷ Evaluate the particle resource
  $MeM \leftarrow MeM + X_i$
  $Pb_i \leftarrow X_i$
end for
Find Gb using algorithm 1
for $t = 1$ to T **do** ▷ Search phase
  for $i = 1$ to N **do**
    Update position X_{new} using Eq.(7)
    similar←False ▷ Similarity screening process
    k←0
    while $X_{new} = MeM[i]$ **do**
      similar←True
      $X_{new,fitness} \leftarrow MeM[i]_{fitness}$
      $X_{new,resource} \leftarrow MeM[i]_{resource}$
      k←k+1
    end while
    if similar=False **then**
      $X_{new,fitness} \leftarrow fitness(X_{new})$
      $X_{new,resource} \leftarrow resource(X_{new})$
      $MeM \leftarrow MeM + X_{new}$
    end if
    Update Pb_i using Algorithm 1
  end for
  if t mod γ =0 **then** ▷ Local search process
    $NewPb \leftarrow Pb_i$
    for $j = 1$ to R **do**
      idx $\leftarrow Rand(1, 2, ..., \mathcal{L})$
      $NewPb[idx] \leftarrow Rand(s - \{Pb[idx]\})$
    end for
    $NewPb_{fitness} \leftarrow fitness(NewPb)$
    $NewPb_{resource} \leftarrow resource(NewPb)$
    Update Pb_i using Algorithm 1 ▷ Find the best solution between NewPb and the current Pb_i
  end if
  Update Gb using Algorithm 1
end for
$\mathcal{M}' \leftarrow Gb$

Output: Optimal quantized model $\mathcal{M}'$

5.2 | CNN architectures for quantization

In our experiments, we employed our proposed BBSO-Quantizer on popular CNN models, namely VGG-16, VGG-19 [51], ResNet-18/-20/-32/-44/-56/-110, DenseNet-40 [52], and the lightweight model MobileNetv2 [53]. Table 3 presents details about the considered models.

Table 2 Summary of datasets used in this study

Dataset	Image Size	Training Images	Test Images	Classes
CIFAR-10	32×32	50000	10000	10
CIFAR-100	32×32	50000	10000	100
Tiny-ImageNet	64×64	100000	10000	200

Table 3 Overview of the CNN models used as baseline to evaluate the proposed BBPSO-Quantizer

Model	FLOPs (M)	Parameter (M)	Size (MB)	Accuracy %		
				CIFAR-10	CIFAR-100	Tiny ImageNet
VGG16	314.03	14.73	58,92	93.02	×	×
VGG19	399.09	20.09	80,36	×	73.11	×
ResNet-20	41.22	0.27	1.08	91.93	62.87	×
ResNet-32	69.76	0.47	1.88	92.34	64.69	×
ResNet-44	98.01	0.66	2.64	92.76	67.79	×
ResNet-56	126.55	0.85	3.40	93.26	67.70	×
Resnet-110	254.99	1.73	6.92	93.50	×	×
DenseNet-40	287.71	1.06	4.24	94.81	×	×
ResNet-18	562.46	11.27	45,08	×	×	63.39
ResNet-50	1310.82	23.91	95,64	×	×	65.47
MobileNetv2	102.43	2.48	9,92	×	×	60.41

5.3 | Evaluation metrics

Evaluation metrics play a critical role in assessing the effectiveness of our BBPSO-Quantizer. In this context, the following evaluation metrics are commonly adopted:

- **Accuracy**: It is a fundamental measure that evaluates the quantized model's performance relative to the original model. A higher accuracy indicates that the quantized model performs well and retains the original model's predictive capabilities.
- **Compression Ratio**: It quantifies the extent to which the model size is reduced through quantization. It is calculated by dividing the original model size (in bits or bytes) by the size of the quantized model. A higher compression ratio implies more efficient use of memory resources, leading to improved storage and transmission efficiency.
- **Average Operation Bit-Width**: It measures the average precision used during computations in the quantized model. It provides insight into the overall efficiency of the quantization process.

Specifically, ΔAccuracy, compression ratio, and Average Operation Bit-Width are defined by Eq. (8), Eq. (9), and Eq. (11), respectively.

$$\Delta Accuracy = accuracy_{quantized} - accuracy_{original} \tag{8}$$

$$compression_{ratio} = \frac{32}{average_{bit}(M_{quantized})} \tag{9}$$

where $average_{bit}()$ denotes the weight's average-bit-width, as defined in Eq. 10:

$$average_{bit}(M_{quantized}) = \frac{\sum_{j=1}^{L} n^j nbr(j)}{\sum_{j=1}^{L} nbr(j)} \tag{10}$$

where n^j denotes the j^{th} layer's weight bit-width. $nbr(j)$ represents the j^{th} layer's parameters number.

$$Average(M_{quantized}) = \left[\frac{\sum_{j=1}^{L} m^j n^j OP(j)}{\sum_{j=1}^{L} OP(j)} \right]^{\frac{1}{2}} \quad (11)$$

where for j^{th} layer, m^j denotes the activation's bit-width, n^j denotes the weight's bit-width and $OP(j)$ represents the number of FLOPs.

5.3.1 | Algorithm parameters

Table 4 provides the details of the hyperparameters used in the BBPSO-Quantizer algorithm. The swarm size (N) determines the number of particles in the swarm. An increased number of particles, coupled with many iterations (T) enables the algorithm to explore a larger portion of the objective space to discover optimal solutions. However, it's important to note that as the number of iterations and/or particles increases, the computational cost also increases. In our experiments, 20 iterations and 15 particles were sufficient to generate good results given a reasonable computing complexity. For fine-tuning, we adopt SGD with a batch size of 128, a weight decay 5e-3, and a momentum of 0.9. As for the epoch and learning schedule, we adopt the following setting:

1. For particle evaluation, each particle is fine-tuned for 2 epochs (ep_{eval}) using a learning rate $lr_{eval} = 0.01$ to recover accuracy for further search before evaluating its accuracy.

2. After the search process, the obtained solution will be fine-tuned for 160 epochs ep_{fine} using learning rate $lr_{fine} = 0.01$ divided by 10 at epochs 80 and 120.

Table 4 Algorithm parameters used to validate the proposed BBPSO-Quantizer

Parameter	Value
Number of iterations T	20
Swarm size N	15
Batch size	128
momentum	0.9
Weight decay	5e-3
ep_{eval}	2
lr_{eval}	0.01
ep_{eval}	160
lr_{fine}	0.01

6 | EXPERIMENTS

6.1 | Results Discussion

Table 5 summarizes the results for all models before and after the application of BBPSO-Quantizer. For each model, the values of accuracy, compression ratio, and average operation bit width are shown. The numerical results in Table 5 demonstrate that BBPSO-Quantizer was able to efficiently quantize original models on the three datasets without greatly affecting the classification accuracy. BBPSO-Quantizer-λ refers to BBPSO-Quantizer with a predefined resource constraint λ.

6.1.1 | Results on CIFAR-10

The evaluation of our BBPSO-Quantizer on the CIFAR-10 shows promising results, as presented in Table 5. Our BBPSO-Quantizer capability in accuracy improvement for several neural network architectures is demonstrated under different average operation bit constraints.

Specifically, for VGG-16, the method achieves improved accuracy by 0.29% with an 11.74x compression ratio under the 3-average operation bit constraint and by 0.56% with a 10x compression ratio under the 4-average operation bit constraint. Figure 3, which illustrates the bit width in VGG-16 under a 4-average operation bit constraint, provides valuable insights into the sensitivity of different layers. This figure demonstrates that

Table 5 Quantization results over CIFAR-10, CIFAR-100 and Tiny-ImageNet

Model	Weight-bits	Activation-bits	Accuracy %	Δ Accuracy	compression ratio	Average
CIFAR-10						
VGG16	32	32	93.02	0.00	1	32
BBPSO-Quantizer-4	M	M	93.55	+0.53	10.00	4.09
BBPSO-Quantizer-3	M	M	93.31	+0.29	11.74	3.03
ResNet-20 original	32	32	91,96	0.00	1	32
BBPSO-Quantizer-4	M	M	92.89	+0.93	9.10	4.02
BBPSO-Quantizer-3	M	M	92.3	+0.34	10.87	3.01
ResNet-32 original	32	32	92,37	0.00	1	32
BBPSO-Quantizer-4	M	M	92.93	+0.56	9.37	3.80
BBPSO-Quantizer-3	M	M	92.47	+0.10	11.15	2.95
ResNet-44 original	32	32	92.72	0.00	1	32
BBPSO-Quantizer-4	M	M	93.20	+0.48	8.33	3.86
BBPSO-Quantizer-3	M	M	93.10	+0.38	10.65	3.09
ResNet-56 original	32	32	93.26	0.00	1	32
BBPSO-Quantizer-4	M	M	93.72	+0.46	8.3	4.03
BBPSO-Quantizer-3	M	M	93.50	+0.24	10.68	3.08
Resnet-110 original	32	32	93.50	0.00	1	32
BBPSO-Quantizer-4	M	M	93.80	+0.40	8.26	4.07
BBPSO-Quantizer-3	M	M	93.68	+0.18	10.88	3.07
DenseNet-40 original	32	32	94.84	0.00	1	32
BBPSO-Quantizer-4	M	M	94.9	-0.06	8.17	4.02
BBPSO-Quantizer-3	M	M	94.6	-0.24	10.95	3.08
CIFAR-100						
VGG19	32	32	73.11	0.00	1	32
BBPSO-Quantizer-4	M	M	73.10	-0.01	9.12	4.06
BBPSO-Quantizer-3	M	M	72.53	-0.42	10.36	2.99
ResNet-44 original	32	32	67.79	0.00	1	32
BBPSO-Quantizer-4	M	M	69.50	+1.71	8	3.96
BBPSO-Quantizer-3	M	M	68.70	+0.91	10.80	2.95
ResNet-32 original	32	32	64.69	0.00	1	32
BBPSO-Quantizer-4	M	M	66.02	+1.33	8.33	3.86
BBPSO-Quantizer-3	M	M	65.60	+0.91	11.99	2.80
ResNet-20 original	32	32	62.87	0.00	1	32
BBPSO-Quantizer-4	M	M	63.90	+1.03	8.33	3.86
BBPSO-Quantizer-3	M	M	63.20	+0.33	11.84	2.97
ResNet-56 original	32	32	67.70	0.00	1	32
BBPSO-Quantizer-4	M	M	67.99	+0.29	8.33	3.86
Tiny-ImageNet						
ResNet-18 original	32	32	63.39	0.00	1	32
BBPSO-Quantizer-3	M	M	63.01	-0.38	10.85	3.08
ResNet50 original	32	32	65.50	0.00	1	32
BBPSO-Quantizer-3	M	M	65.72	+0.22	10.85	3.04
MobileNet V2 original	32	32	60.41	0.00	1	32
BBPSO-Quantizer-3	M	M	60.01	-0.40	10.43	3.04

various layers in the VGG-16 exhibit different levels of sensitivity to quantization, and this sensitivity is reflected in the chosen bit width for each layer.

Notably, the earlier layers of the VGG-16, which are closer to the input, tend to have higher bit widths. This is because these layers are more sensitive to quantization errors, and higher bit-precision is required to preserve accuracy. However, the later layers, which are deeper in the network and closer to the output, are shown to have lower bit widths. This observation indicates that these later layers are less important and can tolerate lower bit precision without significantly impacting the overall network's performance.

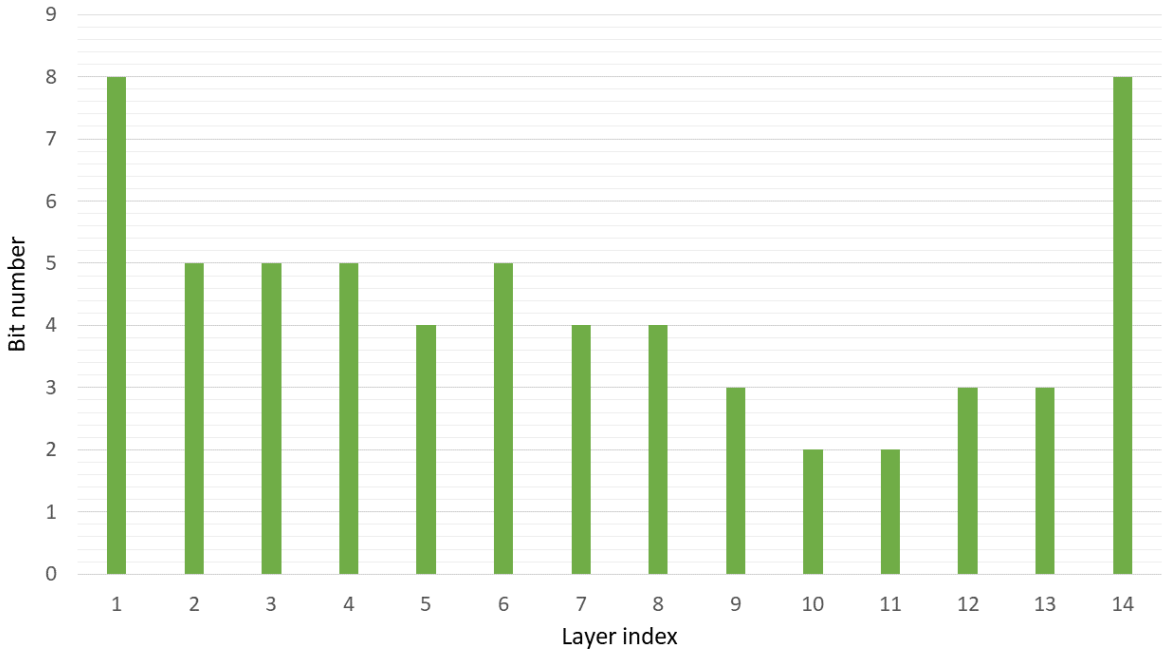


Figure 3 Visualization of bit width for each layer for VGG-16 under 4-average bit constraint

Similarly, BBPSO-Quantizer also improves accuracy for other ResNet architectures such as ResNet-20/-32/-44, and ResNet-56 under different average operation bit constraints. Additionally, DenseNet also benefits from the BBPSO-Quantizer, with accuracy improvement under the specified constraints.

Overall, the consistent accuracy improvements achieved for these various architectures indicate that our BBPSO-Quantizer is not limited to specific network structures and can be effectively applied across a wide range of DNN models. This makes it a valuable tool for enhancing the performance and efficiency of DNN in various applications.

6.1.2 | Results on CIFAR-100

The results obtained on CIFAR-100 further demonstrate the performance of our BBPSO-Quantizer in achieving good results for image classification tasks. The method's ability to enhance accuracy for certain architectures and exhibit relatively low accuracy degradation for others under varying bit constraints highlights its flexibility and efficiency in optimizing quantization.

For instance, in the case of VGG-19, BBPSO-Quantizer causes only -0.01% accuracy degradation under the 4-bit constraint and -0.42% accuracy degradation under the 3-bit constraint. These results indicate that the quantized VGG-19 model still maintains a reasonable level of accuracy even with significantly reduced bit precision. This is crucial as it showcases the ability of the BBPSO-Quantizer to find a balance between accuracy and computational efficiency, making it suitable for resource-constrained environments.

Furthermore, the method achieves accuracy improvement for ResNet-20/-32/-44, and ResNet-56 under the specified constraints. This demonstrates that the BBPSO-Quantizer is effective in enhancing the performance of these ResNet architectures when quantized with reduced bit precision.

In summary, the positive results on CIFAR-10 and CIFAR-100 reinforce the reliability and versatility of the BBPSO-Quantizer in achieving good performance for image classification tasks. The ability to preserve accuracy while quantizing models with fewer bits makes the method a valuable tool for optimizing the efficiency of deep neural networks in various applications.

6.1.3 | Results on Tiny-Imagenet

The results of our BBPSO-Quantizer on the Tiny ImageNet demonstrate its effectiveness in achieving a balance between accuracy and computational efficiency for different neural network architectures. The evaluation was conducted on ResNet-18, ResNet-50, and MobileNetv2 under varying average operation bit constraints. Here are the key findings:

Under 3-average operation bit constraints, our BBPSO-Quantizer causes only 0.38% accuracy degradation for ResNet-18 compared to the fully precision model. Meanwhile, it achieves +0.22% improvement for ResNet-50 and causes only a 0.4% accuracy drop for MobileNetv2.

In conclusion, the BBPSO-Quantizer has demonstrated its effectiveness and efficiency in optimizing the quantization process for various neural network architectures across different datasets, making it a valuable tool in the field of DNNs and model compression.

6.2 | Comparison

The results presented in Table 6 and Table 7 showcase the superiority of our BBPSO-Quantizer over previous methods for neural network quantization on both CIFAR-10 and Tiny-ImageNet, considering both accuracy and compression ratio as evaluation metrics.

The results presented in Table 6 display the consistent superiority of our BBPSO-Quantizer method over existing mixed precision [54,40] and fixed-precision methods [45,55] on CIFAR-10 for various neural network architectures, including VGG-16, ResNet-20, and ResNet-56. Notably, even under high compression ratios, our method continues to show significant accuracy improvements, surpassing the performance of existing techniques.

For instance, our method significantly enhances the accuracy of VGG-16 by +0.29% at an 11.74 compression ratio, while the existing mixed precision method BMPQ [54] causes a -0.34% accuracy drop with a 10.5 compression ratio. This substantial difference in accuracy improvements highlights the effectiveness of our BBPSO-Quantizer in optimizing quantization for different neural network architectures, especially in scenarios where high compression ratios are required.

Additionally, our BBPSO-Quantizer consistently outperforms previous methods such as APoT [45] and LLT [55] on ResNet-20, further emphasizing its superiority in achieving improved accuracy and higher compression ratios on CIFAR-10.

These results demonstrate that our BBPSO-Quantizer provides a more efficient solution for neural network quantization on CIFAR-10, surpassing the performance of existing methods under various compression ratios and network architectures.

Table 6 Comparison of BBPSO-Quantizer with previous quantization methods on CIFAR-10

Model	Method	Weight-bit	Activation-bit	△ accuracy %	Compression ratio
VGG-16	BMPQ [54]	M	M	-0.34	10.50
	BBPSO-Quantizer	M	M	+0.29	11.74
ResNet-20	liu et al. [40]	M	M	-1.31	10.40
	APoT [45]	4	4	+0.70	8.00
	LLT [55]	4	4	-0.25	8.00
	BBPSO-Quantizer	M	M	+0.93	9.10
ResNet-56	Liu et al. [40]	M	M	-0.72	10.34
	BBPSO-Quantizer	M	M	+0.24	10.68

The comparison results on Tiny-ImageNet presented in Table 7 further reinforce the superiority of our proposed BBPSO-Quantizer over previous methods for ResNet-18 and MobileNetV2. The consistent performance of our method on these architectures showcases its effectiveness in achieving both higher compression ratios and accuracy enhancements on the Tiny-ImageNet dataset.

For instance, for ResNet-18, our method achieves an impressive compression ratio of 10.85, compared to the 8.8 compression ratio achieved by the existing mixed precision method BMPQ [54]. Additionally, our method causes only a -0.38% accuracy drop, while the existing method [54] causes a -0.88% accuracy drop. The consistent outperformance of our method for different architectures and datasets showcases its effectiveness in optimizing quantization for DNNs.

Table 7 Comparison of BBPSO-Quantizer with previous quantization methods on Tiny ImageNet

Model	Method	Weight-bit	Activation-bit	△ accuracy %	Compression ratio
ResNet-18	In-Hindsight-Min-Max [56]	8	8	+0.02	4.00
	BMPQ [54]	M	M	-0.88	8.80
	BBPSO-Quantizer	M	M	-0.38	10.85
MobileNetv2	MPLogQ [57]	4.74M	5M	-0.88	-
	In-Hindsight-Min-Max [56]	8	8	-0.33	4.00
	BBPSO-Quantizer	M	M	-0.40	10.43

6.3 | Ablation study

Comparisons in Section 6.2 have demonstrated that our BBPSO-Quantizer is effective for neural network quantization. In this section, we will conduct a detailed performance analysis of the three new mechanisms (the initialization method, similarity screening strategy, and local search strategy). To evaluate the effectiveness of each mechanism, we introduce three variants of BBPSO-Quantizer: BBPSO-Quantizer-IN, BBPSO-Quantizer-SCP, and BBPSO-Quantizer-LSP. Each variant eliminates one of the mechanisms. Specifically, BBPSO-Quantizer-IN replaces the LIGPI method with random initialization, BBPSO-Quantizer-SCP excludes the similarity screening process, and BBPSO-Quantizer-LSP does not involve the local search process.

6.3.1 | Performance of similarity screening process

To evaluate the performance of the similarity screening process introduced in Section 4.3.4 is investigated through ablation experiments on MobileNetv2 on Tiny-ImageNet and ResNet-20 on CIFAR-10. The experiments are conducted on a GPU (Tesla P100-PCIE-16GB), and the summarized GPU times (in hours) taken by our BBPSO-Quantizer and its variant BBPSO-Quantizer-SSP are presented in Table 8. According to the results, BBPSO-Quantizer typically demands less computational time than BBPSO-Quantizer-SSP.

The results demonstrate the importance of the similarity screening process in improving the overall runtime of our algorithm. Without the similarity screening process, searching for mixed precision configurations can be more expensive and time-consuming.

Table 8 Run-time comparison between BBPSO-Quantizer and BBPSO-Quantizer-SCP

Dataset	Model	Method	Search cost (GPU hours)
CIFAR-10	ResNet-20	BBPSO-Quantizer-SCP	7.12
		BBPSO-Quantizer	5.13
Tiny-ImageNet	MobileNetv2	BBPSO-Quantizer-SCP	12.30
		BBPSO-Quantizer	9.40

6.3.2 | Performance of initialization method and local search process in BBPSO-Quantizer

Table 9 provides a comparative analysis of the results achieved by our BBPSO-Quantizer and its variants. The table illustrates the consistent outperformance of our algorithm in terms of accuracy under the same constraint when compared to its variants.

The significantly better accuracies obtained by our algorithm compared to its variants highlight its ability to find better solutions and achieve higher performance. This showcases the efficacy of the proposed enhancements in guiding the search process toward more accurate solutions, ultimately leading to improved performance on the tested datasets.

The comparisons between BBPSO-Quantizer and its three variants (BBPSO-Quantizer-IN, BBPSO-Quantizer-LSP, and BBPSO-Quantizer-SSP) have demonstrated the effectiveness of the initialization method, similarity screening process, and local search strategy in enhancing the search performance.

Table 9 Comparisons between BBPSO-Quantizer and its variants

Dataset	Model	Method	△ accuracy%	average-bit
CIFAR-10	ResNet-20	BBPSO-Quantizer	+0.34	3.01
		BBPSO-Quantizer-IN	+0.25	3.07
		BBPSO-Quantizer-LSP	+0.17	3.03
Tiny-ImageNet	MobileNetv2	BBPSO-Quantizer	-0.40	3.04
		BBPSO-Quantizer-IN	-0.80	3.09
		BBPSO-Quantizer-LSP	-0.60	3.06

6.4 | Statistical analysis

In this section, a comparative analysis of the performance of BBPSO-Quantizer with conventional Bare-Bones Particle Swarm Optimization (BBPSO), and Genetic Algorithm (GA) is conducted.

Each algorithm is run 20 times on three models: MobileNetv2, ResNet-18, and ResNet-50 on the Tiny-ImageNet dataset. Table 10 shows the results in terms of accuracy and statistical comparison based on the Mann–Whitney U test and Friedman test. In the table, mean and standard deviation values of the best and second-best performing algorithms (highest mean accuracy) for each model are highlighted with gray and light gray shading, respectively. Additionally, the symbol '+' signifies that BBPSO-Quantizer outperforms the peer algorithm. Considering the accuracy, BBPSO-Quantizer is better than BBPSO and GA in all models.

Moreover, the average row in the table showcases the mean ranks assigned to each algorithm across diverse models according to the Friedman test. The algorithms are sorted in ascending order according to their accuracy values, and the algorithm with the highest accuracy is assigned a higher rank. Hence, a larger rank indicates better performance. Table 10 highlights the rank associated with the best performance. The average rank results consistently support the statistical superiority of the BBPSO-Quantizer algorithm. All the statistical tests conducted were performed with a p-value of 0.05.

To visually illustrate the performance and stability of different algorithms, accuracy values are depicted in Fig. 4. The x-axis denotes the number of independent runs, while the y-axis represents the accuracy value for each run. Notably, among the 20 independent runs, the accuracy values of BBPSO-Quantizer consistently outperform those of the other algorithms across various models.

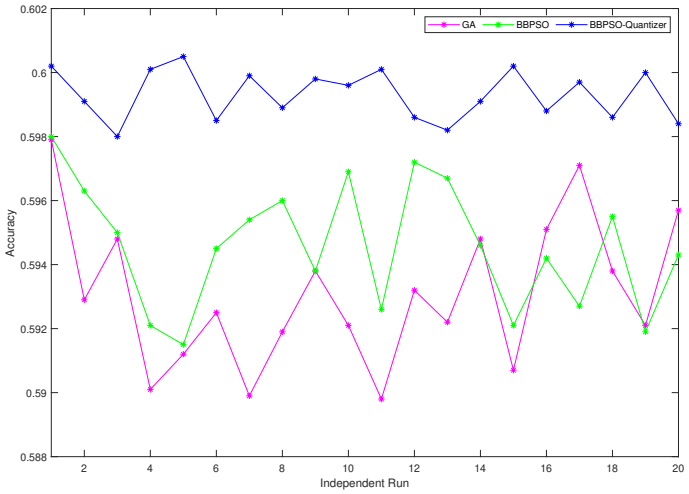
Table 10 Comparison results of BBPSO-Quantizer, BBPSO, and GA in terms of accuracy (Mean and standard deviation) and the average Friedman ranks with scores on Tiny ImageNet dataset

Model	**GA**		**BBPSO**		**BBPSO-Quantizer**
Mobilenetv2	$59.31(2.24e-01)$	(+)	$59.46(1.90e-01)$	(+)	$59.93(7.58e-02)$
Resnet18	$62.28(2.17e-01)$	(+)	$62.51(1.95e-01)$	(+)	$62.94(6.47e-02)$
Resnet50	$65.08(1.34e-01)$	(+)	$65.35(1.95e-01)$	(+)	$65.67(4.95e-02)$
Average Friedman rank	1.0		2.0		**3.0**
Score	3		2		**1**

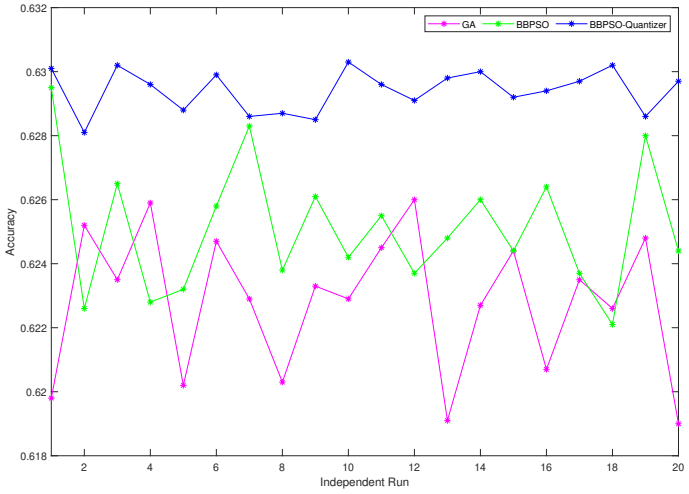
7 | CONCLUSION

In this work, we have presented BBPSO-Quantizer, a method for mixed precision quantization of neural networks using the enhanced BBPSO algorithm. Our method addresses the challenges of reducing model size and computational complexity while maintaining high accuracy, making DNNs suitable for deployment on resource-constrained devices.

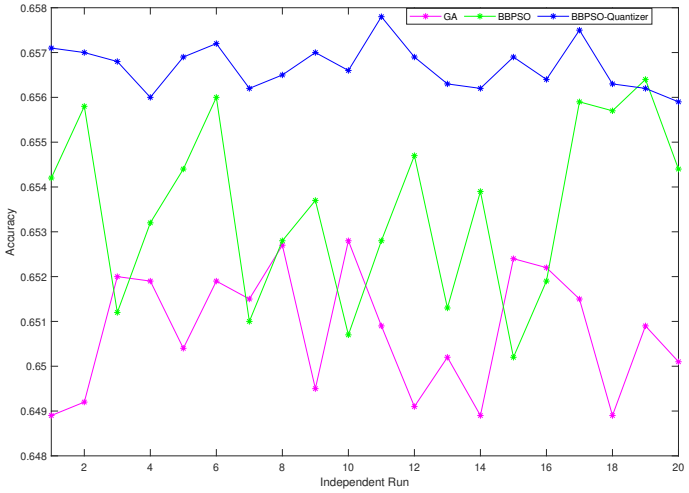
Through extensive experiments on CIFAR-10, CIFAR-100, and Tiny-ImageNet datasets, we have demonstrated the efficiency of BBPSO-Quantizer in optimizing quantization for various CNN architectures, including ResNet, MobileNetv2, VGG, and DenseNet. The results showcase a



(a) Mobilenetv2



(b) Resnet18



(c) Resnet50

Figure 4 Mean Accuracy curves for Mobilenetv2, Resnet18 and Resnet50 models evaluated on Tiny ImageNet dataset

compelling balance between accuracy and computational efficiency, with significant accuracy improvements achieved for different neural network architectures under varying average operation bit constraints.

The incorporation of new population initialization, a robust similarity screening process, and a local search strategy have improved the search performance, guiding the population towards feasible regions. Moreover, the integration of Deb's constraint handling method ensures the optimized solutions meet functional constraints, leading to more reliable and practical quantized models. Additionally, we have conducted ablation studies and compared BBPSO-Quantizer against its variants, confirming the superiority of the proposed mechanisms in improving search performance and overall runtime.

While the study achieved efficient experiments, there are still some limitations. One notable limitation involves deploying quantized models onto real-world hardware, particularly those optimized for edge computing. These hardware are specifically equipped for the efficient execution of quantized operations.

In the future, we aim to address the practical implementation of quantized models on real-world hardware. In addition, we aim to apply BBPSO-Quantizer to tackle other challenging models, such as large language models (LLMs), transformers, and YOLO models. Our plan involves exploring the integration of BBPSO-Quantizer with other compression techniques to further reduce computational costs while ensuring a satisfactory level of accuracy.

CONFLICT OF INTEREST

The authors have no conflicts of interest to declare.

ACKNOWLEDGEMENTS

The authors are grateful to the anonymous reviewers for their insightful comments and suggestions which helped improve the quality of this paper. The research leading to these results has received funding from the Ministry of Higher Education and Scientific Research of Tunisia under grant agreement number LR11ES48. Professor Hussain acknowledges the support of the UK Engineering and Physical Sciences Research Council (EPSRC) (Grants No. EP/M026981/1, EP/T021063/1, EP/T024917/1)

References

1. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. *Communications of the ACM* 2017; 60(6): 84–90.

2. Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. tech. rep., University of Toronto; : 2009.

3. Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision* 2015; 115: 211–252.

4. Sharma V, Gupta SK, Shukla KK, others . Deep learning models for tuberculosis detection and infected region visualization in chest X-ray images. *Intelligent Medicine* 2023.

5. Redmon J, Farhadi A. YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE. ; 2017: 7263–7271.

6. Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer. ; 2016: 21–37.

7. Redmon J, Farhadi A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* 2018.

8. Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. IEEE. ; 2017: 2980–2988.

9. Yuan HS, Chen SB, Luo B, Huang H, Li Q. Multi-branch Bounding Box Regression for Object Detection. *Cognitive Computation* 2022: 1–8.

10. Prinzi F, Insalaco M, Orlando A, Gaglio S, Vitabile S. A Yolo-Based Model for Breast Cancer Detection in Mammograms. *Cognitive Computation* 2023: 1–14.

11. Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* 2018.

12. Jiao X, Yin Y, Shang L, et al. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351* 2019.

13. Sharma I, Sharma A, Gupta SK. Asynchronous and Synchronous Federated Learning-based UAVs. In: IEEE. ; 2023: 105–109.

14. Jain K, Kaushik K, Gupta SK, Mahajan S, Kadry S. Machine learning-based predictive modelling for the enhancement of wine quality. *Scientific Reports* 2023; 13(1): 17042.

15. Liu S, Lin Y, Zhou Z, Nan K, Liu H, Du J. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. ; 2018: 389–400.

16. Tmamna J, Ayed EB, Ayed MB. Deep learning for internet of things in fog computing: survey and open issues. In: 2020 5th International conference on advanced technologies for signal and image processing (ATSIP). IEEE. ; 2020: 1–6.

17. Tmamna J, Ayed EB, Ayed MB. Neural Network Pruning Based on Improved Constrained Particle Swarm Optimization. In: Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part VI 28. Springer. ; 2021: 315–322.

18. Liu Y, Wu D, Zhou W, Fan K, Zhou Z. EACP: An Effective Automatic Channel Pruning for Neural Networks. *Neurocomputing* 2023.

19. Bablani D, Mckinstry JL, Esser SK, Appuswamy R, Modha DS. Efficient and Effective Methods for Mixed Precision Neural Network Quantization for Faster, Energy-efficient Inference. *arXiv preprint arXiv:2301.13330* 2023.

20. Ma Y, Jin T, Zheng X, et al. Ompq: Orthogonal mixed precision quantization. In: Proceedings of the AAAI Conference on Artificial Intelligence. ; 2023: 9029–9037.

21. Yim J, Joo D, Bae J, Kim J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE. ; 2017: 4133–4141.

22. Zhao H, Sun X, Dong J, Chen C, Dong Z. Highlight every step: Knowledge distillation via collaborative teaching. *IEEE Transactions on Cybernetics* 2020; 52(4): 2070–2081.

23. Courbariaux M, Bengio Y, David JP. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems* 2015; 28.

24. Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. Binarized neural networks. *Advances in neural information processing systems* 2016; 29.

25. Rastegari M, Ordonez V, Redmon J, Farhadi A. Xnor-net: Imagenet classification using binary convolutional neural networks. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV. Springer. ; 2016: 525–542.

26. Truong SN. A Ternary Neural Network with Compressed Quantized Weight Matrix for Low Power Embedded Systems. *Engineering, Technology & Applied Science Research* 2022; 12(2): 8311–8315.

27. Liu D, Liu X. Ternary Quantization: A Survey. *arXiv preprint arXiv:2303.01505* 2023.

28. Esser SK, McKinstry JL, Bablani D, Appuswamy R, Modha DS. Learned step size quantization. *arXiv preprint arXiv:1902.08153* 2019.

29. Liu Z, Cheng KT, Huang D, Xing EP, Shen Z. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE. ; 2022: 4942–4952.

30. Zhou S, Wu Y, Ni Z, Zhou X, Wen H, Zou Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160* 2016.

31. Lin Z, Courbariaux M, Memisevic R, Bengio Y. Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009* 2015.

32. Cai Z, He X, Sun J, Vasconcelos N. Deep learning with low precision by half-wave gaussian quantization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE. ; 2017: 5918–5926.

33. Choi J, Wang Z, Venkataramani S, Chuang PJ, Srinivasan V, Gopalakrishnan K. Pact: Parameterized clipping activation for quantized neural networks. arXiv 2018. *arXiv preprint arXiv:1805.06085* 2018.

34. Dong Z, Yao Z, Arfeen D, Gholami A, Mahoney MW, Keutzer K. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems* 2020; 33: 18518–18529.

35. Dong Z, Yao Z, Gholami A, Mahoney MW, Keutzer K. Hawq: Hessian aware quantization of neural networks with mixed-precision. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. IEEE. ; 2019: 293–302.

36. Wang K, Liu Z, Lin Y, Lin J, Han S. Haq: Hardware-aware automated quantization with mixed precision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE. ; 2019: 8612–8620.

37. Yuan Y, Chen C, Hu X, Peng S. Evoq: Mixed precision quantization of dnns via sensitivity guided evolutionary search. In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE. ; 2020: 1–8.

38. Lou Q, Liu L, Kim M, Jiang L. Autoqb: Automl for network quantization and binarization on mobile devices. *arXiv preprint arXiv:1902.05690* 2019; 2(8).

39. Yao Z, Dong Z, Zheng Z, et al. Hawq-v3: Dyadic neural network quantization. In: International Conference on Machine Learning. PMLR. ; 2021: 11875–11886.

40. Liu H, Elkerdawy S, Ray N, Elhoushi M. Layer importance estimation with imprinting for neural network quantization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE. ; 2021: 2408–2417.

41. Tang C, Ouyang K, Wang Z, et al. Mixed-Precision Neural Network Quantization via Learned Layer-Wise Importance. In: European Conference on Computer Vision. Springer. ; 2022: 259–275.

42. Nikolić M, Hacene GB, Bannon C, et al. Bitpruning: Learning bitlengths for aggressive and accurate quantization. *arXiv preprint arXiv:2002.03090* 2020.

43. Yang L, Jin Q. Fracbits: Mixed precision quantization via fractional bit-widths. In: Proceedings of the AAAI Conference on Artificial Intelligence. ; 2021: 10612–10620.

44. Liu Z, Zhang X, Wang S, Ma S, Gao W. Evolutionary quantization of neural networks with mixed-precision. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE. ; 2021: 2785–2789.

45. Li Y, Dong X, Wang W. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144* 2019.

46. Kennedy J, Eberhart R. Particle swarm optimization. In: . 4 of *Proceedings of ICNN'95-international conference on neural networks*. IEEE. ; 1995: 1942–1948.

47. Kennedy J. Bare bones particle swarms. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706). IEEE. ; 2003: 80–87.

48. Deb K. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering* 2000; 186(2-4): 311–338.

49. Song XF, Zhang Y, Gong DW, Gao XZ. A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data. *IEEE Transactions on Cybernetics* 2021; 52(9): 9573–9586.

50. Le Y, Yang X. Tiny imagenet visual recognition challenge. *CS 231N* 2015; 7(7): 3.

51. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: ICLR. ; 2015: 1-14.

52. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE. ; 2017: 4700–4708.

53. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. ; 2018: 4510–4520.

54. Kundu S, Wang S, Sun Q, Beerel PA, Pedram M. Bmpq: bit-gradient sensitivity-driven mixed-precision quantization of dnns from scratch. In: 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE. ; 2022: 588–591.

55. Wang L, Dong X, Wang Y, Liu L, An W, Guo Y. Learnable lookup table for neural network quantization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. IEEE. ; 2022: 12423–12433.

56. Fournarakis M, Nagel M. In-hindsight quantization range estimation for quantized training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE. ; 2021: 3063–3070.

57. Choi D, Kim H. Hardware-Friendly Logarithmic Quantization with Mixed-Precision for MobileNetV2. In: 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS). IEEE. ; 2022: 348–351.