# EvoQ: Mixed Precision Quantization of DNNs via Sensitivity Guided Evolutionary Search

Yong Yuan[1,2], Chen Chen[1,2*], Xiyuan Hu[1,2], Silong Peng[1,2,3]
[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, China
[3]Beijing Visystem Co. Ltd
{*yuanyong2015, chen.chen, xiyuan.hu, silong.peng*}*@ia.ac.cn*

***Abstract*—Network quantization can effectively reduce computation and memory costs without modifying network structures, facilitating the deployment of deep neural networks (DNNs) on edge devices. However, most of the existing methods usually need time-consuming training or fine-tuning and access to the original training dataset that may be unavailable due to privacy or security concerns. In this paper, we introduce a novel method named EvoQ that employs evolutionary search to achieve mixed precision quantization with limited data, which can optimize the resource allocation without adding computation consumption. Considering the shortage of samples and expensive search costs, we use 50 samples to measure the output difference between the quantization model and the pre-trained model for the evaluation of quantization policy, which can save the time obviously while maintaining high accuracy. To improve the search efficiency, we analyze the quantization sensitivity of each layer and utilize the results to optimize the mutation operation. At last, we calibrate the outputs and intermediate features of the quantization model using the selected 50 samples to improve the performance further. We implement extensive experiments on a diverse set of models, including ResNet18/50/101, SqueezeNet, ShuffleNetV2, and MobileNetV2 on ImageNet, as well as SSD-VGG and SSD-ResNet50 on PASCAL VOC. Our method can improve the performance apparently and outperforms the existing post-training quantization methods, demonstrating the effectiveness of EvoQ.**

***Index Terms*—evolutionary algorithm, mixed precision quantization, network compression, deep learning**

## I. Introduction

Over the past few years, deep neural networks (DNNs) have achieved state-of-the-art performance on various tasks such as speech recognition, computer vision, and natural language processing. However, the significant increase of the computational complexity hinders the deployment of DNNs on embedded devices [1] [2]. Many methods have been proposed to address this problem, such as network quantization [3] [4], weight pruning [5], low-rank decomposition [6], compact structure design [8] [9] [10], knowledge distillation [7], and neural architecture search [11] [12] [13]. Among these methods, network quantization that represents the weights and activations with low precision format, replacing most of the floating-point operations with fixed-point operations, can reduce computation, memory, and power consumption remarkably.

*Corresponding author at: 95 Zhongguancun East Road, Beijing 100190, China. E-mail addresses: chen.chen@ia.ac.cn

The majority of literature on network quantization involves training either from scratch or as a fine-tuning step. The relevant methods can achieve extremely low precision quantization, even representing weights with binary values [14] [15] [16]. However, these methods usually need access to the full training dataset, which may be unavailable for different reasons, such as privacy and security concerns. Besides, the training or fine-tuning process usually requires long periods of optimization as well as expensive computation costs. Consequently, it is desirable to apply quantization without the full training dataset. These methods are commonly referred to as post-training quantization, which only needs a small set of samples to accomplish the whole quantization process. Nevertheless, most existing works only manage to quantize weights to 8-bit integer (INT8) [17], and the lower precision post-training quantization incurs considerable accuracy drop. More efforts are needed to improve the performance of post-training quantization below 8 bits.

In DNN models, the importance and structure of different layers differ a lot, resulting in different properties to quantization. Therefore, the methods that assign uniform bit-width to different layers are not optimal. Comparatively, mixed precision quantization can optimize bit-width allocation and improve the representative ability of DNNs without additional computation costs. The layers having much relativity with the performance are assigned more bits. Reversely, the layers having little relativity with the performance are assigned fewer bits. The native mixed precision quantization method can be computationally expensive, as the search space can be exponential. [18] [19] [20] [21] assumed the quantization error per-layer is additive and independent. In reality, these methods are not applicable for low precision quantization, as the additivity and independence are not satisfied when the quantization error is significant. [22] [23] [24] used neural architecture search (NAS) based methods to explore the quantization policy. Nevertheless, these searching based methods can be computation expensive and usually are built on the full dataset.

In this paper, we propose a novel post-training quantization method named EvoQ to overcome the above issues. In particular, we use an evolutionary algorithm to achieve mixed precision quantization with limited data (50 samples). In summary, our contributions are as follows:

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

- We evaluate the quantization policy by measuring the output difference between the quantization model and the pre-trained full precision model using 50 samples, which can speed up the search process significantly.
- We analyze the per-layer quantization sensitivity and utilize the analysis results to optimize the mutation operation, which can improve the search efficiency and help escape from the local minimum.
- We use 50 samples to calibrate the outputs and intermediate features of the quantization model with the pre-trained full precision model, which can improve the performance apparently.
- We extensively test our method on two different tasks and various network structures, showing that our method outperforms the existing post-training quantization methods.

## II. Related Work

### A. Network quantization

Network quantization has been heavily studied by the research communities, and ultra-low precision quantization of DNNs can be achieved nowadays. Courbariaux *et al.* [15] proposed to represent the weights with binary values, which can significantly reduce the storage and eliminate most of the multiplications. Hubara *et al.* [16] binarized the weights and activations simultaneously, replacing the floating-point operations with XNOR and bit-count operations. In addition to weights and activations, Zhou *et al.* [4] quantized the gradients to low precision format, which can accelerate the training further. The above methods usually need time-consuming training and access to the full dataset. Recently, a lot of efforts have been made to enhance the performance of post-training quantization, which does not need time-consuming training and only need a small set of examples. He *et al.* [25] used 1,000 unlabeled samples to re-estimate the statistical parameters of the batch normalization layer for accuracy recovery. Banner *et al.* [27] combined per-channel bit-allocation, bias correction, and analytical clipping for integer quantization together. Choukroun *et al.* [26] adopted multi-quantization for the high MSE layer and refined the quantization scaling factors to achieve better approximation. Zhao *et al.* [28] proposed outlier channel splitting (OCS), which duplicated channels containing outliers, then halved the channel values. The conventional quantization methods usually assign the same bit-width to all layers, while mixed precision quantization can optimize the resource allocation without adding consumption. Darryl *et al.* [18] used the signal-to-quantization-noise ratio (SQNR) to measure the effect of quantization error and determined the bit-width for each layer. Zhou *et al.* [19] theoretical analyzed the correlation between the quantization error and the model accuracy, and then optimized the bit-width allocation for different layers. These methods needed domain expertise for bit-width allocation and assumed the quantization error per-layer is additive and independent.

### B. Neural architecture search

Neural architecture search (NAS) has become an extremely hot research topic in the past two years. Relevant studies try to find optimal network structures and hyper-parameters via reinforcement learning, genetic algorithms, or gradient-based approaches. Zoph *et al.* [12] first utilized reinforcement learning to generate neural network architectures with high accuracy and efficiency. Real *et al.* [13] adopted evolutionary algorithms as the controller, in which genetic operations were used to modify the architecture. Liu *et al.* [11] proposed DARTS, a differentiable architecture search framework, to search in a continuous search space. The automatic methods can generate network structures that outperform the classical hand-crafted models. In addition to architecture search, NAS also can be combined with network quantization and pruning. He *et al.* [29] leveraged reinforcement learning to prune the convolution channels automatically. Wang *et al.* [23] provided a reinforcement learning-based method to search the quantization policy with hardware feedback. Wu *et al.* [24] proposed an effective differentiable neural architecture search (DNAS) framework to determine the bit-width allocation. Chen *et al.* [21] introduced an evolutionary search algorithm to achieve architecture search with quantization. The searching based methods can achieve mixed precision quantization without domain experts, but require a large number of computational resources and the full dataset.

## III. Proposed Method

In this work, we use an evolutionary algorithm to explore mixed precision quantization policy in a heuristic manner. Given a pre-trained full precision model $M$, our target is to find an optimized quantization policy $\Pi(b_1...b_l)$, where $b_i$ denotes the quantization bit-width of the $i$-th layer. The problem can be defined as:

$$\max_{\Pi(b_1...b_l)} F(\Pi(b_1...b_l), M) \quad s.t. \frac{\sum_{i=1}^{l} C_i * b_i}{\sum_{i=1}^{l} C_i} \leq b_{target} \quad (1)$$

where $F(\cdot)$ denotes the quantization policy evaluation function, $C_i$ is the parameter size of the $i$-th layer, $l$ represents the layer numbers, and $b_{target}$ represents the target average bit-width. $b_{min}$ and $b_{max}$ denote the min and max bit-width, repectively. The total search space is exponential and equals to $(b_{max} - b_{min})^l$.

### A. Quantization policy evaluation

Note that the direct way for quantization policy evaluation is to assess the quantization model on the test dataset, which is accurate, but the process is time-consuming for large test dataset. In addition, the test dataset may be unavailable due to privacy concerns. Hence, we need a more efficient way to evaluate the quantization policy. Some of the mixed precision quantization methods assume the quantization error of each

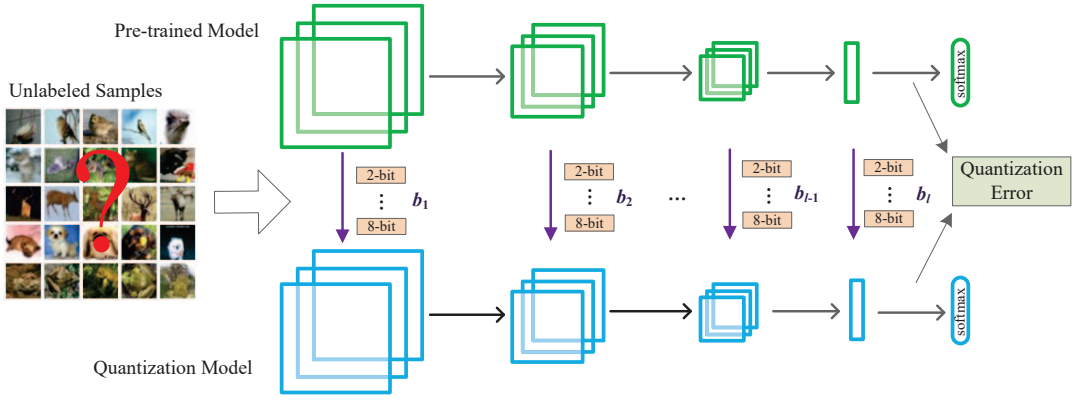


Fig. 1. The illustration of quantization policy evaluation using limited data.

layer is additive and independent [21] [27], then the fitness of the quantization policy can be measured as:

$$F(\Pi(b_1...b_l), M) = 1/\sum_{j=1}^{l} E(b_j) \tag{2}$$

where $E(b_j)$ denotes the quantization error caused by the $j$-th layer. Based on the quantization error per-layer, the evaluation can be finished within $l-1$ addition operations. However, the additivity is unsatisfactory for low precision quantization, as the coupling effect of different layers can not be ignored anymore. Besides, the highly non-linearity of DNNs makes the quantization error difficult to analyze. Considering the above limitations, we propose to evaluate the quantization policy through measuring the output difference between the quantization model and the pre-trained model using $N$ samples:

$$E(\Pi(b_1...b_l)) = \frac{1}{N}\sum_{i=1}^{N}(Q_{\Pi(b_1...b_l)}(x_i) - M(x_i))^2 \tag{3}$$

where $Q_{\Pi(b_1...b_l)}$ denotes the model quantized with policy $\Pi(b_1...b_l)$, $x_i$ denotes the $i$-th input sample. As the model outputs contain more information than hard labels, the quantization policy can be evaluated accurately with few samples. The process can be illustrated in Fig. 1.

### B. Search strategy

To automatically search for high-performing quantization policy, we employ a classical evolutionary algorithm, tournament selection. The procedure is summarized in Algorithm 1. It keeps a population of $P$ quantization policy throughout the experiment. The population is initialized with a uniform quantization policy and its random perturbations. After this, evolution improves the initial population in iterations. Each individual (quantization policy) is evaluated according to (3) using $N$ unlabeled samples. At each evolutionary step, $S$ quantization policies are randomly sampled from the population. The quantization policy with the highest fitness in the sample is selected as the parent. A new quantization policy, called the child, is constructed from the parent by mutation operation. The quantization policy with the worst fitness in the sample is excluded from the population, and the mutated child is pushed into the population. This scheme uses repeated competitions of random individuals to search for an optimized quantization policy. The parameter $S$ controls the aggressiveness of the search: $S = 1$ means random search, and $2 \leq S \leq P$ leads to the evolution of varying greediness.

**Algorithm 1** Evolutionary mixed precision quantization search

**Input:** pre-trained model $M$, population size $P$, sample size $S$, max iterations $T$, target bit-width $b_{target}$, sampled data $D$, searching range $(b_{min}, b_{max})$, mutation probability $p$.

```
population ← ∅
while |population| < P do
    Π(b_1...b_l) ← RANDOM_QUANT(M, b_target)
    Fitness ← F(Π(b_1...b_l), M)
    push Π(b_1...b_l) into population
end while
while iteration < T do
    sample ← ∅
    while |sample| < S do
        sample ← RANDOM_SAMPLE(population)
    end while
    Π_parent ← highest-fitness quantization policy in sample
    Π_worst ← lowest-fitness quantization policy in sample
    Π_child ← MUTATE(Π_parent, p, b_target)
    Fitness ← F(Π(b_1...b_l), M)
    push Π_child into population
    remove Π_worst from population
end while
```

**Output:** Quantization policy with the highest fitness $\Pi_{best}$.

New quantization policies are generated by applying mutation operation on existing quantization policies to search the solutions in random ways. Consider the exponential search space, the efficiency of randomly picking quantization bit-width from $(b_{min}, b_{max})$ can be very low. To optimize the search efficiency, we use the quantization sensitivity of each layer to optimize the mutation direction. We first employ $N$ samples to evaluate the quantization error per-layer as:

$$E(b_j) = \frac{1}{N}\sum_{i=1}^{N}(Q_{b_j}(x_i) - M(x_i))^2 \tag{4}$$

where $Q_{b_j}$ denotes the quantization model that the $j$-th layer is quantized to $b_j$ bits. Per-layer quantization error of MobileNetV2 can be shown in Fig. 2. As we can see, the quantization error of different layers varies a lot, and this is the motivation of mixed precision quantization. Based on the bit-width allocation of the individual, we calculate the relative gain or loss per-layer when we increase or reduce quantization bit-width. For the layers that are more sensitive to current bit-width, we increase their probability for higher bit-width. For the layers that are less sensitive to current bit-width, we increase their probability for lower bit-width. This modification can improve the efficiency of search and helps escape from the local minimum.

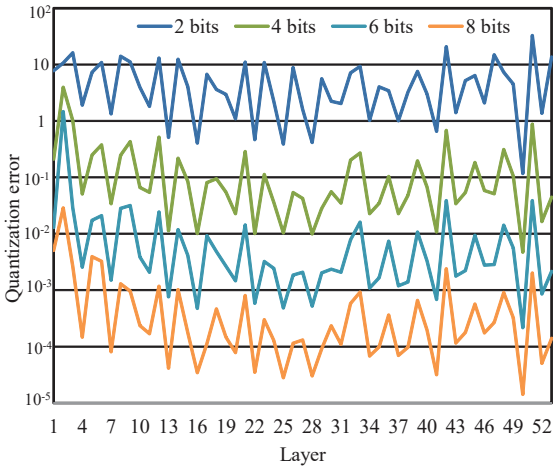


Fig. 2. Quantization error of each layer on MobileNetV2 when weights are quantized to 2/4/6/8 bits.

### C. Feature calibration

The performance of the quantization model can be further improved by calibrating the features using the teacher-student framework. The pre-trained full precision model is considered as a teacher, and the quantization model is considered as a student. Because of the limitation of samples, simply aligning the outputs will lead to the mismatch of intermediate features, resulting in overfitting finally. The intermediate features that have more dimensions and contain affluent information also can be utilized for calibration. The loss function can be formulated as:

$$L = \frac{1}{N}\sum_{i=1}^{N}(\alpha(Q(x_i) - M(x_i))^2 + \beta(O_Q(x_i) - O_M(x_i))^2) \tag{5}$$

where $O_Q$ and $O_M$ denote the intermediate features of the quantization model and the pre-trained model, respectively, $\alpha$ and $\beta$ are the weighting factor. To optimize the quantized weights $w_q$, FP32 version weights $w_f$ are reserved for parameter updating. The FP32 weights are updated based on the gradients of quantized weights, and then FP32 weights are quantized for parameter updating. To address the non-differentiation of activation quantization, the "straight-through estimator" is used for estimating the gradients. The details are demonstrated in Algorithm 2.

**Algorithm 2** Few samples based feature calibration

**Input:** pre-trained model $M$, quantization policy $\Pi_{best}$, max epochs $E$, sampled data $D$, learning rate $\eta$.

$Q \leftarrow QUANT(M, \Pi_{best})$
Initialize the FP32 weight $w_f^0 = w_q^0$.
**for** $t$ = 1,...,$E$ **do**
  **Forward propagation**:
    Compute the features and outputs of $M$ and $Q$.
    Compute the calibration loss $L$.
  **Backward propagation**:
    Compute the gradients of quantized weights: $\frac{\partial L}{\partial w_q^t}$.
  **Parameter update**:
    Update full precision weights: $w_f^{t+1} = w_f^t - \eta^t \frac{\partial L}{\partial w_q^t}$.
    Quantize $w_f^{t+1}$ to update quantized weights $w_q^{t+1}$.
**end for**

**Output:** The optimized quantization model $Q_{best}$.

## IV. Experiments

In this section, we verify the effectiveness of the proposed method on image classification and object detection. In all experiments, batch normalization is folded into the adjacent layer before quantization. We use asymmetric uniform quantization, and perform per-channel quantization and per-layer quantization for weights and activations, respectively. In this work, we only explore the mixed precision quantization policy for weights, and activations are uniformly quantized to 8-bit representation. All experiments are conducted in Pytorch [31].

### A. Experiments on image classification

In this section, we evaluate EvoQ on the large scale image classification dataset, ImageNet2012, which contains over 1.2 million training images, 100k test images, and 50k validation images. Each image is classified into one of the 1000 object categories. We randomly select samples from the training set for sensitivity analysis, quantization policy evaluation, activation range estimation, and feature calibration. All images are resized to 256×256 and then cropped to 224×224 without data augmentation. Extensive experiments are implemented on ResNet18/50/101, as well as compact network architectures SqueezeeNet, ShuffleNetV2, and MobileNetV2. We obtain the pre-trained full precision model from the official community of PyTorch. In evolutionary search, population size $P = 16$, sample size $S = 8$, max iterations $T = 1000$, mutation probability $p = 0.1$, target bit-width $b_{target} = 4$, and the quantization bit-width ranges in $(2, 8)$. In feature calibration, SGD with a momentum of 0.9 is set as the optimizer, and the learning rate is set to 0.0001.

We randomly select 50, 100, 200, and 400 samples from the training dataset to evaluate the effect of sample numbers on EvoQ. As is shown in Table I, increase the sample numbers only can improve the performance on MobileNetV2 slightly. Thus, we only use 50 samples in the following experiments, which can save 2000× times comparing to the test dataset. The evolution processes of ResNet18/50/101, SqueezeNet,

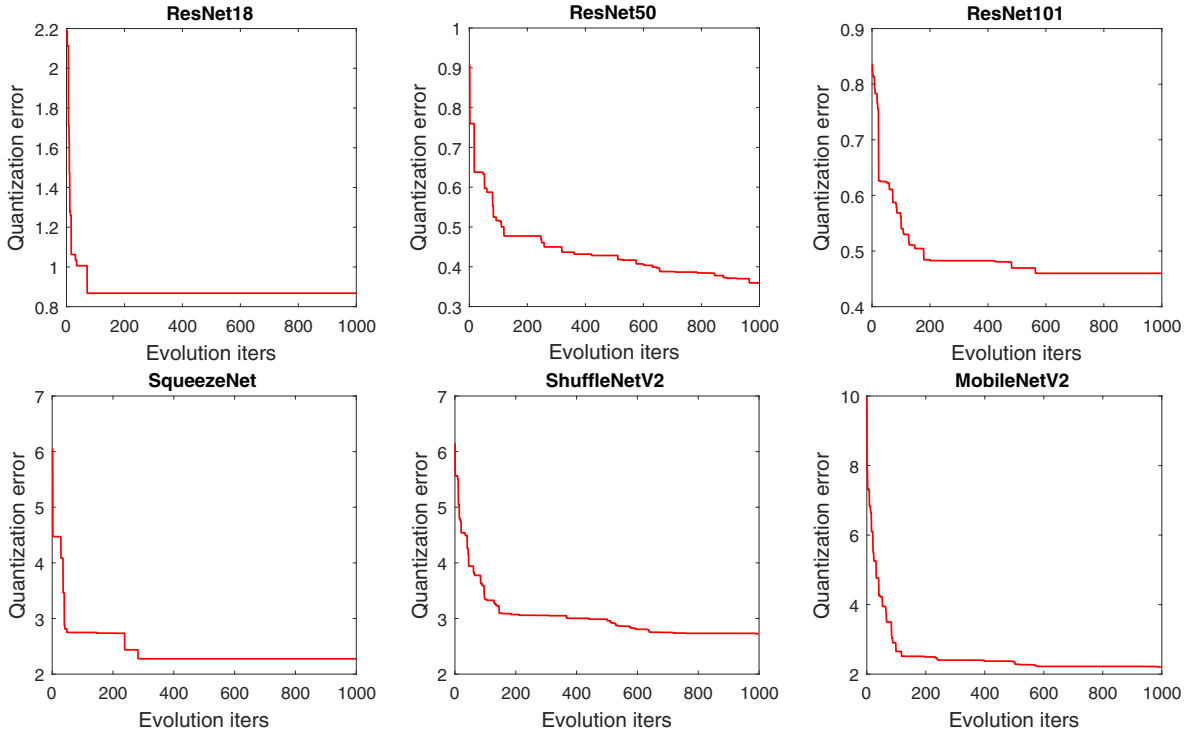


Fig. 3. The search process of EvoQ on ResNet18, ResNet50, ResNet101, SqueezeNet, ShuffleNetV2 and MobileNetV2.

ShuffleNetV2, and MobileNetV2 can be shown in Fig. 3. The quantization error decreases rapidly in the early iterations, and the optimization process can be accomplished within 300 iterations for ResNet18 and SqueezeNet. We visualize the quantization policy of ShuffleNetV2 and MobileNetV2 in Fig. 4. From the figure, we can observe that depthwise convolution layers are assigned more bits than pointwise convolution layers, and former layers usually have more bits than latter layers. Intuitively, this is because the depthwise convolution layers and the former layers usually have fewer parameters. We compare EvoQ with uniform precision quantization, OMSE [15], and ACIQ [13]. As presented in Table II and Table III, EvoQ can improve the performance apparently comparing to uniform precision quantization, especially for compact network structures. EvoQ outperforms the existing methods except ResNet101 on ACIQ. Note that the weights on ACIQ are quantized channel-wisely with non-uniform quantization, which is time-consuming than asymmetric uniform quantization. The performance of ResNet50/101 drops less than 1% comparing to the pre-trained full precision model. Of all the structures, the Top-1 accuracy of MobileNetV2 improves from 10.12% to 68.90%, demonstrating the effectiveness of EvoQ.

TABLE I
THE EFFECT OF SAMPLE NUMBERS ON MOBILENETV2 (MIXED PRECISION 4-BIT QUANTIZATION).

| Samples | 50 | 100 | 200 | 400 |
|---|---|---|---|---|
| Top-1/Top-5 | 68.90/88.85 | 69.02/88.88 | 69.04/88.91 | 68.94/88.84 |

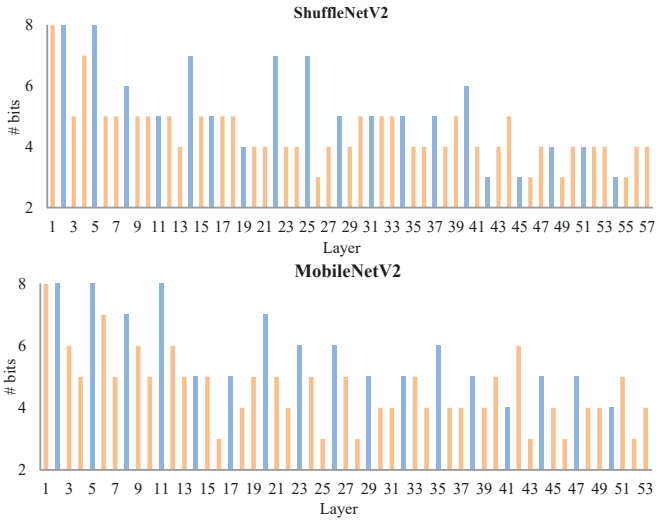


Fig. 4. Quantization policy of ShuffleNetV2 and MobileNetV2. The blue block represents depthwise convolution layers, and the adjacent layers (except the first layer) represent pointwise convolution layers.

### B. Experiments on object detection

To verify the effectiveness of our proposed method further, we implement experiments on object detection, which is a much complicated task than image classification. We employ EvoQ on PASCAL VOC, a generic object detection dataset composed of 21 categories. We adopt the classical object detector SSD [30], which is trained with the combined training set VOC 2007 and VOC 2012, and tested on the VOC

TABLE II
QUANTIZATION RESULTS OF RESNET-18, RESNET-50, AND RESNET-101 ON IMAGENET.

| | Weights | ResNet-18 | | | ResNet-50 | | | ResNet-101 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Top-1 | Top-5 | Size | Top-1 | Top-5 | Size | Top-1 | Top-5 | Size |
| UniQ | 4 bits | 56.18 | 79.70 | 5.85MB | 63.70 | 85.54 | 12.81MB | 65.17 | 86.54 | 22.33MB |
| OMSE [15] | 4 bits | 67.42 | 87.72 | 5.90MB | 72.60 | 90.85 | 12.92MB | 73.60 | 91.53 | 22.52MB |
| ACIQ [13] | MP | 68.30 | - | 6.10MB | 75.30 | - | 13.12MB | **76.90** | - | 23.34MB |
| EvoQ | MP | **68.55** | **88.55** | 5.85MB | **75.51** | **92.60** | 12.81MB | 76.76 | 93.25 | 22.33MB |
| Original | 32 bits | 69.76 | 89.08 | 46.8MB | 76.15 | 92.87 | 102.5MB | 77.37 | 93.56 | 178.7MB |

* "UniQ" denotes uniform precision quantization, and "MP" denotes mixed precision quantization.

TABLE III
QUANTIZATION RESULTS OF SQUEEZENET, SHUFFLENETV2, AND MOBILENETV2 ON IMAGENET.

| | Weights | SqueezeNet | | | ShuffleNet-V2 | | | MobileNet-V2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Top-1 | Top-5 | Size | Top-1 | Top-5 | Size | Top-1 | Top-5 | Size |
| UniQ | 4 bits | 40.50 | 65.92 | 0.63MB | 40.18 | 62.64 | 1.15MB | 10.12 | 22.73 | 1.78MB |
| OMSE [15] | 4 bits | 55.36 | 78.48 | 0.74MB | - | - | - | - | - | - |
| EvoQ | MP | **55.66** | **78.91** | 0.63MB | **66.39** | **86.45** | 1.15MB | **68.90** | **88.85** | 1.78MB |
| Original | 32 bits | 58.19 | 80.62 | 5.0MB | 69.36 | 88.32 | 9.2MB | 71.88 | 90.29 | 14.2MB |

2007 test set. The backbones of SSD are based on VGG16[1] and ResNet50[2], and the performance is measured by mean Average Precision (mAP). As on ImageNet, we randomly select 50 samples from the training set for sensitivity analysis, quantization policy evaluation, activation range estimation, and feature calibration. The hyperparameters are set as on ImageNet. Different from image classification, we extract the backbone of SSD and only optimize the bit-width allocation of the backbone. As the existing post-training quantization methods did not perform experiments on object detection, we only compare EvoQ with uniform precision quantization. As is presented in Table IV, EvoQ achieves 76.75 mAP and 78.42 mAP on SSD-VGG and SSD-ResNet50, respectively, which only causes 0.74% and 1.14% mAP degradation comparing to the original model.

TABLE IV
QUANTIZATION RESULTS OF SSD-VGG AND SSD-RESNET50 ON PASCAL VOC.

| | Weights | SSD-VGG | | SSD-ResNet50 | |
|---|---|---|---|---|---|
| | | mAP | Size | mAP | Size |
| UniQ | 4 bits | 70.62 | 13.15MB | 63.15 | 7.31MB |
| EvoQ(no FC) | MP | 74.80 | 13.15MB | 75.74 | 7.31MB |
| EvoQ | MP | **76.75** | 13.15MB | **78.42** | 7.31MB |
| Original | 32 bits | 77.49 | 105.2MB | 79.56 | 58.5MB |

*FC denotes feature calibration.

## C. Ablation Study

In this section, we present an ablation study for (a) sensitivity guided mutation and (b) mixed precision quantization on ShuffleNetV2 and MobileNetV2 to verify the effectiveness of EvoQ.

[1]https://github.com/amdegroot/ssd.pytorch

[2]https://github.com/ShuangXieIrene/ssds.pytorch

*a) Sensitivity guided mutation:* To optimize the search efficiency, we utilize quantization sensitivity per-layer to improve the mutation operation. We compare sensitivity guided mutation with random mutation on ShuffleNetV2 and MobileNetV2. As is shown in Fig. 5, with the guidance of quantization sensitivity, the quantization error decreases more rapidly than random search in the early iterations. At most times, sensitivity guided mutation can obtain better quantization policies than random search. As can be seen in Table V, sensitivity guided mutation outperforms random mutation by 1.30% and 1.74% on ShuffleNetV2 and MobileNetV2, respectively.

TABLE V
ABLATION STUDY FOR SENSITIVITY GUIDED MUTATION ON SHUFFLENETV2 AND MOBILENETV2.

| | Weights | ShuffleNet-V2 | | MobileNet-V2 | |
|---|---|---|---|---|---|
| | | Top-1 | Top-5 | Top-1 | Top-5 |
| EvoQ(no FC)$_{random}$ | MP | 60.44 | 81.85 | 64.82 | 86.34 |
| EvoQ(no FC)$_{guide}$ | MP | **61.74** | **82.82** | **66.56** | **87.36** |
| Original | 32 bits | 69.36 | 88.32 | 71.88 | 90.29 |

*b) Mixed precision quantization:* Mixed precision quantization can optimize the bit-width allocation without additional consumption. We implement experiments to study the benefit of mixed precision quantization on ShuffleNetV2 and MobileNetV2. We first compare the performance of uniform quantization and mixed precision quantization without feature calibration, and then we apply feature calibration on uniform precision quantization and mixed precision quantization. The experimental results are shown in Table VI. The performance of mixed precision quantization improves apparently than uniform precision quantization. Feature calibration can improve the model performance significantly, while mixed precision quantization also outperforms uniform precision quantization by 0.51% and 6.73% on ShuffleNetV2 and MobileNetV2, respectively.

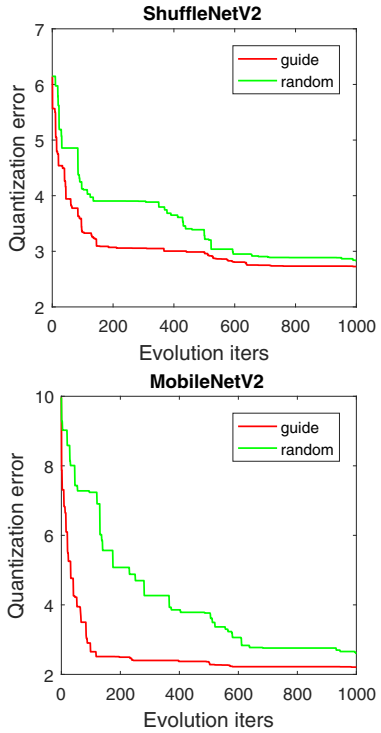


Fig. 5. The search process of random mutation and sensitivity guided mutation on ShuffleNetV2 and MobileNetV2.

TABLE VI
ABLATION STUDY FOR MIXED PRECISION QUANTIZATION ON SHUFFLENET-V2 AND MOBILENET-V2.

| | Weights | ShuffleNetV2 | | MobileNetV2 | |
|---|---|---|---|---|---|
| | | Top-1 | Top-5 | Top-1 | Top-5 |
| UniQ | 4 bits | 40.18 | 62.64 | 10.12 | 22.73 |
| EvoQ(no FC) | MP | **61.74** | **82.82** | **66.56** | **87.36** |
| UniQ+FC | 4 bits | 65.88 | 85.95 | 62.17 | 84.34 |
| EvoQ | MP | **66.39** | **86.45** | **68.90** | **88.85** |
| Original | 32 bits | 69.36 | 88.32 | 71.88 | 90.29 |

## V. CONCLUSIONS

In this paper, we introduce a novel post-training quantization method named EvoQ, which uses 50 samples to perform an evolutionary search for mixed precision quantization. Firstly, we propose an efficient method to evaluate the fitness of the quantization policy, which can save 2000× times on ImageNet. Secondly, we optimize the mutation operation with sensitivity guidance to accelerate the search process. Thirdly, we utilize the selected 50 samples to calibrate the outputs and intermediate features of the quantization model for further improvement. Experiments on image classification and object detection demonstrate the effectiveness of the proposed method. In future research, we will extend our method on activations to achieve more comprehensive resource allocation.

## ACKNOWLEDGMENT

This work was supported by the National Science Foundation of China under Grant NSFC.61906194 and the National Key R&D Program of China under Grant 25904.

## REFERENCES

[1] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges," *IEEE Signal Processing Magazine*, Vol.35, No.1, pp.126-136, 2018.

[2] J. Cheng, P. Wang, G. Li, Q. H. Hu, and H. Q. Lu, "Recent Advances in Efficient Computation of Deep Convolutional Neural Networks," *Frontiers of Information Technology & Electronic Engineering*, Vol.19, No.1, pp.19-64, 2018.

[3] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International Conference on Machine Learning*, pp.1737-1746, 2015.

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

[5] S. Han, H. Mao and W.J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[6] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, pp.1269-1277, 2014.

[7] G. Hinton, O. Vinyals and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, 2015.

[8] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 MB model size," arXiv preprint arXiv:1602.07360, 2016.

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.4510-4520, 2018.

[10] N. Ma, X. Zhang, H. T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," In *Proceedings of the European Conference on Computer Vision*, pp.116-131, 2018.

[11] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," arXiv preprint arXiv:1806.09055, 2018.

[12] B, Zoph, Q. V. Le, "Neural architecture search with reinforcement learning," arXiv preprint arXiv:1611.01578, 2016.

[13] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," In *Proceedings of the aaai conference on artificial intelligence*, Vol.33, pp.4780-4789, 2019.

[14] M. Courbariaux, Y. Bengio and J.P. David, "BinaryConnect: training deep neural networks with binary weights during propagations," in *International Conference on Neural Information Processing Systems*, pp.3123-3131, 2015.

[15] I. Hubara, D. Soudry and R.E Yaniv, "Binarized Neural Networks," in *Advances in Neural Information Processing Systems*, pp.4107-4115, 2016.

[16] M. Rastegari, V. Ordonez and J. Redmon, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *Proceedings of the European Conference on Computer Vision*, pp.525-542, 2016.

[17] S. Migacz, "8-bit inference with TensorRT," In *GPU Technology Conference*, p.7, 2017.

[18] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," In *International Conference on Machine Learning*, pp.2849-2858, 2016.

[19] Y. Zhou, S. M. Moosavi-Dezfooli, N. M. Cheung, and P. Frossard, "Adaptive quantization for deep neural network," In*Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[20] N. M. Ho, R. Vaddi, and W. F. Wong, "Multi-objective Precision Optimization of Deep Neural Networks for Edge Devices," In *2019 Design, Automation & Test in Europe Conference & Exhibition*, pp.1100-1105, 2019.

[21] W. Zhe, J. Lin, V. Chandrasekhar, and B. Girod, "Optimizing the Bit Allocation for Compression of Weights and Activations of Deep Neural Networks," In *2019 IEEE International Conference on Image Processing*, pp.3826-3830, 2019.

[22] Y. Chen, G. Meng, Q. Zhang, X. Zhang, L. Song, S. Xiang, and C. Pan, "Joint neural architecture search and quantization," arXiv preprint arXiv:1811.09426, 2018.

[23] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-Aware Automated Quantization with Mixed Precision," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.8612-8620, 2019.

[24] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer, "Mixed precision quantization of convnets via differentiable neural architecture search," arXiv preprint arXiv:1812.00090, 2018.

[25] X. He and J. Cheng, "Learning Compression from Limited Unlabeled Data," In *Proceedings of the European Conference on Computer Vision*, pp.752-769, 2018.

[26] Y. Choukroun, E. Kravchik and P. Kisilev, "Low-bit Quantization of Neural Networks for Efficient Inference," In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[27] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolution networks for rapid-deployment," In *Advances in Neural Information Processing Systems*, pp.7948-7956, 2019.

[28] R. Zhao, Y. Hu, J. Dotzel, J., C. De Sa, and Z. Zhang, "Improving Neural Network Quantization without Retraining using Outlier Channel Splitting," In *International Conference on Machine Learning*, pp.7543-7552, 2019.

[29] Y. He, J. Lin, Z. Liu, H. Wang, L. J. Li, and S. Han, "AMC: Automl for model compression and acceleration on mobile devices," In *Proceedings of the European Conference on Computer Vision*, pp.784-800, 2018.

[30] W. Liu, D.Anguelov, D. Erhan, D., C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," In *Proceedings of the European Conference on Computer Vision*, pp.21-37, 2016.

[31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, *et al.*, "A. Automatic differentiation in PyTorch,' In *Proceedings of the Conference on Neural Information Processing Systems Workshops*, 2019.