©SHUTTERSTOCK.COM/DMITRIY PRAZYEL

Evolutionary Multi-Objective Model Compression for Deep Neural Networks

Zhehui Wang and Tao Luo
*Agency for Science, Technology and Research (A*STAR), SINGAPORE*

Miqing Li,
University of Birmingham, UK

Joey Tianyi Zhou, Rick Siow Mong Goh, and Liangli Zhen
*Agency for Science, Technology and Research (A*STAR), SINGAPORE*

Abstract—While deep neural networks (DNNs) deliver state-of-the-art accuracy on various applications from face recognition to language translation, it comes at the cost of high computational and space complexity, hindering their deployment on edge devices. To enable efficient processing of DNNs in inference, a novel approach, called Evolutionary Multi-Objective Model Compression (EMOMC), is proposed to optimize energy efficiency (or model size) and accuracy simultaneously. Specifically, the network pruning and quantization space are explored and exploited by using architecture population evolution. Furthermore, by taking advantage of the orthogonality between pruning and quantization, a two-stage pruning and quantization co-optimization strategy is developed, which considerably reduces time cost of the architecture search. Lastly, different dataflow designs and parameter coding schemes are considered in the optimization process since they have a significant impact on energy consumption and the model size. Owing to the cooperation of the evolution between different architectures in the population, a set of compact DNNs that offer trade-offs on different objectives (e.g., accuracy, energy efficiency and model size) can be obtained in a single run. Unlike most existing approaches designed to reduce the size of weight parameters with no significant loss of accuracy, the proposed method aims to achieve a trade-off between desirable objectives, for meeting different requirements of various edge devices. Experimental results demonstrate that the proposed approach can obtain a diverse population of compact

Digital Object Identifier 10.1109/MCI.2021.3084393
Date of current version: 15 July 2021

Corresponding author: Liangli Zhen (e-mail: zhenll@ihpc.a-star.edu.sg).

1556-603X/21©2021IEEE

DNNs that are suitable for a broad range of different memory usage and energy consumption requirements. Under negligible accuracy loss, EMOMC improves the energy efficiency and model compression rate of VGG-16 on CIFAR-10 by a factor of more than 8.9 × and 2.4 ×, respectively.

I. Introduction

Deep neural networks (DNNs) are artificial neural networks with more than three layers (*i.e.*, more than one hidden layer), which progressively extract higher-level features from the raw input in the learning process. They have delivered the state-of-the-art accuracy on various real-world problems, such as image classification, face recognition, and language translation [1]. The superior accuracy of DNNs, however, comes at the cost of high computational and space complexity. For example, the VGG-16 model [2] has about 138 million parameters, which requires over 500 MB memory for storage and 15.5G multiply-and-accumulates (MACs) to process an input image with 224 × 224 pixels. In myriad application scenarios, it is desirable to make the inference on edge devices rather than on cloud, for reducing the latency and dependency on connectivity and improving privacy and security. Many of the edge devices that draw the DNNs inference have stringent limitations on energy consumption, memory capacity, etc. The large-scale DNNs [3], [4] are usually difficult to be deployed on edge devices, thus hindering their wide application.

Efficient processing of DNNs for inference has become increasingly important for the deployment on edge devices. For generating efficient DNNs, many neural architecture search (NAS) approaches have been developed in recent years [5]–[7]. One way of carrying out NAS is to search from scratch [8], [9]. In contrast, model compression[1] [10] searches for the optimal networks starting from a well-trained network. For instance, to reduce the storage requirement of DNNs, Han *et al.* proposed a three-stage pipeline (*i.e.*, pruning, trained quantization, and Huffman coding) to compress redundant weights [10]. Wang *et al.* suggested removing redundant convolution filters to reduce the model size [11]. Rather than reducing the model size, a few attempts [12], [13] are conducted to compress DNNs directly by taking the energy consumption as the feedback signals. They have achieved promising results in reducing the size of weight parameters (or energy consumption). However, these approaches require the model to achieve approximately no loss of accuracy, rendering the solution less flexible.

In practice, different users often have distinct preferences on desirable objectives, *e.g.*, accuracy, model size, energy efficiency, and latency, when they select the optimal DNN model for their applications. In this paper, a novel approach, called Evolutionary Multi-Objective Model Compression (EMOMC), is proposed to optimize energy efficiency/model size and accuracy simultaneously. By considering network pruning and quantization, the model compression is formulated as a multi-objective problem under different dataflow designs and parameter coding schemes. Each candidate architecture can be regarded as an individual in the evolutionary population. Owing to the cooperation and interplay of the evolution between different architectures in the population, a set of compact DNNs that offer trade-offs on different objectives (*e.g.*, accuracy, energy efficiency, and model size) can be obtained in a single run. Unlike most existing approaches which aim to reduce the size of weight parameters or the energy consumption with no significant loss of accuracy, the proposed approach attempts to achieve a good balance between desired objectives, for meeting the requirements of different edge devices. Experimental results demonstrate that the proposed approach can obtain a diverse population of compact DNNs for customized requirements of accuracy, memory capacity, and energy consumption.

The novelty and main contributions of this work can be summarized as follows:

❏ The model compression problem is formulated as a multi-objective problem. The optimal solutions are searched in the network pruning and quantization space using a population-based algorithm.

❏ To speed up the population evolution, a two-stage pruning/quantization co-optimization strategy is developed based on the orthogonality between pruning and quantization.

❏ The trade-offs between accuracy, energy efficiency, and model size in model compression are explored by considering different dataflow designs and parameter coding schemes. The experimental results demonstrate that the proposed method can obtain a set of diverse Pareto optimal solutions in a single run. Also, it achieves a considerably higher energy efficiency than current state-of-the-art methods.

II. Preliminaries

Network pruning and quantization are two commonly used model compression techniques to improve the energy efficiency in model inference and/or to shrink the size of the model. Moreover, the dataflow design employed by edge devices and the coding scheme applied to store the weight matrix both have a significant impact on the performance of model compression.

A. Network Pruning and Quantization

For making the training easy, the networks are usually over-parameterized [14]. Pruning is a widely-used model compression technique that can effectively reduce the energy consumption of edge devices and shrink the model size [10]. Network pruning removes some of the redundant parameters in the network by setting their values as zeros. A well-trained neural network usually contains a large number of weights whose values are relatively small compared to other parameters. In most cases, these parameters are not particularly important when performing model inference. Hence, one can sort all the parameters in the model and replace those parameters with the least absolute values by zeros, while the accuracy of the model can still be maintained. For instance, the pruning amount to be 33%, then one-third of the parameters in the model will be replaced by zeros. In the inference process, if the processing elements (PEs)[2] whose input

[1] The technique aims to shrink the size of the neural network model without a significant drop of accuracy.

[2] The PE is a basic unit to conduct computation in processors.

Dataflow design is one of the most important features in hardware accelerators, which allows the system to reuse the data among different processing elements.

weight parameters are zeros, the computation process can be skipped in those PEs, thus reducing energy consumption.

Quantization is another critical model compression technique that is used to accelerate DNNs and reduce model size [10]. It involves mapping data to a small set of quantization levels and aims at minimizing the error between the reconstructed data from the quantization levels and the original data. The quantization level reflects the precision and ultimately the number of bits representing a parameter. After the quantization, the low precision parameters may still store enough information for model inference, and the accuracy of the model can be maintained. In practical implementations, if the weights are quantized, one can use multipliers with simpler structures, thus reducing energy consumption. For instance, a high precision parameter with 32-bit float point (32FP) data type requires 23 bit × 23 bit multipliers. Such type of multiplier contains 506 adders in total. If quantizing the activation from 32FP to 16-bit float point (16FP) and quantizing the weights from 32FP to 8-bit integer (8INT), only 10 bit × 8 bit multipliers are required, each of which contains only 72 adders in total. The fewer adders in multipliers, the lower energy consumption for computation.

Pruning and quantization can reduce not only the energy consumption on the computation process but also the energy consumption on the data movement, which is roughly proportional to the total amount of data transmitted from the memory module in terms of bits [15]. For instance, if pruning 80% of the parameters in the model and quantizing all the parameters from 16 bits to 8 bits, then about 90% of the energy consumption on data movement can be reduced.

B. Dataflow Design in Hardware Accelerators

The dataflow design decides how data is reused among different PEs. Since a large portion of the energy consumption of hardware accelerators is on the data movement, the dataflow design needs to be considered when optimizing the energy efficiency. Algorithm 1 shows the computation procedure of a typical convolutional layer. It contains six loops, each of which corresponds to one dimension either in the weight filter or in the feature map. More specifically, C_O and C_I are the numbers of output and input channels, X and Y are the width and height of the feature map, and F_X and F_Y are the width and height of the weight filter. In each iteration of the innermost loop, a basic arithmetic operation called multiply-accumulate (MAC) is performed. In one convolutional layer, there are $C_O{\cdot}C_I{\cdot}X{\cdot}Y{\cdot}F_X{\cdot}F_Y$ MAC operations in total.

In typical hardware accelerators,[3] there are a set of processing elements. Each PE can execute one MAC operation independently. How to map the MAC operation into each PE and how the data flow among those PEs become key considerations in the design of hardware accelerators. Theoretically, there are many mapping methods, resulting in different dataflow designs. For example, suppose that the device has an array of PEs, one can unroll any one of the loops in Algorithm 1 and map each iteration in the unrolled loop into each PE of the array. Similarly, if the device has a matrix of PEs, one can unroll any two loops in Algorithm 1 and map the MAC operations into each PE in the matrix. Thus, with six loops as shown in Algorithm 1, there are $C_6^2 = 15$ possible dataflow designs in total. To simplify the problem, in this work, only four popular dataflow designs are evaluated, as shown in Table I. These dataflow designs are named as $A:B$, where A and B stand for the names of the two unrolled loops.

Figure 1 shows the schematic diagram of those four popular dataflow designs, where only four PEs are involved in each example. Each PE contains one multiplier and one adder, which can execute one MAC operation each time. The PE also contains register files, which can temporarily store input or output data. In X:Y, the MAC operation results are stored in registers at output ports of PEs. At each iteration, the last MAC operation result is read from registers. In C_I:C_O, at each iteration, the input feature map is reused C_O times, and C_I MAC operation results are summed up. In F_X:F_Y, $F_X{\cdot}F_Y$ weights are stored in registers at input ports of PEs. At each iteration, $F_X{\cdot}F_Y$ MAC operation results are summed up. In X:F_X, F_X weights are stored in registers at input ports of PEs. At each iteration, the weights are reused X times, and F_X MAC operation results are summed up.

Algorithm 1 **Computation of a typical convolutional layer.**

```
for $c_o$ in range($C_O$) do
  for $c_i$ in range($C_I$) do
    for $x$ in range($X$) do
      for $y$ in range($Y$) do
        for $f_x$ from -($F_X$-1)/2 to ($F_X$-1)/2 do
          for $f_y$ from -($F_Y$-1)/2 to ($F_Y$-1)/2 do
            $O[c_o][x][y]$
            $+= I[c_i][x+f_x][y+f_y] \times W[c_o][c_i][f_x][f_y]$
```

TABLE I **Popular dataflow designs that are applied in literature.**

DATAFLOW	APPLIED BY	DATAFLOW	APPLIED BY
$X:Y$	[16] [17]	$C_I:C_O$	[18] [19]
$F_X:F_Y$	[20]	$X:F_X$	[21] [22]

[3] The devices that are specialized to execute a certain task, such as the graphics processing unit (GPU), field-programmable gate array(FPGA), and application-specific integrated circuit (ASIC).

C. Coding Scheme for the Parameters

After pruning, the filter in the model becomes a sparse matrix, which means that it contains plenty of zero elements. To store the sparse matrix into the memory, many coding schemes are developed, and the choice of the coding scheme mainly depends on the characteristics of the matrix. In this work, three coding schemes are considered. The first one is the normal coding scheme, where it stores those zero elements in the same way as those non-zero elements. In other words, it keeps the space for all the zero elements in the matrix. Therefore, the storage size of the normal coding scheme is $N \cdot q_i$, where N is the total number of weight elements in the matrix and q_i is the quantization depth of the weight.

In some cases, it is a waste of the memory capacity to store all these zero weights. For saving the memory space, various new coding schemes are proposed to shrink the size of the sparse matrix. One common coding scheme is called the Coordinate (COO) coding. In the COO coding scheme, the non-zero elements are stored, along with the row index and the column index, and zero elements are ignored. Another popular coding scheme is called the compressed sparse row (CSR) coding. In the CSR coding scheme, only the values of the non-zero elements are stored along with the column index and the row offset. To further save the memory space, one version of the CSR coding scheme only stores the relative distance between two non-zero elements, which is shown in Figure 2. For this example, the weight matrix has eight elements, and three of them are non-zero elements. Only the values of these three elements and their relative positions in the array are stored. The second non-zero element is three slots away from the first non-zero element, and the integer 2 is recorded as a relative row index for the second element. It is assumed that each non-zero element requires three bits to store the relative row index. If one non-zero element is far away from the previous element, zero-padding elements are inserted into the array to avoid overflow. Therefore, the storage size of the CSR coding scheme with relative positions is $n \cdot (p_i + 3)$, where n is the number of non-zero elements and q_i is the quantization depth of the weight.

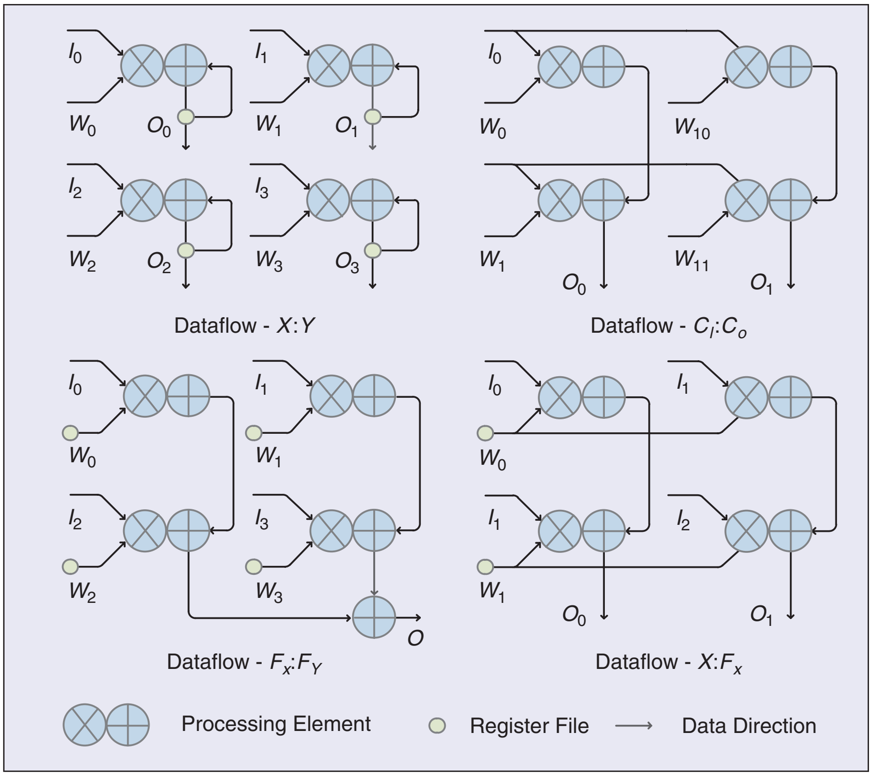

FIGURE 1 Examples of the four popular dataflow designs [15]. For simplicity, only four PEs are shown in each example. W_k/W_{kk}, I_k, and O_k are the elements in the weight, the input feature map, and the output feature map, respectively.

III. Related Work

A. Model Compression

Model compression aims to compress and accelerate DNN models. Different approaches target different objectives, such as model size, number of floating-point operations per second (FLOPs), latency, and energy efficiency. The initial intention of model compression is to alleviate the on-chip storage limit for complicated CNN models [10]. Since then, many approaches have been proposed to shrink the model size of CNNs [23], [24]. There are two major branches in this area. The first branch focuses on the computation cost, and they target the number of FLOPs [25]. For example, Li *et al.* [26] proposed to prune whole filters from CNNs, avoiding sparse connectivity patterns and reducing the computational cost significantly. Lemaire *et al.* proposed a budgeted regularized pruning framework for deep CNNs [25], which makes the compressed model less computation-intensive. The second branch targets the inference speed [27], [28]. For instance, He *et al.* leveraged reinforcement

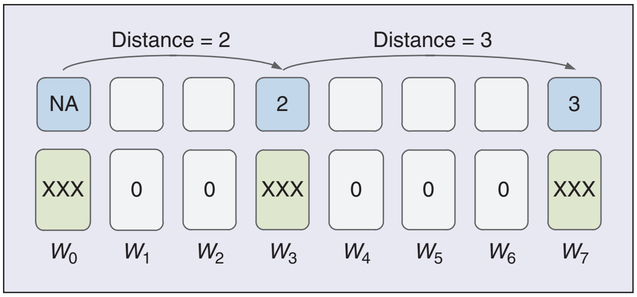

FIGURE 2 The CSR coding scheme with relative positions, which stores the values of non-zero elements and their relative positions in the array.

Evolutionary multi-objective optimization has been widely used to search for the optimal solutions, in the presence of trade-offs between multiple conflicting objectives.

learning to provide the model compression policy, which can accelerate the inference on mobiles considerably.

Recently, edge devices have become increasingly popular for AI applications. However, considering the large amount of energy consumed for the model inference, the deployment of CNN on edge devices becomes challenging. To solve this problem, some scholars proposed model compression approaches to reduce the energy consumption directly, using quantization [13] and/or pruning [12] techniques. In [12], an energy-aware network pruning approach is proposed to reduce the overall energy across all layers by 3.7 × for AlexNet [29] and 1.6 × for GoogLeNet [30].

From the above, it can be seen that model compression is essentially a multi-objective optimization problem, with several objectives to be considered, including accuracy, energy consumption, model size, etc. Previous studies rarely deal with multiple objectives at the same time. A common way adopted in literature is to optimize only one of the objectives while setting the remaining ones to be hard constraints. In this work, the evolutionary multi-objective optimization technique is applied to tackle these objectives simultaneously.

B. Evolutionary Multi-Objective Optimization

In the real-world systems, there exist plenty of problems having two or more (often conflicting) objectives which one needs to consider simultaneously. Such problems are called the multi-objective optimization problems (MOPs). Without loss of generality, a multi-objective optimization problem (MOP) can be formulated as the following minimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))^T \\ \text{s.t.} \quad & g_j(\mathbf{x}) \leq 0, j \in \{1, 2, \ldots, J\}, \\ & h_k(\mathbf{x}) = 0, k \in \{1, 2, \ldots, K\}, \\ & \mathbf{x} \in \Omega, \end{aligned} \tag{1}$$

where J denotes the number of inequality constraints, K is the number of equality constraints, $\Omega \subseteq \mathbb{R}^n$ is the decision space, $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is a candidate solution, and $F : \Omega \to \mathbb{R}^M$ consists of M (conflicting) objective functions.

Let **a** and **b** be two feasible solutions for an MOP defined in Equation (1), one can say that **a** dominates **b** if $\forall u \; f_u(\mathbf{a}) \leq f_u(\mathbf{b})$ and $\exists v \; f_v(\mathbf{a}) < f_v(\mathbf{b})$, where $u, v \in \{1, 2, \ldots, M\}$. A solution is Pareto optimal if it is not dominated by any other solutions. Due to the conflict of the objectives in MOPs, there are a set of Pareto optimal solutions, which represent the best possible trade-offs among different objectives. The optimal solution set in the decision space is called the Pareto set (PS), and its mapping in the objective space is called the Pareto front (PF).

In the literature, many approaches have been developed to solve MOPs since the 1950s [31]. Among them, evolutionary algorithms (EAs) stand out thanks to the nature of population-based search that aims to approximate the whole Pareto front in a single execution. Also, EAs are typically exempt from the characteristics of the PF than conventional mathematical programming techniques [31]. They can handle the MOPs with discontinuous and non-convex PFs well.

Since the seminal work, called Vector Evaluated Genetic Algorithm (VEGA) [32], was proposed by Schaffer in 1985, a large number of multi-objective evolutionary algorithms (MOEAs) have been developed and adopted in various applications. In MOEAs, the selection strategy of individuals in the population plays a key role in the evolutionary process. Since the optimal solutions are those non-dominated to each other in the whole search space, Pareto dominance naturally becomes a viable criterion for selecting promising solutions during the evolutionary process. The Pareto dominance criterion, however, may fail to provide sufficient selection pressure, making the algorithm hard to converge. This situation can be usually encountered when the objective space is enormous, *e.g.*, in many-objective optimization problems [33]–[35]. To push the population towards the PF, Goldberg proposed a mechanism called Pareto ranking [36] for the selection in MOEAs. A niche method is then used in the Nondominated Sorting Genetic Algorithm (NSGA) [37] to maintain stable sub-populations. Later on, in its new version Nondominated Sorting Genetic Algorithm-II (NSGA-II) [38], a crowding degree comparison operator is adopted to make the ranking scheme more effective and efficient. NSGA-II is widely used to solve MOPs, despite its limitations in handling the MOPs with more than three objectives [39]. Recently, many MOEAs tend to consider other selection strategies since they may converge fast towards the PF, such as indicator-based MOEAs, decomposition-based MOEAs, and bi-goal criterion MOEAs [33].

Recently, there have been a few attempts to exploit MOEAs to search for efficient neural architectures. For instance, Lu *et al.* proposed a method, called NSGA-Net [40], which formulates the neural architecture search as a multi-objective problem and uses the NSGA-II algorithm to solve it. NSGA-Net considers two objectives: the classification error and the computation cost (measured by the number of MACs). It has achieved promising results compared with other neural architecture search methods, *e.g.*, DARTS [5] and ENAS [41], on the CIFAR-10 dataset [42].

This work studies how the evolutionary multi-objective (EMO) method can be used in model compression, given its multi-objective nature.

IV. Our Proposed Method

In real-world applications, users usually have different preferences on the prediction model's objectives, including accuracy, energy efficiency, model size, etc. In this section, the evolutionary multi-objective model compression method is presented. The model compression problem is formulated as a multi-objective problem (MOP), which has several objective functions and constraints [43]. Then, an evolutionary algorithm is adopted to solve the MOP. The goal of the optimization is to find a set of Pareto optimal solutions that represents various trade-offs on the desired objectives, thus enabling the deployment of the AI models on edge devices with different resource constraints.

A. Problem Formulation

This work aims to compress a well-trained model to achieve high accuracy, low energy consumption, and low model size. By providing different pruning amount, p, and quantization depth, $\mathbf{q}$, the compressed model should result in different accuracy, energy consumption, and model size. The goal of the optimization is to reduce the energy consumption or model size while at the same time making the accuracy of the model as high as possible. The relationship between the accuracy, the pruning amount, p, and the quantization depth $\mathbf{q} = [q_1, q_2, \dots, q_L]$ is denoted as

$$\text{Accuracy} = f_1(p, \mathbf{q}), \tag{2}$$

where $f_1(\cdot)$ represents the accuracy score of the model obtained by pruning p of the weight parameters in each layer of the original model, then quantizing the parameters in the i-th layer with the depth of q_i bits, and L denotes the number of layers in the original model.

The energy consumption of the inference is constrained by the battery's capacitance of edge devices. Exceeding the energy budget of the edge device will greatly limit the implementation of AI applications. From the perspective of users, it is usually acceptable to trade a bit of loss of accuracy for a large amount of reduction on energy consumption, especially for edge devices. For a trained model, the energy consumption in inference is also related to the exact dataflow design d applied on the edge devices. The relationship among the pruning amount p, the quantization depth $\mathbf{q}$, and the dataflow design d is denoted as follows:

$$\text{Energy} = f_2(p, \mathbf{q}, d). \tag{3}$$

The model size is constrained by the capacities of on-chip memory modules in edge devices. If the model size exceeds the limitation, the model inference procedure requires to load and save weights/features maps through the off-chip memory. Given the fact that off-chip memory access consumes much larger energy consumption than the on-chip memory access [10], the energy consumption of the inference process increases tremendously. Furthermore, the app stores are sensitive to the size of the binary files, *e.g.*, App Store has the restriction "apps above 100 MB will not download until you connect to Wi-Fi" [10]. Hence, it is important to shrink the size of the model and to make sure that the entire model can be fit into the memory constraint of the edge devices. For a given model, the model size highly depends on the coding scheme c applied to store the weights. The relationship between the model size, the pruning amount p, the quantization depth $\mathbf{q}$ and the coding scheme c is defined as

$$\text{Model Size} = f_3(p, \mathbf{q}, c). \tag{4}$$

There are $L + 3$ variables, and L denotes the number of layers in the original model. The value of the variable p is a real number that indicates the pruning amount in all the layers of the model. The value of the variable $\mathbf{q}_i$ is an integer that reflects the quantization depth in the i-th layer of the model. The constraints on these variables are as follows:

$$\begin{aligned} &p_l \leq p \leq p_u, \\ &q_l \leq q_i \leq q_u, \\ &d \in \{d_1, d_2, d_3, d_4\}, \\ &c \in \{c_1, c_2, c_3\}, \end{aligned} \tag{5}$$

where p_l and p_u are the upper and lower bounds of the pruning amount, q_l and q_u are the upper and lower bounds of the quantization depth, d_1, d_2, d_3, and d_4 correspond to the four dataflow designs of $X{:}\,Y, C_I{:}\,C_O, F_X{:}\,F_Y$ and $X{:}\,F_X$, and c_1, c_2 and c_3 indicate three parameter coding schemes of the normal coding, COO and CSR, respectively. In this work, the pruning amount is assumed to be from 0% to 100%, and the quantization depth of each layer ranges from 1 bit to 23 bits.

Two bi-objective optimization problems are studied. In the first problem, it explores possible combinations of pruning amount and quantization depth, and aims to maximize the model accuracy f_1 and minimize the energy consumption f_2, assuming the dataflow design to be d. Mathematically, the bi-objective problem can be formulated as following:

$$\begin{cases} \max\ f_1(p, \mathbf{q}), \\ \min\ f_2(p, \mathbf{q}, d), \end{cases} \text{s.t.} \begin{cases} p_l \leq p \leq p_u, \\ q_l \leq q_i \leq q_u, \\ d \in \{d_1, d_2, d_3, d_4\}. \end{cases} \tag{6}$$

The second bi-objective problem considers to maximize the accuracy f_1 and minimize the model size f_3 simultaneously, assuming the coding scheme to be c, namely, the following problem:

$$\begin{cases} \max\ f_1(p, \mathbf{q}), \\ \min\ f_3(p, \mathbf{q}, c), \end{cases} \text{s.t.} \begin{cases} p_l \leq p \leq p_u, \\ q_l \leq q_i \leq q_u, \\ c \in \{c_1, c_2, c_3\}. \end{cases} \tag{7}$$

Note that this work formulates two bi-objective optimization problems rather than a three-objective optimization problem. There are two reasons. Firstly, if one optimizes the energy consumption and the model size simultaneously (*i.e.*, different dataflow designs and different coding schemes will be considered at the same time), the decision space will be increased considerably, making the optimization much harder and consuming more computation resource. Secondly, as the evaluation of each individual has a high computational cost, the population size cannot be a large number. Typically, the population size is set to be smaller than 100. A three-objective space will lead to the solution set to be much more sparse than a bi-objective space.

B. Multi-Objective Optimization and Speedup

Instead of pruning the model directly in one step, a more effective approach employed is to prune the model in multiple steps. If pruning the model in one step, the accuracy will decrease apparently, and it will be too difficult to restore the model [44]. Figure 3 demonstrates the comparison between the multi-step pruning method and the single-step pruning method. The model compression of a well-trained VGG-16 model is tested on the CIFAR-10 dataset [15], [42]. For the multi-step pruning, it gradually increases the pruning amount from 0 to 95% in 32 steps. In each step, the model is pruned partially and re-trained by one epoch. In the single-step pruning, the model is pruned by 95% immediately, and then re-trained by 32 epochs. As shown in Figure 3, it can be seen that the multi-step pruning method outperforms the single-step pruning method in terms of accuracy with a large margin.

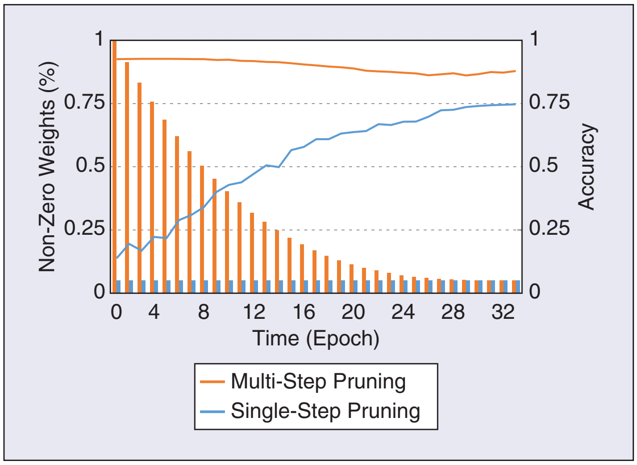

FIGURE 3 The comparison between multi-step pruning and single-step pruning, tested on CIFAR-10 using VGG-16 (figure adopted from [15]).

A challenge in the multi-step pruning process is that it usually has high computational complexity. Specifically, each step requires fine-tuning the model by one or several epochs. If one attempts to find the optimal pruning amount and quantization depth for a model, the multi-step pruning process will considerably delay the optimization progress. To obtain the accuracy of the compressed model at a given pruning amount and quantization depth, the model needs to be compressed first, which usually includes many training epochs. Due to the large search space, it is almost impossible to pre-store all the compressed models under any combinations of pruning amount and quantization depth. For example, the parameters in each layer of the model can be quantized from 23 bits to 1 bit. The pruning amount in each layer can range from 0 to 100%. In general, an L-layer model can have 100×23^L possible combinations of pruning amount and quantization depth, assuming 1% pruning amount granularity.

The EMO technique is adopted to solve this problem. However, since an evolutionary algorithm is essentially a stochastic search, it may need thousands of trials (candidate solutions) to find a high-quality solution. Once a new solution (architecture) is produced, it takes a substantial amount of time to perform the training for the evaluation. Consequently, it may make the EMO-based search impossible.

To address this issue, by taking advantage of the orthogonality between pruning and quantization [45], a two-stage pruning and quantization co-optimization method is proposed, which can effectively reduce the computational cost. Specifically, the optimization process is divided into two stages. In the first stage, it prunes the model by multiple independent loops. In each loop, it starts from a well-trained model, prunes the model with a different pruning amount, fine-tunes the model, and saves the pruned model into a library. The set of pruning amounts cover all the possible pruning amounts which can be referenced by the multi-objective solver. This is to guarantee that no pruning process is required in the second stage. In the second stage, the multi-objective solver starts to explore the design space and tries to find the optimal combinations of pruning amount and quantization depth. During this process, the solver needs to know the accuracy, energy consumption, and model size under a given combination of pruning amount and quantization depth. At this step, one just needs to load the corresponding pruned model from the library and quantize it.

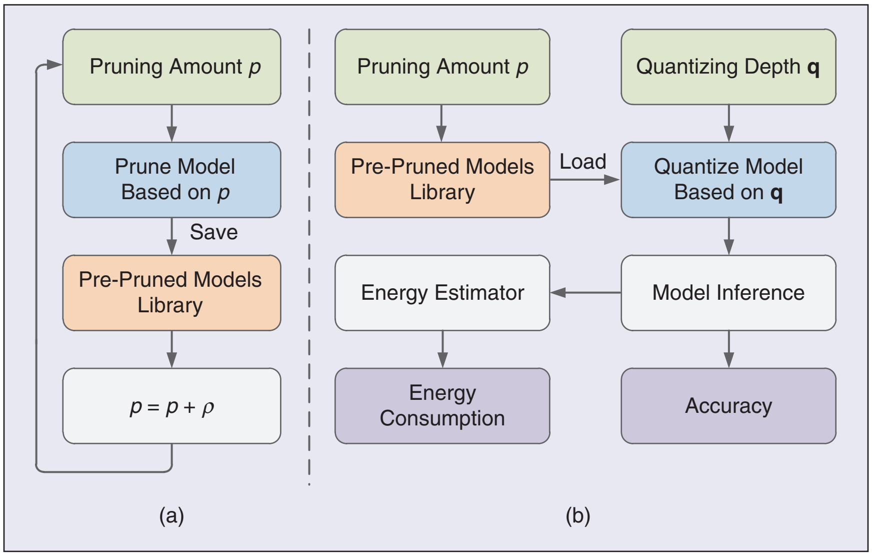

FIGURE 4 The process of the proposed two-stage pruning and quantization co-optimization method. (a) Stage-I: prune the model and save the pre-pruned models to the library; (b) Stage-II: load the pre-pruned model from the library, quantize the parameter, and calculate the accuracy as well as the energy consumption/the model size.

Figure 4 shows an overview of the proposed approach. Instead of pruning and quantizing the models at the same time, these two actions are taken into two different stages. In the first stage, it only prunes the model. Specifically, assuming ρ as granularity, it prunes the model by $100/\rho$ times. At the i-th time step, it starts from the well-trained model and prunes the model gradually using the multi-step pruning method, until the target pruning amount reaches $i \cdot \rho$. After that, it saves the compressed model into a pre-pruned models library. In the second stage, it loads one of the pre-pruned models from the library based on the required pruning amount p, and then quantizes the parameters on the pre-pruned model based on the required quantization depth $\mathbf{q}$. Since pruning and quantization are two orthogonal operations, the final compressed model will be equivalent to the compressed model that is pruned and quantized at the same time. Lastly, it obtains the accuracy by performing the model inference and read the energy consumption from an energy estimator.

Pruning and quantization are two popular techniques in DNN model compression, which show substantial improvement in energy efficiency.

The proposed approach can efficiently speed up the optimization process. To obtain the accuracy and energy consumption under a given pruning amount and quantization depth, it does not need to fine-tune the model anymore. Before the optimization process, it completes the procedures in stage-I and saves only 100 compressed models into the library, assuming 1% granularity. The number of saved models is much less than 100×23^L, *i.e.*, the number of possible compressed models in the whole exploration space. For each combination of pruning amount and quantization depth, the time cost of evaluating the individual is roughly equal to the inference time cost of the model.

V. Experimental Results and Analysis

The proposed method is evaluated on three baseline CNN models: MobileNet [46], VGG-16 [2] and LeNet-5 [47], which have different characteristics. MobileNet is a neural network specially designed for mobile and embedded vision applications. VGG is a typical deep neural network, which was in the first place on the image localization and the second place on the image classification task in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2014. LeNet-5 is a simple network for handwritten and machine-printed character recognition. It consists of only two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, two fully-connected layers, and a Softmax classifier. MobileNet and VGG-16 are tested for color image classification on the CIFAR-10 dataset [42], and LeNet-5 is applied to recognize handwriting digits in the MNIST dataset [47].

A. Experimental Setting

The NSGA-II algorithm in the python-based tool Pymoo [43] is used to solve the formulated multi-objective problem. The neural network is implemented in PyTorch[4]. During the network training, the initial learning rate is set to be 0.01, and it decays by half every 30 epochs. The batch size is set to be 256. During the multi-objective optimization process, the population size is set to be 40, and it runs 250 generations in each execution. The multi-objective optimization and network training are performed on an NVIDIA Titan Xp graphics processing unit (GPU) card. Four dataflow designs are considered as they are the most commonly used dataflow designs $X:Y, C_I:C_O, F_X:F_Y$, and $X:F_X$. The resource requirement is calculated based on the Xilinx Virtex UltraScale FPGA and the energy consumption from the Xilinx XPE toolkit [48]. In the

[4] PyTorch Open Source Toolkit at https://github.com/pytorch/pytorch.

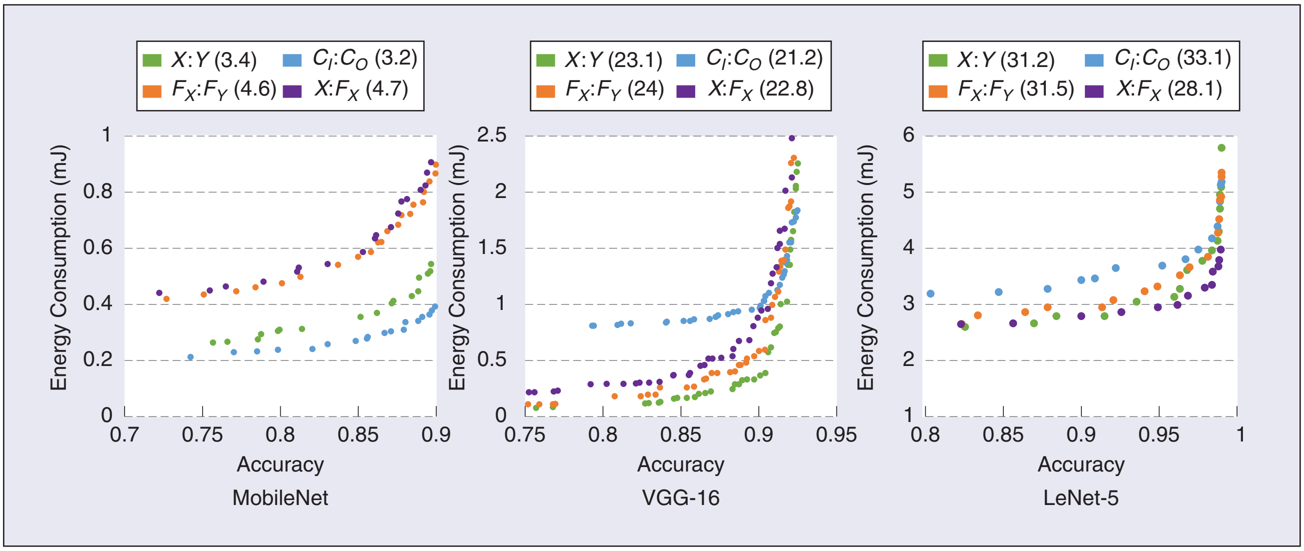

FIGURE 5 The solution sets obtained from the bi-objective optimization of accuracy and energy consumption on CIFAR-10 (MobileNet and VGG-16) and MNIST (LeNet-5). The four different dataflow designs are marked with different colors. In the legends, the quoted number after the dataflow design indicates its energy consumption (mJ) on the original model before the model compression.

implementation, the multipliers and adders are implemented on LUTs (lookup tables). An $M \times N$ multiplier requires $M/2 \times (N+1)$ LUTs [49]. To save the memory space, there is no need to keep the feature map in local memory after the computation of each layer. Hence, the size of the local memory modules must support the weights in all layers and the temporary feature maps. The pruning and quantization approaches are described in [10]. For pruning, the ℓ_1-norm based unstructured pruning method is adopted and a mask is added to filter out the pruned weight. For quantization, the linear (uniform) quantization method is adopted and a scaling factor is used to lower the precision of the weights.

B. Bi-Objective Optimization of Accuracy and Energy Consumption

Due to the proposed two-stage pruning and quantization co-optimization method, one can complete the model compression and obtain the solution set efficiently. The entire optimization process includes two stages. The first stage is for the pre-processing, which takes around 24 hours. The second stage is for the multi-objective optimization. The solver can generate optimal solutions within one hour by using a single NVIDIA Titan Xp graphics processing unit (GPU) card. Figure 5 shows the solution sets obtained from the bi-objective optimization of accuracy and energy consumption, under the four different dataflow designs. Each point in the figure corresponds to one compressed model in the solution set obtained by the bi-objective optimization. From the results, one can see that:

❑ The points marked in different colors cover a large range of accuracy scores and energy consumption, which means that EMOMC obtains a solution set with a high diversity for the model compression of the three baseline CNN models, under the four dataflow designs. For example, under the dataflow design of $X:Y$, the accuracy scores of MobileNet range from around 75% to 90%, and the energy consumption from around 0.2 mJ to 0.58 mJ. It offers the right trade-offs between the two objectives for meeting the constraints of various edge devices.

❑ From the perspective of energy consumption, if searching solutions from the one with the highest energy consumption to the one with the lowest energy consumption, the loss on accuracy is negligible at the first few points. For instance, under the dataflow design of $X:Y$, the energy consumption of VGG-16 decreases from around 2.3 mJ to 0.5 mJ with an accuracy drop less than 2%. However, after a certain threshold, the accuracy loss becomes extremely large. By considering the model's accuracy, if searching for solutions from the one with the highest accuracy to the one with the lowest accuracy, the reduction of energy consumption is remarkable at the first few points. However, after a certain threshold, the energy consumption becomes relatively stable.

❑ Different models prefer different dataflow designs. Specifically, $C_I:C_O$ achieves the highest energy efficiency among the four dataflow designs for MobileNet. However, it is inferior to other dataflow designs for VGG-16. The reason is that the convolution layers of different models have different shapes. In addition to energy consumption, the latency and cost of edge devices also depend on dataflow designs. The selection of dataflow designs involves many factors, which makes it very difficult in practice. This work explores the optimization results on the four popular dataflow designs.

C. Bi-Objective Optimization of Accuracy and Model Size

Figure 6 demonstrates the solution sets obtained from the bi-objective optimization of accuracy and model size, under three

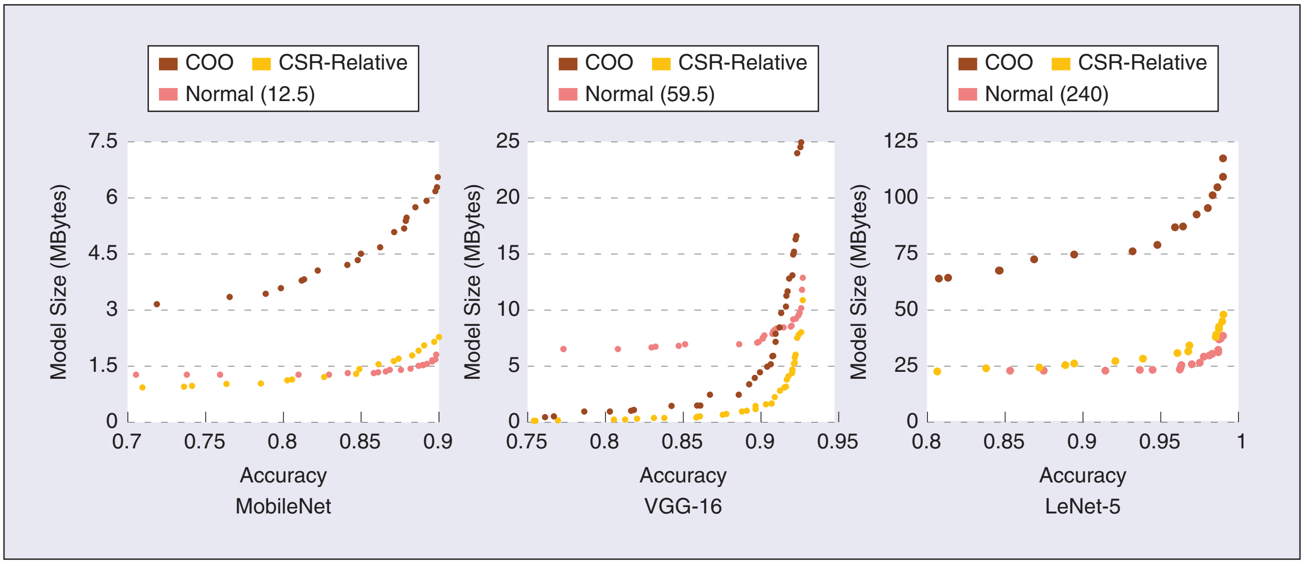

FIGURE 6 The solution sets obtained from the bi-objective optimization of accuracy and model size on CIFAR-10 (MobileNet and VGG-16) and MNIST (LeNet-5). The three different coding schemes are marked with different colors. In the legends, the quoted number after normal coding scheme indicates the size of the original model before the model compression.

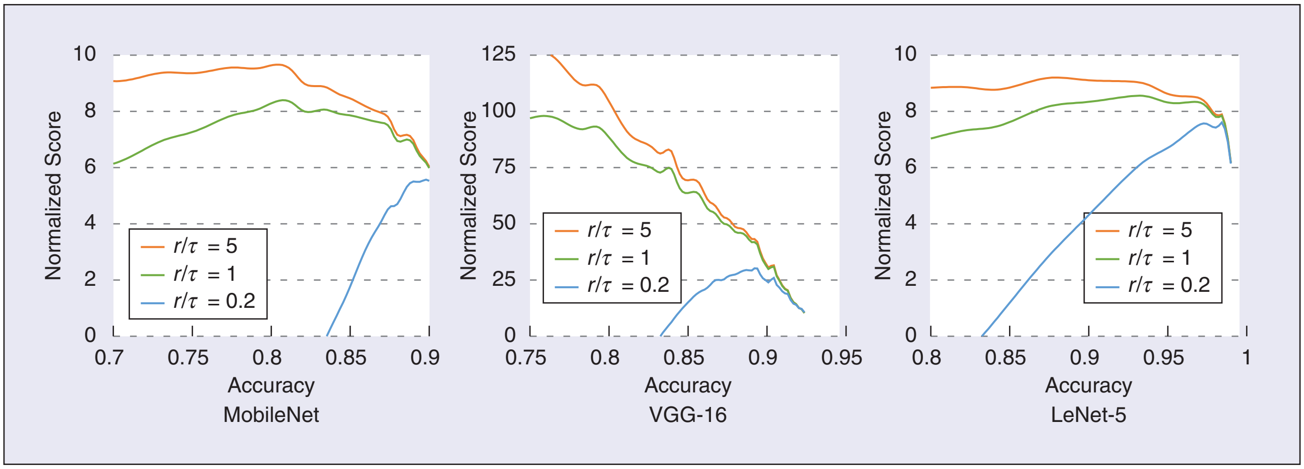

FIGURE 7 The aggregation scores on CIFAR-10 (MobileNet and VGG-16) and MNIST (LeNet-5) under three different reward over penalty ratios. The scores are individually normalized by the aggregation score obtained by the uncompressed models.

different parameter coding schemes. Each point stands for one compressed model in the solution set obtained by EMOMC. The results demonstrate that in terms of diversity the solution sets show similar patterns with the bi-objective optimization of accuracy and energy consumption.

Furthermore, it can be observed that although COO and CSR are developed to store a sparse matrix, sometimes they do not save the memory space for the compressed models, compared with the normal coding scheme. For example, if pursuing high model accuracy, the normal coding scheme is the best one among the three coding schemes for MoblieNet. The reason is that although the COO and CSR coding schemes only store non-zero elements, they still need several extra bits to record the position of each non-zero element. If attempting to keep the model accuracy at a high level, the compression rate cannot be high, making the memory space saved from the sparsity of the filter less than the overhead of those extra bits. In this case, the normal coding scheme is a better choice. However, if allowing a certain level of accuracy loss, then CSR is the best among the three coding schemes.

D. Aggregation of Accuracy and Energy Efficiency

Theoretically, higher accuracy comes with higher energy consumption. Most previous model compression approaches only allow a negligible loss of accuracy. For applications on edge devices, it will be acceptable to sacrifice a little bit of accuracy to achieve substantial improvement in energy efficiency. For VGG-16, as shown in Figure 5, if 2% of accuracy loss is acceptable, the energy consumption can be reduced by around 80%. In the solution sets displayed in Figure 5, there are some knee points if considering the balance of both the model accuracy and the energy consumption. To help users select the model for deployment on edge devices, a new metric called aggregation score is defined as:

$$\text{AScore} = (f_1 \cdot r + (1 - f_1) \cdot \tau)/f_2, \tag{8}$$

where f_1 is the accuracy of the model, and f_2 is the corresponding energy consumption. When classifying an image, if the result is correct, a reward r can be obtained; otherwise, a penalty τ is performed. By giving a fixed amount of energy budget, the number of images that can be classified is inversely proportional to the energy consumed per image f_2. From Equation (8), it can be seen that one of the key parameters in this aggregation score system is the ratio between the reward and the penalty r/τ, which indicates the significance of accuracy. The selection of the optimal solution highly depends on the ratio r/τ.

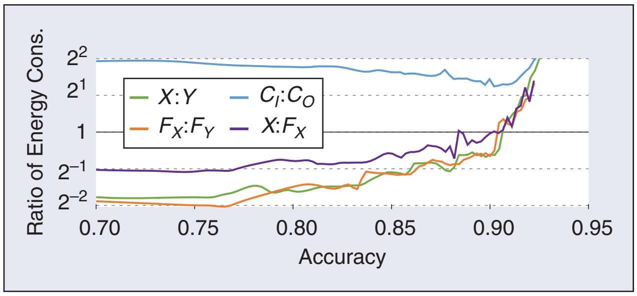

FIGURE 8 The energy consumption of VGG-16 over the energy consumption of MobileNet under different accuracy scores on CIFAR-10.

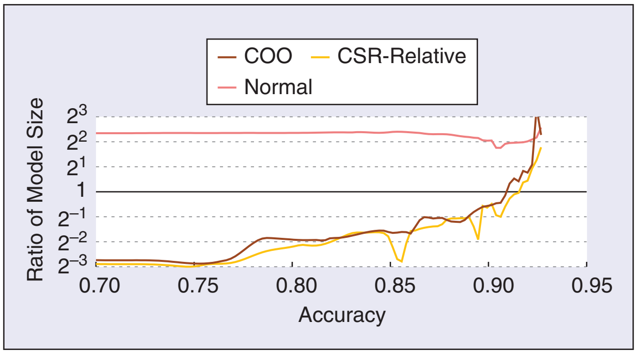

FIGURE 9 The model size of VGG-16 over the model size of MobileNet under different accuracy scores on CIFAR-10.

TABLE II Energy consumption comparison of the compressed models obtained by EMOMC and the peer methods for VGG-16 on CIFAR-10.

	ACCURACY	ENERGY CONSUMPTION (MJ)				EFFICIENCY IMPROVEMENT (×)				AGGREGATION SCORE		
METHOD	LOSS	$X:Y$	$C_I:C_O$	$F_X:F_Y$	$X:F_X$	$X:Y$	$C_I:C_O$	$F_X:F_Y$	$X:F_X$	$r/\tau = 5.0$	$r/\tau = 1.0$	$r/\tau = 0.2$
EMOMC (OURS)	0.3%	1.7	1.7	2.3	2.6	14.0	12.2	10.4	8.9	125.6	97.0	30.0
PRUNING FILTERS [26]	0.2%	18.5	19.6	18.5	19.3	1.2	1.1	1.3	1.2	1.2	1.2	1.4
PLAY AND PRUNE [50]	0.1%	9.5	12.6	9.5	11.2	2.4	1.7	2.5	2.0	2.2	2.2	2.5

TABLE III Model size comparison of the compressed models obtained by MOMC and the peer methods for VGG-16 on CIFAR-10.

	ACCURACY	MODEL SIZE (MB)			COMPRESSION RATE (×)		
METHOD	LOSS	NORMAL	CSR	COO	NORMAL	CSR	COO
EMOMC (OURS)	0.3%	9.8	8.0	24.5	6.1	7.4	2.4
PRUNING FILTERS [26]	0.2%	34.7	37.9	57.3	1.7	1.6	1.0
PLAY AND PRUNE [50]	0.1%	15.1	16.5	24.6	3.9	3.6	2.4

Figure 7 displays the aggregation scores of different solutions under three different values of r/τ. Each curve is plotted based on the results from one execution of the multi-objective optimization. As the multi-objective solver generates discrete points (*i.e.*, solutions), they are plotted in a line using a smooth function called cspline, which connects consecutive points by natural cubic splines after rendering the data monotonic. The scores are individually normalized by the aggregation score obtained by original uncompressed models. From the results, it can be observed that most accuracy-energy curves have one peak point. For the complex neural networks such as VGG-16, the highest scores will be 125 × higher than the uncompressed model due to its high compression rate. For the simpler networks such as MobileNet and LeNet, the highest aggregation score is around 9 × more than the original model.

E. On Selection of Neural Networks

On the same dataset, the selection of an optimal neural network depends on how one compresses the model. For instance, MobileNet is specially designed for computation efficiency; although its accuracy is slightly lower than VGG-16, it uses much fewer hardware resources than VGG-16, in terms of energy efficiency and model size. However, this statement is true only for the original uncompressed MobileNet and VGG-16. After model compression, VGG-16 may be more efficient than MobileNet. Figures 8 and 9 show the ratios of energy consumption and model size between VGG-16 and MobileNet, in four dataflow designs and three coding schemes. The results show that apart from dataflow design $C_I{:}\,C_O$ and the normal coding scheme, VGG-16 consumes around 50% less energy and occupies around 50% less memory space than the MobileNet when the accuracy is below 88%. This observation shows that although MobileNet is designed for computation efficiency, one should select a compressed model from a more complex neural network such as VGG-16. It is more efficient than the compressed model from simpler neural networks, in terms of energy efficiency and model size. The reason is that the number of the parameters or the precision of the parameters in VGG-16 can be lower than those of MobileNet after the mode compression.

F. Comparison to the State-of-the-Art

Tables II and III report the results of different model compression methods, in terms of the energy consumption and the aggregation scores, and model size, respectively. Table II shows that under negligible accuracy loss (typically, less than 0.5% accuracy loss), EMOMC improves the energy efficiency and model compression rate by a factor of 11.4 × and 5.3 ×, on average. There are two reasons for such improvements. Firstly, the evolutionary multi-objective solver optimizes the problem generation by generation. By allowing a certain range of accuracy loss, it can generate many intermediate results, and these results contribute to the improvements in energy efficiency or compression rate. Compared with previous methods which take accuracy loss as a hard constraint, EMOMC is more likely to find better results. Secondly, the exploration space of the model compression process is significantly reduced by adopting both pruning and quantization techniques. Without the proposed two-stage pruning and quantization co-optimization strategy, previous approaches suffer from too high computation cost to explore and exploit such a huge search space. In addition to energy efficiency and compression rate, the proposed method also shows an average 84.2 × improvement on aggregation scores.

In practice, one needs to select an optimal solution (from the solution set obtained by an EMO algorithm) for the machine learning task on a specific device. For instance, after solving the bi-objective optimization problem of accuracy and energy efficiency, a set of solutions can be obtained which trade-off the two objectives. The constraint on the energy can be calculated based on the energy capacity and battery life. Then, the solution that achieves the highest accuracy will be selected as the optimal solution for the task. Alternatively, one can select the knee point from the solution set as the preferred solution.

VI. Conclusion

In this paper, an evolutionary multi-objective model compression approach is proposed to accelerate and compress DNNs by optimizing multiple objectives (*e.g.*, accuracy, energy efficiency, and model size) simultaneously. As the evaluation of each architecture is extremely time-consuming during the evolution, a two-stage pruning and optimization co-optimization strategy is developed to speed up the architecture searching process. Extensive experimental results demonstrate that the proposed method can obtain a set of diverse networks in a single execution. Furthermore, the proposed method outperforms the peer methods in terms of energy efficiency and model size for model compression of three popular DNNs.

Acknowledgement

This work is partly supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (No. A18A1b0045 and No. A1687b0033).

References

[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge MA: MIT Press, 2016, no. 2.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1804.09081

[3] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Adv. Neural Inf. Process. Syst.*, 2019, pp. 5753–5763.

[4] L. Zhen, P. Hu, X. Peng, R. S. M. Goh, and J. T. Zhou, "Deep multimodal transfer learning for cross-modal retrieval," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. doi: 10.1109/TNNLS.2020.3029181.

[5] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2018.

[6] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 350–364, 2020. doi: 10.1109/TEVC.2019.2924461.

[7] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervized deep neural networks for learning meaningful representations," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 89–103, 2018. doi: 10.1109/TEVC.2018.2808689.

[8] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, 2020. doi: 10.1109/TNNLS.2019.2919608.

[9] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, 2020.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015. [Online]. Available: https://arxiv.org/abs/1510.00149

[11] Y. Wang, C. Xu, J. Qiu, C. Xu, and D. Tao, "Towards evolutionary compression," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2476–2485.

[12] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 5687–5695.

[13] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019.

[14] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017. doi: 10.1109/JPROC.2017.2761740.

[15] Z. Wang, T. Luo, J. T. Zhou, and R. S. M. Goh, "EDCompress: Energy-aware model compression with dataflow," 2020. [Online]. Available: https://arxiv.org/abs/2006.04588

[16] Z. Du et al., "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. Annu. Int. Symp. Computer Architecture*, 2015, pp. 92–104.

[17] M. Song et al., "Towards efficient microarchitectural design for accelerating unsupervized GAN-based deep learning," in *Proc. IEEE Int. Symp. High Performance Comput. Architecture*, 2018, pp. 66–77.

[18] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. Annu. Int. Symp. Comput. Architecture*, 2017, pp. 1–12.

[19] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer CNN accelerators," in *Proc. Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2016, pp. 1–12.

[20] J. Qiu et al., "Going deeper with embedded FPGA platform for convolutional neural network," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2016, pp. 26–35.

[21] Y.-H. Chen et al., "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH Comput. Architecture News*, vol. 44, no. 3, pp. 367–379, 2016. doi: 10.1145/3007787.3001177.

[22] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *Proc. Int. Conf. Field Programmable Logic Appl.*, 2016, pp. 1–9.

[23] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1379–1387.

[24] F. Manessi, A. Rozza, S. Bianco, P. Napoletano, and R. Schettini, "Automated pruning for deep neural network compression," in *Proc. Int. Conf. Pattern Recognit.*, 2018, pp. 657–664.

[25] C. Lemaire, A. Achkar, and P.-M. Jodoin, "Structured pruning of neural networks with budget-aware regularization," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 9108–9116.

[26] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2016. [Online]. Available: https://arxiv.org/abs/1608.08710

[27] 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 *Proc. Eur. Conf. Comput. Vision*, 2018, pp. 784–800.

[28] Z. Liu, J. Xu, X. Peng, and R. Xiong, "Frequency-domain dynamic pruning for convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1043–1053.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[30] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 1–9.

[31] K. Deb, "Multi-objective optimization," in *Search Methodologies*. Springer-Verlag, 2014, pp. 403–449.

[32] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. Int. Conf. Genetic Algorithms*, 1985, pp. 93–100.

[33] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 48, no. 1, 2015. doi: 10.1145/2792984.

[34] L. Zhen, M. Li, R. Cheng, D. Peng, and X. Yao, "Adjusting parallel coordinates for investigating multi-objective search," in *Proc. Int. Conf. Simulated Evol. Learn.*, Shenzhen, China, 2017, pp. 224–235.

[35] M. Li, L. Zhen, and X. Yao, "How to read many-objective solution sets in parallel coordinates," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 88–100, 2017. doi: 10.1109/MCI.2017.2742869.

[36] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing, 1989.

[37] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994. doi: 10.1162/evco.1994.2.3.221.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002. doi: 10.1109/4235.996017.

[39] C. A. C. Coello, S. G. Brambila, J. F. Gamboa, M. G. C. Tapia, and R. H. Gómez, "Evolutionary multiobjective optimization: Open research areas and some challenges lying ahead," *Complex Intell. Syst.*, vol. 6, no. 2, pp. 221–236, 2020.

[40] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genetic Evol. Comput. Conf.*, 2019, pp. 419–427.

[41] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[42] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (Canadian Institute for Advanced Research)," Tech. Rep., 2010. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html

[43] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89,497–89,509, 2020.

[44] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017. [Online]. Available: https://arxiv.org/abs/1710.01878

[45] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017. [Online]. Available: https://arxiv.org/abs/1710.09282

[46] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[47] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner et al., "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791.

[48] Xilinx, "*Vivado design suite user guide*," Technical Publication, 2018.

[49] E. G. Walters, "Array multipliers for high throughput in Xilinx FPGAs with 6-input LUTs," *Computers*, vol. 5, no. 4, p. 20, 2016. doi: 10.3390/computers5040020.

[50] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri, "Play and Prune: Adaptive filter pruning for deep model compression," 2019. [Online]. Available: https://arxiv.org/abs/1905.04446