Joint Optimization of Dimension Reduction and Mixed-Precision Quantization for Activation Compression of Neural Networks

Yu-Shan Tai, Student Member, IEEE, Cheng-Yang Chang, Student Member, IEEE, Chieh-Fang Teng, Yi-Ta Chen, Student Member, IEEE, and An-Yeu (Andy) Wu, Fellow, IEEE

Abstract—Recently, deep convolutional neural networks (CNNs) have achieved eye-catching results in various applications. However, intensive memory access of activations introduces considerable energy consumption, resulting in a great challenge for deploying CNNs on resource-constrained edge devices. Existing research utilizes dimension reduction and mixed-precision quantization separately to reduce computational complexity without paying attention to their interaction. Such naive concatenation of different compression strategies ends up with sub-optimal performance. To develop a comprehensive compression framework, we propose an optimization system by jointly considering dimension reduction and mixed-precision quantization, which is enabled by independent group-wise learnable mixed-precision schemes. Group partitioning is guided by a well-designed automatic group partition mechanism that can distinguish compression priorities among channels, and it can deal with the trade-off between model accuracy and compressibility. Moreover, to preserve model accuracy under low bit-width quantization, we propose a dynamic bit-width searching technique to enable continuous bit-width reduction. Our experimental results show that the proposed system reaches 69.03%/70.73% with average 2.16/2.61 bits per value on Resnet18/MobileNetV2, while introducing only approximately 1% accuracy loss of the uncompressed full-precision models. Compared with individual activation compression schemes, the proposed joint optimization system reduces 55%/9% (-2.62/-0.27 bits) memory access of dimension reduction and 55%/63% (-2.60/-4.52 bits) memory access of mixed-precision quantization, respectively, on Resnet18/MobileNetV2 with comparable or even higher accuracy.

Index Terms—Activation compression, convolutional neural network, dimension reduction, mixed-precision quantization.

I. INTRODUCTION

Nowadays, deep convolutional neural networks (CNNs) have been widely applied in different fields and reached remarkable performance, such as face recognition [1], image classification [2], disease detection [3]. However, these models typically require extremely huge memory footprints and computations to reach the eye-catching accuracies in each scenario. As a result, directly deploying huge CNNs on resource-constrained edge devices is challenging, hindering them from real-world applications. To alleviate this issue, model compression techniques have gained more and more attention in recent years.

During the entire CNN computation, data movements of intermediate meta data, e.g., activations or feature maps, account for more than half of the total energy cost. Take GoogleNet as an example, up to 68% of energy consumption results from the data communication between the deep learning accelerator (DLA) and off-chip memories [4][5]. Consequently, activation compression (AC) is emerging as an orthogonal approach to model compression. Unlike CNN weights, which are static after off-line training, activations are dynamic and highly dependent on the input data; therefore, the compressibility of activations is not guaranteed, thus challenging and requiring special treatment.

To exploit the redundancy in activations, transform-based dimension reduction (DR) [6]-[11] has been proposed to enhance data sparsity, as shown in Fig. 1(a). Instead of...
compressing raw activations directly, transform-based DR leverages discrete cosine transform (DCT) [6][7] or principal component analysis (PCA) [8]-[11] to project data to another domain, where important and unimportant segments are decorrelated. Consequently, the unimportant portion could be removed to achieve high sparsity without sacrificing the accuracy.

Another popularly used technique is fixed-precision quantization [12]-[17], which assigns equal bit-width to the activations of entire model, as shown in Fig. 1(b). However, it neglects the characteristic among different layers. To improve the compressibility by exploiting layer-wise redundancy, mixed-precision quantization (MP) [19]-[24] allocates specific bit-width for each layer, as shown in Fig. 2(a). To follow the wave, some hardware manufactures and academia also started to implement mixed-precision on chips, such as Apple A12 Bionic chip [25], NVIDIA Automatic Mixed Precision (AMP) [26], and BISMO [27]. However, exhaustive searching the optimal bit-width allocation from an exponential number of discrete combinations regarding the model depth is time-consuming. To solve this issue, the authors of [22]-[24] employ a differentiable search architecture. By adding a specific bit-width to each searching branch, the authors of [22]-[24] relax the search space continuous and make gradient descent feasible.

Although the aforementioned methods are commonly used to achieve fabulous results, most frameworks regard DR [6]-[11] and MP [20]-[24] as two independent techniques without suitable joint search space. However, the optimal pruning strategy of the floating-point model is not guaranteed to be the best for the quantized one. Consequently, combining these approaches without considering their interactive effects ends up with sub-optimal compression rate [18]. To rapidly obtain the optimal configuration for different orthogonal compression approaches, the authors of [18] utilize the evolutionary search with an accuracy predictor to accelerate the evaluation process. The joint system [18] successfully outperforms the corresponding sequential design in both accuracy and compression ratio. Nevertheless, the non-trivial dataset collection, requiring to train a large supernet for 2,400 GPU hours, hinders the joint system from practical applications. Moreover, the evolutionary search takes hundreds of iterations to reach the optimal solution. Another joint optimization framework [19] manually designs different thresholds to partition channels and leverages differentiable search to allocate specific compression policies. However, the handcrafted design fails to reach optimal configurations. It becomes intolerable when the model architecture gets large and complex, thus confining its scalability.

In summary, although transform-based DR and MP have been exploited extensively in the literature, some issues are still needed to be addressed:

1) **Threshold-based DR ignores layer-wise characteristics:** Existing transform-based DR methods exploit the redundancy among channels by assigning the same threshold of accumulated eigenvalues to each layer [8][9]. However, they neglect that both the eigenvalue distributions and the activation sizes vary among CNN layers. Without considering the layer-wise variance makes the compressing process sub-optimal.

2) **Hand-crafted and static MP policy:** In existing differential MP methods, the bit-width of each searching branch is set manually and fixed during the training phase. Therefore, the options usually consist of low bit-width branches to achieve high compression ratio. Nevertheless, directly applying aggressive quantization on models may result in dramatic accuracy degradation. Moreover, the fixed branches limit the search space, sacrificing the compressibility of quantization.

3) **Neglect mutual effects between DR and MP:** Due to the vast joint search space of DR and MP, most works only focus on one of them. We observe that combining DR and MP can potentially improve the model compressibility. However, naïve concatenation neglects the interaction between DR and MP, leading to a sub-optimal compression rate. Consequently, how to efficiently co-optimize DR and MP is still an open problem.

In this paper, we propose a joint optimization system of DR and MP based on the differentiable search architecture. The system overview and layer-wise visualization are shown in Fig. 1(c) and Fig. 2(b), respectively. Our main contributions are:

1) **Greedy dimension reduction (Greedy DR) with a newly-designed selection metric:** We design a selection metric considering both the accuracy drop and activation sizes. Based on the selection metric, we use greedy algorithm, which iteratively applies DR to the most profitable layer in each round, to massively reduce DRAM access with minimum accuracy drop. Our experimental results show that Greedy DR reaches 67.2%/70.2% with average 4.40/2.72 bits per value on Resnet18/MobileNetV2.

2) **Learnable mixed-precision (MP) scheme with dynamic bit-width searching (DBS):** To focus on AC in the differentiable searching process, we design a learnable mixed-precision scheme with customized memory overhead loss. Furthermore, to achieve aggressive quantization without sacrificing accuracy, we propose a dynamic bit-width searching strategy to reach continuous bit-width reduction and enlarge the search space. Combined with the abovementioned techniques, our simulation results get 68.3%/70.4% with average 2.32/2.66 bits per value on Resnet18/MobileNetV2.

3) **Joint Optimization of group-wise DR and MP:** To jointly optimize DR and MP seamlessly, we further introduce a group-wise mechanism. We propose an automatic group partition (AGP) mechanism, a combination of the aforementioned selection metric for Greedy DR and an outlier detection technique. With AGP, our system can dynamically update partition during the searching phase. Moreover, with the proposed learnable MP schemes, our system automatically learns the optimal group-wise compression policy in few epochs rather than time-consuming iterative searching. Equipped with the group-wise mechanism, the proposed joint system finally
reaches 69.03%/70.73% with average 2.16/2.61 bits per value on Resnet18/MobileNetV2.

The rest of the paper is organized as follows. In Section II, the existing activation compression methods are introduced. Section III presents the Greedy DR with the proposed selection metric. After that, Section IV elaborates the proposed joint optimization system of group-wise DR and MP, which contains the learnable MP scheme with DBS and AGP mechanism. Section V demonstrates the simulation results and analysis of our joint optimization system. Finally, the conclusions are drawn in Section VI.

II. RELATED WORKS

A. Transform-based Dimension Reduction (DR)

Since the sparsity of activations is dynamic and highly dependent on the input data, techniques sensitive to data distribution are not suitable for AC. To overcome the restriction of sparsity pattern and further enhance the compression ratio, transform-based methods are introduced, as shown in Fig. 1(a). Since activations are generated by mapping relatively small-sized inputs to high dimensions, there exists redundancy among channels [9]. Consequently, rather than directly removing small-valued elements as conventional pruning methods [17][18], activations can be decorrelated into important and unimportant components with suitable transformations, e.g., DCT [6][7] and PCA [8]-[11]. After that, the unimportant channels can be removed to reduce storage requirements without sacrificing the model accuracy.

In [7], the authors utilize DCT to project activations to frequency domain. Furthermore, to fold the transformation matrix into convolution layer, the authors of [6] implement 1D-DCT on the channel domain rather than the spatial domain and utilize a mask to facilitate channel reordering. However, the accuracy is sensitive to the design of the mask, thus limiting its applicability.

Instead of using DCT as the transformation matrix, some other works choose PCA as an alternative. In [8], the authors adopted a pre-computed PCA matrix to transform activations. Let \( A \) stand for the activation, whose size is \( n \times d \times w \times h \), \( n, d, w, h \) represent the batch size, channel number, width, and height, respectively. Since PCA operates on the channel domain, \( A \) would need to be reshaped to \( A_c \in d \times (n \times w \times h) \) first. Then, the corresponding transformation matrix \( U \) can be obtained by PCA:

\[
U, \Sigma = PCA(A_c),
\]

where \( U \) stands for the orthogonal basis and \( \Sigma = \{ \sigma_1^2, \sigma_2^2, ..., \sigma_N^2 \} \) is its corresponding eigenvalues in descending order. After multiplying \( A_c \) with \( U \) following by reshaping the multiplication result, we obtain the transformed activation \( A' \). If the activations are needed to be fetched from the off-chip memory, they could be reconstructed by the symmetric inverse process. Since the PCA matrix is data-dependent, it is more likely to reach higher compressibility than fixed DCT matrix. Moreover, the importance of each channel could be verified by its eigenvalue, where larger \( \sigma \) implies higher amount of information. Thus, threshold-based dimension reduction [8][9] utilizes this concept by keeping a minimal number of channels with accumulated eigenvalue reaching a pre-defined threshold \( T_{\text{Thres}} \), which can be specified as:

\[
d_l^* = \arg \min_{d_l} \frac{\sum_{k=1}^{K} \sigma_k^2}{\sum_{i=1}^{l} \sigma_i^2} \geq T_{\text{Thres}}, \quad l \in \{1,2,...,L\}.
\]

\( d_l \) and \( d_l^* \) denote the number of channels of layer \( l \) before and after dimension reduction, respectively, and \( L \) stands for the total number of layers. By removing these unimportant channels, we can further reduce the channel dimension of the activations to a smaller scale, directly decreasing the memory access requirements.

Although PCA transformation can address the lack of sparsity and locality, there still exists some room for improvement. Specifically, threshold-based dimension reduction, which reduces the dimension of each layer as Eq. (2), applies the same threshold to the entire model. Since the eigenvalue distributions differ layer by layer, the impacts on model accuracy of DR are also different. Moreover, the spatial size of feature maps \( (w \times h) \) significantly influences the compression strength. Ignoring the layer-wise difference would not only make the compression process inefficient but also hurt the model accuracy. Therefore, our proposed Greedy DR tackle these issues by a well-designed selection metric, achieving layer-wise design with consideration of both accuracy drop and activation sizes.

B. Mixed-Precision Quantization (MP)

Besides dimension reduction, quantization is another prevalent compression technique. Compared with full-precision representation, quantization methods allocate less bit-width to save computation cost and memory usage, as shown in Fig. 1(b). The most commonly used strategy is fixed-precision quantization, which allocates equal bit-width among the entire network. Nevertheless, without considering the potential difference among layers, fixed-precision quantization results in a sub-optimal trade-off between accuracy and compressibility [12]-[17]. To consider the layer-wise redundancy and enhance compression ratio, mixed-precision quantization gains more attraction recently [18]-[24], as shown in Fig. 2(a).

However, the vast and non-differentiable search space makes it challenging to find the optimal layer-wise bit-width allocation. Related research can be divided into two scopes, which are non-differentiable and differentiable approaches. In the non-differentiable category, the authors of [20][21] resort to reinforcement learning to train an agent to determine the suitable mixed-precision policy, while the authors of [18] utilize the evolutionary algorithm to search the optimal compression strategy. Nevertheless, these methods require a long searching time. Consequently, differentiable methods [22]-[24], which force the model to learn suitable bit-width allocation automatically, are introduced to serve as another practical alternative for mixed-precision quantization.

In [22], the authors design a hypernet composed of \( N^a \) and \( N^b \) parallel branches for weights and activations, respectively. Each branch denotes a bit-width option for exploration, and the output during the training phase is the weighted sum of these
branches. Specifically, the corresponding output can be formulated as follows:

\[ y = \sum_{i=1}^{N^a} \pi_i^a Q_i^a (W) * \sum_{j=1}^{N^b} \pi_j^b Q_j^b (A), \]

\[ \pi_i^a = \exp(\alpha_i) / \sum_k \exp(\alpha_k), \pi_j^b = \exp(\beta_j) / \sum_k \exp(\beta_k). \]

\[ W \text{ and } A \text{ denote the weight and activation, } Q_i^a \text{ and } Q_j^b \text{ stand for uniform quantization operator with bit-width } b_i^a \text{ and } b_j^b. \]

The quantization operators, \( Q_i^a \) and \( Q_j^b \), contain the quantization and dequantization process:

\[ x_q = \text{clamp}(\lfloor x/s_i \rfloor, -2^{b_i-1}, 2^{b_i-1} - 1), \]

\[ x_{\text{rec}} = x_q \times s_i = Q_i(x) \]

where \( \text{clamp}(x, l, h) \) is used to clamp \( x \) in \([l, h]\) and \( s_i \) denotes the scaling factor. \( \alpha_i \) as well as \( \beta_i \) are the corresponding architecture parameters. Through \( \alpha_i \) and \( \beta_i \), the optimization can be done by gradient descent due to the relaxed continuous search space. Moreover, since the objective of [22] is to reduce the computation complexity, bit operations are summed as the penalty loss to navigate the direction of optimization. During inference, only the branches with the highest \( \pi_i^a \) or \( \pi_j^b \) of each layer would be selected as the final bit-width.

Though differentiable mixed-precision quantization successfully allocates layer-wise bit-width efficiently, there still exist some challenges. First, most mixed-precision strategies ignore the channel-wise redundancy, which fails to achieve optimal bit-width allocation. Second, the search space is confined by the hand-crafted bit-width of branches, i.e., fixed \( b_i^a \) and \( b_j^b \). For high compression ratios, the low bit-width are usually set prior to the search process. However, the aggressive quantization may induce significant accuracy loss. On the other hand, directly increasing the number of branches, \( N^a \) and \( N^b \), not only leads to extraordinary memory overheads but also suffers accuracy degradation due to the unconcentrated distribution of \( \alpha_i / \beta_i \) and non-continuous bit-width drops. Consequently, we design a novel group-wise mechanism to further exploit the channel-wise redundancy and enhance compression rates. Moreover, to enlarge the search space and facilitate continuous bit-width conversion, we propose a dynamic bit-width searching (DBS) strategy to replace the hand-crafted branches.

C. Co-design of Dimension Reduction (DR) and Mixed-Precision Quantization (MP)

The aforementioned compression techniques, DR and MP, are two orthogonal methods which can be applied simultaneously to improve the overall model compression ratio. Most research just focuses on optimizing the two methods individually and then concatenating them directly. Nevertheless, sequential concatenation [17] of the two isolated designs leads to sub-optimal results. Since the optimal DR strategy for floating models is not always the best for the quantized ones, different methods should be co-optimized to enhance the compressibility [18]. However, joint optimization would introduce extraordinary search space, leading to unaffordable time and energy costs for an exhaustive search.

To alleviate this issue, the authors of [18] utilize the evolutionary algorithm with an accuracy predictor to accelerate the evaluation process of joint optimization of architecture, pruning ratio, and bit-width. With consideration of interaction among these orthogonal compression approaches, the joint system achieves higher accuracy and compression ratio compared with the corresponding sequential design. However, collecting the training data, pairs of architectures and their corresponding accuracy, for the predictor is non-trivial. In fact, the preparation requires to train a large supernet for 2,400 GPU hours. Furthermore, the evolutionary search requires hundreds of iterations to reach the optimal solution. Another joint system [19] utilizes differentiable search to learn the optimal compression policy instead. Despite saving much training cost compared with [18], [19] requires manually-designed thresholds for channel partitioning. The hand-crafted policy introduces extra overhead for parameters tuning, suffering from sub-optimal configurations and impractical for practical applications. Therefore, instead of finding the optimal strategy from a huge pool and notorious parameter tuning, we propose to extend the differentiable search framework [22]-[24] to learn the optimal compression policy with dynamic group partition during training. Without huge training requirements and time-consuming iterative searching, our method achieves the compressed model combining DR and MP by a relatively small amount of training epochs, greatly reducing the searching cost.

III. IMPROVED MECHANISM OF DIMENSION REDUCTION

In this section, we would elaborate the proposed improved design specialized for dimension reduction. First, we introduce the Greedy DR with a newly-designed selection metric [11]. Then, we present the corresponding experimental results compared with previous works. In Section IV, we will further combine Greedy DR with MP to achieve joint optimization.

A. Proposed Greedy Dimension Reduction (Greedy DR)

Though threshold-based dimension reduction [8][9] removes unimportant channels, this strategy leads to severe accuracy drop under high compression ratios. To tackle this issue by analyzing layer-wise difference, we propose a greedy selection metric that considers both accuracy drop and memory reduction to achieve a better trade-off.

Our strategy is to greedily choose one layer for dimension reduction at each step and iterate until the percentage of the remaining channels is less than the pre-defined target ratio \( T_{\text{Greedy}} \). To achieve our goal, we design a selection metric to prioritize which layer is most suitable for DR:

\[ S_i = \Delta_{\text{accuracy}} / \Delta N, \]

where \( \Delta_{\text{accuracy}} \) and \( \Delta N \) represent the accuracy reduction and saved storage after operating DR on layer \( i \). Small \( S_i \) implies sacrificing minor accuracy and reducing substantial memory footprints. Take Fig. 3 as an example. There are \( L \) projection matrices \( \{P_{1:d'_1}, P_{2:d'_2}, \ldots, P_{L:d'_L}\} \), and each \( P_{l:d'_l} \) contains \( d'_l \) rows. By computing the selection metrics of each layer, we obtain \( S = \{S_1, S_2, \ldots, S_L\} \). If the minimum occurred
we can observe we keep \( \sigma_h \), negligible compared with total model. Note that the technical details and \( \sigma_h \), a = , -

Fig. 3. Greedy DR with proposed selection metric.

at layer 2, we would remove the last row of \( \mathbf{P}_{d,l} \). Afterward, \( d_2' \) is updated to \( d_2' - 1 \). During the whole process, we keep greedily selecting layers with minimal \( S_l \) to remove the least significant channel until reaching the desired threshold \( T_{Greedy} \). Since evaluating the accuracy drop in each step is time-consuming for large-scale tasks (e.g., ImageNet) and sometimes labeled data may be unavailable, a simple yet effective alternative evaluation metric is required. As mentioned in [10], the product of the layer-wise cumulated eigenvalue is highly related to final accuracy. Therefore, we utilize the percentage of eigenvalue on layer \( l \) to approximate the accuracy drop induced by DR:

\[
\Delta \text{accuracy} = \frac{\sigma_{l,d}}{\sum_{c=1}^{d_l} \sigma_{l,c}}.
\]

As for \( \Delta N \), we measure the saved storage after removing one channel to quantify the compression gain, which can be formulated as the size of a single feature map:

\[
\Delta N = w \times h.
\]

Compared to Eq. (2), the proposed selection metric for Greedy DR reaches a better trade-off since it takes both accuracy drop and compression strength into consideration. In next subsection, we would compare the performance of our greedy-based method with prior threshold-based work to evaluate its effectiveness and then analyze the different distribution of DR between them.

### B. Experimental Results

In this subsection, we demonstrate the effectiveness of proposed Greedy DR. Note that the technical details and simulation results of the joint optimization system of DR and MP are presented in Section IV. In the following experiments, we simulate our proposed method on Pytorch pre-trained ResNet18 and MobileNetV2, whose accuracies are 69.758% and 71.818% with full-precision, respectively. The dataset we implement is ImageNet (ILSVRC 2012) [28]. We randomly sample 50,000 and 200 out of 1,281,167 training data to fine-tune our models and pre-compute the PCA matrix, respectively. After DR, we fine-tune the models for 6 epochs. The learning rates are set as \( 1e-3 \) and \( 1e-5 \) for ResNet18 and MobileNetV2. Note the computation is negligible compared with total model training, which requires training the whole training dataset for hundreds of epochs. The values of \( T_{Thres}, T_{Greedy} \) for the experiment are set as \([0.91,0.92,0.93], [0.53,0.55,0.6]\) for ResNet18 and \([0.987,0.99,0.993], [0.32,0.33,0.34,0.36]\) for MobileNetV2. We indicate the benefits of activation compression by lowering the average bits per value, which is calculated by dividing the total number of bits required to send to off-chip memory by the number of activation elements. The bit-width of weight and batch size are set as 8 and 64. We choose stochastic gradient descent (SGD) as our optimizer. All the experiments are operated with PyTorch1.9.0 and Python3.9.

#### 1) Comparison between Proposed Greedy DR and Prior Works

In this part, we evaluate the compression performance between the proposed greedy DR and conventional threshold-based DR [8] as shown in Fig. 4. Each point indicates DR under different thresholds \( T_{Thres} \) or \( T_{Greedy} \), where lower thresholds correspond to smaller average bits per value. We can observe that threshold-based DR effectively reduces bit-width requirements, but accuracy suffers from catastrophic drops as \( T_{Thres} \) decreases.

On the other hand, after replacing the DR mechanism with the proposed Greedy DR, the performance can be further improved. By considering accuracy drop and bits reduction simultaneously, we can achieve a better trade-off than Eq. (2). In summary, our method reaches 67.2%/70.2% with average 4.40/2.72 bits per value on Resnet18/MobileNetV2, which reduces 12%/11% (0.6/-0.33 bits) memory access of threshold-based DR with comparable or even higher accuracy.

![Fig. 4. Performance comparison of threshold-based DR and Greedy DR on (a) ResNet18 and (b) MobileNetV2.](image-url)

![Fig. 5. The visualization of channel distribution of threshold-based DR and Greedy DR on (a) ResNet18 with \( T_{Thres} = 0.99, T_{Greedy} = 0.34 \) with \( T_{Thres} = 0.99, T_{Greedy} = 0.34 \).](image-url)
2) **Visualization of Dimension Reduction Policy**

To further analyze the difference between proposed Greedy DR and threshold-based DR, we visualize the number of remaining channels among different layers. We take ResNet18 with $T_{\text{Thres}} = 0.92$, $T_{\text{Greedy}} = 0.55$ and MobileNetV2 with $T_{\text{Thres}} = 0.99$, $T_{\text{Greedy}} = 0.34$ as examples and illustrate in Fig. 5. From this figure, Greedy DR tends to remain a greater number of channels for deep layers than threshold-based DR. Since the size of deep layers is smaller than shallow layers, they are more likely to receive large selection metrics. Specifically, the size of the first feature map of ResNet18 is 112×112 while that of the last layer is 7×7. Consequently, we can conclude that compressing shallow layers leads to higher number of bits reduction and makes our Greedy DR more efficient.

IV. PROPOSED JOINT OPTIMIZATION SYSTEM OF DR AND MP

Based on the improved design of DR mentioned in Section III, we further propose a joint optimization system of DR and MP for comprehensive compression. We first present the overview of the proposed DR and MP joint optimization framework, as shown in Fig. 6. Next, we introduce an automatic group partition (AGP) mechanism to exploit the channel-wise redundancy, combining the selection metric mentioned in Section III.A with an outlier detection technique. Finally, we elaborate the proposed learnable MP scheme with a dynamic bit-width searching (DBS) strategy, which automatically learns the optimal group-wise compression policy.

A. Proposed Joint Optimization System

Our whole framework is shown in Fig. 6, which achieves joint optimization of DR and MP by operating the following three steps iteratively:

1) **Sort channel importance by PCA:** To reduce computation cost, we sample a small amount of data from the training dataset to pre-compute PCA transformation matrix $U$ first. Then, after undergoing convolution ($\ast$) and batch normalization (BN), the transformed activation $A'_I$ can be obtained after multiplied by $U$.

2) **Group partition:** With the transformed activation $A'_I$, we can further divide channels into groups according to their sensitivity and importance. To avoid manually-designed thresholds in [19], we propose an automatic group partition (AGP) mechanism, which would be elaborated in Section IV.B. After the group partition process, $A'_I$ are separated into $\{A'_{1,1}, A'_{1,2}, \ldots, A'_{1,G}\}$ for group 1 to $G$, while group 1 contains the most important channels, and group $G$ comprises the most unimportant ones.

3) **Group-wise DR and MP schemes:** After group partitioning by AGP, we can apply group-wise compression policies to exploit channel redundancies. Since channels in group $G$ are those most suitable for dimension reduction, they are directly pruned, i.e., the values of them are seen as zero with no memory overhead. As for the other groups, we utilize the independent learnable MP schemes, whose details are specified in Section IV.C, to learn the group-wise optimal bit-width allocation. Finally, the output $A''_I$ can be reformulated by concatenating $A''_{I,g}$ from group 1 to $G-1$.

To jointly optimize DR and MP, our system undergoes the aforementioned three steps during each iteration to recalculate $U$ and reallocate the channels for each group. Since the model weights update during each training epoch, the renewed process keeps our compression policy in accordance with the current model state. Therefore, the channels belonging to one group may change to another one in the next iteration, which means those chosen for DR (group $G$) can be recovered (group 1, 2, ..., $G-1$) in the following iterations. In other words, our system can dynamically modify the compression policy for each channel during the searching phase, therefore achieving co-optimization of DR and MP.

To sum up, our system can reduce the channel dimensions and further operate group-wise mixed-precision on the remaining channels. Since the granularity of MP is relevant to hardware overhead, users can decide the value of $G$ according to different hardware implementations, and the corresponding experiments under different parameter settings would be specified in Section V.B.

B. Proposed Automatic Group Partition (AGP) Mechanism

To maximize the benefits of group-wise compression policy, the group partition mechanism should take the channel-wise variety into account. Therefore, we combine the selection metric mentioned in Section III.A with an outlier detection technique to propose the automatic group partition (AGP) mechanism. The whole partition process is shown in Fig. 7, which includes three steps:

1) **Selection metric of group $G$ (DR):** The proposed selection metric, as specified in Section III.A, can identify compression priority among channels regarding both the accuracy drop and compression strength. To select the channels most suitable for DR, which are the members of group $G$, we select channels with minimal selection metric iteratively until reaching the user-designed DR threshold $T_{\text{Greedy}}$.

2) **Outlier detection of group 1 to $G$-1 (MP):** After determining the channels for DR, we further divide the remaining ones into groups to exploit channel-wise redundancy. For those channels belonging to the same group, they share a learnable MP scheme and thus the same compression policy, e.g., equal bit-width and scaling.
factor. To initial the group partition, we use channel-wise mean square error (MSE) between 8-bit and full-precision activations to evaluate the quantization sensitivities. Then, we implement K-means algorithm to cluster channels with similar MSE into 1 to G-1 groups for each layer. However, initialization has a great impact on the final results of K-means algorithm. If the allocation is sub-optimal, there may exist some channels with relatively larger or smaller quantization errors than their group members. In other words, the channels are outliers and suitable for group reallocation. Consequently, we utilize z-score of MSE to detect outliers. Z-score is a commonly used indicator to identify the extension of deviation, and the z-score of a data point \( x_i \) can be formulated as follows:

\[
  z_i = \frac{x_i - \bar{x}}{\sigma},
\]

where \( \bar{x} \) and \( \sigma \) stand for the mean and standard deviation. An element with z-score of 1 is one standard deviation above the mean. Therefore, data with absolute z-score larger than 1 deviate from the mean and thus require reorganization. We perceive channels with absolute z-scores larger than 1 as outliers, and update their group partition in next step.

3) Update group partition of outliers: For outliers with positive z-scores, they own larger MSE than their group members. Therefore, we incrementally reallocate them to more important groups to reduce their MSE. For instance, if the outliers belong to group 2 originally, then they will be updated to group 1. On the other hands, outliers with negative z-scores, implying potential for aggressive compression, are moved to less important groups. Moreover, since our joint system recalculates the selection metric for DR in step 1 during every iteration, some channels selected to DR in the current iteration may restore in later iterations. For stable training, we allocate those recovered channels to group G-1, which is the least important group.

By means of the proposed AGP mechanism, our system is able to automatically partition channels into groups given their sensitivity and importance. Moreover, due to the dynamic update process, the channels chosen for DR are not only determined by Greedy DR but also influence by the global information in the learning process during searching phase. Likewise, the group-wise learnable MP scheme is also influenced by Greedy DR. With iterations of update, the end-to-end joint optimization system can renew the compression policy considering DR and MP simultaneously.

C. Proposed Learnable MP Scheme with Dynamic Bit-width Searching (DBS)

To enable differentiable MP searching targeted for AC and maintain model accuracy under aggressive quantization, we propose the learnable MP scheme with dynamic bit-width searching. In the following context, we first introduce our proposed learnable MP scheme. Then, we elaborate the dynamic bit-width searching strategy, which is incorporated with the learnable MP scheme.

1) Learnable MP scheme

To achieve mixed-precision quantization targeted for activation compression, we introduce a learnable MP scheme based on the prior differentiable method [22], as illustrated in Fig. 8. Since our target is to reduce the data movement of activations, we only focus on MP for activations. Therefore, we use \( N_i, Q_{i,g}, b_{i,g}, \beta_{i,g} \) and \( \pi_{i,g} \) to replace \( N^\beta_i, Q_i^\beta, b_i^\beta, \beta_i \) and \( \pi_i^\beta \) mentioned in Section II, where \{ \( b_{1,g}, b_{2,g}, ..., b_{N_i,g} \) \} are sorted in ascending order, and script \( g \) denotes group \( g \) out of \( G \) groups in total. Moreover, we apply the learnable range used in [24], assigning a trainable parameter \( \alpha_{i,g} \) to clamp the range of activation to \( [-\alpha_{i,g}, \alpha_{i,g}] \) and thus determining the scaling factor \( s_{i,g} \) as \( \frac{\alpha_{i,g}}{2^{\pi_{i,g}} - 1} \). The input of each learnable MP scheme is the transformed activation \( A'_{i,g} \) with size \( n \times d_{i,g} \times w_i \times h_i \). The whole training process can be divided into the searching phase and fine-tuning phase.

a) Searching phase: To search for the optimal bit-width distribution, our joint system updates the architecture parameters \( \beta_{i,g} \) and the learnable range \( \alpha_{i,g} \) in each iteration during searching phase. Moreover, the AGP mechanism mentioned in Section IV.B renews the group partition in this phase. As for the output \( \mathbf{A}^Q_{i,g} \), it can be formulated as the weighted sum of all branches:

\[
\mathbf{A}^Q_{i,g} = \sum_{i=1}^{N} \pi_{i,g} Q_{i,g}(A'_{i,g}).
\]

b) Fine-tuning phase: To reduce the dependency of model on the other irrelevant branches, we fix the group partition, \( \beta_{i,g} \) as well as \( \alpha_{i,g} \) and only train the branches with the largest \( \pi_{i,g} \) of each layer. The output \( \mathbf{A}^Q_{i,g} \) can be specified as the output of the branch with the largest \( \pi_{i,g} \):

\[
\mathbf{A}^Q_{i,g} = Q_{k,g}(A'_{i,g}), \quad k = \arg\max_i(\pi_{i,g}).
\]
Unlike the approaches of [22], we focus on activation compression rather than computation complexity reduction. Therefore, we design a memory overhead loss $L_{memory}$ to encourage low bit-width allocations, which is the expected sum of activations bits over total $L$ layers:

$$L_{memory} = p \sum_{l=1}^{L} \sum_{g=1}^{G} \sum_{i=1}^{N} \pi_{i,g} b_{l,g} d_{l,g} h_{l,w_{l}}.$$  \hspace{1cm} (13)

where $p$ is the hyperparameter to adjust the penalty strength. Moreover, considering labeled data are hard to access and sometimes involve privacy issues, we adopt knowledge distillation to learn from the output distribution of the original model:

$$L_{KD} = D_{KL}(\mathbf{o} || \mathbf{o}'),$$  \hspace{1cm} (14)

where $D_{KL}(\cdot || \cdot)$ is the Kullback-Leibler divergence, and $\mathbf{o}$ as well as $\mathbf{o}'$ stand for the output vectors of the original model and that of the compressed model. Finally, our loss function for training can be written as:

$$L = L_{memory} + L_{KD}.$$  \hspace{1cm} (15)

With the proposed learnable MP scheme, we can leverage gradient descent to automatically search the optimal bit-width combination for AC within several epochs. In the following section, we would further elaborate on the proposed dynamic bit-width searching (DBS) technique, which is operated in the learnable MP scheme.

### 2) Dynamic Bit-width Searching (DBS)

In this section, we will introduce a novel searching technique and incorporate it with the learnable MP scheme. Prior works tend to set low bit-width branches to achieve high compression ratio; however, aggressive quantization with small $b_{l,g}$ sometimes results in irrecoverable information loss. Although large $b_{l,g}$ would maintain accuracy well, it would lead to poor model compression ratio. Moreover, just increasing the number of searching paths $N$ would lead to extra computation burden and inefficient training due to unconcentrated $\pi_{i,g}$ and sharp bit-width reduction. To solve this issue, we propose a dynamic bit-width search (DBS) technique to replace fixed $b_{l,g}$, which not only enables continuous bit-width conversion but also enlarges search space without additional memory overheads.

Since activations under 8 bits quantization are shown to maintain original performance, we initialize each group with $N = 3$ and $\{b_{1,g}, b_{2,g}, b_{3,g}\} = \{6,7,8\}$ to replace low bit-width branches. To break the limitation of fixed search space, we dynamically update $b_{l,g}$ during the searching phase according to the distribution of the architecture parameters $\beta_{l,g}$. An example is illustrated in Fig. 8. If the maximum occurred at $\beta_{1,g}$, and the minimum appeared at $\beta_{3,g}$, this implies the tendency to move toward low bit-width allocation. Therefore, we can update the search space by changing the bit-width $b_{l,g}$ of each branch:

$$\{b'_{l,g}, b'_{2,g}, b'_{3,g}\} \leftarrow \{b_{1,g} - 1, b_{1,g}, b_{2,g}\},$$  \hspace{1cm} (16)

$$\{\beta'_{1,g}, \beta'_{2,g}, \beta'_{3,g}\} \leftarrow \{\beta_{2,g}, \beta_{1,g}, \beta_{2,g}\},$$  \hspace{1cm} (17)

where $b'_{l,g}$ and $\beta'_{l,g}$ stand for the updated values of $b_{l,g}$ and $\beta_{l,g}$, respectively. We remove the branch with the largest bit-width $b_{g}$ and introduce a small bit-width one $b'_{g}$. Then, $\beta_{g}$ is assigned to the new branch $\beta_{g}'$ to form a bell-shaped distribution. For stable training, we update branches only when the model shows a strong tendency to reduce bit-width. Specifically, the search space will be updated when the occurrences of $\beta_{1,g} > \beta_{2,g} > \beta_{3,g}$ reaches pre-defined patience, which is set as 3 in our following experiments.

Utilizing DBS, we can force the bit-width to decrease continuously and smoothly, which avoids sharp conversion and thus preserves model accuracy better. Moreover, this approach breaks the restriction of limited search space, achieving optimal bit-width allocation automatically.

### V. EXPERIMENTAL RESULTS OF THE JOINT OPTIMIZATION SYSTEM

In this section, we present the simulation results. For ResNet18, we set the learning rate as $1e^{-3}$ and the number of searching epochs is 15. As for MobileNetV2, the learning rate is $1e^{-3}$ and the number of searching epochs is 30. Both models would be fine-tuned for additional 6 epochs with fixed compression policy. The detailed analysis of computation overhead is elaborated in Section V.D. The values of penalty $p$ are set as $[0.05, 0.1, 0.2]$ for ResNet18 and $[0.001, 0.01, 0.02]$ for MobileNetV2, respectively. The patience for DBS is set as 3. Since the goal of this paper is to alleviate the bandwidth bottleneck of DRAM access, we focus on activation compression rather than computation complexity reduction. Therefore, we quantize the weights of both ResNet18 and MobileNetV2 to 8 bits in all experiments. The other experimental setups and parameter settings are the same as Section III.B.
Since previous research [22] emphasize complexity reduction, we make some modification to prior work for a fair comparison. First, we use our proposed memory overhead loss $L_{\text{memory}}$ as Eq. (13) to replace bit operations as the penalty loss. Secondly, we apply the learnable range used in [24] to determine the values of scaling factor. Next, prior methods tend to set a fixed bit-width options in the search space. Thus, we set $\{b_1^p, b_2^p, b_3^p, b_4^p\} = \{2,3,4,5\}$ and $\{b_1^o, b_2^o, b_3^o, b_4^o\} = \{3,4,5,6\}$ for ResNet18 and MobileNetV2, which are the empirical optimal allocation under our settings. Lastly, instead of training from scratch, we fine-tune a pre-trained model with fewer data and a smaller batch size. After that, we call the modified version conventional MP and use it as our baseline.

### A. Comparison between DR/MP Co-design with DBS and Prior Works

In this section, we would validate the effectiveness of DR/MP co-design with dynamic bit-width searching (DBS). First, we apply the selection metric mentioned in Section III.A to greedily select channels for DR. After that, we operate the learnable MP scheme in Section IV.C on the remaining channels to achieve layer-wise bit-width assignments. The above-mentioned system is the special case with $G = 2$ in Fig. 6, where DR and MP are operated on group 2 and group 1, respectively. The corresponding accuracies of DR/MP systems under different $T_{\text{greedy}}$ are specified in Table I, where the percentages in parentheses are the increment over the results of the w/o co-design version. For the co-design system, group partition would be dynamically updated during the searching phase while that is fixed in models w/o co-design. Moreover, we set $\{b_1^p, b_2^p, b_3^p, b_4^p\}$ of the systems w/o DBS to $\{3,4,5,6\}$ and $\{5,6,7,8\}$ for ResNet18 and MobileNetV2 for fair comparison. We ensure that the final bit-width allocation acquired by the system w/ DBS is within the search space of the system w/o DBS. In other words, system w/o DBS is possible to end up with the identical bit-width allocation as the system w/ DBS. Since we aim to evaluate the benefits of continuous bit-width reduction in this section, this setting is to remove the influence of the other benefit of DBS, i.e., enlarging the search space. The influence on search space is discussed later in Section V.C.

We can observe the naive concatenation of DR/MP without co-design results in terrible accuracy due to the ignorance of mutual interference. Conversely, models with co-design can substantially improve the accuracy, validating the necessity of joint optimization. Next, we visualize the layer-wise bit-width allocation of the co-design framework w/ and w/o DBS in Fig. 9. For ResNet18, though the two strategies end up with similar bit-width allocation, the accuracy without DBS drops significantly. Since Greedy DR removes most redundancy, the remaining channels are imperative to maintain model performance and vulnerable to aggressive quantization. Therefore, sharp bit-width conversion would lead to catastrophic information loss. Specifically, the system w/o DBS undergoes quantization started from low bit-widths, i.e., $\{3,4,5,6\}$. However, the bit-widths for exploration of system w/ DBS begins from $\{6,7,8\}$, and then decreases to $\{5,6,7\}$, $\{4,5,6\}$, $\{3,4,5\}$, and so forth. With DBS, our model learns to capture information with relatively high bit-widths during the early stage. After training the model well, we reduce the bit-widths to increase the quantization strength gradually. With continuous bit-width switching, the proposed DBS technique effectively solves the above problem, enabling the model to preserve accuracy with low bit-width allocation. As for MobileNetV2, a high percentage of channels are removed with low $T_{\text{greedy}}$. Thus, the bit-width values are relatively high due...
to low redundancy after DR, concentrated between 6 and 8. Consequently, this phenomenon alleviates the impact of sharp conversion on MobileNetV2 and enables it to maintain better accuracy. However, models with DBS can still obtain improvements, especially for those with small $T_{\text{Greedy}}$. Since under similar bit-width distribution, models with fewer remaining channels after DR are more fragile, which require gradual bit-width reduction to restore accuracy. Besides continuous bit-width reduction, another benefit of DBS is to enlarge the search space to automatically reach optimal bit-width allocation, which would be elaborated in Section V.C.

Next, we plot the Pareto frontier of our co-design system under different $T_{\text{Greedy}}$ to compare performance with conventional MP, as shown in Fig. 10. In these figures, each line denotes performance under certain $T_{\text{Greedy}}$, and the points at the same line are obtained by modifying the values of $p$. Since lower $T_{\text{Greedy}}$ stands for more aggressive DR, reducing $T_{\text{Greedy}}$ makes the frontier shift left. As for the values of $p$, it determines the strength of quantization. Thus, larger $p$ leads to lower average bit per value, and the accuracy also drops more. Compared with conventional MP, our co-design system gains better performance under different parameter settings. In summary, the co-design system with DBS reaches 68.3%/70.4% with average 2.32/2.66 bits per value on ResNet18/MobileNetV2, which reduces 32%/51% (-1.07/-2.80 bits) memory access of conventional MP with higher accuracy.

B. Comparison with Prior Works

To demonstrate the improvement of the compression ratio in our joint optimization system, we validate its effectiveness by comparison with Greedy DR and conventional MP. We operate different $T_{\text{Greedy}}$ on proposed joint optimization system of group-wise DR and MP, and the model performance as well as the group partition visualization are shown in Fig. 11. From these results, we can observe a similar phenomenon as mentioned in Section III.B.2, where low $T_{\text{Greedy}}$ and high $p$ result in aggressive DR and quantization, respectively. As for the number of groups, the model with large $G$ implies mixed-precision with fine granularity, and therefore improves the compressibility. We implement our system with $G = 2, 3, 4$, where $G = 2$ is equal to the case of DR/MP co-design mentioned in Section V.A. From these experiment results, we find those simulations with $G = 4$ perform best under different $T_{\text{Greedy}}$, which accord with our previous assumption. Furthermore, we compare our proposed joint optimization system with the manually-designed method [19], and the corresponding results are shown in Table II. From this table, we can observe our system reaches smaller average bits per value under similar accuracy than [19], indicating the effectiveness of proposed AGP mechanism.

In summary, the joint optimization system with group-wise DR and MP reaches 69.03%/70.73% with average 2.16/2.61 bits per value on ResNet18/MobileNetV2, which reduces 55%/63% (-2.62/-4.52 bits) memory access of conventional MP under similar or higher accuracy, respectively.

C. Visualization and Analysis of Joint Compression Policy

To further analyze the group-wise mechanism, we visualize the group partition of ResNet18 with $T_{\text{Greedy}} = 0.53$ and MobileNetV2 with $T_{\text{Greedy}} = 0.34$ under $G = 4$ in Fig. 11(d).
and (h), respectively. Since channels belonging to group 4 are those least significant and would be removed finally, the distribution of model redundancy can be inferred from the group partition. For ResNet18, pruned channels are concentrated in the shallow layers, similar to the distribution under Greedy DR in Fig. 5(a). Since the activation sizes in shallow layers are larger than in deeper layers, they are easier to obtain a small selection metric and assigned to group 4. As for MobileNetV2, aggressive dimension reduction occurs at depthwise layers, which is similar to the distribution for edge devices in [20]. The result implies more redundancy in depthwise layers than in pointwise layers, and our system is able to perceive the difference and allocate suitable bit-width respectively. Moreover, the number of channels assigned to group 2 is relatively smaller compared with other groups in MobileNetV2, which implies $G = 3$ is enough to represent the channel-wise variety. The phenomenon explains the comparable performance under $G = 3$ and $G = 4$ in Fig. 11(e)–(g).

To analyze how the compression policy changes during the searching phase, we further present the group partition and bit-width distribution during the early, middle, and final searching phases of ResNet18 in Fig. 12 and Fig. 13, respectively. Since the searching phase is saturated after 15 epochs, we demonstrate the results of epoch 1, 7, and 15. Since the model weights are updated during the searching phase, the activation values and the corresponding channel sensitivity also change accordingly. Therefore, as mentioned in the AGP mechanism, we recalculate the selection metric during each iteration of searching phase to keep track of current model state. From Fig. 12, we can perceive those channels assigned to group 4 are different at the three stages. Since members of group 4 are selected by the selection metric, the different distribution infers the channel sensitivity changes during the searching phase. For as the other groups, they also differ from the initialization, showing the AGP mechanism can facilitate group updating.

As for mixed-precision quantization, we demonstrate the group-wise bit-width distribution in Fig. 13. Since channels assigned to group 4 are directly pruned, it is unnecessary to specify their bit-width. In epoch 1, the bit-width distribution is pre-defined in $\{6, 7, 8\}$. After training for more epochs, we can notice the bit-width distribution declines continuously. For the final bit-width assignments, as shown in Fig. 13(c), the distribution varies dramatically among different groups and layers, ranging from 3 to 7 bits. This phenomenon indicates group-wise mechanism is effective in further exploiting redundancy among channels, allowing various bit-width allocations in a certain layer. Moreover, we can observe that optimal bit-width ranges differently from layer to layer, which is neglected in previous works with hand-crafted searching branches. With the proposed dynamic bit-width searching, our system enlarges the search space and automatically reaches the suitable bit-width allocation.

### D. Analysis of Training and Inference Computation Overhead

From here, we have demonstrated the effectiveness of the proposed joint optimization system. In this section, we further analyze the additional computation overhead during training and inference, respectively.

As mentioned in Section III.B and Section V, we just use 50,000 out of 1,281,167 training data of ImageNet. Moreover, 21 epochs (15 for searching and 6 for fine-tuning) and 36 epochs (15 for searching and 6 for fine-tuning) are used to train ResNet18 and MobileNetV2, respectively. As for the pre-training process, however, models are trained by the whole dataset for thousands of epochs. Take ResNet18 as an example, it requires 600,000 iterations, roughly 2,344 epochs, for training from scratch. Therefore, the induced computation overhead of

![Fig. 12. The visualization of group partition of ResNet18 with $T_{\text{Greedy}} = 0.53$, $p = 0.1$, $G = 4$ on (a) epoch 1, (b) epoch 7, and (c) epoch 15.](image1.png)

![Fig. 13. Dynamic group-wise bit-width distribution of ResNet18 with $T_{\text{Greedy}} = 0.55$, $p = 0.1$, $G = 4$ on (a) epoch 1, (b) epoch 7, and (c) epoch 15.](image2.png)

<table>
<thead>
<tr>
<th>Model</th>
<th>$G$</th>
<th>$\text{Acc.}$</th>
<th>$\text{GPU Times}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet18</td>
<td>2</td>
<td>68.45%</td>
<td>2h10min</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>68.97%</td>
<td>2h20min</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>69.03%</td>
<td>2h22min</td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>2</td>
<td>69.89%</td>
<td>5h10min</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>70.58%</td>
<td>6h04min</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>70.73%</td>
<td>6h30min</td>
</tr>
</tbody>
</table>
our system is: $\frac{50,000}{1,281,167} \times 100\% = 3.5 \times 10^{-2} \%$ of the pretrained model, which is negligible. We demonstrate the required GPU hours with single Tesla V100 GPU for ResNet18 and single NVIDIA RTX A5000 GPU for MobileNetV2 in Table III. Models with $G$ larger than 2 require longer searching time due to the K-means step and group update of AGP mechanism, while models with $G = 2$ are just partitioned by the selection metric. While the pre-training process requires days of training, our optimization can be done within a tolerable timing cost.

Next, we further analyze the additional computation overhead during inference. Due to the negligible overhead of group concatenation, we focus on the cost induced by PCA and its inverse transformation. As mentioned in [8], the pre-computed PCA matrix can be folded into convolution and BN operations. Thus, the extra computation comes from the inverse transformation. We calculate the relative computation under different DR threshold $T_{\text{Greedy}}$, as shown in Table IV. The relative computation can be specified as:

$$\frac{C_{\text{Original}} + C_{\text{PCA}}}{C_{\text{Original}}} \times 100\% \quad (18)$$

$$\frac{C_{\text{Folded}} + C_{\text{Inverse}}}{C_{\text{Original}}} \times 100\% \quad (19)$$

where $C_{\text{Original}}$, $C_{\text{PCA}}$, $C_{\text{Inverse}}$, and $C_{\text{Folded}}$ stand for the number of multiplication operations of original model, PCA transformation, inverse transformation, and after folding PCA transformation into convolution. When $T_{\text{Greedy}}$ is set to 1, it means no dimension reduction and the overhead is purely induced by the inverse transformation. However, as the threshold decreases, dimension reduction can reduce the matrix size and make the computation of our method even less than the original model. Consequently, our method won’t introduce much computation overhead during inference.

From the simulation results shown above, the proposed joint optimization system successfully compress activation with little training and inference overheads. While prior works require thousands of GPU hours of training, our methods can efficiently search a good trade-off between model accuracy and compressibility in less than 7 GPU hours.

### VI. Conclusion

In this work, we propose a joint optimization system of dimension reduction and mixed-precision quantization, which has specialized group-wise DR and MP co-design for activation compression. By adopting the proposed automatic group partition mechanism, our system takes both accuracy and compressibility into account and enables dynamically-updated partition. Moreover, to preserve model accuracy under aggressive quantization, we design a learnable MP scheme with dynamic bit-width searching, achieving continuous bit-width conversion and enlarging the search space. The proposed joint optimization system just takes few epochs to learn the optimal strategy for DR and MP, dramatically enhancing the efficiency compared with time-consuming iterative searching. Finally, the experimental results show that our method achieves 69.03%/70.73% with average 2.16/2.61 bits per value on ResNet18/MobileNetV2.

### VII. References


Yu-Shan Tai received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2021. She is currently pursuing the Ph.D. degree in electronics engineering with National Taiwan University, Taipei, Taiwan. Her research interests include deep learning applications and neural network compression.

Cheng-Yang Chang received the B.S. degree in electronic engineering from National Taiwan University, Taipei, Taiwan, in 2019. He is currently pursuing the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan. His research interests include hyperdimensional computing, multi-task learning, and collaborative learning.

Chieh-Fang Teng received his B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2017 and Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University in 2022. He is currently a hardware architect for AI system at MediaTek Inc., Taiwan. His research interests are in the areas of VLSI architecture for DSP, deep learning, neural network compression, and machine learning assisted wireless communication systems design.

Yi-Ta Chen received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 2017. He is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University. His research interests include the machine learning engine for affective computing, biosignal processing and feature extraction for affective computing, and SW/HW co-design for SDN data plane.

An-Yeu (Andy) Wu (Fellow, IEEE) received the B.S. degree in electrical engineering from National Taiwan University (NTU), Taipei, Taiwan, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland of College Park, MD, USA, in 1992 and 1995, respectively. In 2000, he joined the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering (GIEE), NTU, as a Faculty Member, where he is currently a Distinguished Professor and also has been the Director of GIEE since 2016. His research interests include VLSI architectures for signal processing and communications and adaptive/multi-rate signal processing. He has published more than 250 refereed journal and conference papers in the above research areas, together with five book chapters and 20 granted US patents. From 2007 to 2009, he was on leave from NTU and served as the Deputy General Director of the SoC Technology Center (STC), Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan. Dr. Wu was elevated to IEEE Fellow for his contributions to DSP algorithms and VLSI designs for communication IC/SoC in 2015. From 2016 to 2019, he served as Director of the Graduate Institute of Electronics Engineering (GIEE), National Taiwan University.