SQUANT: ON-THE-FLY DATA-FREE QUANTIZATION VIA DIAGONAL HESSIAN APPROXIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization of deep neural networks (DNN) has been proven effective for compressing and accelerating DNN models. Data-free quantization (DFQ) is a promising approach without the original datasets under privacy-sensitive and confidential scenarios. However, current DFQ solutions degrade accuracy, need synthetic data to calibrate networks, and are time-consuming and costly. This paper proposes an on-the-fly DFQ framework with sub-second quantization time, called SQuant, which directly optimizes weight parameters and achieves higher accuracy without any dataset. With the theoretical analysis of the second-order information of DNN task loss, we decompose and approximate the Hessian-based optimization objective into three diagonal items, which have different areas corresponding to three dimensions of weight tensor: element-wise, kernel-wise, and output channel-wise. Then, we progressively optimize the objective by composing these three diagonal Hessian approximations and propose a novel data-free optimization objective, Constrained Absolute Sum of Error (or CASE in short), which surprisingly has no need of any datasets or even awareness of network architecture. We also design an efficient algorithm without back-propagation to further reduce the computation complexity of the objective solver. Finally, without finetuning and synthetic datasets, SQuant accelerates the data-free quantization process to a sub-second level with $> 30\%$ accuracy improvement over other data-free PTQ works, with the evaluated models under 4-bit quantization.

1 INTRODUCTION

With the widespread application of DNN, more and more DNN models are deployed on both computation-constrained and memory-constrained environments, e.g., smartphones, IoT devices, and self-driving cars. The desire for lightweight and energy-efficient DNN deployment solutions is increasing. Quantization is one of the most promising techniques to reduce computational time and memory consumption simultaneously. Quantization converts weights and activations to their lower bit formats. For a model with 32-bit precision, 8-bit quantization could decrease the memory overhead by $4\times$ and reduce the multiplication computation cost by $16\times$. There are two kinds of quantization: Post-Training Quantization (PTQ) (Banner et al., 2018b; Choukroun et al., 2019; Zhao et al., 2019; Nagel et al., 2020) and Quantization-Aware Training (QAT) (Gupta et al., 2015; Jacob et al., 2018; Wang et al., 2019; Zhuang et al., 2021). QAT requires to simulate quantization in the training process, which invokes time-consuming retraining and hyper-parameter tuning. In contrast, PTQ directly quantizes well-trained models without retraining. However, they still need training dataset to calibrate (Nagel et al., 2020) quantized models, often unavailable due to privacy and security issues, such as medical and confidential data.

Data-free quantization (DFQ) has recently been presented as a promising way to quantize models without original dataset (Banner et al., 2018a; Nagel et al., 2019; Cai et al., 2020; Zhang et al., 2021; Xu et al., 2020; Liu et al., 2021; Qin et al., 2021; Choi et al., 2020). From a deployment perspective, DFQ is the most attractive quantization method since we can apply it to any trained models as a black box post-processing step. However, current DFQ methods cannot achieve high accuracy and fast processing time simultaneously. Traditionally, DFQ (Nagel et al., 2019) uses rounding with weight equalization and bias correction, leading to the rounding-to-nearest strategy. Such a strategy causes significant accuracy loss, especially in low-bit settings. To bridge the accuracy gap between data-free and data-driven quantization, a series of data-generative DFQ methods are proposed. They

use gradient-based methods to generate fake datasets for trained models. With synthetic data, they can employ a data-driven calibration and fine-tuning strategy to improve accuracy. However, data generation is time-consuming for many gradient-based methods since they require multiple iterations to generate each input. They often spend hours to generate a calibration dataset and fine-tune the network (Xu et al., 2020; Liu et al., 2021; Zhang et al., 2021).

To solve this dilemma, we propose SQuant, a fast and accurate quantization DFQ framework for convolutional neural networks, employing the constrained absolute sum of error (CASE) of weights as the rounding metric. By leveraging Hessian information of network loss due to quantization, we propose a novel diagonal Hessian approximation, which decomposes the optimization objective into three data-free sub-objectives: element-wise, kernel-wise, and output channel-wise corresponding to the dimensions of weight tensor. These approximate objectives eliminate the requirement of data generation. We also propose a SQuant progressive algorithm with linear complexity to solve the optimization objective, which further accelerates DFQ time into a sub-second level. For example, SQuant only needs an average of 4 ms and 76 ms for quantizing a layer and the overall network of ResNet18, respectively. SQuant can execute on inference-only devices on the fly with constrained computation and memory resources as it does not require back-propagation nor fine-tuning. Compared with SOTA DFQ methods, SQuant achieves higher accuracy on all evaluated models under the 4/6/8-bit settings. SQuant only introduces 0.1% accuracy loss on average under the 8-bit setting. Under lower bit precision, the advantage of SQuant further expands. SQuant only introduces 1.8% accuracy loss on average under the 6-bit setting. Under the 4-bit setting, SQuant can achieve more than 30% accuracy improvement compared with other data-free PTQ methods. In a word, SQuant pushes the accuracy and processing time of DFQ to a new frontier.

2 PRELIMINARIES

2.1 NOTATIONS

We specifically use x, y and w to denote the input, output and weight variables, respectively. Constant and scalar are denoted by italic letters, e.g., c, M. Column vector and flattened matrix are denote by bold lowercase letters, e.g., $\mathbf{w}$, and matrices (or tensors) are represented by uppercase letters, e.g., $\mathbf{W}$. The subscript and superscript can further represent the element indices and the layer of network, respectively, e.g., $\mathbf{W}^{\ell}_{i,j}$. $\mathbb{E}[\cdot]$ denotes the expectation operator, and the network loss function is represented by $\mathscr{L}(\cdot)$.

For convenience in this paper, we call the row of FC (fully connected layer) weights as output channel and the column of FC as input channel, which are the counterparts to Conv (convolution layer) weight. We use M, N, and K to denote output channel size, input channel size and kernel height $\times$ kernel width respectively. Specifically, FC has the shape of $(M, N, 1)$.

2.2 QUANTIZATION

Most previous works adopt the rounding-to-nearest approach for quantizing deep neural network weights by rounding weight elements w to the nearest quantization grid values with a fixed-point data type. The quantization and de-quantization for weight $\widehat{w}$ can be described as $\widehat{w} = s \cdot \text{clip}(\lfloor \frac{w}{s} \rceil, min, max)$, where s denotes the quantization scale parameter and, min and max are the lower and upper thresholds for the clipping function $\text{clip}(\cdot)$. The operator $\lfloor \cdot \rceil$ represents the rounding-to-nearest. The rounding-to-nearest approach intuitively adopts the metrics of the mean squared error (MSE) of weights.

2.3 HESSIAN-BASED OPTIMIZATION FOR NEURAL NETWORKS

The Hessian-based approach is one of the most promising optimizations to further improve quantization (Dong et al., 2019b;a; Nagel et al., 2020; Shen et al., 2020; Qian et al., 2020; Wu et al., 2020; Hubara et al., 2020; Li et al., 2021; Yao et al., 2021) or pruning (Yu et al., 2021) performance. Some of those works exploit the Hessian matrix to approximate loss degradation due to the quantization

perturbation $\Delta\mathbf{W}$ by

$$\mathbb{E}[\mathscr{L}(\mathbf{X},\mathbf{Y},\mathbf{W}+\Delta\mathbf{W})-\mathscr{L}(\mathbf{X},\mathbf{Y},\mathbf{W})]\approx\mathbb{E}[\Delta\mathbf{W}\cdot\mathbf{g}^{\mathbf{W}}+\frac{1}{2}\Delta\mathbf{W}\cdot\mathbf{H}^{\mathbf{w}}\cdot\Delta\mathbf{W}^{T}], \tag{1}$$

where the equation comes from second-order Taylor series expansion, $\mathbf{g}^{\mathbf{w}}$ is the gradient and $\mathbf{H}^{\mathbf{w}}$ is the full network Hessian matrix (**`H-N`**) w.r.t. $\mathbf{w}$. Since a well-trained model has already converged, the gradient term will be close to 0 and thus can be safely ignored. However, computing $\mathbf{H}^{\mathbf{w}}$ is infeasible because of the large memory overhead and computation complexity. To tackle this problem, we approximate $\mathbf{H}^{\mathbf{w}}$ as a layer-wise Hessian matrix $\mathbf{H}^{\mathbf{w}^{\ell}}$ (**`H-L`**) under the assumption of cross-layer independence (Dong et al., 2017; Nagel et al., 2020), i.e., $\mathbf{H}^{\mathbf{w}^{\ell}}=\mathbf{x}^{\ell}\mathbf{x}^{\ell^{T}}\otimes\nabla^{2}_{\mathbf{y}^{\ell}}\mathscr{L}$, where $\otimes$ denotes Kronecker product of two matrices, $\nabla^{2}_{\mathbf{y}^{\ell}}\mathscr{L}$ is the Hessian of the task loss w.r.t. $\mathbf{y}^{\ell}$.

For the m-th output channel of Conv or FC, $\mathbf{H}^{\mathbf{w}^{\ell}}$ can be approximatively simplified into (output) channel-wise (Nagel et al., 2020; Yu et al., 2021; Wu et al., 2020; Qian et al., 2020), called **`H-C`**:

$$\mathbf{H}^{\mathbf{w}^{\ell}_{m}}\approx\nabla^{2}_{\mathbf{y}^{\ell}}\mathscr{L}_{m,m}\cdot\mathbf{x}^{\ell}\mathbf{x}^{\ell^{T}}=l_{m}\cdot\mathbf{x}^{\ell}\mathbf{x}^{\ell^{T}}, \tag{2}$$

where $\nabla^{2}_{\mathbf{y}^{\ell}}\mathscr{L}$ is approximately a diagonal matrix. Then the final optimization objective is

$$\Delta\hat{\mathbf{W}}^{\ell}_{m,:}=\underset{\Delta\mathbf{W}^{\ell}_{m,:}}{\arg\min}\quad\Delta\mathbf{W}^{\ell}_{m,:}\,\mathbb{E}[\mathbf{H}^{\mathbf{w}^{\ell}_{m}}]\Delta\mathbf{W}^{\ell}_{m,:}{}^{T} \tag{3}$$

$$=\underset{\Delta\mathbf{W}^{\ell}_{m,:}}{\arg\min}\quad\Delta\mathbf{W}^{\ell}_{m,:}\,\mathbb{E}[\mathbf{x}^{\ell}\mathbf{x}^{\ell^{T}}]\Delta\mathbf{W}^{\ell}_{m,:}{}^{T} \tag{4}$$

$$=\underset{\Delta\mathbf{W}^{\ell}_{m,:}}{\arg\min}\,\mathbb{E}[(\Delta\mathbf{W}^{\ell}_{m,:}\mathbf{x}^{\ell})^{2}]. \tag{5}$$

Each sub-problem deals with a single output channel $\Delta\mathbf{W}^{\ell}_{m,:}$.

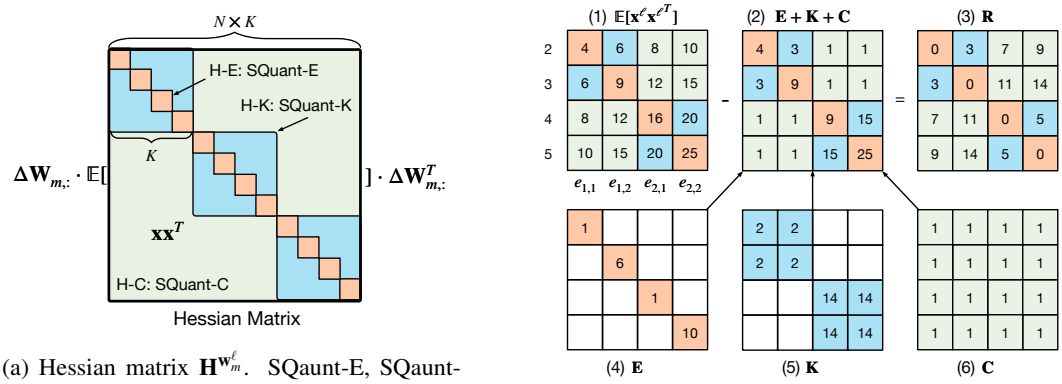
3 METHODOLOGY

3.1 DIAGONAL HESSIAN APPROXIMATION

Although a good quantization strategy can be obtained via minimizing Eq. (5), which is the mean square error (MSE) between preactivation produced from original and quantized weights for each output channel. It is an NP-hard combinatorial optimization problem in which even approaching an acceptable local minimum requires significant efforts. Previous data-free quantization methods adopt the rounding-to-nearest mechanism, which essentially assumes $\mathbb{E}[\mathbf{x}^{\ell}\mathbf{x}^{\ell^{T}}]$ is a diagonal matrix shown in Fig. 1a **`H-E`**. It just rounds the weight vector to the nearest representable quantization grid value in a fixed-point grid, so that it is completely multiplication and data-free. However, as many previous works pointed out (Nagel et al., 2020), rounding-to-nearest is not optimal because it ignores most elements of the matrix $\mathbb{E}[\mathbf{x}^{\ell}\mathbf{x}^{\ell^{T}}]$ except for the diagonal.

To avoid the combinatorial optimization problem and utilize the Hessian information, we propose **SQuant** framework, which progressively optimizes Eq. (5) in three diagonal Hessian matrices corresponding to the dimensions of weight tensor in DNN. Under the data-free scenario, we design three approximate sub-objectives shown in Fig. 1a:

- **SQuant-E (a.k.a. Rounding)** for element-wise optimization covers the diagonal elements of $\mathbf{H}^{\mathbf{w}^{\ell}_{m}}$ (**`H-E`**).
- **SQuant-K** for kernel-wise optimization covers the diagonal blocks of $\mathbf{H}^{\mathbf{w}^{\ell}_{m}}$ (**`H-K`**), which is a superset of **`H-E`**.
- **SQuant-C** for output channel-wise optimization covers the whole $\mathbf{H}^{\mathbf{w}^{\ell}_{m}}$ (**`H-C`**), which is a superset of **`H-E`** and **`H-K`**.



(a) Hessian matrix $\mathbf{H}^{\mathbf{w}^\ell_m}$. SQaunt-E, SQaunt-K, and SQuant-C are three approximation sub-objectives, which cover the area of **`H-E`**, **`H-K`** and **`H-C`** respectively.

(b) An example of sub-objectives approximation and composition with $\mathbf{x}^{\ell^T} = [2,3,4,5]$.

Figure 1: Hessian matrix and objective approximation.

In this work, we propose a new approximation of the Hessian matrix to cover non-diagonal elements and decompose Eq. (4) into three items corresponding to the three weight dimensions illustrated in Fig. 1a, as shown below,

$$\mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}] = \mathbf{E} + \mathbf{K} + \mathbf{C} + \mathbf{R} \approx \mathbf{E} + \mathbf{K} + \mathbf{C}, \tag{6}$$

where $\mathbf{R}$ is the approximate error matrix, $\mathbf{C} = c_m \mathbf{J}_{NK}$,

$$\mathbf{K} = \begin{bmatrix} k_1\mathbf{J}_K & & \\ & \ddots & \\ & & k_N\mathbf{J}_K \end{bmatrix}, \text{ and } \mathbf{E} = \begin{bmatrix} e_{1,1} & & \\ & \ddots & \\ & & e_{N,K} \end{bmatrix}.$$

Here, $\mathbf{J}_{NK}$ is an all-one matrix with dimension $NK = N \times K$ and c_m is a constant value for m-th output channel. $\mathbf{K}$ is a diagonal block matrix, where $\mathbf{J}_K$ represents an all-one matrix with dimension K. The n-th diagonal block corresponds to n-th kernel in convolution and has the same constant k_n. $\mathbf{E}$ is a diagonal matrix whose elements are $e_{n,i}$, which are also constant values corresponding to diagonal i-th element of n-th kernel in $\mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}]$.

Fig. 1b illustrates an example about the approximation and composition of the three sub-objectives in Eq. (6). For each inference, the elements in a kernel scan the same feature map. As a result, the corresponding $\mathbf{x}$ has nearly the same expectation value in the center area, with a small perturbation in the marginal area due to the padding mechanism. Therefore, kernel-wise elements achieve a pretty low approximate error based on convolution. Fig. 1b (3) matrix shows the approximate error matrix, $\mathbf{R} = \mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}] - (\mathbf{E} + \mathbf{K} + \mathbf{C})$, which is zero for diagonal elements and has low errors for the diagonal kernel block and higher for the non-diagonal elements. Substituting Eq. (6) into Eq. (4) yields

$$\Delta\mathbf{W}^\ell_{m,:}\, \mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}] \Delta\mathbf{W}^{\ell\ T}_{m,:} \approx \sum_{n,i} e_{n,i} \Delta\mathbf{W}^{\ell\ 2}_{m,n,i} + \sum_n k_n \Delta\mathbf{W}^\ell_{m,n,:} \mathbf{J}_K \Delta\mathbf{W}^{\ell\ T}_{m,n,:} + c_m \Delta\mathbf{W}^\ell_{m,:} \mathbf{J}_{NK} \Delta\mathbf{W}^{\ell\ T}_{m,:}. \tag{7}$$

3.2 Data-free Optimization Objective

For any $\mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}]$, we can always find a decomposition that satisfies $e_{n,i}, k_n, c_m > 0$ and present decomposition method in Appendix A.1. Since that all constants and all $\Delta\mathbf{W}\mathbf{J}\Delta\mathbf{W}^T$ items are > 0 in Eq. (7), it can be easily found that there are no training samples needed to minimize,

$$\underset{\Delta\mathbf{W}^\ell_{m,:}}{\arg\min} \quad \sum_{n,i} (\Delta\mathbf{W}^\ell_{m,n,i})^2 + \sum_n \Delta\mathbf{W}^\ell_{m,n,:} \mathbf{J}_K \Delta\mathbf{W}^{\ell\ T}_{m,n,:} + \Delta\mathbf{W}^\ell_{m,:} \mathbf{J}_{NK} \Delta\mathbf{W}^{\ell\ T}_{m,:} \tag{8}$$

$$= \underset{\Delta\mathbf{W}^\ell_{m,:}}{\arg\min} \quad \sum_{n,i} (\Delta\mathbf{W}^\ell_{m,n,i})^2 + \sum_n (\sum_i \Delta\mathbf{W}^\ell_{m,n,i})^2 + (\sum_{n,i} \Delta\mathbf{W}^\ell_{m,n,i})^2. \tag{9}$$

Next, we transform Eq. (9) into a discrete optimization problem based on quantization and introduce how to optimize and compose the three approximate sub-objectives in order. Without loss of generality, we assume all weights has been scaled with the scale parameter s_m for $\mathbf{W}^\ell_{m,:}$.

SQuant-E For element-wise item, which is the first item in Eq. (9), the problem is reduced into

$$\underset{\Delta \mathbf{W}^{\ell}_{m,:}}{\arg\min} \sum_{n,i} (\Delta \mathbf{W}^{\ell}_{m,n,i})^2 \Leftrightarrow \forall \Delta \mathbf{W}^{\ell}_{m,n,i},\ |\Delta \mathbf{W}^{\ell}_{m,n,i}| \leq r_e = 0.5, \tag{10}$$

which is essentially the rounding method when $r_e = 0.5$. Rounding does not introduce any approximate error and has $O(1)$ complexity for each element in weight. However, rounding drops all non-diagonal Hessian information, and it is unlikely to find an optimal global solution.

SQuant-K For kernel-wise sub-objective, which maps to the second item in Eq. (9), we have,

$$\underset{\Delta \mathbf{W}^{\ell}_{m,:}}{\arg\min} \sum_{n} (\sum_{i} \Delta \mathbf{W}^{\ell}_{m,n,i})^2 \Leftrightarrow \forall \Delta \mathbf{W}^{\ell}_{m,n,i},\ |\sum_{i} \Delta \mathbf{W}^{\ell}_{m,n,i}| \leq r_k = 0.5, \tag{11}$$

where $|\sum_i \Delta \mathbf{W}^{\ell}_{m,n,i}|$ is the Constrained Absolute Sum of Error (CASE) of each kernel-wise weight matrix in the Conv layer and $r_k = 0.5$ because of the discrete quantization. **SQuant** is based on the insight of **S**um of **S**igned error instead of the accumulation of absolute (or squared, unsigned) error.

Relaxation and Flipping Obviously, $r_e = 0.5$ is against $r_k = 0.5$ because rounding ($r_e = 0.5$) only guarantees the upper-bound $\dot{r}_k = 0.5 \cdot K$ for SQuant-K. We adopt two methods, relaxation and flipping, to reduce the kernel CASE. Some elements need to relax the constraint r_e to a larger number such as 1.0, since that rounded elements can be flipped (Nagel et al., 2020) (from rounding up to rounding down and vice versa) with ± 1 mutation. Specifically, we need flip $k = \lfloor \text{CASE} \rceil$ elements, whose perturbation has the same sign as $\sum_i \Delta \mathbf{W}^{\ell}_{m,n,i}$ to reduce CASE ≤ 0.5. For example, if $\sum_i \Delta \mathbf{W}^{\ell}_{m,n,i} = 3.2$, we need to flip 3 elements with positive perturbation (rounding up) to negative (rounding down).

Top-k Perturbation However, any k elements can satisfy Eq. (11). Fortunately, based on Eq. (10), SQuant-K can only select k elements with the top-k largest perturbation because they will have the smallest perturbation after flipping and achieve the minimum $\sum_{n,i} (\Delta \mathbf{W}^{\ell}_{m,n,i})^2$ under the relaxation of SQuant-K. We will prove the top-k algorithm is the only solution for the SQuant-E&K in Appendix B.1 to simultaneously satisfy the first two items of Eq. (9).

SQuant-C For output channel-wise item, the third item in Eq. (9), we have the CASE of output channel,

$$\underset{\Delta \mathbf{W}^{\ell}_{m,:}}{\arg\min} (\sum_{n,i} \Delta \mathbf{W}^{\ell}_{m,n,i})^2 \Leftrightarrow \forall \Delta \mathbf{W}^{\ell}_{m,n,i},\ |\sum_{n,i} \Delta \mathbf{W}^{\ell}_{m,n,i}| \leq r_c = 0.5. \tag{12}$$

Similarly, SQuant-C also need to relax $r_k = 1.0$ and "flip" the "SQuanted" kernel after SQuant-K. Notice that we can flip only one element for a kernel to satisfy the constraint $r_k = 1.0$ and $r_c = 0.5$.

Flipping Set Finally, we need to work out a flipping set $\mathbf{f}_m$ to satisfy the overall objective Eq. (9) by composing three optimization sub-objectives in order (SQuant-E $\rightarrow$ SQuant-K $\rightarrow$ SQuant-C) with constraints relaxation. After optimization, the $\mathbf{f}_m$ will be

$$\forall (m,n,i) \notin \mathbf{f}_m,\ |\Delta \mathbf{W}^{\ell}_{m,n,i}| \leq 0.5; \quad \forall (m,n,j) \in \mathbf{f}_m,\ 0.5 \leq |\Delta \mathbf{W}^{\ell}_{m,n,j}| < 1.0, \tag{13}$$

where $\mathbf{f}_m$ is the index set of flipped elements for m-th output channel. In the next section, we will introduce an efficient SQuant algorithm with a linear computation complexity to find a proper $\mathbf{f}_m$ based on Eq. (9).

3.3 Progressive SQuant Algorithm

Progressive Algorithm We design a progressive algorithm solver illustrated in algorithm 1 from SQuant-E to SQuant-K and finally SQuant-C to meet our stated optimization objective, i.e., reducing the CASE of weight perturbation. We start by rounding the weight and updating its perturbation $\Delta \mathbf{W}^{\ell}$ to satisfy $r_e = 0.5$ (line 4-5). Then we run the SQuant-K (line 6) to relax r_e to $r_e = 1.0$ and satisfy $r_k = 0.5$ and update kernel perturbation (line 7). The follow-up SQuant-C (line 8) further relaxes r_k to $r_k = 1.0$ and satisfy $r_c = 0.5$. Finally, we quantized the weight (line 9).

The key insight of the progressive algorithm is to gradually calibrate the deviation from the optimal global solution introduced by the fine-grained diagonal sub-objective. To calibrate the SQuant-E, SQuant-K flips certain rounded elements, then SQuant-C flips SQuanted kernels after SQuant-K.

SQuant-K Flip Algorithm The goal of flip algorithm is to find a proper element set $\mathbf{f}$ to flip and minimize the CASE depicted in algorithm 2. First, SQuant-K needs to compute the accumulated

Algorithm 1: Progressive SQuant Algorithm.

Input: Weight tensor $\mathbf{W}$ of layer ℓ, scale factor $\mathbf{s}$ of layer ℓ.
Output: Quantized weight tensor $\mathbf{C}$ of layer ℓ.

1 **foreach** $m \in [1,..,M]$ **do** `// SQuant-C: SQuant` M `output channels.`
2 **foreach** $n \in [1,..,N]$ **do** `// SQuant-K: SQuant` N `kernels.`
3 **foreach** $i \in [1,...,K]$ **do** `// SQuant-E: Round` K `elements.`
4 $\mathbf{E}_{m,n,i} = \lfloor \mathbf{W}_{m,n,i}/\mathbf{s}_m \rceil$ `// Scale and SQuant-E.`
5 $\Delta\mathbf{E}_{m,n,i} = \mathbf{E}_{m,n,i} - \mathbf{W}_{m,n,i}$ `// Element perturbation.`
6 $\mathbf{K}_{m,n,:} = \text{SQuantFlip}(\mathbf{E}_{m,n,:}, \Delta\mathbf{E}_{m,n,:})$`// SQuant-K.`
7 $\Delta\mathbf{K}_{m,n,:} = \text{UpdatePerturbation}(\Delta\mathbf{E}_{m,n,:})$ `// Kernel perturbation.`
8 $\mathbf{C}_{m,:} = \text{SQuantFlip}(\mathbf{K}_{m,:}, \Delta\mathbf{K}_{m,:}) \cdot \mathbf{s}_m$`// SQuant-C.`
9 **return C**

perturbation (e), i.e., signed CASE (line 2). Since the optimization objective, CASE, is an absolute value, we conduct weight selection by its signs. We select weights with positive perturbation to decrease the positive e and vice versa for negative e. Therefore, we set 0 for the elements with a different sign (line 3) to disable them. Obviously, SQuant-K need only $k = \lfloor |e| \rceil$ elements and reduce $|e| < r_k = 0.5$ (line 4). Finally, SQuant-K flips k weights with the largest $|\mathbf{p}|$ (line 5-6). For now, we have SQuanted the kernel and tuned CASE to $|e| \leq r_k = 0.5$.

SQuant-C Flip Algorithm SQuant-K and SQuant-C can utilize the same flip function with different shapes and partitions. Assume a convolution layer with the shape of (M,N,K). SQuant-C can reshape the weight tensor with a shape of (M,NK). Specifically, for FC and Conv with a kernel size of $(1,1)$, we can regard them as tensors with the shape of (M,N) and skip the SQuant-K. As mentioned in section 3.2, SQuant-C flips only one element in each kernel. We illustrate the kernel perturbation update algorithm (line 7 of algorithm 1) for SQuant-C to flip kernel in Appendix B.2.

As a result, SQuant successfully identifies the optimum combination $\mathbf{f}$ under a low computation complexity, which we analyse next.

Complexity Analysis The original optimization problem described by Eq. Eq. (4) is NP-hard with $O(M \cdot 2^{(NK)^2})$. Based on our fine-grain objective approximation and composition, the new optimization objective is to reduce CASE, whose complexity is $O(M \cdot 2^N)$ for SQuant-C, $O(MN \cdot 2^K)$ for SQuant-K and $O(MNK)$ for SQuant-E. SQuant is optimized to a top-k algorithm with a great complexity reduction to $O(n \cdot log(k))$ for each sub-problem. Our experiments show that after SQuant-K pre-optimization, SQuant-C only requires a tiny top-k number, such as $k = 32$ can satisfy all cases. For a 3×3 kernel with 9 elements, SQuant-K only need $k = 4$ because the kernel CASE must $\leq 4.5 \cdot K$. Finally, their complexity can reduce to linear, $O(M \cdot N \cdot 5)$ for SQuant-C and $O(MN \cdot 9 \cdot 2)$ for SQuant-K.

From the overall perspective of the optimization, SQuant-K has MN sub-problems, while SQuant-C has M sub-problems. Fortunately, because of the independence of sub-problems, SQuant is GPU friendly, allowing each sub-problem to be accelerated in parallel.

Algorithm 2: SQuant Flip Algorithm.

Input: Rounded/SQuanted Weight $\mathbf{w}$;
Weight perturbation $\mathbf{p}$.
Output: Updated Quantized Weight $\mathbf{w}$.

def `SQuantFlip(`$\mathbf{w}$, $\mathbf{p}$`):`
$e = \sum_i \mathbf{p}_i$`// Accumulated perturbation (signed CASE).`
$\mathbf{p}[e \cdot \mathbf{p} < 0] = 0$ `// Disable Elements/kernels with different sign from` e`.`
$k = \lfloor |e| \rceil$ `// Flip` k `elements/kernels based on the CASE.`
$\mathbf{f} = \text{TopK}(|\mathbf{p}|, k).\text{indices}$`// Indices of` k `largest perturbation.`
$\mathbf{w}[\mathbf{f}] = \text{Flip}(\mathbf{w}[\mathbf{f}])$`// Flip` k `elements/kernels with same sign as` e`.`
return w

Arch	Method	No BP	No FT	W-bit	A-bit	Top-1
ResNet18	Baseline	–	–	32	32	71.47
	DFQ	✓	✓	4	4	0.10
	ZeroQ	✗	✓	4	4	26.04
	DSG	✗	✓	4	4	34.53
	GDFQ	✗	✗	4	4	60.60
	SQuant	✓	✓	4	4	**66.19**
	DFQ	✓	✓	6	6	67.30
	ZeroQ	✗	✓	6	6	69.74
	DSG	✗	✓	6	6	70.46
	GDFQ	✗	✗	6	6	70.13
	SQuant	✓	✓	6	6	**70.77**
	DFQ	✓	✓	8	8	69.70
	ZeroQ	✗	✓	8	8	71.43
	GDFQ	✗	✗	8	8	70.68
	SQuant	✓	✓	8	8	**71.45**
ResNet50	Baseline	–	–	32	32	77.74
	ZeroQ	✗	✓	4	4	8.20
	DSG	✗	✓	4	4	23.10
	GDFQ	✗	✗	4	4	55.65
	SQuant	✓	✓	4	4	**70.98**
	ZeroQ	✗	✓	6	6	75.56
	DSG	✗	✓	6	6	76.07
	GDFQ	✗	✗	6	6	76.59
	SQuant	✓	✓	6	6	**77.11**
	ZeroQ	✗	✓	8	8	77.67
	DSG	✗	✓	8	8	77.68
	GDFQ	✗	✗	8	8	77.51
	SQuant	✓	✓	8	8	**77.71**

Table 1: Results of data-free methods with ResNet18 and ResNet50. "No BP" means that no back-propagation algorithm is used to generate data, "No FT" means no fine-tuning (retraining) for weight quantization.

Arch	Method	No BP	No FT	W-bit	A-bit	Top-1
Inception V3	Baseline	–	–	32	32	78.81
	ZeroQ	✗	✓	4	4	12.00
	DSG	✗	✓	4	4	34.89
	GDFQ	✗	✓	4	4	70.39
	SQuant	✗	✗	4	4	**73.36**
	ZeroQ	✗	✓	6	6	75.14
	DSG	✗	✓	6	6	76.52
	GDFQ	✗	✗	6	6	77.20
	SQuant	✓	✓	6	6	**78.30**
	ZeroQ	✗	✓	8	8	78.70
	DSG	✗	✓	8	8	78.81
	GDFQ	✗	✗	8	8	78.62
	SQuant	✓	✓	8	8	**78.81**
Squeeze Next	Baseline	–	–	32	32	69.38
	ZeroQ	✗	✓	4	4	0.10
	SQuant	✓	✓	4	4	**43.97**
	ZeroQ	✗	✓	6	6	39.83
	DSG	✗	✓	6	6	60.50
	SQuant	✓	✓	6	6	**67.37**
	ZeroQ	✗	✓	8	8	68.01
	DSG	✗	✓	8	8	69.27
	SQuant	✓	✓	8	8	**69.29**
Shuffle Net	Baseline	–	–	32	32	65.07
	ZeroQ	✗	✓	6	6	39.92
	DSG	✗	✓	6	6	44.88
	GDFQ	✗	✗	6	6	60.12
	SQuant	✓	✓	6	6	**60.28**
	ZeroQ	✗	✓	8	8	64.46
	GDFQ	✗	✗	8	8	64.03
	SQuant	✓	✓	8	8	**64.67**

Table 2: Results of data-free methods with InceptionV3, SqueezeNext, and ShuffleNet

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

For demonstrating the strength of SQuant, we evaluate the SQuant as well as four SOTA methods, DFQ (Nagel et al., 2019), ZeroQ (Cai et al., 2020), DSG (Zhang et al., 2021; Qin et al., 2021), and GDFQ (Xu et al., 2020), with 5 different CNN models including ResNet-18 & 50 (He et al., 2016), Inception V3 (Szegedy et al., 2016), SqueezeNext (Gholami et al., 2018) and ShuffleNet (Zhang et al., 2018) on the golden standard dataset ImageNet (Krizhevsky et al., 2012).

In our experiments, SQuant is dedicated to weight quantization, including setting min&max range without clipping and selecting the grid point with per-channel quantization, which is friendly to the accelerator. Based on the BN-based approach, we adopt a simple rounding method and a wide quantization range for activation suggested by DFQ (Nagel et al., 2019) without breaking the data-free premise and clip activation tensors in a layerwise manner (per-tensor). We utilize a uniform distribution as the initialization for the activation quantization range. All DFQ algorithms are implemented with PyTorch (Paszke et al., 2019) and evaluated on Nvidia GPU A100-40GB. Unless otherwise stated, we employ both weight and activation quantization in all experiments. Also, uniform quantization grids are used in all experiments, and hyperparameters, e.g., $r_e = r_k = 1.0$ and $r_c = 0.5$, for all SQuant experiments are the same.

4.2 COMPARISON TO SOTA METHODS

Table 1 and Table 2 show the results on the ImageNet datasets for various bit-width choices, comparing our SQuant against other data-free PTQ methods. Among these methods, ZeroQ, DSG, and GDFQ are data-generative approaches with back-propagation. The former two are PTQ methods, while the last is a QAT method, which retrains the network with the synthetic data. DFQ is the only pure data-free method with weight equalization and bias correction.

Experiments show that SQuant significantly outperforms all other SOTA DFQ methods, even with synthetic dataset calibrating their networks. The 8-bit quantization preserves better network accuracy than the lower bit quantization does because of higher precision. The benefit of SQuant becomes more prominent as the bit-width decreases. SQuant outperforms the PTQ methods, i.e. DFQ, ZeroQ and DSG, on all datasets and 4-bit settings by more than 30%. It is noteworthy that SQuant surpasses GDFQ in all cases and even surpasses more than 15% in ResNet50 in 4bit, although GDFQ is a quantization-aware training method.

Comparing GDFQ with ZeroQ and DSG in 4-bit, we find that the weight perturbation has much more influence than activation perturbation, especially in lower bit cases, because all methods adopt the similar activation quantization method, i.e., rounding. But GDFQ makes much effort to calibrate the weights than ZeroQ and DSG for much higher accuracy. SQuant also optimizes the weight perturbation to achieve much higher accuracy in 4-bit with a direct optimization objective. These results clearly illustrate the effectiveness of our CASE-based optimization framework, which is to reduce the CASE of weight perturbation.

4.3 SQUANT EFFICIENCY

The trade-off between efficiency and accuracy is challenging for previous DFQ methods. Before SQuant, DFQ is the fastest one since it does not require back-propagation and finetuning, but it performs poorly, especially in the low bit cases. GDFQ performs relatively good but takes hours to complete 400 epochs that produce synthetic data from weights and then finetune the network. SQuant employs the direct optimization objective, minimizing weight CASE, pushes the quantization procedure to a sub-second level. Table 3 shows the quantization time of the 5 models using SQuant. The efficient algorithm design also contributes to the surprising results. Note that the results in Table 3 are the sum of all layer quantization time, and it will be faster if we quantize the weight of the layer in parallel. A single weight layer takes SQuant just 5 milliseconds on average because SQuant does not involve complex algorithms, such as back-propagation and finetune. That means we can implement the SQuant algorithm for inference-only devices such as smartphones and IoT devices and quantize the network on the fly.

4.4 ABLATION STUDY

SQuant granularity We investigate the effect of SQuant-K and SQuant-C, which have different granularities to optimize CASE. As Table 4 shown, they both have much better effectiveness over SQuant-E (rounding), and the joint optimization drives ResNet18 to higher accuracy. SQuant-E&C has a lower accuracy than SQuant-E&K because SQuant-C has a more significant approximation error than SQuant-K. As the analysis in section 3, the lower granularity ignores more Hessian information and thus is more possible to miss the optimal global solution. This ablation study shows that the more Hessian information we utilize, the higher accuracy we achieve.

Comparison to data-free AdaRound DSG evaluates with a novel data-driven PTQ method, AdaRound (Nagel et al., 2020), which also utilizes the Hessian-based approach to set the weights rounding up or down with a approximation assumption. That provides an opportunity to compare SQuant and AdaRound under the data-free scenario. Results in Table 5 show that SQuant has better accuracy than data-free AdaRound because SQuant directly optimizes the CASE objective instead of

Arch	ResNet18	ResNet50	InceptionV3	SqueezeNext	ShuffleNet
Layers	21	54	95	112	50
Time	76 ms	260 ms	388 ms	588 ms	234 ms

Table 3: SQuant quantization time on GPU A100

Method	W-bit	A-bit	Top-1
Baseline	32	32	71.47
SQuant-E	4	4	43.71
SQuant-E&C	4	4	57.41
SQuant-E&K	4	4	64.41
SQuant-E&K&C	4	4	66.19

Table 4: SQuant ablation results with ResNet18.

Method	No BP	No SD	W-bit	A-bit	Top-1
Baseline	–	–	32	32	71.47
ZeroQ + AdaRound	✗	✗	3	32	49.86
DSG + AdaRound	✗	✗	3	32	56.09
SQuant	✓	✓	3	32	**60.78**
ZeroQ + AdaRound	✗	✗	4	32	63.86
DSG + AdaRound	✗	✗	4	32	66.87
SQuant	✓	✓	4	32	**69.75**

Table 5: ResNet18 results of ZeroQ and DSG combined AdaRound, and SQuant. "No SD" means no synthetic data.

the data-driven MSE of the output activation adopted by AdaRound. It is hard for DSG+AdaRound to find the optimal solution with an excessively long optimization path and gradient-based approaches. Even AdaRound tries a shorter path to finetune the weights layerwise. SQuant still outperforms them in terms of quantization time and accuracy.

5 Related Work

Data-driven Quantization Quantization-aware training (QAT) (Gupta et al., 2015; Jacob et al., 2018; Wang et al., 2019; Zhuang et al., 2021) is one of the most promising techniques to retrain networks and mitigate the accuracy drop introduced by quantization. However, the training procedure is time-consuming and costly. Therefore, post-training quantization (PTQ) (Banner et al., 2018b; Choukroun et al., 2019; Zhao et al., 2019; Nagel et al., 2020) has earned lots of attention due to the absence of any finetuning or retraining process, at the expense of accuracy.

Data-free Quantization Recently, several methods for CNN quantization without the original training datasets have been proposed. These methods are known as data-free quantization (DFQ), including PTQ (Banner et al., 2018a; Nagel et al., 2019; Cai et al., 2020; Zhang et al., 2021) and QAT (Xu et al., 2020; Liu et al., 2021; Qin et al., 2021; Choi et al., 2020). DFQ (Nagel et al., 2019) and ACIQ (Nagel et al., 2019) rely on weight equalization or bias correction without requiring synthetic data. However, they produce a pretty lower accuracy in lower bit-width (such as 6bit and 4bit). Other works synthesize the data to calibrate or finetune the network based on the batch normalization statistics (Cai et al., 2020) or adversarial knowledge distillation techniques (Liu et al., 2021; Choi et al., 2020). They achieve better accuracy using a time-consuming process.

6 Conclusion

This paper approximates and composes the original Hessian optimization objective into weight CASE with a data-free premise. Surprisingly, CASE only involves the weight perturbation and requires no knowledge of any datasets or network architecture. Based on that, we proposed the on-the-fly SQuant framework. We used a progressive algorithm to directly reduce CASE and significantly improve accuracy than other DFQ methods. SQuant considerably reduces optimization complexity and accelerates the data-free quantization procedure, which previously requires back-propagation with massive computation and memory resources consumption seen in other works. In summary, SQuant outperforms other data-free quantization approaches in terms of accuracy and pushes the quantization processing time to a sub-second level.

References

Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Aciq: Analytical clipping for integer quantization of neural networks. 2018a.

Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment. *arXiv preprint arXiv:1810.05723*, 2018b.

Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13169–13178, 2020.

Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 710–711, 2020.

Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, pp. 3009–3018, 2019.

Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *arXiv preprint arXiv:1705.07565*, 2017.

Zhen Dong, Zhewei Yao, Yaohui Cai, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *arXiv preprint arXiv:1911.03852*, 2019a.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 293–302, 2019b.

Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. Squeezenext: Hardware-aware neural network design. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1638–1647, 2018.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pp. 1737–1746. PMLR, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*, 2020.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.

Yuang Liu, Wei Zhang, and Jun Wang. Zero-shot adversarial quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1512–1521, 2021.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1325–1334, 2019.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037, 2019.

Xu Qian, Victor Li, and Crews Darren. Channel-wise hessian aware trace-weighted quantization of neural networks. *arXiv preprint arXiv:2008.08284*, 2020.

Haotong Qin, Yifu Ding, Xiangguo Zhang, Aoyu Li, Jiakai Wang, Xianglong Liu, and Jiwen Lu. Diverse sample generation: Pushing the limit of data-free quantization. *arXiv preprint arXiv:2109.00212*, 2021.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8815–8821, 2020.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Ziwei Wang, Jiwen Lu, Chenxin Tao, Jie Zhou, and Qi Tian. Learning channel-wise interactions for binary convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 568–577, 2019.

Yikai Wu, Xingyu Zhu, Chenwei Wu, Annie Wang, and Rong Ge. Dissecting hessian: Understanding common structure of hessian in neural networks. *arXiv preprint arXiv:2010.04261*, 2020.

Shoukai Xu, Haokun Li, Bohan Zhuang, Jing Liu, Jiezhang Cao, Chuangrun Liang, and Mingkui Tan. Generative low-bitwidth data free quantization. In *European Conference on Computer Vision*, pp. 1–17. Springer, 2020.

Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pp. 11875–11886. PMLR, 2021.

Shixing Yu, Zhewei Yao, Amir Gholami, Zhen Dong, Michael W Mahoney, and Kurt Keutzer. Hessian-aware pruning and optimal neural implant. *arXiv preprint arXiv:2101.08940*, 2021.

Xiangguo Zhang, Haotong Qin, Yifu Ding, Ruihao Gong, Qinghua Yan, Renshuai Tao, Yuhang Li, Fengwei Yu, and Xianglong Liu. Diversifying sample generation for accurate data-free quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15658–15667, 2021.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International conference on machine learning*, pp. 7543–7552. PMLR, 2019.

Bohan Zhuang, Mingkui Tan, Jing Liu, Lingqiao Liu, Ian Reid, and Chunhua Shen. Effective training of convolutional neural networks with low-bitwidth weights and activations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

A APPROXIMATION AND DECOMPOSITION

A.1 $\mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}]$ DECOMPOSITION

In this section, we present the decomposition method for $\mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}]$ illustrated in the algorithm 3. First, we construct three matrices with the shape of (NK, NK), $\mathbf{C}' = \mathbf{J}_{NK}$,

$$\mathbf{K}' = \begin{bmatrix} \mathbf{J}_K & & \\ & \ddots & \\ & & \mathbf{J}_K \end{bmatrix}, \text{ and } \mathbf{E}' = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}.$$

Here, $\mathbf{J}_{NK}$ is an all-one matrix with dimension $NK = N \times K$ and $\mathbf{J}_K$ represents an all-one matrix with dimension K. The n-th diagonal block corresponds to n-th kernel in convolution and has the same constant k_n. $\mathbf{E}$ is a diagonal matrix whose diagonal elements are 1.

Algorithm 3: $\mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}]$ Decomposition.

Input: $\mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}]$, $\mathbf{H}$;
Channel-wise matrix, $\mathbf{C}'$.
Kernel-wise matrix, $\mathbf{K}'$.
Element-wise matrix, $\mathbf{E}'$.
Output: Matrix $\mathbf{E}$,$\mathbf{K}$,$\mathbf{C}$ and $\mathbf{R}$.

$\mathbf{H}' = |\mathbf{H}|$
$c_m = (1-\varepsilon) min(\mathbf{H}')$ `// Set a small value < minimum number of` $\mathbf{H}'$ `for` $\mathbf{C}$`.`
$\mathbf{C} = c_m \mathbf{C}'$
foreach $n \in [1, ..., N]$ **do** `// Kernel-wise.`
  $k_n = (1-\varepsilon'_n) min(\mathbf{H}'_{n,:} - c)$
  $\mathbf{K}_{n,:} = k_n \mathbf{K}'_{n,:}$
  foreach $i \in [1, ..., K]$ **do** `// Element-wise.`
    $e_{n,i} = \mathbf{H}'_{n,i} - c - k$
    $\mathbf{E}_{n,i} = e_{n,i} \mathbf{E}'_{n,i}$
$\mathbf{R} = \mathbf{H} - \mathbf{E} - \mathbf{K} - \mathbf{C}$
return $\mathbf{E}, \mathbf{K}, \mathbf{C}, \mathbf{R}$

In algorithm 3, $0 < \varepsilon, \varepsilon' < 1$, and we can get the four matrices $\mathbf{E}, \mathbf{K}, \mathbf{C}, \mathbf{R}$. Evidently, algorithm 3 can make $c_m > 0$, $k_n > 0$, and $e_{n,i} > 0$ for any $\mathbb{E}[\mathbf{x}^\ell \mathbf{x}^{\ell^T}] = \mathbf{E} + \mathbf{K} + \mathbf{C} + \mathbf{R}$. Without loss of generality, we set $\varepsilon = \frac{3}{4}$, $\varepsilon'_1 = \frac{1}{3}$ and $\varepsilon'_2 = \frac{1}{15}$ for the example in Fig. 1b and the decomposition process shown in Fig. 2.

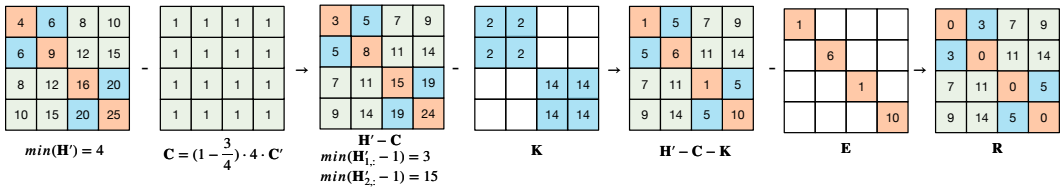


Figure 2: Decomposition process for the example in Fig. 1b.

Specifically, for activation after ReLU function, all elements of $\mathbf{H} > 0$, when the ε and ε' are infinitesimal values ($\varepsilon, \varepsilon' \to 0$), the approximation has the minimum error for channel-wise SQuant-C and kernel-wise SQuant-K.

B SQUANT ALGORITHM

B.1 PROOF OF TOK-k PERTURBATION ALGORITHM

Proof. We will prove the SQuant-E&K will lead to the top-k algorithm. We have the composition SQuant-E and SQuant-K optimization objective for n-th kernel,

$$\underset{\Delta \mathbf{W}^{\ell}_{m,n,:}}{\arg\min} \quad \sum_i (\Delta \mathbf{W}^{\ell}_{m,n,i})^2 + (\sum_i \Delta \mathbf{W}^{\ell}_{m,n,i})^2, \tag{14}$$

which is the first two itmes of Eq. (9). Without loss of generality, we assume $e = \sum_i \Delta \mathbf{W}^{\ell}_{m,n,i} > 0$, then SQuant needs to flip $k = \lfloor e \rceil$ elements transform $\sum_i \Delta \mathbf{W}^{\ell}_{m,n,i}$ to $e-k$ and is still constant $e-k$ regardless of which k elements are. Therefore, the k elements are only determined by the first item of Eq. (14), $\sum_i (\Delta \mathbf{W}^{\ell}_{m,n,i})^2$. We denote $\mathbf{f}$ as the index set of the k flipped elements and the original perturbation $\mathbf{O} = \Delta \mathbf{W}^{\ell}_{m,n,:}$ for n-th kernel. Substituting $\mathbf{f}$ and $\mathbf{O}$ in Eq. (14), we have the optimization objective for $\mathbf{f}$ after flipping k elements,

$$\underset{\mathbf{f}}{\arg\min} \quad \sum_t (|\mathbf{O}_t|)^2 + \sum_j (1-|\mathbf{O}_j|)^2 + (e-k)^2, \quad t \notin \mathbf{f},\ j \in \mathbf{f} \tag{15}$$

$$= \underset{\mathbf{f}}{\arg\min} \quad \sum_i (|\mathbf{O}_i|)^2 - \sum_j (|\mathbf{O}_j|)^2 + \sum_j (1-|\mathbf{O}_j|)^2, \quad j \in \mathbf{f} \tag{16}$$

$$= \underset{\mathbf{f}}{\arg\min} \quad \sum_j [(1-|\mathbf{O}_j|)^2 - |\mathbf{O}_j|)^2], \quad j \in \mathbf{f} \tag{17}$$

$$= \underset{\mathbf{f}}{\arg\min} \quad \sum_j (1-2|\mathbf{O}_j|), \quad j \in \mathbf{f} \tag{18}$$

$$= \underset{\mathbf{f}}{\arg\max} \quad \sum_j (|\mathbf{O}_j|), \quad j \in \mathbf{f}. \tag{19}$$

Therefore, the Eq. (19) is essentially the top-k perturbation algorithm. We can easily extend the top-k algorithm in SQuant-C and design the perturbation update algorithm in B.2.

□

B.2 PERTURBATION UPDATE ALGORITHM

Algorithm 4: Perturbation Update Algorithm.

Input: Weight perturbation $\mathbf{p}$;
Output: Updated weight perturbation $\mathbf{p}$;

```
def UpdatePerturbation(p):
    e = Σ_i p_i // Accumulated perturbation (signed CASE).
    p[e · p < 0] = 0 // Disable Elements/kernels with different sign from e.
    k = ⌊|e|⌉ // Flip k elements/kernels based on the CASE.
    f = TopK(|p|, k).indices// Indices of k largest perturbation.
    if k > |e| then
        // Over SQuant.
        i = f_k // The k-th (last) element of f.
        v = p_i // 0.5 ≤ |v| < 1.0
    else
        // Under SQuant.
        i = TopK(|p|, k+1).indices[k+1] // The (k+1)-th largest element of p.
        v = p_i // |v| ≤ 0.5
    p = 0 // Disable all elements.
    p_i = v // The only flip element candidate of the kernel for SQuant-C.
    return p
```

SQuant-K initializes all elements as flip candidates. After SQuant-K, we update the flip candidates for SQuant-C as shown in algorithm 4 based on the insight top-k perturbation algorithm (B.1).

Over SQuant First we define the situation of $k > |e|$ as "Over SQuant" (line 6). For example, if we have a kernel with $e = +1.6$, we need to SQuant it to -0.4 to satisfy in $(-0.5, 0.5]$ with flipping $k = 2$ elements $\{2.6, 2.7\}$ to $\{2, 2\}$. Obviously, when SQuant-C needs this kernel to calibrate, the last element 2.7 should be the first and the only candidate (line 7,8) to flip back to the original rounded number 3 to make the $e = +0.6$, due to it has the largest element perturbation in the k fliped elements and the smallest element perturbation $(|-0.3| < 0.5)$ after it flips back. It is vice versa for $e < 0$.

Under SQuant For "Under SQuant" (line 9), we need to make the first un-flipped element as the flip candidate (line 10, 11) for SQuant-C, and will lead the kernel to "Over SQuant" with absolute kernel perturbation in $(0.5, 1.0)$ when SQuant-C flips this element of the kernel.

Finally, each kernel has only one candidate flip element for SQuant-C to satisfy Eq. (9). In practice, it is easy to fuse the perturbation update algorithm with the flip algorithm without extra overhead.