HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neural Networks

Zhen Dong, Zhewei Yao, Daiyaan Arfeen,
Amir Gholami, Michael W. Mahoney, Kurt Keutzer
University of California, Berkeley,
{zhendong, zheweiy, daiyaanarfeen, amirgh, mahoneymw, and keutzer}@berkeley.edu

Abstract

Quantization is an effective method for reducing memory footprint and inference time of Neural Networks. However, ultra low precision quantization could lead to significant degradation in model accuracy. A promising method to address this is to perform mixed-precision quantization, where more sensitive layers are kept at higher precision. However, the search space for a mixed-precision quantization is exponential in the number of layers. Recent work has proposed a novel Hessian based framework [9], with the aim of reducing this exponential search space by using second-order information. While promising, this prior work has three major limitations: (i) they only use a heuristic metric based on top Hessian eigenvalue as a measure of sensitivity and do not consider the rest of the Hessian spectrum; (ii) their approach only provides relative sensitivity of different layers and therefore requires a manual selection of the mixed-precision setting; and (iii) they do not consider mixed-precision activation quantization. Here, we present HAWQ-V2 which addresses these shortcomings. For (i), we theoretically prove that the right sensitivity metric is the average Hessian trace, instead of just top Hessian eigenvalue. For (ii), we develop a Pareto frontier based method for automatic bit precision selection of different layers without any manual intervention. For (iii), we develop the first Hessian based analysis for mixed-precision activation quantization, which is very beneficial for object detection. We show that HAWQ-V2 achieves new state-of-the-art results for a wide range of tasks. In particular, we present quantization results for InceptionV3 (7.57MB with 75.98% accuracy), ResNet50 (7.99MB with 75.92% accuracy), and SqueezeNext (1MB with 68.68% accuracy), all without any manual bit selection. Furthermore, we present results for object detection on Microsoft COCO, where we achieve 2.6 higher mAP than direct uniform quantization and 1.6 higher mAP than the recently proposed method of FQN, with a smaller model size of 17.9MB.

1 Introduction

Deep convolutional Neural Networks (NNs) have achieved great success in recent years. However, many of these models, particularly those with state-of-the-art performance, have a high computational cost and memory footprint. This slows inference and training in the cloud, and prohibits their deployment on edge devices.

Quantization [1, 6, 15, 36, 34, 17, 35, 8, 32, 33, 29, 9, 10, 5, 7] is a very promising approach to address this problem by reducing the memory bottleneck, thus allowing the use of lower precision computational units in hardware. By replacing floating point weights in the model with low precision fixed-point values, quantization can shrink the model size without changing the original network architecture. The gains in speed and power consumption directly depend on how aggressively we

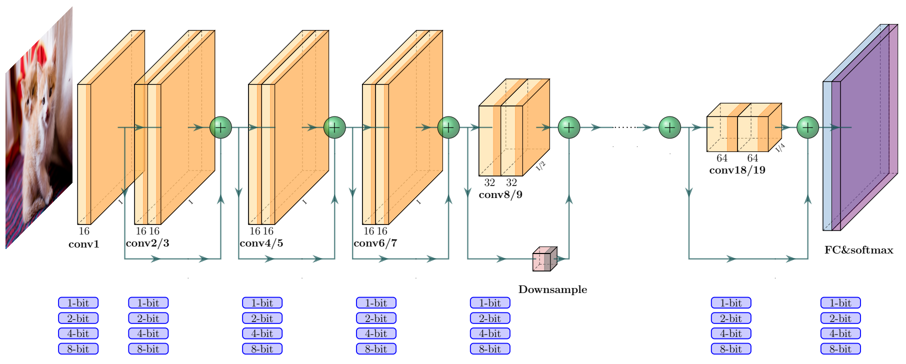

Figure 1: *Mixed Precision Illustration of ResNet20. Here we show the network architecture and list four possible bit precision setting for each layer.*

can perform quantization without losing generalization/accuracy of the model. Despite significant advances, performing ultra low-bit quantization results in non-trivial degradation on accuracy.

Notable recent work on quantization includes using non-uniform quantizers [32, 19], progressive quantization-aware fine-tuning [34, 9], and mixed-precision quantization [30, 29, 9, 27]. Despite the use of non-uniform quantization (which is generally difficult for efficient implementation on hardware), the accuracy degradation is still significant for ultra-low precision. A promising approach to address this is through mixed-precision quantization, where some layers are kept at higher precision, and other layers at lower precision. However, a major problem with this approach is that the search space for determining a good mixed-precision quantization setting is exponentially large in the number of NN layers. This is schematically shown in Figure 1, where four possible bit settings of 1/2/4/8 are considered for each layer of ResNet20. Finding a mixed-precision setting using these possible bitwidths, has a search space of size $4^{20} \approx 1 \times 10^{12}$. It is computationally impossible to test all of these mixed-precision settings and choose a particular setting with good generalization and good hardware performance (in terms of latency and power). The recent work of [29] proposed a reinforcement learning based method to address this. However, this consumes orders of magnitude larger time for bit precision selection (Table 4). Another notable approach is differentiable neural architecture search (DNAS) based methods [30]. But these searching methods can require a large amount of computational resources, and the quality of quantization is very sensitive to the initialization of their search parameters and therefore unpredictable. This makes the deployment of these methods in online learning scenarios especially challenging, as in these applications a new model is trained every few hours and needs to be quantized for efficient inference.

To address these issues, recent work introduced HAWQ [9], which aims to assign higher bit-precision to layers that are more sensitive, and lower bit-precision to less sensitive layers. The sensitivity is measured based on the top Hessian eigenvalue of each layer. This can reduce the exponential search space for mixed-precision quantization, since a layer with higher Hessian eigenvalues cannot be assigned lower bits, as compared to another layer with smaller Hessian eigenvalues. However, there are several shortcomings of this approach: (i) HAWQ uses a heuristic metric based on top Hessian eigenvalue as a measure of sensitivity, and it ignores the rest of the Hessian spectrum; (ii) HAWQ only provides relative sensitivity of different layers, and it still requires a **manual** selection of the mixed-precision setting; and (iii) HAWQ does not consider mixed-precision activation quantization.

Here, we address these challenges and introduce the HAWQ-V2 method. Our main contributions are the following.

1. We perform a theoretical analysis (Lemma 1) showing that a better sensitivity metric is the average Hessian trace, instead of the top eigenvalue heuristic used in HAWQ [9].
2. The HAWQ framework [9] only provides relative sensitivity, and thus it requires manual intervention to select the precise bit-precision setting for each layer. We address this by using a Pareto-frontier based method to automatically determine the bit precision of different layers without any manual selection (Figure 4).
3. We implement a fast algorithm to compute the Hessian trace information using Hutchinson's algorithm in PyTorch. A common concern with the application of Hessian-based methods is the

computational cost [31], but we show this is not an issue here. For example, we can compute Hessian trace for all 54 layers in ResNet50 in less than 30 minutes with 4 GPUs (only 33s per block on average). This is quite fast compared to manual mixed-precision methods which often take weeks of tuning.

4. We achieve new state-of-the-art results for a wide range of models. We present quantization results for InceptionV3, ResNet50, and SqueezeNext (Table 1). Furthermore, we present results for object detection on the Microsoft COCO dataset, where HAWQ-V2 achieves 2.6 higher mAP than direct uniform quantization and 1.6 higher mAP than the recently proposed method of FQN [18], with even smaller model size 17.9MB (Table 2).
5. We develop mixed-precision activation quantization, as described in §2.2. We propose a fast method for computing Hessian information w.r.t. activations, and we show that mixed-precision activation can boost the performance of our object detection model to 34.4 mAP (Table 2).

Outline: In § 2, we discuss theoretical analysis and the relationship between the Hessian spectrum and quantization; and we then discuss the Pareto frontier and our automatic precision selection method. Then, in § 3, we show the results of the trade-off between speed and convergence in the Hutchinson algorithm; and we test HAWQ-V2 with various models on both image classification and object detection tasks. Finally, we provide a conclusion in § 4.

2 Methodology

For a supervised learning framework, the goal is to minimize the empirical risk loss,

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i, \theta), \tag{1}$$

where $\theta \in R^d$ is the learnable model parameters, and $f(x, y, \theta)$ is the loss for a datum $(x, y) \in (X, Y)$. Here, $N = |X|$ is the cardinality of the training set. Assume that the NN can be partitioned into L layers as $\{B_1, B_2, \cdots, B_L\}$, with corresponding learnable parameters $\{W_1, W_2, \cdots, W_L\}$. Furthermore, we denote mini-batch gradient of the loss w.r.t. model parameters as $g = \frac{1}{N_B}\sum_{i=1}^{N_B}\frac{\partial f}{\partial \theta}$, and sub-sampled Hessian w.r.t. parameters as $H = \frac{1}{N_B}\sum_{i=1}^{N_B}\frac{\partial^2 f}{\partial \theta^2}$, where N_B is the mini-batch.

2.1 Sensitivity Metric

HAWQ uses the top Hessian eigenvalue to determine the relative sensitivity order of different layers [9]. However, a NN model contains millions of parameters, and thus millions of Hessian eigenvalues. Therefore, just measuring the top eigenvalue can be sub-optimal. As a simple example, consider two functions $F_1(x, y) = 100x^2 + y^2$ and $F_2(x, y) = 100x^2 + 99y^2$. The top Hessian eigenvalues of F_1 and F_2 are the same (i.e., 200). However, it is clear that F_2 is more sensitive than F_1 since F_2 has much larger function value change along y-axis. Below, we perform a theoretical analysis and show that a better metric is to compute the average Hessian trace (i.e., average of all Hessian eigenvalues) instead of just the top eigenvalue, and later in Section 3.4 we perform an empirical ablation study which supports this finding. Note that in practice the trace and top eigenvalues can be significantly different, as shown in Figure 6 in Appendix C.

Assumption 1 *Assume that:*

- *The model is twice differentiable and has converged to a local minima such that the first and second order optimality conditions are satisfied, i.e., the gradient is zero and the Hessian is positive semi-definite.*
- *If we denote the Hessian of the i^{th} layer as H_i, and its corresponding orthonormal eigenvectors as v_1^i, v_2^i, ..., $v_{n_i}^i$, then the quantization-aware fine-tuning perturbation, $\Delta W_i^* = \arg\min_{W_i^* + \Delta W_i^* \in Q(\cdot)} L(W_i^* + \Delta W_i^*)$, satisfies*

$$\Delta W_i^* = \alpha_{bit} v_1^i + \alpha_{bit} v_2^i + \ldots + \alpha_{bit} v_{n_i}^i. \tag{2}$$

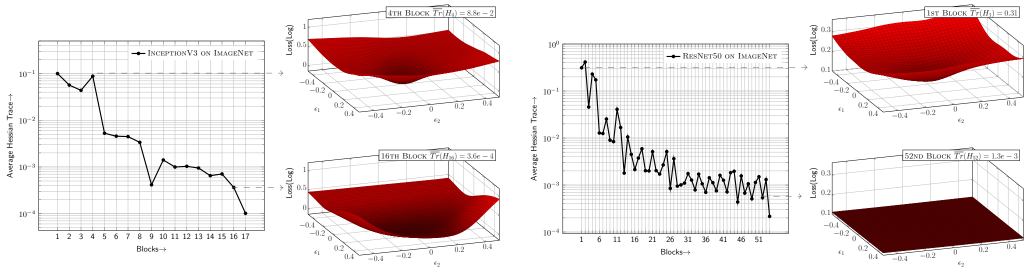

Figure 2: *Average Hessian trace of different blocks in InceptionV3 and ResNet50 on ImageNet, along with the loss landscape of the block 4 and 16 in InceptionV3 (block 1 and 52 in ResNet50). As one can see, the average Hessian trace is significantly different for different blocks.*

Here, n_i is the dimension of W_i, W_i^* is the converging point of i^{th} layer, and $Q(\cdot)$ is the quantization function which maps floating point values to reduced precision values. Note that α_{bit} is a constant number based on the precision setting and quantization range.[1]

- *The third-order term, $\|\frac{\nabla^3 \mathcal{L}}{\nabla W_i^3}\| \; \|\Delta W_i^*\|^3/6$ in the Taylor expansion series is small.*

Given this assumption, we establish the following lemma.

Lemma 1 *Under Assumption 1, when we quantize two layers (denoted by B_1 and B_2) with same amount of perturbation, namely $\|\Delta W_1^*\|_2^2 = \|\Delta W_2^*\|_2^2$, we will have:*

$$\mathcal{L}(W_1^* + \Delta W_1^*, W_2^*, \cdots, W_L^*) \le \mathcal{L}(W_1^*, W_2^* + \Delta W_2^*, W_3^*, \cdots, W_L^*), \tag{3}$$

if

$$\frac{1}{n_1} Tr(\nabla^2_{W_1} \mathcal{L}(W_1^*)) \le \frac{1}{n_2} Tr(\nabla^2_{W_2} \mathcal{L}(W_2^*)). \tag{4}$$

Proof Denote the gradient and Hessian of the first layer as g_1 and H_1, correspondingly. By Taylor's expansion we have:

$$\mathcal{L}(W_1^* + \Delta W_1^*) = \mathcal{L}(W_1^*) + g_1^T \Delta W_1^* + \frac{1}{2} \Delta W_1^{*T} H_1 \Delta W_1^* = \mathcal{L}(W_1^*) + \frac{1}{2} \Delta W_1^{*T} H_1 \Delta W_1^*.$$

Here, we have used the fact that the gradient at the optimum point is zero and that the loss function is locally convex. Also note that $\mathcal{L}(W_1^*) = \mathcal{L}(W_2^*)$ since the model has the same loss before we quantize any layer. Based on the assumption, ΔW_1^* can be decomposed by the eigenvectors of the Hessian. As a result we have:

$$\Delta W_1^{*T} H_1 \Delta W_1^* = \sum_{i=1}^{n_1} \alpha_{bit,1}^2 v_i^{1T} H_1 v_i^1 = \alpha_{bit,1}^2 \sum_{i=1}^{n_1} \lambda_i^1,$$

where (λ_i^1, v_i^1) is the corresponding eigenvalue and eigenvector of Hessian. Similarly, for the second layer we will have: $\Delta W_2^{*T} H_2 \Delta W_2^* = \alpha_{bit,2}^2 \sum_{i=1}^{n_2} \lambda_i^2$, where λ_i^2 is the i^{th} eigenvalue of H_2. Since $\|\Delta W_1^*\|_2 = \|\Delta W_2^*\|_2$, we have $\sqrt{n_1}\alpha_{bit,1} = \sqrt{n_2}\alpha_{bit,2}$. Therefore, we have:

$$\mathcal{L}(W_2^* + \Delta W_2^*) - \mathcal{L}(W_1^* + \Delta W_1^*) = \alpha_{bit,2}^2 n_2 (\frac{1}{n_2} \sum_{i=1}^{n_2} \lambda_i^2 - \frac{1}{n_1} \sum_{i=1}^{n_1} \lambda_i^1) \ge 0.$$

It is easy to see that the lemma holds since the sum of eigenvalues equals to the trace of the matrix. $\square$

It should be noted that the proof still holds for cases where $||\Delta W_1^*||_2^2 \neq ||\Delta W_2^*||_2^2$. In such cases, Eq. 4 becomes:

$$\frac{||\Delta W_1^*||_2^2}{n_1} Tr(\nabla^2_{W_1} \mathcal{L}(W_1^*)) \le \frac{||\Delta W_2^*||_2^2}{n_2} Tr(\nabla^2_{W_2} \mathcal{L}(W_2^*)), \tag{5}$$

indicating that $\overline{Tr}(H_i)||\Delta W_i^*||_2^2$ can be used as a measure of sensitivity.

At first, computing the Hessian trace may seem a prohibitive task, as we do not have direct access to the elements of the Hessian matrix. Furthermore, forming the Hessian matrix explicitly is not

[1]We assume α_{bit} a constant for simplicity. It can be relaxed to random coefficients with the same second moment, i.e., α_{bit} can be random variables for different directions $(v_1^i,\ v_2^i, ..., v_{n_i}^i)$ but with same $\mathbf{E}[\alpha_{bit}^2]$.

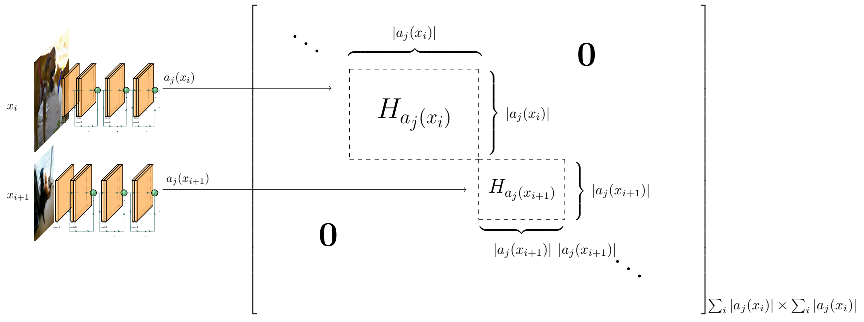

Figure 3: *Illustration of the structure of Hessian w.r.t to activations (H_{a_j}). It is evident that different sized inputs x_i will produce different sized blocks $H_{a_j(x_i)}$ which appear on the diagonal of H_{a_j}.*

computationally feasible. However, it is possible to leverage the extensive literature in Randomized Numerical Linear Algebra (RandNLA) [22, 23] which address this type of problem. In particular, the seminar works of [2, 3] have proposed randomized algorithms for fast trace estimation, using so-called matrix-free methods which do not require the explicit formation of the Hessian operator. Here, we are interested in the trace of a symmetric matrix $H \in R^{d\times d}$. Then, given a random vector $z \in R^d$ whose component is i.i.d. sampled Gaussian distribution ($N(0,1)$) (or Rademacher distribution), we have:

$$Tr(H) = Tr(HI) = Tr(H\,\mathbb{E}[zz^T]) = \mathbb{E}[Tr(Hzz^T)] = \mathbb{E}[z^THz], \tag{6}$$

where I is the identity matrix. Based on this, the Hutchinson algorithm [2] can be used to estimate the Hessian trace:

$$Tr(H) \approx \frac{1}{m}\sum_{i=1}^{m} z_i^T H z_i = Tr_{Est}(H). \tag{7}$$

We show empirically in §3.1 that this algorithm has good convergence properties, resulting in trace computation being orders of magnitude faster than training the network itself.

We have incorporated the above approach and computed the average Hessian trace for different layers of InceptionV3 and ResNet50, as shown in Figure 2. As one can see, there is a significant difference between average Hessian trace for different layers. To better illustrate this, we have also plotted the loss landscape of InceptionV3 and ResNet50 by perturbing the pre-trained model along the first and second eigenvectors of the Hessian for each layer. It is clear that different layers have significantly different "sharpness." (In Appendix C, we also show the average Hessian trace for different blocks of SqueezeNext and RetinaNet, as well as their corresponding loss landscape; see Figure 7.)

2.2 Mixed Precision Activation

The above analysis is not restricted to weights, and in fact it can be extended to mixed-precision activation quantization. In § 3, we will show that this is particularly useful for tasks such as object detection. The theoretical results remain the same, except that the Hessian here is with respect to activations instead of model parameters. In the matrix-free Hutchinson algorithm, we need the result of the following Hessian-vector product to compute the Hessian trace:

$$z^T H_{a_j} z = z^T \left(\nabla^2_{a_j} \frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i, \theta)\right) z, \tag{8}$$

where a_j is the activations of the j^{th} layer. Here, $H_{a_j} \in \mathbb{R}^{(\sum_{i=1}^N |a_j(x_i)|)\times(\sum_{i=1}^N |a_j(x_i)|)}$, where $|a_j(x_i)|$ is the size of the activation of the j^{th} layer for i^{th} input. This is because a_j is a concatenation of $a_j(x_i), \forall i$. See Figure 3 for illustration of the matrix H_{a_j} and its shape. Not only is it prohibitive to compute Hessian matrix, the Hessian-vector product is also infeasible since even generating the random vectors $z \in \mathbb{R}^{\sum_{i=1}^N |a_j(x_i)|}$ is prohibitive, let alone computing its product with H_{a_j}. Furthermore, note that a_j depends on x_i, and for many tasks such as object detection on Microsoft COCO, x_i does not have a fixed size. As a result, the activation size of each layer depends on the input data and is not fixed, which further complicates computing Hessian trace w.r.t. activations.

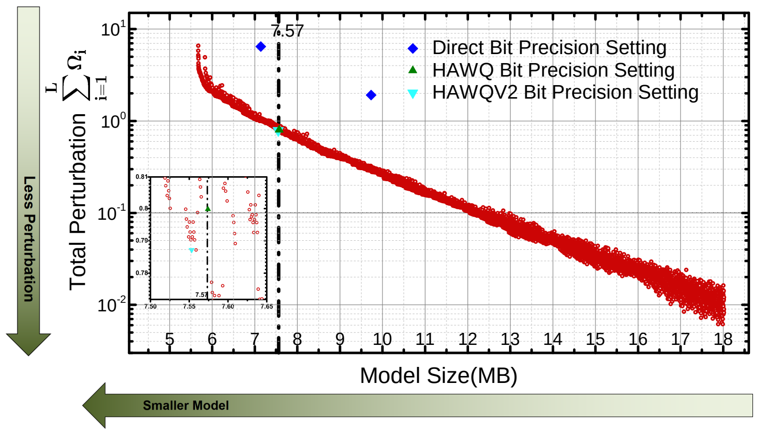

Figure 4: *Pareto Frontier: The trade-off between model size and the sum of Ω metric (of Eqn. (10)) in InceptionV3. Here, L is the number of blocks in the model, and each point in the figure stands for a specific bit precision setting. We show the precision setting used in Direct quantization and* HAWQ. *To achieve fair comparison, we set constraint on* HAWQ-V2 *to have the same model size as* HAWQ.

However, H_{a_j}, has a very interesting structure. As illustrated in Figure 3, it is block diagonal, with $H_{a_j(x_i)}$ being the blocks, where $H_{a_j(x_i)} = \nabla^2_{a_j(x_i)} \frac{1}{N} f(x_i, y_i, \theta)$. This is due to the fact that different inputs are independent of each other. As a result, we can compute the Hessian-trace for the layer's activations *for one input* at a time, and then average the resulting Hessian-traces of each block diagonal part, i.e.,

$$z^T H_{a_j} z = \frac{1}{N} \sum_{i=1}^{N} z_i^T H_{a_j(x_i)} z_i, \tag{9}$$

where z_i is the corresponding components of z w.r.t. the i^{th} input, i.e., x_i. We note that usually this trace computation converges very fast, and it is not necessary to average over the entire dataset. See Figure 8 in Appendix for more details.

2.3 Automatic Bit Selection

An important limitation of relative sensitivity analysis is that it does not provide the specific bit precision setting for different layers. This is true even if we use the average Hessian trace, instead of the top Hessian eigenvalue. For example, we show the average Hessian trace for different blocks of InceptionV3 in Figure 2. We can clearly see that block 1 to block 4 have the largest average Hessian trace, and block 9 or block 16 have orders of magnitude smaller average Hessian trace. However, although we know the first four blocks are more sensitive, we still cannot determine whether to assign 8-bit or 4-bit for these layers.

Denote by $\mathcal{B}$ the set of all admissible bit precision settings that satisfy the relative sensitivity analysis based on the average Hessian trace discussed above. Compared to the original exponential search space, applying the sensitivity constraint makes the cardinality (size) of $\mathcal{B}$ significantly smaller. As an example, the original mixed-precision search space for ResNet50 is $4^{50} \approx 1.3 \times 10^{30}$ if bit-precisions are chosen among $\{1, 2, 4, 8\}$. Using the Hessian-trace sensitivity constraint significantly reduces this search space to $|\mathcal{B}| = 2.3 \times 10^4$ (details on how to calculate the size of $|\mathcal{B}|$ are included in Appendix B). However, this search space is still prohibitively large, especially for deeper models such as ResNet152. In the HAWQ paper [9], the authors manually chose the bit precision among this reduced search space, but this manual selection is undesirable.

We found that this problem can be efficiently addressed using a Pareto frontier approach. The main idea is to sort each candidate bit-precision setting in $\mathcal{B}$ based on the total second-order perturbation that they cause, according to the following metric:

$$\Omega = \sum_{i=1}^{L} \Omega_i = \sum_{i=1}^{L} \overline{Tr}(H_i) \cdot \|Q(W_i) - W_i\|_2^2, \tag{10}$$

where i refers to the i^{th} layer, L is the number of layers in the model, $\overline{Tr}(H_i)$ is the average Hessian trace, and $\|Q(W_i) - W_i\|_2$ is the L_2 norm of quantization perturbation. The intuition is that a bit precision setting with minimal second-order perturbation to the model should lead to good generalization after quantization-aware fine-tuning. Given a target model size, we sort the elements of $\mathcal{B}$ based on their Ω value, and we choose the bit precision setting with minimal Ω. While this approach cannot theoretically guarantee the best possible performance, we have found that in practice it can generate bit precision settings that exceed current state-of-the-art results with a small time cost (as shown in Section 3.1). An important benefit of this approach is that it removes the manual precision selection process used in our previous work on HAWQ [9].

We show the process for choosing the exact bit precision setting of InceptionV3 in Figure 4 (details in Appendix E). Each red dot denotes a specific bit precision setting for different blocks of InceptionV3. For each target model size, HAWQ-V2 chooses the bit precision setting with minimal Ω value. With green triangles, we have also denoted the bit precision setting that was manually selected in the HAWQ paper [9]. The automatic bit precision setting of HAWQ-V2 exceeds the accuracy of HAWQ, as will be discussed in the next section.

3 Empirical Results

3.1 Hutchinson's Method for Trace Estimation

In Figure 5, we show the convergence plot for the Hutchinson's algorithm as we increase the number of iterations used for the Hessian trace estimation. It can be clearly seen that the trace converges rapidly as we increase the number of data points over 512, over which the sub-sampled Hessian is computed. We can see that 50 Hutchinson iterations are sufficient to achieve an accurate approximation with low variance. Based on the convergence analysis, we are able to calculate all the average Hessian traces, shown in Figure 2, corresponding to 54 blocks in a ResNet50 model, within 30 minutes (33s per block on average) using 4 GPUs. The Hutchinson algorithm, in addition to the automatic bit precision selection, makes HAWQ-V2 a significantly faster algorithm than previous searching based algorithms [29] (up to $120\times$ faster as shown in Table 4 in Appendix C).

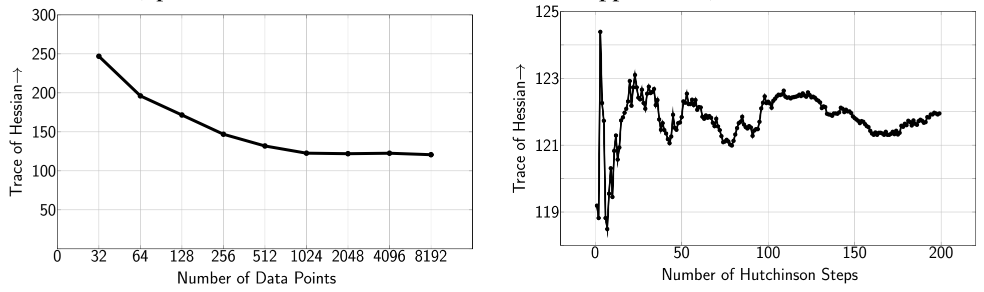

Figure 5: *Relationship between the convergence of Hutchinson and the number of data points (Left) as well as the number of steps (Right) used for trace estimation on block 21 in ResNet50.*

3.2 ImageNet

As shown in Table 1, we first apply HAWQ-V2 on ResNet50 [14], and compare HAWQ-V2 with other popular quantization methods [35, 8, 32, 13, 29, 9]. It should be noted that [35, 8, 32, 13] followed traditional quantization rules which set the precision of the first and last layers to 8-bit, and quantized other layers to an identical precision. Both [29, 9] are mixed-precision quantization methods. Also, [29] uses reinforcement learning methods to search for a good precision setting, while HAWQ uses second-order information to guide the precision selection as well as the block-wise fine-tuning. HAWQ achieves 75.48% with a 7.96MB model size. Keeping model size the same, HAWQ-V2 can achieve 75.92% accuracy without any heuristic knowledge or manual efforts.

We then show results on InceptionV3 [28]. Direct quantization of InceptionV3 (i.e., without use of second-order information), results in 7.69% accuracy degradation. Using the approach proposed in [16] results in more than 2% accuracy drop, even though it uses higher bit precision. HAWQ [9] results in a 2% accuracy gap with a compression ratio of $12.04\times$. HAWQ-V2 **automatically** generates the exact precision setting for the whole network, and still achieves better accuracy than the manual method of HAWQ.

Table 1: *Quantization results on ImageNet. We abbreviate quantization bits used for weights as "w-bits," quantization bits used for activations as "a-bits," top-1 testing accuracy as "Top-1," and weight compression ratio as "W-Comp." Furthermore, we compare* HAWQ-V2 *with direct quantization method of [9] ("Direct") and other state-of-the-art quantization methods. Here "MP" refers to mixed-precision quantization, and we show the lowest bit-precision used in a mixed-precision setting. Compared to [16, 25], we achieve higher compression ratio with higher testing accuracy.*

(a) ResNet50 on ImageNet.

Method	w-bits	a-bits	Top-1	W-Comp	Size(MB)
Baseline	32	32	77.39	1.00×	97.8
Dorefa [35]	2	2	67.10	16.00×	6.11
Dorefa [35]	3	3	69.90	10.67×	9.17
PACT [8]	2	2	72.20	16.00×	6.11
PACT [8]	3	3	75.30	10.67×	9.17
LQ-Nets [32]	3	3	74.20	10.67×	9.17
Deep Comp. [13]	3	MP	75.10	10.41×	9.36
HAQ [29]	MP	MP	75.30	10.57×	9.22
HAWQ [9]	2 MP	4 MP	75.48	12.28×	7.96
HAWQ-V2	2 MP	4 MP	**75.92**	12.24×	7.99

(b) InceptionV3 on ImageNet

Method	w-bits	a-bits	Top-1	W-Comp	Size(MB)
Baseline	32	32	77.45	1.00×	91.2
IntOnly [16]	8	8	75.40	4.00×	22.8
RVQ [25]	3 MP	3 MP	74.14	10.67×	8.55
Direct [9]	2 MP	4 MP	69.76	15.88×	5.74
HAWQ [9]	2 MP	4 MP	75.52	12.04×	7.57
HAWQ-V2	2 MP	4 MP	**75.98**	12.04×	7.57

(c) SqueezeNext on ImageNet

Method	w-bits	a-bits	Top-1	W-Comp	Size(MB)
Baseline	32	32	69.38	1.00×	10.1
Direct [9]	3 MP	8	65.39	9.04×	1.12
HAWQ [9]	3 MP	8	68.02	9.26×	1.09
HAWQ-V2	3 MP	8	**68.68**	9.40×	1.07

We also apply HAWQ-V2 to quantize deep and highly compact models such as SqueezeNext [11]. We choose the wider SqueezeNext model which has a baseline accuracy of 69.38% with 2.5 million parameters (10.1MB in single precision). We can see from Table 1 that direct quantization of SqueezeNext (i.e., without use of second-order information), results in 3.98% accuracy degradation. HAWQ results in a 1MB model size, with 1.36% top-1 accuracy drop. By applying HAWQ-V2 on SqueezeNext, we can achieve a 68.68% accuracy with an unprecedented model size of 1.07MB. Furthermore, in Table 4 we show the timing of HAWQ-V2 as compared to [29]. As one can see, despite using second order information, HAWQ-V2 is orders of magnitude faster and results in significantly more accurate models.

3.3 Microsoft COCO

In order to show the generalization capability of HAWQ-V2, we also test object detection task Microsoft COCO 2017 [21]. RetinaNet [20] is a single stage detector that can achieve state-of-the-art mAP with a very simple network architecture. As shown in Table 2, we use the pretrained RetinaNet with ResNet50 backbone as our baseline model, which can achieve 35.6 mAP with 145MB model size. We first show the result of direct quantization where no Hessian information is used. Even with quantization-aware fine-tuning and channel-wise quantization of weights, directly quantizing weights and activations in RetinaNet to 4-bit causes a significant 4.1 mAP degradation. FQN [18] is a recently proposed quantization method which reduces this accuracy gap to 3.1 mAP with the same compression ratio as Direct method. We implement HAWQ to perform mixed-precision quantization, which results in 33.5 mAP. However, using HAWQ-V2 achieves a state-of-the-art performance of 34.1 mAP, which is 0.6 mAP higher than [9] and 1.6 mAP higher than [18] with an even smaller model size.

It should also be noted that we found the activation quantization to be sensitive for object detection models. For instance, increasing activation quantization bit precision to 6-bit, can results in a 34.8 mAP. One might argue that using 6-bit for activation leads to higher activation memory, which can be a problem for extreme cases such as on micro-controllers where every bit counts. For these situations, we can use mixed-precision activation as discussed in §2.2, with the same automatic bit-precision selection method using Pareto optimal curve. As can be seen in Table 2, mixed-precision activation quantization can achieve very good trade-off between accuracy and compression. With only marginal change to activation compression ratio, it can achieve 34.4 mAP, which significantly outperforms uniform 4-bit activation, and is even close to a uniform 6-bit activation quantization.

Table 2: *Quantization results of RetinaNet-ResNet50 on Microsoft COCO 2017. We show results of direct quantization, mixed-precision quantization [9], as well as a state-of-the-art quantization method for object detection [18].* HAWQ-V2 *can outperform previous results by a large margin. We also show that* HAWQ-V2 *with mixed-precision activations can achieve even better mAP, with a slightly lower activation compression ratio.*

Method	w-bits	a-bits	mAP	W-Comp	A-Comp	Size(MB)
Baseline	32	32	35.6	1.00×	1.00×	145
Direct	4	4	31.5	8.00×	8.00×	18.13
FQN [18]	4	4	32.5	8.00×	8.00×	18.13
HAWQ	3 MP	4	33.5	8.10×	8.00×	17.90
HAWQ-V2	3 MP	4	**34.1**	8.10×	8.00×	17.90
HAWQ-V2	3 MP	4 MP	**34.4**	8.10×	7.62×	17.90
HAWQ-V2	3 MP	6	**34.8**	8.10×	5.33×	17.90

3.4 Ablation Study

Here, we perform three ablation studies. First, we show why it is important to choose the bit-precision setting that results in the smallest model perturbation as done in Figure 4. The results are shown in Table 3(a), where the ablation row uses a bit precision setting with large model perturbation. As one can see, the HAWQ-V2 approach achieves more than 1% higher accuracy with a smaller model size.

Second, we measure the importance of using the Hessian trace to weight the sensitivity $\Omega_i = \overline{Tr}(H_i)\|\Delta W_i\|_2^2$ in Eq. 10. The results are shown in Table 3(b), where we compare with using only parameter perturbation as the sensitivity metric $\Omega_i = \|\Delta W_i\|_2^2$. As we can see, HAWQ-V2 with average Hessian trace is 0.85% better than L2-Sensitivity, while achieving a smaller model size.

Finally, we also compare HAWQ-V2 with a sensitivity that is weighted by Top-1 Hessian eigenvalue. The results are shown in Table 5 in Appendix D. As expected, the trace-weighted metric in HAWQ-V2 achieves higher accuracy.

Table 3: *The effectiveness of metric in Eq. 10. Experiments are for SqueezeNext on ImageNet.*

(a) Accuracy v.s. Total Perturbation

Method	w-bits	a-bits	Top-1	Size(MB)	Perturb.
Baseline	32	32	69.38	10.1	0
Large Perturbation (Ablation)	3 MP	8	67.46	1.09	3.2
Min Perturbation (HAWQ-V2)	3 MP	8	**68.68**	**1.07**	1.1

(b) $\overline{Tr}(H_i)\|\Delta W_i\|_2^2$ v.s. $\|\Delta W_i\|_2^2$

Method	w-bits	a-bits	Top-1	W-Comp	Size(MB)
Baseline	32	32	69.38	1.00×	10.1
L2-Sensitivity (Ablation)	3 MP	8	67.83	9.18×	1.10
Trace-Sensitivity (HAWQ-V2)	3 MP	8	**68.68**	**9.40×**	**1.07**

4 Conclusions

In this work, we performed a theoretical analysis showing that a better sensitivity metric is the average Hessian trace, instead of the top Hessian eigenvalue heuristic used in HAWQ [9], and we presented an automatic mixed-precision quantization method to avoid the manual bit selection. Moreover, we developed mixed-precision activation, and we proposed a very efficient method for computing the Hessian trace by using matrix-free algorithms. HAWQ-V2 achieves state-of-the-art results on image classification for InceptionV3, ResNet50 and SqueezeNext, and on object detection for RetinaNet-ResNet50. As part of future work, one could consider co-designing quantization for a particular hardware and include not only the Hessian but also hardware-specific metrics such as latency/power consumption [29]. Another direction is to avoid using any floating point number, which is typically used to perform arithmetic on hardware with existing methods [16]. Furthermore, one could explore to make training more efficient by incorporating Hessian to adjust precision throughout training [24], or use it to compress communication in distributed training [26, 12].

Acknowledgments

This work was supported by a gracious fund from Intel corporation, and in particular Intel VLAB team. We also acknowledge gracious support from Google Cloud, Google TFTC team, Samsung SAIT, and Amazon AWS. We also gratefully acknowledge the support of NVIDIA Corporation for their donation of two Titan Xp GPU used for this research. Michael W. Mahoney would also like to acknowledge the UC Berkeley CLTC, ARO, NSF, and ONR for providing partial support of this work. Our conclusions do not necessarily reflect the position or the policy of our sponsors, and no official endorsement should be inferred.

Broader Impact

Deep learning models are rapidly increasing in size and this has created a challenge for deploying these models in practice. Our work addresses this problem by developing a novel compression method with minimal impact on accuracy. Our work is applicable to a wide range of NN models, as depicted in our empirical evaluation. Furthermore, our method can help reduce both the computation and the memory bottleneck, so that state-of-the-art neural network models can be deployed even onto edge devices. This helps to realize wider applications of technology, especially in area such as deep learning based daily applications on smart phones, or basic and always-on applications on embedded chips.

References

[1] Krste Asanovic and Nelson Morgan. *Experimental determination of precision requirements for back-propagation training of artificial neural networks*. International Computer Science Institute, 1991.

[2] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):8, 2011.

[3] Zhaojun Bai, Gark Fahey, and Gene Golub. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1-2):71–89, 1996.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[5] 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*, pages 13169–13178, 2020.

[6] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5918–5926, 2017.

[7] Ting-Wu Chin, Pierce I-Jen Chuang, Vikas Chandra, and Diana Marculescu. One weight bitwidth to rule them all. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[8] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian aware quantization of neural networks with mixed-precision. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[10] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph Hassoun. Near-lossless post-training quantization of deep neural networks via a piecewise linear approximation. *arXiv preprint arXiv:2002.00104*, 2020.

[11] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. SqueezeNext: Hardware-aware neural network design. *Workshop paper in CVPR*, 2018.

[12] Vipul Gupta, Dhruv Choudhary, Ping Tak Peter Tang, Xiaohan Wei, Xing Wang, Yuzhen Huang, Arun Kejariwal, Kannan Ramchandran, and Michael W. Mahoney. Fast distributed training of deep neural networks: Dynamic communication thresholding for model and data parallelism, 2020.

[13] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations*, 2016.

[14] 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*, pages 770–778, 2016.

[15] Lu Hou and James T Kwok. Loss-aware weight quantization of deep networks. *arXiv preprint arXiv:1802.08635*, 2018.

[16] 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*, pages 2704–2713, 2018.

[17] Soroosh Khoram and Jing Li. Adaptive quantization of neural networks. In *International Conference on Learning Representations*, 2018.

[18] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2810–2819, 2019.

[19] Zhenyu Liao, Romain Couillet, and Michael W Mahoney. Sparse quantized spectral clustering. *arXiv preprint arXiv:2010.01376*, 2020.

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[22] M. W. Mahoney. *Randomized algorithms for matrices and data*. Foundations and Trends in Machine Learning. NOW Publishers, Boston, 2011.

[23] M. W. Mahoney, J. C. Duchi, and A. C. Gilbert, editors. *The Mathematics of Data*. IAS/Park City Mathematics Series. AMS, IAS/PCMI, and SIAM, 2018.

[24] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[25] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. Value-aware quantization for training and inference of neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 580–595, 2018.

[26] Cèdric Renggli, Saleh Ashkboos, Mehdi Aghagolzadeh, Dan Alistarh, and Torsten Hoefler. Sparcml: High-performance sparse communication for machine learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2019.

[27] 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 *AAAI*, pages 8815–8821, 2020.

[28] 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*, pages 2818–2826, 2016.

[29] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

[30] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[31] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael Mahoney. Pyhessian: Neural networks through the lens of the hessian. *arXiv preprint arXiv:1912.07145*, 2019.

[32] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[33] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. *Proceedings of Machine Learning Research*, 2019.

[34] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless CNNs with low-precision weights. *International Conference on Learning Representations*, 2017.

[35] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[36] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. *arXiv preprint arXiv:1712.01048*, 2017.

A Quantization Details

During the forward pass, each element in a weight or activation tensor X will be quantized as follows:

$$X' = \text{Clamp}(X, q_0, q_{2^k-1}),$$
$$X^I = \lfloor \frac{X' - q_0}{\Delta} \rceil, \text{ where } \Delta = \frac{q_{2^k-1} - q_0}{2^k - 1},$$
$$Q(X) = \Delta X^I + q_0,$$

where $\lfloor \cdot \rceil$ is the round operator, Δ is the distance between adjacent quantized points, X^I is a set of integer indices, $[q_0, q_{2^k-1}]$ stands for the quantization range of the floating point tensor, and the function Clamp sets all elements smaller than q_0 equal to q_0, and all elements larger than q_{2^k-1} to q_{2^k-1}. It should be noted that $[q_0, q_{2^k-1}]$ can be a subinterval of $[min, max]$, in order to get rid of outliers and better represent the majority of the given tensor. During inference, the expensive floating point arithmetic can be replaced by efficient integer arithmetic for the matrix multiplication with X^I, and then followed by a gathered dequantization operation, which will significantly accelerate the computation process. Since we use the quantization-aware fine-tuning scheme, in the backward pass, the Straight-Through Estimator (STE) [4] is used for computing the gradient for X.

B Search Space

Suppose the number of different mixed-precision settings is $\mathcal{B}$, and the number of different progressive quantization-aware fine-tuning orders is $\mathcal{C}$. The whole search space can be written as $\mathcal{B} \times \mathcal{C}$. We have:

$$\mathcal{B} = m^L.$$

$$\mathcal{C} = \sum_{i=1}^{L} i! \times S(L, i) \to L!.$$

where m is the number of quantization precision options, L is the number of layers in a given model, $S(L, i)$ stands for Stirling numbers of the second kind, which have a growth speed between $O(L!)$ and $O(L^L)$. In the case of layer-wise progressive fine-tuning, where only one layer can be fine-tuned at a time, $\mathcal{C}$ degrades to $L!$.

Given two layers B_i and B_j with average Hessian trace $Tr(B_i)/n_i > Tr(B_j)/n_j$, if we set quantization precision $q_i \geq q_j$, then based on that, we are able to order all L layers in the model according to their average Hessian trace. Considering the situation that j precision options are used out of total number m, the mixed-precision problem can be reduced to an integer partition problem, namely, to partition the ordered layers into j different groups, which results in $\binom{j-1}{L-1}$ possible solutions. Since there are $\binom{j}{m}$ different combinations of the j precision options, the total size of search space is $\sum_{j=1}^{m} \left(\binom{j}{m} \cdot \binom{j-1}{L-1}\right)$.

C Extra Results

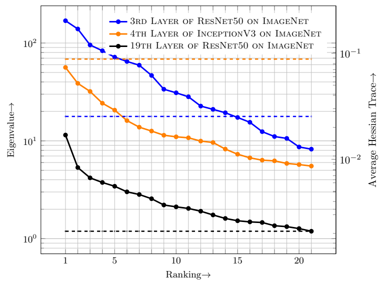

Figure 6: *The top-20 eigenvalues of ResNet50 (3rd/19th layers) and InceptionV3 (4th layer). The dotted lines represent corresponding values of average Hessian trace. As we can see, the trace of Hessian is not dominated by the top-1 eigenvalue, indicating Hessian trace and top-1 eigenvalue to be different metrics.*

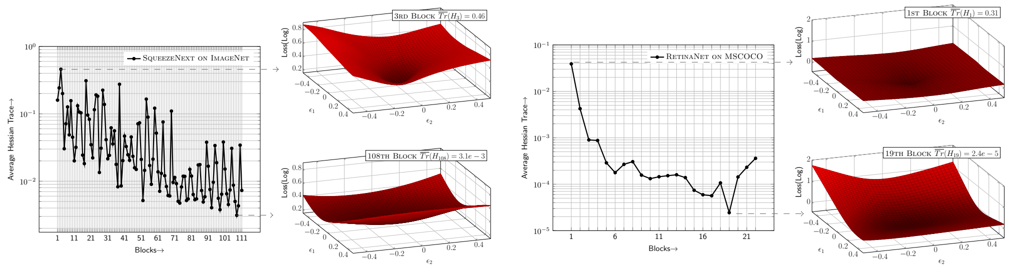

Figure 7: *Average Hessian trace of different blocks in SqueezeNext and RetinaNet, along with the loss landscape of block 3 and 108 in SqueezeNext, and block 1 and 19 in RetinaNet. It should be noted that block 1 to block 17 in RetinaNet are the ResNet50 backbone, block 18 to block 20 are FPN, and block 21 and block 22 are classification and regression head, respectively. As one can see, the average Hessian trace is significantly different for different blocks. We assign higher bits for blocks with larger average Hessian trace, and fewer bits for blocks with smaller average Hessian trace. For reference, in Figure 2 we showed a similar plot but for InceptionV3 and ResNet50.*

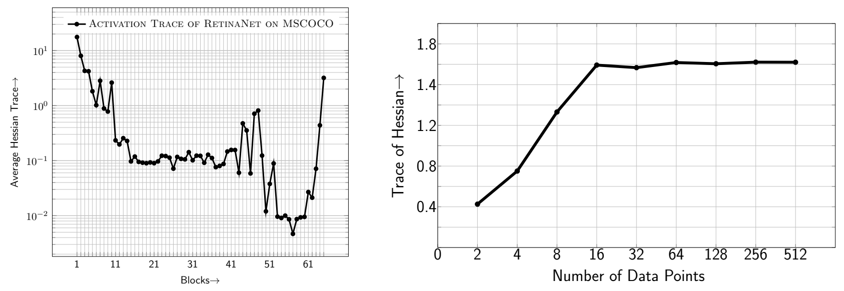

Figure 8: *(Left) Average Hessian trace w.r.t. activations in RetinaNet. As we can see, the average Hessian trace varies significantly across activations of different blocks. We use this information to perform mixed-precision activation quantization as discussed in § 2.2. (Right) we show the relationship between the convergence of Hutchinson and the number of data points used for trace estimation on block 5 in RetinaNet. We used 128 data points with 50 Hutchinson steps to plot the left figure.*

Table 4: *Comparison between* HAWQ-V2 *and HAQ on ResNet50, InceptionV3 and SqueezeNext. HAQ method can take an order of magnitude more time to find the right bit precision configuration, while our fully automatic approch* HAWQ-V2 *can still achieve significantly higher accuracy with better stability.*

Model	Method	Top-1 Accuracy	W-Comp	Size(MB)	Search Time(hours)	Speed Up
ResNet50	Baseline Model	**77.39**	1.00×	97.8		
ResNet50	HAQ [29]	75.30	10.57×	9.22	10	
ResNet50	HAWQ-V2	**75.92**	12.24×	7.99	**0.5**	$> \mathbf{20}\times$
InceptionV3	Baseline Model	**77.45**	1.00×	91.2		
InceptionV3	HAQ [29]	71.60	10.00×	9.12	50	
InceptionV3	HAWQ-V2	**75.98**	12.04×	7.57	**0.4**	$> \mathbf{125}\times$
SqueezeNext	Baseline Model	**69.38**	1.00×	10.1		
SqueezeNext	HAQ [29]	65.87	10.00×	1.01	50	
SqueezeNext	HAWQ-V2	**68.68**	9.40×	1.07	**0.9**	$> \mathbf{55}\times$

D Additional Ablation Study

In Table 5, we show the effectiveness of using average Hessian trace over top eigenvalues. HAWQ uses a heuristic metric $S_i = \lambda_i / n_i$ to select mixed-precision bitwidths, where λ_i is the top eigenvalue of the i^{th} layer and n_i is the parameter size. However, as we can see, only using λ_i as sensitivity metric in HAWQ (represented as HAWQ-Ablation) can lead to significant accuracy degradation. In contrast, the theoretically derived metric, average Hessian trace, can achieve better results than HAWQ with the same compression ratio. To achieve fair comparison, we constrain HAWQ-V2 to assign the same quantization precision for layers in the same block of InceptionV3, as what HAWQ did, and we make their activation bit settings to be the same (referred as HAWQ-V2-blockwise).

Table 5: *The effectiveness of average Hessian trace. The experiments are for InceptionV3 on ImageNet. We abbreviate quantization bits used for weights as "w-bits," quantization bits used for activations as "a-bits," top-1 testing accuracy as "Top-1," and weight compression ratio as "W-Comp." Here "MP" refers to mixed-precision quantization, and we show the lowest bit-precision used in a mixed-precision setting. Blockwise means to assign the same bitwidth for layers within the same block. Compared to* HAWQ *and* HAWQ-*Ablation,* HAWQ-V2 *can achieve higher compression ratio with higher accuracy.*

Method	Metric	w-bits	a-bits	Top-1	W-Comp	Size(MB)
Baseline	NA	32	32	77.45	1.00×	91.2
HAWQ	λ_i / n_i	2 MP	4 MP	75.52	12.04×	**7.57**
HAWQ-Ablation	λ_i	2 MP	4 MP	73.35	10.56×	8.65
HAWQ-V2-blockwise	$\overline{Tr}(H_i)$	2 MP	4 MP	**75.73**	12.04×	**7.57**

E Pareto Frontier

In order to generate Figure 4 when the size of $\mathcal{B}$ is non-trivial, we break the network into L/a groups, with each group containing a layers. Then we break the x-axis (model size) of the Pareto frontier plot into b intervals. We start with the first a layers of the network. For each interval on the x-axis, we choose top t configurations that have the lowest overall sensitivity based on Equation 10.

Then we consider the next set of a layers. This is similar to the first step, except that now we need to jointly consider the top $t \times b$ configurations selected for the first a layers. Suppose we have p possible bit configurations for the next a layers, we choose new top $t \times b$ configurations out of all $t \times b \times p$ possible bit configurations for the first $2a$ layers.

This process needs to be performed iteratively for all the L/a groups. Our experiments show that the accuracy is not sensitive to the hyperparameters, and we typically set t, b, a to be 5, 200, 5, respectively. This process can help us to tackle deep networks that have very large size of $\mathcal{B}$. We can even expand $\mathcal{B}$ to be the whole space m^L (m represents the number of quantization precision options), since in that case, the complexity of this algorithm becomes $(L/a) \times t \times b \times m^a$ where a is a constant number irrelevant to L. After we obtain Figure 4, we choose the bit configuration with lowest sensitivity given the target model size.