RDO-Q: Extremely Fine-Grained Channel-Wise Quantization via Rate-Distortion Optimization

Zhe Wang[1], Jie Lin[1⋆], Xue Geng[1], Mohamed M. Sabry Aly[2], and Vijay Chandrasekhar[1,2]

[1] Institute for Infocomm Research, A*STAR, Singapore 138632
wangz@i2r.a-star.edu.sg, jie.dellinger@gmail.com,
geng_xue@i2r.a-star.edu.sg, vijay.cmu@gmail.com
[2] Nanyang Technological University, 50 Nanyang Ave, Singapore 639798
msabry@ntu.edu.sg

Abstract. Allocating different bit widths to different channels and quantizing them independently bring higher quantization precision and accuracy. Most of prior works use equal bit width to quantize all layers or channels, which is sub-optimal. On the other hand, it is very challenging to explore the hyperparameter space of channel bit widths, as the search space increases exponentially with the number of channels, which could be tens of thousand in a deep neural network. In this paper, we address the problem of efficiently exploring the hyperparameter space of channel bit widths. We formulate the quantization of deep neural networks as a rate-distortion optimization problem, and present an ultra-fast algorithm to search the bit allocation of channels. Our approach has only linear time complexity and can find the optimal bit allocation within a few minutes on CPU. In addition, we provide an effective way to improve the performance on target hardware platforms. We restrict the bit rate (size) of each layer to allow as many weights and activations as possible to be stored on-chip, and incorporate hardware-aware constraints into our objective function. The hardware-aware constraints do not cause additional overhead to optimization, and have very positive impact on hardware performance. Experimental results show that our approach achieves state-of-the-art results on four deep neural networks, ResNet-18, ResNet-34, ResNet-50, and MobileNet-v2, on ImageNet. Hardware simulation results demonstrate that our approach is able to bring up to 3.5× and 3.0× speedups on two deep-learning accelerators, TPU and Eyeriss, respectively.

Keywords: Deep Learning, Quantization, Rate-Distortion Theory

1 Introduction

Deep Learning [20] has become the de-facto technique in Computer Vision, Natural Language Processing, Speech Recognition, and many other fields. However,

⋆ Jie Lin and Vijay Chandrasekhar did this work when they were with Institute for Infocomm Research, Singapore. Jie Lin is the corresponding author.

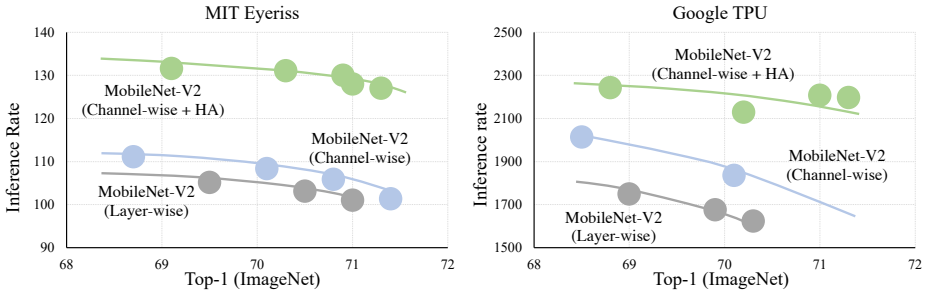

Fig. 1. Channel-wise bit allocation plus hardware-aware constraints (HA) achieves the best performance on Eyeriss and TPU. Channel-wise bit allocation outperforms layer-wise bit allocation because of higher quantization precision. Inference Rate: number of images processed per second.

the high accuracy of deep neural networks [19] comes at the cost of high computational complexity. Due to the large model size and huge computational cost, deploying deep neural networks on mobile devices is very challenging, especially on tiny devices. It is therefore important to make deep neural networks smaller and faster through model compression [12], to deploy deep neural networks on resource-limited devices.

Quantization [12] is one of the standard techniques for neural network compression. One problem existed in prior works is that they typically use equal bit width to quantize weights and activations of all layers, which is sub-optimal because weights and activations in different layers react differently on quantization. They should be treated independently and quantized with un-equal bit widths. Moreover, most of prior works only consider the model size and accuracy in their methods and do not consider the system-level performance when deploying quantized models on hardware platforms. As illustrated in prior works [30], a well quantized network can not guarantee superior performance on hardware platforms. To address these two issues, the recently proposed mixed-precision quantization methods assign un-equal bit widths across layers, and optimize the hardware metrics directly in the quantization mechanism. For example, HAQ [30] proposed a reinforcement learning method to learn the bit widths of weights and activations across layers and minimized latency and energy in their objective function directly.

Although noticeable improvement has been obtained by mixed-precision quantization, the layer-wise bit allocation scheme is still sub-optimal, since all channels in a CONV layer are quantized with equal bit width. In fact, different channels react very distinctively to quantization. Higher precision can be obtained if allocating un-equal bit widths to channels. However, the challenge is that the hyperparameter space of channel bit widths increases exponentially with the number of channels. Given N channels and C bit widths, the search complexity is $O(C^N)$, where in a deep neural network, N can be tens of thousand or even

more. Such huge search space could make it unaffordable for a heuristic search method, like reinforcement learning [27], to find solution within limited time.

In this paper, we propose a new approach to efficiently explore the hyperparameter space of channel bit widths. We apply the classic coding theories [28], and formulate the quantization of weights and activations as a rate-distortion optimization problem. Since the output distortion is highly related to accuracy, by minimizing the output distortion induced by quantization, our approach is able to well maintain the accuracy at very low bit widths. We then search the optimal bit allocation across channels in a rate-distortion optimized manner. Through utilizing the additivity property of output distortion, we present an ultra-fast algorithm with linear time complexity to find the optimal channel-wise bit allocation, by using Lagrangian formulation. Our algorithm only costs a few minutes on CPU for a deep neural network.

What's more, we present an alternative way to improve the system-level performance when deploying quantized networks on target hardware platforms. Prior works typically optimize the hardware metrics of a whole network directly, and need real-time feedback from simulators in their learning procedure, which could cause additional overhead to optimization. Instead, our approach improves hardware performance by restricting the size of each individual layer, and does not require feedback from simulators. Our key insight is that the volume of weights and activations in some layers is particularly significant, which exceeds the capacity of on-chip memory. As a result, these layers significantly prolong the inference time due to the necessity of slow data access to off-chip memory. We thus constrain the size of these large layers to ensure that all variables can be stored on-chip.

To our best knowledge, only one prior work, AutoQ [22], finds channel-wise bit allocation, and optimizes the performance on hardware platforms simultaneously. AutoQ employs reinforcement learning to solve the bit allocation problem, which is time-consuming, and could fall into a local optimum in their heuristic search method. Our approach adopts Lagrangian formulation for fast optimization, and is able to find global optimal solution in a rate-distortion optimized manner. We summarize the main contributions of our paper as following:

- We formulate the quantization of deep neural networks as a rate-distortion optimization problem, and optimize the channel-wise bit allocation for higher accuracy. We present an ultra-fast algorithm with linear time complexity to efficiently explore the hyperparameter space of channel bit widths.
- We present a simple yet effective way to improve the performance on target hardware platforms, through restricting the size of each individual layer and incorporating hardware-aware constraints into our objective function. The hardware-aware constraints can be integrated seamlessly, without causing additional overhead to optimization.
- Our approach achieves state-of-the-art results on various deep neural networks on ImageNet. Hardware simulation results demonstrate that our approach is able to bring considerable speedups for deep neural networks on two hardware platforms.

Table 1. A comparison with prior mixed-precision quantization works.

Approach	Bit Allocation Scheme	Hardware -Aware	Optimization	Complexity
ReLeQ [7]	Layer-Wise	No	Reinforcement Learning	High
HAQ [30]	Layer-Wise	*Yes*	Reinforcement Learning	High
DNAS [31]	Layer-Wise	No	Neural Architecture Search	High
DQ [29]	Layer-Wise	No	Training from Scratch	High
HAWQ [6]	Layer-Wise	No	Training from Scratch	High
ALQ [23]	Layer-Wise	No	Training from Scratch	High
AQ [18]	*Element-Wise*	No	Closed-Form Approximation	*Low*
PTQ [1]	*Channel-Wise*	No	Analytic Solution	*Low*
FracBits [32]	*Channel-Wise*	No	Training from Scratch	High
DMBQ [34]	*Channel-Wise*	No	Training from Scratch	High
AutoQ [22]	*Channel-Wise*	*Yes*	Reinforcement Learning	High
RDO-Q (Ours)	*Channel-Wise*	*Yes*	Lagrangian Formulation	*Low*

2 Related Works

We discuss prior mixed-precision quantization works related to our work. ReLeQ [7] proposed an end-to-end deep reinforcement learning (RL) framework to automate the process of discovering quantization bit widths. Alternatively, HAQ [30] leveraged reinforcement learning to determine quantization bit widths, and employed a hardware simulator to generate direct feedback signals to the RL agent. DNAS [31] proposed a differentiable neural architecture search framework to explore the hyperparameter space of quantization bit widths. Differentiable Quantization (DQ) [29] learned quantizer parameters, including step size and range, by training with straight-through gradients, and then inferred quantization bit widths based on the learned step size and range. Hessian AWare Quantization (HAWQ) [6] introduced a second-order quantizatino method to select the quantization bit width of each layer, based on the layer's Hessian spectrum. Adaptive Loss-aware Quantization (ALQ) [23] directly minimized network loss w.r.t. quantized weights, and used network loss to decide quantization bit widths. Layer-wise bit allocation scheme is employed in all the above methods.

Adaptive Quantization (AQ) [18] found a unique, optimal precision for each network parameter (element-wise), and provided a closed-form approximation solution. Post Training Quantization (PTQ) [1] adopted channel-wise bit allocation to improve quantization precision, and provided an analytic solution to find quantization bit widths, assuming that parameters obey certain distributions. FracBits [32] generalized quantization bit widths to arbitrary real numbers to make them differentiable, and learned channel-wise (or kernel-wise) bit allocation during training. Distribution-aware Multi-Bit Quantization (DMBQ) [34] proposed loss-guided bit-width allocation strategy to adjust the bit widths of weights and activations channel-wisely. AQ, PTQ, FracBits, and DMBQ all did not take the impact on hardware platforms into account. AutoQ [22] proposed a hierarchical deep reinforcement learning approach to find quantization bit widths

of channels and optimize hardware metrics (e.g., latency and energy) simultaneously. Different with AutoQ, our approach provides an alternative way to quickly explore the hyperparameter space of bit widths with linear time complexity, and is able to find global optimal solution in a rate-distortion optimized manner. Table 1 illustrates the differences between the mixed-precision quantization approaches.

One prior work [9] interpreted neural network compression from a rate-distortion's perspective. The main focus of [9] was giving an upper bound analysis of compression and discussing the limitations. [9] did not give a way to search bit allocation, and there was no practical results provided.

3 Approach

We utilize classic coding theories [28], and formulate the quantization of deep neural networks as a rate-distortion optimization problem [37, 36]. The differentiability of input-output relationships for the layers of neural networks allows us to relate output distortion to the bit rate (size) of quantized weights and activations. We add hardware-aware constraints into the objective function to improve the performance on hardware platforms. We will discuss the formulation of our approach and its optimization in this section.

3.1 Formulation

Let $\mathcal{F}$ denote a deep neural network. Given an input $\mathbf{I}$, we denote $\mathbf{Y}$ as the output of $\mathcal{F}$, i.e. $\mathbf{Y} = \mathcal{F}(\mathbf{I})$. When performing quantization on weights and activations, a modified output vector $\widehat{\mathbf{Y}}$ would be received. The output distortion is measured by the distance between $\mathbf{Y}$ and $\widehat{\mathbf{Y}}$, which is defined as

$$\delta = \|\mathbf{Y} - \widehat{\mathbf{Y}}\|_2^2 \tag{1}$$

Here Euclidean distance $\|.\|_2$ is adopted. Our approach allocates different quantization bit widths to weight channels and activation layers. We aim to minimize the output distortion under the constraint of bit rate (size). Given the bit rate constraint r, the rate-distortion optimization problem is formulated as

$$\min \delta = \|\mathbf{Y} - \widehat{\mathbf{Y}}\|_2^2 \;\; s.t. \sum_{i=1}^{l} \sum_{j=1}^{n_i} R_{i,j}^w + \sum_{i=1}^{l} R_i^a \leq r, \tag{2}$$

where $R_{i,j}^w$ denotes the bit rate of weight channel j in layer i, R_i^a denotes the bit rate of activations in layer i, n_i denotes the number of channels in layer i, and l denotes the total number of layers. Specifically, $R_{i,j}^w$ equals to the quantization bit width of channel j in layer i, denoted as $B_{i,j}^w$, multiplied by the number of weights in that channel; R_i^a equals to the quantization bit width of activations in layer i, denoted as B_i^a, multiplied by the number of activations in that layer.

We noticed that output distortion is highly related to network accuracy. By minimizing output distortion induced by quantization, our approach is able to maintain the accuracy at very high compression ratio.

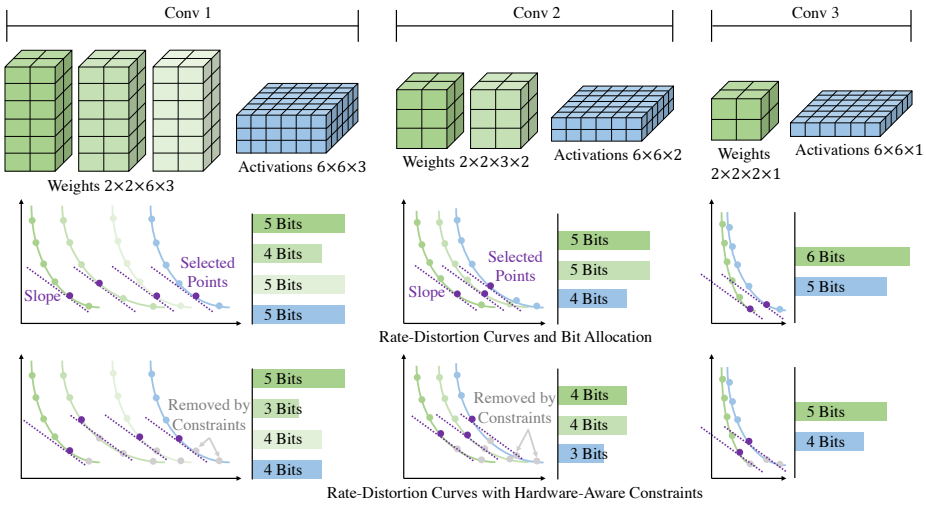

Fig. 2. Examples of finding optimal bit allocation w/ and w/o hardware-aware constraints.

3.2 Optimizing Channel-Wise Bit Allocation

We explored the additivity property of output distortion when performing quantization on weight channels and activation layers, and found that the additivity property holds, similar to the observation made in [39, 37]. Utilizing the additivity property, we develop an efficient Lagrangian formulation method to solve the bit allocation problem.

Specifically, let $\delta_{i,j}^{w}$ and δ_{i}^{a} denote the output distortion caused by quantizing an individual weight channel and an individual activation layer, respectively. The output distortion δ, caused by quantizing all weight channels and activation layers, equals the sum of output distortion due to the quantization of each individual item

$$\delta = \sum_{i=1}^{l} \sum_{j=1}^{n_i} \delta_{i,j}^{w} + \sum_{i=1}^{l} \delta_{i}^{a} \tag{3}$$

Equation (3) can be derived mathematically by linearizing the output distortion using Taylor series expansion with the assumption that the neural network is continuously differentiable and quantization errors can be considered as small deviations. The mathematical derivation of Equation (3) is provided in supplementary material.

We then apply Lagrangian formulation [26] to solve objective function (2). The Lagrangian cost function of (2) is defined as

$$\mathcal{J} = \delta - \lambda \cdot \Big(\sum_{i=1}^{l} \sum_{j=1}^{n_i} R_{i,j}^{w} + \sum_{i=1}^{l} R_{i}^{a} - r \Big), \tag{4}$$

in which λ decides the trade-off between bit rate and output distortion. Setting the partial derivations of $\mathcal{J}$ to zero with respect to each $R^w_{i,j}$ and R^a_i and utilizing the additivity property in (3), we obtain the optimal condition

$$\frac{\partial \delta^w_{i,1}}{\partial r^w_{i,1}} = ... = \frac{\partial \delta^w_{i,n_i}}{\partial r^w_{i,n_i}} = \frac{\partial \delta^a_i}{\partial r^a_i} = \lambda, \tag{5}$$

for all $1 \leq i \leq l$. Equation (5) expresses that the slopes of all rate-distortion curves (output distortion versus bit rate functions) should be equal to obtain optimal bit allocation with minimal output distortion. According to (5), we are able to solve objective function (2) efficiently by enumerating slope λ and then choosing the point on each rate-distortion curve with slope equal to λ as solution.

The algorithm works as follows. Before optimization, we quantize each weight channel and activation layer with different bit widths and calculate the output distortion caused by quantization to generate the rate-distortion curve for each weight channel and activation layer. After that, we assign a real value to λ, and select the point with slope equal to λ on each curve. The selected points on all curves correspond to a group of solution for bit allocation. In practice, we explore multiple values for λ until the network bit rate exceeds constraint r. We randomly select 50 images from ImageNet dataset to calculate output distortion caused by quantization. Given the number of λ evaluated, t, and the total number of bit widths, b, the time complexity of optimization is $O((l + \sum_{i=1}^{l} {}_i) \cdot t \cdot b)$. The algorithm has only linear time complexity, which can find the answer in a few minutes on a normal CPU.

3.3 Choice of Quantizer

We adopt uniform quantizer in our approach. The quantization step size Δ is defined as a value of a power of 2, ranging from 2^{-16} to 2^0, where the one with minimal quantization error is selected. We clip all weights by $(-2^{b-1} \cdot \Delta, (2^{b-1} - 1) \cdot \Delta)$ and all activations by $(0, (2^b - 1))$, in which b is the quantization bit width. Note that our approach is compatible with other quantizers, including both uniform quantizer and non-uniform quantizer (e.g., K-Means [10]). Since the focus of this paper is not the design of quantizer, we only evaluate uniform quantizer in our approach. It is worth mentioning that applying a non-uniform quantizer could further improve the accuracy, but non-uniform quantizers are more complicated for computation, and require additional resources (e.g., look-up tables) for implementation. Similar as prior mixed-precision methods [30, 22, 31], our approach employs uniform quantizer, as it is more hardware-friendly and is straightforward for implementation.

3.4 Improving Performance on Hardware

We consider improving inference rate as a guide to the design of our quantization mechanism. Inference rate is defined as the maximum number of images that a neural network can process per unit time. Memory access, especially data

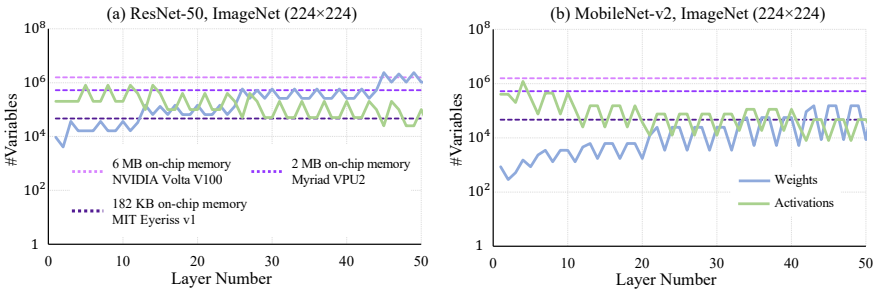

Fig. 3. The number of weights and activations across layers and the on-chip capacity on different hardware platforms.

movement from off-chip memory to on-chip memory, dominates inference time, rather than convolutional operations [13, 12]. We thus aim to maintain as many weights and activations as possible stored on-chip, and avoid data movement from off-chip memory to improve inference speed.

Our key insight is that the volume of weights and activations in some layers is particularly significant. As a result, part of weights and activations can not be stored on-chip, which leads to significant memory-access traffic to off-chip DRAM. Fig. 3 illustrates the number of parameters across layers and the on-chip memory capacity on different hardware platforms. As we can see, on-chip memory capacity is very limited, and the size of some layers exceeds the capacity. To this end, we restrict the quantization bit widths in these large layers to make sure that the size of these layers is less than on-chip memory capacity. Specifically, for layer i, we have an independent bit rate constraint,

$$\sum_{j=1}^{n_i}\left(K_{i,j}^{w}\cdot B_{i,j}^{w}\right)+K_i^a\cdot B_i^a\leq m_{on}, \tag{6}$$

in which $K_{i,j}^{w}$ denotes the number of weights of channel j in layer i, K_i^a denotes the number of activations in layer i, and m_{on} denotes the on-chip memory capacity. In practice, we relax (6) into two items, and incorporate them to objective function (2),

$$B_{i,j}^{w}\leq\frac{m_{on}}{\sum_j K_{i,j}^{w}+\frac{\beta}{1-\beta}K_i^a},\ B_i^a\leq\frac{\alpha m_{on}}{\frac{1-\beta}{\beta}\sum_j K_{i,j}^{w}+K_i^a}, \tag{7}$$

for all $1\leq i\leq l$ and $1\leq j\leq n_i$, where α and β are two hyperparameters, ranging from 0 to 1. Incorporating constraints (7) into (2), we have the objective function with the bit rate of each weight channel and activation layer constrained to improve hardware performance

$$\begin{aligned} &\min \delta = \|\mathbf{Y} - \widehat{\mathbf{Y}}\|_2^2 \;\; s.t. \sum_{i=1}^{l}\sum_{j=1}^{n_i} R_{i,j}^w + \sum_{i=1}^{l} R_i^a \le r, \\ &B_{i,j}^w \le \frac{m_{on}}{\sum_j K_{i,j}^w + \frac{\beta}{1-\beta} K_i^a}, \; B_i^a \le \frac{\alpha \cdot m_{on}}{\frac{1-\beta}{\beta}\sum_j K_{i,j}^w + K_i^a} \end{aligned} \tag{8}$$

Note that the optimization of (8) is the same as that of (2). The only difference is that in (8) we have different search range for quantization bit widths. In (2), the range is from 1 to b where $b = 16$ is the maximal bit width, while in (8), it is from 1 to $\frac{m_{on}}{\sum_j K_{i,j}^w + \frac{\beta}{1-\beta} K_i^a}$ for weight channels, and is from 1 to $\frac{\alpha m_{on}}{\frac{1-\beta}{\beta}\sum_j K_{i,j}^w + K_i^a}$ for activation layers. Incorporating bit rate constraints into objective function (2) does not increase the search time. Actually, it even slightly decreases the time as it reduces the search range of bit widths. Fig. 2 illustrates examples of the optimization procedure with and without constraints (7).

3.5 Discussion

Prior works [30, 22] typically use hardware simulators to guide the design of quantizatino mechanism. Implementing a simulator is complicated and calculating simulation results costs time. Alternatively, we provide a simple yet effective way to improve performance on hardware platforms. We directly restrict the bit rate of each layer to have weights and activations saved on-chip. Our method is easy to implement and does not cause additional overhead to optimization. Another advantage is that, once the rate-distortion curves are generated, our approach is able to find the bit allocation under any network size, by just changing the slope λ. This is better than most prior works which need to re-run the whole method every time searching the bit allocation for a network size.

4 Experiments

We report experimental results in this section. We first show quantization results on four deep neural networks, ResNet-18 [14], ResNet-34, ResNet-50 and MobileNet-v2 [25], on the ImageNet dataset [5]. We then report the results of inference rate on two hardware platforms, Google TPU [16] and MIT Eyeriss [3].

4.1 Parameter Settings

We set hyperparameters α and β as values between 0 and 1, where the combination with best hardware performance is chosen. We enumerate slope λ from -2^{-20} to -2^{20} until network size meets constraint r. Similar as prior works [30, 22], we fine-tune the model after quantization up to 100 epochs with learning rate 0.0001. Our approach is able to obtain high accuracy after 2 epochs, and can almost converge after 5 to 10 epochs. We fine-tune 100 epochs to make sure that

Method	ResNet-18	ResNet-34	ResNet-50
Original [14]	69.3%	73.0%	75.5%
LQ-Nets [33]	64.9%	68.8%	71.5%
PTG [41]	-	-	70.0%
DSQ [11]	65.2%	70.0%	-
QIL [17]	65.7%	70.6%	-
ALQ [23]	66.4%	71.0%	-
APoT [21]	67.3%	70.9%	73.4%
SAT [15]	65.5%	-	73.3%
LSQ [8]	67.6%	71.6%	73.7%
AUXI [40]	-	-	73.8%
DMBQ [34]	67.8%	72.1%	-
RDO-Q (Ours)	**68.8%**	**72.6%**	**75.0%**

Table 2. Top-1 image classification accuracy at 2 bits on ImageNet.

Method	Top-1 Accuracy	Top-5 Accuracy
Original [25]	71.8%	90.2%
HAQ [30]	67.0% (-4.8)	87.3 (-3.1)%
DQ [29]	69.7% (-2.1)	-
DSQ [11]	64.8% (-7.0)	-
AutoQ [22]	69.0% (-2.8)	89.4% (-0.8)
SAT [15]	71.1% (-0.7)	89.7% (-0.5)
LLSQ [35]	67.4% (-4.4)	88.0% (-2.2)
RDO-Q (Ours)	**71.3% (-0.5)**	**90.0% (-0.2)**

Table 3. Results on MobileNet-v2 at 4 bits on ImageNet.

the quantized network completely converges. Straight-through estimator (STE) [2] is applied to perform back-propagation through non-differentiable quantization functions in fine-tuning. We randomly select 50 images from ImageNet to generate the rate-distortion curves. We noticed that using more images to generate the curve doesn't affect the final accuracy.

4.2 Quantization Results

Table 2 and Table 3 list the results on four deep neural networks, ResNet-18, ResNet-34, ResNet-50, and MobileNet-v2, when weights and activations are quantized to very low bit widths (i.e., 2 bits or 4 bits). As our approach allocates unequal bit widths to different weight channels and activation layers, we report the results when networks are quantized to the target size on average for fair comparison, same as prior works [29, 22]. Our approach, named as Rate-Distortion-Optimized Quantization (RDO-Q), improves state-of-the-arts on the four neural networks. Specifically, our approach outperforms SOTAs by 1.0%, 0.5%, and 1.2%, on ResNet-18, ResNet-34, and ResNet-50, respectively.

Method	Accuracy top-1	Google TPU	MIT Eyeriss
ResNet-50			
Original [14]	75.5%	361	5.6
DoReFa+PACT [4]	76.5%	646	12.6
DoReFa+PACT [4]	72.2%	920	13.7
RDO-Q+HA (Ours)	76.5%	769	13.9
RDO-Q+HA (Ours)	76.2%	904	15.0
RDO-Q+HA (Ours)	75.0%	1254	17.0
MobileNet-V2			
Original [25]	71.1%	1504	64
DoReFa+PACT [4]	71.2%	1698	104
DoReFa+PACT [4]	70.4%	1764	108
HAQ [30]	71.2%	2067	124
HAQ [30]	68.9%	2197	128
RDO-Q+HA (Ours)	71.3%	2197	127
RDO-Q+HA (Ours)	71.0%	2207	128
RDO-Q+HA (Ours)	70.9%	2256	130

Table 4. Inference Rate on Google TPU and MIT Eyeriss. We show the results of our approach with hardware-aware (HA) constrains in this table.

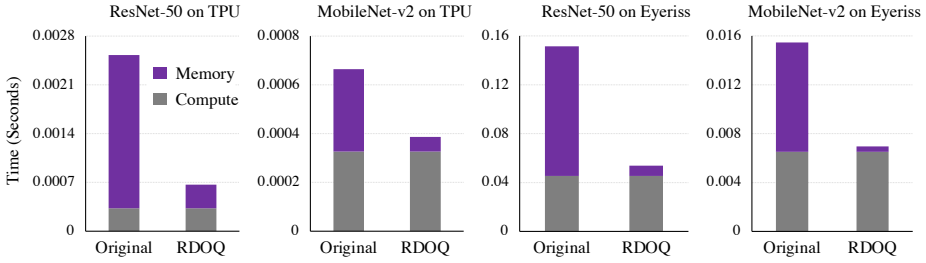

Fig. 4. A breakdown for the compute time and memory time on two hardware platforms, Google TPU and MIT Eyeriss.

4.3 Performance on Hardware Platforms

We examined the inference rate on two hardware platforms, Google TPU [16] and MIT Eyeriss [3], both of which are state-of-the-art architectures, inspired by current embedded and high-performance neural-network-targeted accelerators. We adopt the SCALE-Sim software [24] to simulate the time cycles of ResNet-50 and MobileNet-v2, when mapped into the two considered hardware platforms.

Table 4 illustrates the inference rate on TPU and Eyeriss. Our approach significantly improves the inference rate, compared with originally uncompressed neural networks. We speed up the inference rate by 3.5x and 3.0x for ResNet-50 on TPU and Eyeriss, and by 1.5x and 2.0x for MobileNet-V2 on TPU and Eyeriss, without hurting the accuracy (loss $\leq 0.5\%$). We also compare our approach with the competitive mixed-precision quantization method, HAQ [30],

and the competitive equal bit quantization method, DoReFa+PACT [38, 4]. Our approach outperforms both HAQ and DoReFa+PACT. We notice that although equal bit quantization method DoReFa+PACT obtains superior quantization results, the performance on hardware platforms is not high. This is because DoReFa+PACT is not hardware-aware as they do not optimize the hardware performance in their quantization mechanism.

Note that both TPU and Eyeriss do not support computation with mixed precision. The computation on TPU is with 8-bit integers, and that on Eyeriss is with 16-bit integers. We clarify that we pad quantized parameters to 8-bit or 16-bit integers when we do the computation on TPU or Eyeriss, respectively. Although TPU and Eyeriss do not support computation with mixed precision, they still benefit from mixed-precision quantization, because the bottleneck of deep neural networks is memory access and mixed-precision quantization helps to further reduce the network size to reduce memory. Fig. 4 illustrates a breakdown of the inference time for memory access and computation on TPU and Eyeriss. We believe that hardware with specialized integer arithmetic units can further improve the performance, since the computation can also be more efficient. As the main focus of our paper is the quantization algorithm, we did not implement a hardware architecture to support mixed-precision computation.

4.4 Time Cost

Table 5 lists the time of our approach to find channel-wise bit allocation on four deep neural networks. Our approach takes about a few minutes on a normal CPU (Intel Core i7 6600U CPU with 2.60 GHZ) to find the solution. We also evaluated HAQ [30] — the competitive mixed-precision quantization method built upon reinforcement learning. As we can see, the reinforcement-learning-based approach requires several days on multiple GPUs to search the bit allocation for one time, which is orders of magnitude slower. Our approach provides an alternative way to quickly explore the hyperparameter space of bit widths, and is particularly suitable for the case without powerful computation resources.

4.5 Distributions of Bit Rate Across Layers

Fig. 5 illustrates the distribution of the bit rate under different hardware-aware constraints. Intuitively, by balancing the size between layers, our approach assigns lower bit widths to large layers and higher bit widths to small layers, to meet the constraints. Fig. 6 illustrates the number of activations that hardware platforms can accommodate, under different bit widths given to the activation layer. We can see that on both ResNet-50 and MobileNet-v2, some layers have more than 1 million activations, and the bit widths assigned to these layers have to be very small when the on-chip memory capacity is only a few KBs.

4.6 Discussion of Additivity Property

Based on our mathematical analysis, the additivity property holds if quantization errors can be considered as small deviations. In that case, second (or higher)

Method	HAQ [30]	RDO-Q
Device	GPU × 4	CPU × 1
ResNet-18	-	2 minites
ResNet-34	-	3 minites
ResNet-50	117 hours	7 minites
MobileNet-v2	79 hours	6 minites

Table 5. Time cost to find channel-wise bit allocation.

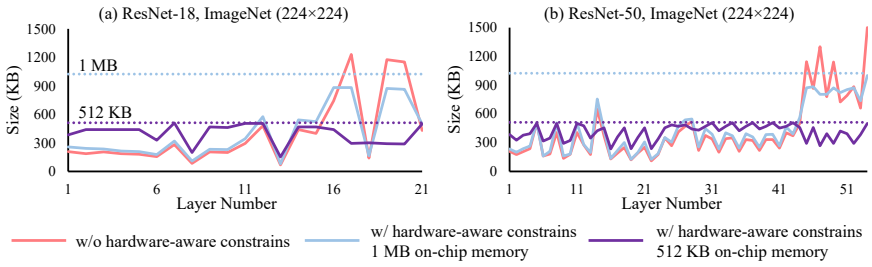

Fig. 5. Distributions of bit rate across layers given different memory constraints.

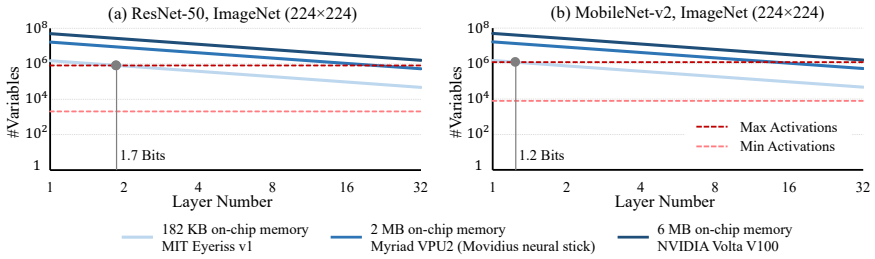

Fig. 6. The number of variables that the on-chip memory on specific hardware platforms can accommodate, give different bit widths per variable. The minimum and maximum numbers of activations that a single layer can have are highlighted.

order items in Taylor series expansion are small values, and we can use the zero and first order items to approximate output distortion. We tested the Mean Square Error (MSE) of quantized parameters in practice, and found that MSEs are really small deviations. Moreover, we evaluated the relation between the output distortion and the sum of individual distortion using real examples. The practical results also consist with our theoretical analysis. The mathematical analysis of additivity property is provided in supplementary material.

4.7 Implementation Details of Optimization

We formulate the quantization of weight channels and activation layers as a rate-distortion optimization problem, as illustrated in Section 3. We utilize the addtivity of the output distortion and apply the classical Lagrangian formulation

to solve (2). The optimal condition in Equation (4) expresses that the slopes of all output distortion versus bit rate curves should be equal. Based on this equation, the optimziation problem can be solved by enumerating λ and selecting the point with slope equal to λ on each rate-distortion curve.

Specifically, we first generate the rate-distortion curve for each channel and activation layer. In our case, the rate-distortion curves are comprised of discrete points. For example, if the range of bit width is from 1-bit to 8-bits, then a rate-distortion curve is a discrete curve with 8 points. We enumerate λ and select the point on each curve with slope equal to λ. We may enumerate different λ to find the best solution with minimal output distortion under the size constraint. Assume that we have N curves and M points on each curve. The total time complexity to generate rate-distortion curves and find optimal bit allocation is $O(I \cdot M \cdot N \cdot C + K \cdot M \cdot N)$, where I denotes the number of images used to generate rate-distortion curves, C is a constant which denotes the cost to perform the inference, and K is the total number of slope λ to be evaluated.

5 Conclusion

Channel-wise bit allocation brings higher quantization precision and superior accuracy. Our approach provides an ultra-fast way to explore the hyperparameter space of channel bit widths with linear time complexity, using Lagrangian Formulation. The quantization of deep neural networks is formulated as a rate-distortion optimization problem, and the fast optimization method is proposed, by utilizing the additivity of output distortion. Moreover, we consider the impact on hardware platforms in the design of our quantization mechanism, and present a simple yet effective method to improve hardware performance. We restrict the bit rate of each layer to allow as many weights and activations as possible saved on-chip, and add hardware-aware constraints in our objective function to improve inference rate on target hardware platforms. The hardware-aware constraints can be incorporated into our objective function seamlessly, without incurring additional overhead for optimization. Extensive experiments show that our approach improves state-of-the-arts on four deep neural networks. Hardware simulation results demonstrate that our approach is able to accelerate deep learning inference considerably on two hardware platforms.

Acknowledgments

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its Funds (Project Number A1892b0026, A19E3b0099, and C211118009). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the A*STAR.

References

1. Banner, R., Nahshan, Y., Hoffer, E., Soudry, D.: Post-training 4-bit quantization of convolution networks for rapid-deployment. arXiv preprint arXiv:1810.05723 (2018)
2. Bengio, Y., Leonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. In: arXiv:1308.3432 (2013)
3. Chen, Y.H., Krishna, T., Emer, J.S., Sze, V.: Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE Journal of Solid-State Circuits **52**(1), 127–138 (2016)
4. Choi, J., Wang, Z., Venkataramani, S., Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, K.G.: Pact: Parameterized clipping activation for quantized neural networks. In: arXiv (2018)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR (2009)
6. Dong, Z., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Hawq: Hessian aware quantization of neural networks with mixed-precision. In: arXiv (2019)
7. Elthakeb, A.T., Pilligundla, P., Mireshghallah, F., Yazdanbakhsh, A., Esmaeilzadeh, H.: Releq: A reinforcement learning approach for deep quantization of neural networks. In: NeurIPS Workshop on ML for Systems (2018)
8. Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.: Learned step size quantization. In: ICLR (2020)
9. Gao, W., Liu, Y.H., Wang, C., Oh, S.: Rate distortion for model compression: From theory to practice. In: International Conference on Machine Learning. pp. 2102–2111. PMLR (2019)
10. Gersho, A., Gray, R.: Vector quantization and signal compression. In: Kluwer Academic Publishers (1991)
11. Gong, R., Liu, X., Jiang, S., Li, T., Hu, P., Lin, J., Yu, F., Yan, J.: Differentiable soft quantization: Bridging full-precision and low-bit neu- ral networks. In: ICCV (2019)
12. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In: ICLR (2016)
13. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. arXiv preprint arXiv:1506.02626 (2015)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
15. Jin, Q., Yang, L., Liao, Z., Qian, X.: Neural network quantization with scale-adjusted training. BMVC (2020)
16. Jouppi, N., et al.: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture. pp. 1–12. ISCA '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3079856.3080246, http://doi.acm.org/10.1145/3079856.3080246
17. Jung, S., Son, C., Lee, S., Son, J., Han, J., Kwak, Y., Hwang, S.J., Choi, C.: Learning to quantize deep networks by optimizing quantiza- tion intervals with task loss. In: CVPR (2019)
18. Khoram, S., Li, J.: Adaptive quantization of neural networks. In: ICLR (2018)
19. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012)
20. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. In: Nature (2015)

21. Li, Y., Dong, X., Wang, W.: Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In: ICLR (2020)
22. Lou, Q., Guo, F., Kim, M., Liu, L., Jiang, L.: Autoq: Automated kernel-wise neural network quantizations. In: ICLR (2020)
23. Qu, Z., Zhou, Z., Cheng, Y., Thiele, L.: Adaptive loss-aware quantization for multi-bit networks. In: arXiv (2020)
24. Samajdar, A., Zhu, Y., Whatmough, P.N., Mattina, M., Krishna, T.: Scale-sim: Systolic CNN accelerator. CoRR **abs/1811.02883** (2018), http://arxiv.org/abs/1811.02883
25. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: arXiv (2018)
26. Shoham, Y., Gersho, A.: Efficient bit allocation for an arbitrary set of quantizers (speech coding. In: IEEE Transactions on Acoustics, Speech, and Signal Processing (1988)
27. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
28. Taubman, D.S., Marcellin, M.W.: Jpeg2000 image compression fundamentals, standards and practice (2002)
29. Uhlich, S., Mauch, L., Cardinaux, F., Yoshiyama, K., Garcia, J.A., Tiedemann, S., Kemp, T., Nakamura, A.: Mixed precision dnns: All you need is a good parametrization. arXiv preprint arXiv:1905.11452 (2019)
30. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: HAQ: Hardware-aware automated quantization with mixed precision. In: CVPR (2019)
31. Wu, B., Wang, Y., Zhang, P., Tian, Y., Vajda, P., Keutzer, K.: Mixed precision quantization of convnets via differentiable neural architecture search. In: ICLR (2019)
32. Yang, L., Jin, Q.: Fracbits: Mixed precision quantization via fractional bit-widths. arXiv preprint arXiv:2007.02017 (2020)
33. Zhang, D., Yang, J., Ye, D., Hua, G.: Lq-nets: learned quantization for highly accurate and compact deep neural networks. In: ECCV (2018)
34. Zhao, S., Yue, T., Hu, X.: Distribution-aware adaptive multi-bit quantization. In: CVPR (2021)
35. Zhao, X., Wang, Y., Cai, X., Liu, C., Zhang, L.: Linear symmetric quantization of neural networks for low-precision integer hardware. ICLR (2020)
36. Zhe, W., Lin, J., Aly, M.S., Young, S., Chandrasekhar, V., Girod, B.: Rate-distortion optimized coding for efficient cnn compression. In: DCC (2021)
37. Zhe, W., Lin, J., Chandrasekhar, V., Girod, B.: Optimizing the bit allocation for compression of weights and activations of deep neural networks. In: ICIP (2019)
38. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. In: arXiv preprint arXiv:1606.06160 (2016)
39. Zhou, Y., Moosavi-Dezfooli, S.M., Cheung, N.M., Frossard, P.: Adaptive quantization for deep neural network. In: AAAI (2018)
40. Zhuang, B., Liu, L., Tan, M., Shen, C., Reid, I.: Training quantized neural networks with a full-precision auxiliary module. In: CVPR (2020)
41. Zhuang, B., Shen, C., Tan, M., Liu, L., Reid, I.: Towards effective low-bitwidth convolutional neural networks. In: cvpr (2018)