NIPQ: Noise Injection Pseudo Quantization for Automated DNN Optimization

Sein Park, Junhyuk So, Juncheol Shin, and Eunhyeok Park
Graduate School of Artificial Intelligence
POSTECH
seinpark, junhyukso, jchshin, eh.park@postech.ac.kr

Abstract

The optimization of neural networks in terms of computation cost and memory footprint is crucial for their practical deployment on edge devices. In this work, we propose a novel quantization-aware training (QAT) scheme called noise injection pseudo quantization (NIPQ). NIPQ is implemented based on pseudo quantization noise (PQN) and has several advantages. First, both activation and weight can be quantized based on a unified framework. Second, the hyper-parameters of quantization (e.g., layer-wise bit-width and quantization interval) are automatically tuned. Third, after QAT, the network has robustness against quantization, thereby making it easier to deploy in practice. To validate the superiority of the proposed algorithm, we provide extensive analysis and conduct diverse experiments for various vision applications. Our comprehensive experiments validate the outstanding performance of the proposed algorithm in several aspects.

1 Introduction

Although the accuracy or output quality has been used as a decisive metric to evaluate the superiority of neural networks, the computation cost and memory footprint are also critical measures to determine their practical utility in reality [1, 2]. To improve the usability of advanced neural networks for large-scale servers as well as embedded platforms, neural network optimization is gaining increasing attention these days. Quantization, one of the representative optimization techniques, reduces the memory footprint by limiting the precision in smaller bit-widths. In particular, linear quantization can easily utilize hardware acceleration via integer arithmetic [3, 4, 5, 6] or bit-serial acceleration [7, 8, 9]. If marginal accuracy degradation is acceptable, one can enjoy the benefit of low footprint and high performance via linear quantization. Moreover, diverse studies have been actively conducted to reduce accuracy loss within the given resource budget to popularize the benefit of quantization.

To mitigate the accuracy degradation, quantization-aware training (QAT) has emerged as a major quantization scheme that trains a neural network with quantization operators to adapt the network to the low-precision representation. While the quantization operator is not differentiable, the straight-through estimator (STE) [10] allows the backpropagation of the quantized data based on linear approximation. This approximation works well in redundant networks with moderate precision (>4-bit). Thus, not only early studies [11, 12, 13] but also advanced ones [14, 15, 16] have proposed diverse QAT schemes based on STE and shown that complex neural networks (i.e., ResNet-18 [17]) can be quantized into 4-bit without accuracy loss.

However, STE bypasses the approximated gradient, not the true gradient, and many studies have pointed out that it could incur instability and bias during training. Bi-Real net [18] and DSQ [19] show that replacing STE with high-order polynomial or multiple soft-step functions results in better convergence in low-precision representations. In the case of HLHLp [20] and PROFIT [21], they propose specialized training pipelines to mitigate the instability induced by STE approximation. In
particular, in the case of PROFIT, the instability derived from STE is a major source of accuracy degradation for the optimized networks (e.g., MobileNet-v2/v3 [22, 23]). More recently, NICE [24], PQ [25], and DiffQ [26] have proposed alternative QAT approaches that finetune the network with pseudo quantization noise (PQN) [27] instead of using STE. While those studies are only applicable to weight quantization, they show the potential of pseudo quantization. In this study, we present a unified QAT scheme for both activation and weight based on pseudo quantization, resulting in better output quality than the previous STE-based approaches.

On the other hand, another hot topic in quantization studies is layer-wise bit-width optimization because the layer-wise sensitivity varies significantly and this enables the highest accuracy within the given constraints to be achieved. Finding the optimal number of bits is a significant challenge since there are multiple bit-width candidates per layer. Various methods, such as RL-based [28, 29, 30], Hessian-based [31, 32, 33], and differentiable [34, 16] methods, have been proposed. Still, there is room for improvement given the requirement of complicated hyper-parameter tuning, hand-crafted bit-width assignment and the instability of STE approximation. In this paper, our fully automated optimization process for layer-wise bit-width assignment considers per-layer quantization sensitivity.

In this paper, we propose a novel QAT algorithm called noise-injection pseudo quantization (NIPQ). NIPQ is implemented based on PQN and designed to tune quantization hyper-parameters automatically. When we train a network with NIPQ with a penalty loss term, all the network parameters (e.g., the layer-wise bit-width and quantization interval) are jointly optimized without the instability induced by STE approximation. In addition, the QAT process of NIPQ inherently regularizes the sum of the Hessian trace of a neural network, which enables the network to endure additional noise with minimal quality degradation. In short, NIPQ has four representative advantages. First, both activation and weight are quantized based on a unified framework. Second, All the quantization hyper-parameters (e.g., layer-wise bit-width and quantization interval) are jointly optimized. Third, network robustness is enhanced to make the optimized network easier to deploy in practice. Finally, NIPQ shows state-of-the-art accuracy with minimal cost in various applications.

2 Related Work

Due to its practical usefulness, various studies have been proposed for multi-bit linear quantization [13, 14, 35, 15]. Most of these studies are based on STE and demonstrate their performance in well-known networks such as AlexNet [36] and ResNet. The accuracy has been improved significantly by learning quantization hyper-parameters in a differentiable form. However, in optimized networks such as MobileNet-v2, accuracy loss induced by STE instability has been reported for both activation [19] and weight [21]. In order to mitigate the accuracy loss, complex pipeline and non-linear approximation have been proposed. These attempts have succeeded in quantizing the optimized network in low precision but have significantly increased the complexity and cost of QAT. In this work, NIPQ implements a QAT process without STE, enabling stable convergence without additional cost or complexity.

Mixed-precision studies focus on allocating layer-wise or group-wise bit-width in consideration of precision sensitivity of each layer to minimize accuracy drop within a given resource constraints. This goal is achieved via various methods, e.g., RL-based [28, 29, 30], Hessian-based [31, 32, 33], and differentiable [34, 16] algorithms. However, RL-based and Hessian-based methods are relatively complex and require a lot of parameter adjustments, and differentiable algorithms still suffer from STE approximation error. In this work, we reinterpret the differentiable bit-width tuning in terms of PQN-based QAT and greatly simplify mixed-precision quantization.

Robust quantization [37, 38, 39] aims to guide the convergence of the network toward a smooth and flat loss surface based on additional regularization. The robustness of neural networks is highly beneficial for deploying noisy devices or low-precision ALUs. In the case of NIPQ, it inherently improves the robustness during QAT with PQN. In particular, it is observed for the first time that the robustness of activation is enhanced as well as weight (Section 5).

The most relevant studies are NICE [24] and DiffQ [26]. Those studies are also based on PQN and have shown the potential of PQN-based QAT. However, both studies are only applicable for weight, and the hyper-parameter tuning is not automated in NICE. In NIPQ, we design a PQN-based unified QAT pipeline applicable to both activation and weight and fully automate the tuning of quantization hyper-parameters.
3 Noise injection pseudo quantization (NIPQ)

NIPQ is a QAT scheme that follows the assumption of PQN \[27\] where the pseudo-noise is injected to approximate the distortion of the quantization operator. In the NIPQ training pipeline, the network is updated with the injected noise and becomes robust to it, making the network tolerant to the quantization. This alternative approach enables training the parameters of a neural network without using STE approximation, resulting in stable convergence and better output quality. In addition, NIPQ is judiciously designed to optimize the quantization hyper-parameters (e.g., layer-wise bit-width and quantization interval) automatically. When we train a network with NIPQ with a penalty scalar \(\lambda\), the quantization configuration is automatically tuned to minimize the target cost (e.g., the total model size or the number of bit-operations \[33, 16\]), while maintaining the quality of output. This automated optimization is designed on top of two key insights of the training with scalable noise and a truncation boundary. First, we define the NIPQ algorithm and provide a detailed explanation of the insights for clarity.

3.1 NIPQ Implementation

Figure 1 shows the PyTorch-like pseudo-code for the NIPQ algorithm. For brevity, we explain NIPQ for data having a non-negative range. As shown in the figure, NIPQ performs two distinguishable actions depending on the operation mode. In noise injection mode, the pseudo-noise is injected to approximate the quantization noise induced by the rounding operator. In quantization mode, the LSQ-like operation \[15\] is used to quantize the input data into the low-precision representation. As for quantization hyper-parameters, NIPQ utilizes two per-layer learnable parameters, \(\alpha\) and \(\text{bit}\) for the truncation boundary and number of bit-width, respectively. The number of available quantization levels is \(N_{\text{lv}} = 2^{\text{bit}}\). Note that we adopt the continuous approximation of bit-width \[16, 26\]. It allows us to tune the layer-wise bit-width for minimizing loss through gradient descent, which greatly simplifies the hyper-parameter tuning process. The continuous bit-width is used during noise injection mode, while the rounded bit-width is used during quantization mode.\[4\] The data outside the truncation boundary \(\alpha\) is truncated to \(\alpha\), while the data within the quantization interval \([0, \alpha]\) is quantized; therefore, the quantization step size \(\Delta\) is equal to \(\frac{\alpha}{2^{\text{bit}} - 1}\).

While the behaviors of the two operation modes are different, their noise models are designed to be similar. In pseudo-noise injection mode, the iid samples of PQN are added to the target tensor as follows:

\[\tilde{x} = x + N(x|\text{bit}, \alpha),\]  

where \(N(x|\text{bit}, \alpha)\) represents the sampling function of the random variable for noise. According to previous studies on PQN \[23, 27\], the magnitude of noise should be equal to the step size \(\Delta\). Thereby,\[1\] According to our empirical analysis, the noise injection for bit-width (3rd line in Figure 1) can be replaced by STE without accuracy degradation.
the random variable is modeled as uniform distribution $U[-\Delta/2, \Delta/2]$ or a Gaussian distribution with zero-centered $\Delta/2$ variance. After adding the pseudo-noise, the output is truncated in the range of $[0, \alpha]$ as follows:

$$
\hat{x} = \begin{cases} 
0 & \text{if } \hat{x} < 0, \\
\alpha & \text{if } \hat{x} > \alpha, \\
\hat{x} & \text{otherwise.}
\end{cases}
$$

(2)

This noise-injected and truncated output $\hat{x}$ is used for the following operations of a neural network. The truncation function could be implemented based on the existing clamping function, but it is crucial that it be able to bypass the gradient of the truncated elements to $\alpha$. By training the network with noise injection mode, the internal representation of networks turns robust to the quantization operator.

In quantization mode, the quantization operator is used instead of pseudo noise. The input data $x$ is mapped to the low-precision representation through the transformation functions as follows:

$$
\bar{x} = \text{clamp}(x/\Delta, 0, 2^{\text{bit}} - 1)
$$

(3)

$$
\bar{x} = \text{round}(\bar{x})
$$

(4)

$$
\check{x} = \bar{x} \times \Delta.
$$

(5)

Note that NIPQ in quantization mode follows the LSQ quantization function, except the step size is not a single learnable parameter but expressed by the relationship between $\alpha$ and bit. By default, quantization mode is used during inference. However, when this mode is used during training, the conventional STE is used to bypass the gradient except for bit-width, which is not updated in this mode. The usage of quantization mode during training is provided in Section 3.3.

### 3.2 Automated Quantization Hyper-parameter Tuning via Training with Pseudo-noise

One of the most attractive characteristics of NIPQ is the automated layer-wise tuning of quantization hyper-parameters during the QAT process. Unlike the previous mixed-precision quantization methods, NIPQ does not require any unnecessary additional computation to analyze layer-wise sensitivity against quantization. This automated optimization is achievable solely by relying on the convergence property of training with scalable noise and the truncation value; the scale of noise tends to converge to zero (Lemma 1), and the truncation boundary tends to increase continuously (Lemma 2).

Lemma 1 and Lemma 2 can be validated based on the numerical basis and intuitive analysis. For Lemma 1, let us consider the situation of training a neural network with an additive uniform noise $U[-\epsilon, \epsilon]$ whose magnitude is determined by a learnable parameter $\epsilon$. The objective function can be expressed and approximated via Taylor expansion as follows:

$$
E_{\delta \sim U[\epsilon, \epsilon]}[L(x, w + \delta)] 
\approx E_{\delta \sim U[\epsilon, \epsilon]} \left[ L(x, w) + \delta \nabla_{w} L(x, w) + \frac{1}{2} \delta^{T} \nabla_{w}^{2} L(x, w) \delta \right] 
$$

(6)

$$
= L(x, w) + \frac{\epsilon^{2}}{2} \text{Tr} \left\{ \nabla_{w}^{2} L(x, w) \right\},
$$

(7)

where $x$ and $w$ represent the input and parameter, respectively, and the term related to the first derivative is removed since $E[\delta] = 0$ and the off-diagonal elements of the second derivative term become 0 because it relates to the expectation of multiplication of two i.i.d. samples. When the loss is converged to the minima point, the sum of eigenvalues $\nabla_{w}^{2} L(x, w)$ should have a non-negative value. To minimize the average loss after training, the magnitude of noise $\epsilon$ should converge to 0. Otherwise, when $\epsilon$ has non-zero values, the loss surface is guided to be converged to the minima having a lower Hessian trace value [39].

Lemma 2 was validated in a previous study, PACT [14], introduced the truncation boundary as a learnable parameter and updated it through gradient descent. In PACT, the quantization operator is defined as follows:

$$
\check{x} = 0.5 \cdot (|x| - |x - \alpha| + \alpha),
$$

(9)

$$
Q(x) = \text{round}(\check{x} \cdot (n_{lv} - 1)/\alpha) \cdot \alpha/(n_{lv} - 1).
$$

(10)

In this equation, the input values larger than $\alpha$ are truncated to $\alpha$, and their corresponding gradients are bypassed to $\alpha$ which enables $\alpha$ to be updated through back propagation. Because PACT forces
the truncation of the values larger than $\alpha$, the gradient descent continuously increases the value of $\alpha$.

In the NIPQ framework, the scale of noise is proportional to the truncation boundary, because $\Delta = \frac{\alpha}{\sigma^2}$. According to Lemma 1 and Lemma 2, the scale of noise tends to be reduced during training while the truncation boundary tends to be increased. Due to these convergence counterpart trends, the truncation boundary converges to a stable point, balancing the trade-off of the pseudo-quantization error and truncation error. While the dynamic range of the quantized data is sacrificed, this balanced truncation greatly reduces the quantization error in the given bit-width, resulting in higher accuracy at the same bit-width.

In addition to the truncation boundary, NIPQ also adjusts the layer-wise bit-width automatically. To restrict the storage/computation cost of a network, an additional penalty term should be introduced. For instance, in the experimental section, the activation and weight bit-width is bounded directly to the target bit-width by introducing the penalty loss with scale terms of $\lambda_a$ and $\lambda_w$, respectively. If the $i$-th layer’s weight bit-width is given as $b_{w,i}$, the average bit-width of weight is tuned for the target bit $b_t$ as follows:

$$
\min L_{target} + \lambda_w \cdot h((\frac{\sum_i w_i \cdot e_i}{\sum_i e_i}) - b_t),
$$

where $L_{target}$ is the target objective loss function, $e_i$ is the number of elements in the $i$-th layer, and $h(\cdot)$ represents the Huber loss. As explained, the noise scale tends to decrease. When $\alpha$ is converted, the bit-width tends to increase to reduce the noise scale. When the average bit-width is restricted, the layer-wise bit-width is assigned considering the sensitivity of the layer against quantization. For instance, if the $i$-th layer is more vulnerable to quantization than the $j$-th layer, the sum of the eigenvalues of Hessian of the $i$-th layer should be larger than that of the $j$-th layer. Within the restricted bit-width resource, a larger bit-width is preferred to be assigned to the $i$-th layer than the $j$-th layer to minimize the error scale $\epsilon$ which is inversely proportional to $2^{b_{t,i}} - 1$. Therefore, the layerwise bit-width is automatically assigned considering the layer-wise sensitivity within the available resource budget, and the corresponding $\alpha$ is also jointly updated to balance the pseudo-noise error and truncation error. This carefully designed framework enables the quite easy automated tuning of quantization hyper-parameters. Note that the activation bit-width could likewise be assigned automatically. Besides, when the number of bit-operations [16] [33] needs to be restricted, it is used as a constraint loss directly, instead of using the average bit-width loss.

### 3.3 Practical Details

In practice, empirical engineering could be helpful to achieve higher accuracy in addition to the automated optimization process with theoretical support. For $\alpha$, we apply Softplus nonlinearity to force the value to have a non-negative range. In the case of bit, the actual bit-width is expressed as $\text{bit} = 2 + \sigma(b) \cdot 12$, where $\sigma(\cdot)$ is a sigmoid function and $b$ is an unbounded continuous parameter. As a result, the bit-width could be in $[2, 14]$. We use Gaussian noise instead of uniform noise, because empirically, it shows better results. A similar observation was reported in the study on DiffQ [20].

The QAT pipeline consists of two stages: pre-training with noise injection mode and post-training with quantization mode. The first stage produces the robust network as a good initial condition, and the second stage specializes the network for the quantization operator. In the early stage, the quantization parameters and network parameters are jointly updated, and the network becomes robust to the injected noise. In the later stage, the network converges for the specific noise of the quantization operator, and the internal statistics (e.g., running mean and variance of batch normalization) are adjusted for the quantization operator. Note that bit is fixed, and only $\alpha$ is updated in this stage. This two-step training greatly improves accuracy, and we adapt this pipeline over the entire application.

### 4 Sensitivity-aware Layer-wise Mixed-precision Quantization

Mixed-precision quantization aims to improve accuracy while minimizing the overall storage footprint and computational cost. To maintain accuracy within the resource budget, more bit-width should be assigned to the sensitive layer to minimize overall quantization errors. Recently, the Hessian of the loss function has often been used as a metric of sensitivity against quantization [31] [32] [33]. The smaller the sum of the Hessian trace, the lower the sensitivity, and vice versa. Figure 2 visualizes the assigned bit-width of activation and weight and the corresponding sum of the Hessian trace estimated...
Figure 2: Layer-wise assigned bit-width of activation (top) and weight (bottom) and corresponding sensitivity measure (the sum of Hessian trace) of MobileNet-v2 on CIFAR-100 dataset [40].

Figure 3: Loss landscapes [41] of quantized MobileNet-V2 fine-tuned by STE-based LSQ [15] (left) and NIPQ (right) on CIFAR-100 dataset.

via the Hutchinson algorithm [32] when applying NIPQ to MobileNet-v2 on the CIFAR-100 dataset. As shown in the figure, the more sensitive the layer is, the more bit-width is assigned. Note that NIPQ does not have any additional stages that measure the sensitivity of the layer. Instead, we just train the network with an additional penalty term to restrict the average precision to 3-bit. The NIPQ algorithm allocates precision by itself, considering the sensitivity of the target layer, thereby quantizing the network with the highest accuracy as efficiently as possible within the resource budget.

5 Robust Quantization for Practical Deployment

Another strength of NIPQ is the enhanced robustness of the network against unexpected noise. The robustness of the network brings diverse advantages to deploying the network in practice. For instance, the analog current sum-based device has significant energy efficiency but has inevitable noise induced by process variation or temperature drift, which causes instability of output. Even in the case of digital NPU, the low-precision implementation is fragmented because there are hundreds of hardware manufacturers [37]. When we deploy a neural network after QAT to the target device, the implementation difference could introduce unexpected distortion of the quantization configuration, resulting in accuracy degradation. The robustness of the network allows accuracy to be maintained in this environment, so securing this property is a significant advantage.

As explained in Section 3.2, noise injection-based QAT regularizes the sharpness of the loss surface. Figure 3 visualizes the sharpness of the loss surface by measuring the change of loss values after adding noise on top of the trained weight along the two random vectors. As shown in the figure,
NIPQ converges to a flat and smooth loss surface. NIPQ is designed to force the network to adapt to the pseudo-noise that generalizes the quantization noise, but it is also quite helpful to enhance the robustness of network parameters.

In addition, NIPQ QAT approximates the quantization interval by corresponding; this approximation enhances the robustness of the quantization parameters as well as the network parameters. Figure 4 shows the result of measuring the accuracy while changing the quantization step size or the truncation interval. The more robust the network, the more it can endure the change of the quantization configuration. As shown in the figure, NIPQ-based quantization shows comparable or superior results to the previous best algorithm for robustness, KURE. It is especially worthy that existing studies have focused on improving the robustness of weight only [38, 37] but that NIPQ also improves the robustness of activation by a large margin. To the best of our knowledge, this is the first time a study has reported activation robustness, which is a crucial characteristic in deploying networks in a noisy environment.

6 Importance of Truncation for Quantization

The last line of related study is DiffQ [26], which is based on PQN-based QAT for weight quantization. The key difference from NIPQ is that linear quantization is applied based on the min-max value of weight instead of truncation. However, to minimize quantization errors with the limited bit-width, the presence of truncation is extremely helpful. In general, the data distribution of a natural network follows a bell-shaped distribution, where the majority of data is concentrated near zero. Because min-max quantization increases the quantization interval near zero, the quantization error greatly increases. Figure 5 compares the accuracy between NIPQ with truncation and NIPQ with min-max quantization for weight quantization of ResNet-18 (left) and MobileNet-v2 (right) on CIFAR-100 dataset. The colored lines represent the mean and variance of accuracy with 10 times repetition.
Table 1: Top-1 accuracy (%) of quantized networks on ImageNet dataset. MP-BOPs represents the mixed-quantization with bit-operations (BOPs) constraint while MP-N that with N-bit average bit-width. '*' denotes the first and last layers remaining 8-bit, and KD denotes the knowledge distillation [42].

| Method    | Bit-width | BOPs(G) | Top-1   | Method    | Bit-width | BOPs(G) | Top-1   |
|-----------|-----------|---------|---------|-----------|-----------|---------|---------|
| FP        | 32        | 1857.6  | 70.54   | FP        | 32        | 306.8   | 72.6    |
| FP+KD     | 32        | 1857.6  | 72.17   | FP+KD     | 32        | 306.8   | 73.41   |
| PACT*     | 4         | 34.7    | 69.2    | DSQ*      | 4         | 15.8    | 64.8    |
| LSQ       | 4         | 34.7    | 69.3    | LSQ*      | 4         | 15.8    | 70.46   |
| DJPQ      | MP-BOPs   | 35.0    | 69.3    | DJPQ      | MP-BOPs   | 7.9     | 69.3    |
| HAQ       | MP-BOPs   | 34.4    | 69.2    | HAQ       | MP-BOPs   | 8.3     | 69.5    |
| HAWQ      | MP-BOPs   | 34.0    | 68.5    | DuQ+KD    | 4         | 5.3     | 69.86   |
| HAWQ-V3   | MP-BOPs   | 34.0    | 68.5    | NIPQ      | MP-4      | 12.9    | 71.63   |
| HAWQ-v3   | MP-BOPs   | 72.0    | 70.2    | NIPQ+KD   | MP-4      | 13.0    | 72.46   |
| NIPQ      | MP-BOPs   | 35.7    | 69.76   | NIPQ      | MP-BOPs   | 5.3     | 70.48   |
| NIPQ+KD   | MP-BOPs   | 35.7    | 70.71   | NIPQ      | MP-BOPs   | 8.3     | 72.01   |

| Method    | Bit-width | BOPs(G) | Top-1   | Method    | Bit-width | BOPs(G) | Top-1   |
|-----------|-----------|---------|---------|-----------|-----------|---------|---------|
| FP        | 32        | 218.7   | 74.52   | FP+KD     | 32        | 218.7   | 74.52   |
| PACT*     | 4         | 3.46    | 67.98   | NIPQ      | MP-4      | 9.18    | 70.27   |
| PACT+KD   | 4         | 3.46    | 70.16   | NIPQ+KD   | MP-4      | 9.70    | 72.24   |
| DuQ*      | 4         | 3.46    | 69.50   | NIPQ      | MP-BOPs   | 3.57    | 70.91   |
| DuQ+KD    | 4         | 3.46    | 71.01   | NIPQ+KD   | MP-BOPs   | 3.31    | 72.19   |

quantization when the weight is quantized. As shown in the figure, NIPQ with truncation shows much higher accuracy in the same bit-width. NIPQ has a mechanism of tuning the truncation boundary in a direction to minimize errors, so it is very effective in maintaining the quality of output. In particular, the learning mechanism of truncation enables activation quantization based on PQN. NIPQ is designed as a more versatile quantization algorithm, successfully quantizing both activation and weight on a single framework.

7 Quantization Results of Large-scale Vision Applications

To demonstrate the outstanding performance of NIPQ, we apply NIPQ for large-scale vision applications, including ImageNet [44] classification, multi-scale super-resolution, and VOC [45] object detection. The details of the experimental setups will be provided in the supplementary.

First, we apply quantization to diverse networks on the ImageNet classification task, a well-known large-scale dataset, and performs comparisons with various existing studies. Existing studies report mixed results with/without using well-known teacher-student-based knowledge distillation, so we report the accuracy for both cases, with EfficientNet-B0 as a teacher when necessary. Only the input image has 8-bit precision, and every layer in the network, including the first and last layers, is quantized via NIPQ.

Table 1 shows top-1 accuracy of the quantized networks. NIPQ shows outstanding results for the optimized but hard to quantize networks such as MobileNet-v2/v3, as well as redundant networks such as ResNet-18. As shown in the table, existing methods are inferior to the proposed method with high accuracy in the same bit-width or bit-operations. These outstanding results come from two facts: first, PQN-based QAT allowed us to converge to a more robust space without STE-oriented instability, and second, within the resource budget, the quantization hyper-parameters could be automatically tuned without the intervention of any hand-crafted manipulations. The benefit of these properties is maximized in optimized networks such as MobileNet-v2/v3. Note that when the average precision is constrained, NIPQ tries to increase accuracy with additional operations and vice versa. The automated tuning allows us to quantize the network considering our target goal. Besides, while NIPQ shows promising results, quantized MobileNet-v3 suffers from high accuracy degradation in

---

2Due to the lack of resources, the results are achieved with insufficient fine-tuning. Please note that there is room for improvement in accuracy with longer fine-tuning, and we will revisit the results.
Table 2: PSNR comparison of quantized EDSR [46] of scale 4 and scale 2.

| Network | Dataset | Bit-width | DoReFa [13] | TFLite [6] | PACT [14] | PAMS [47] | DDTB [48] | NIPQ |
|---------|---------|-----------|-------------|------------|-----------|-----------|-----------|------|
| EDSRx2  | Set5 [49] | 3         | 37.13       | 37.33      | 37.36     | 36.76     | 37.51     | 37.69 |
|         |         | 4         | 37.22       | 37.64      | 37.57     | 37.67     | 37.72     | 37.76 |
|         | Set14 [50] | 3         | 32.73       | 32.98      | 32.99     | 32.5      | 33.17     | 35.25 |
|         |         | 4         | 32.82       | 33.24      | 33.2      | 33.3      | 33.35     | 33.31 |
|         | BSD100 [51] | 3         | 31.57       | 31.76      | 31.7      | 31.8      | 31.89     | 31.97 |
|         |         | 4         | 31.63       | 31.94      | 31.93     | 31.94     | 32.01     | 32.03 |
|         | Urban100 [52] | 3         | 30.48       | 30.57      | 29.5      | 31.01     | 31.22     |       |
|         |         | 4         | 30.17       | 31.11      | 31.09     | 31.1      | 31.39     | 31.41 |
| EDSRx4  | Set5 [49] | 3         | 30.76       | 31.05      | 30.98     | 27.25     | 31.52     | 31.65 |
|         |         | 4         | 30.91       | 31.54      | 31.32     | 31.59     | 31.85     | 31.8  |
|         | Set14 [50] | 3         | 26.66       | 27.92      | 27.8      | 25.24     | 28.18     | 28.27 |
|         |         | 4         | 27.78       | 28.2       | 28.07     | 28.2      | 28.39     | 28.36 |
|         | BSD100 [51] | 3         | 26.97       | 27.12      | 27.09     | 25.38     | 27.3      | 27.37 |
|         |         | 4         | 27.04       | 27.31      | 27.21     | 27.32     | 27.44     | 27.43 |
|         | Urban100 [52] | 3         | 24.85       | 24.85      | 24.82     | 22.76     | 25.33     | 25.43 |
|         |         | 4         | 24.93       | 25.28      | 25.05     | 25.32     | 25.69     | 25.58 |

Table 3: mAP comparison of Yolov5-S [53] on PASCAL VOC dataset [45].

| Bit-width (Weight / Activation) | FP/FP  | 8/5   | 8/4   | 8/3   | 5/8   | 4/8   | 3/8   | 5/7   | 4/4   | 3/3   |
|-------------------------------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| DoReFa [13]                   | 0.857  | 0.628 | 0.62  | 0.565 | 0.593 | 0.541 | 0.359 | 0.588 | 0.498 | 0.288 |
| PACT [14]                     | 0.857  | 0.846 | 0.835 | 0.799 | 0.845 | 0.835 | 0.806 | 0.838 | 0.811 | 0.708 |
| LSQ [15]                      | 0.857  | 0.851 | 0.838 | 0.8   | 0.85  | 0.843 | 0.815 | 0.843 | 0.823 | 0.761 |
| NIPQ                          | 0.857  | 0.846 | 0.838 | 0.819 | 0.851 | 0.850 | 0.836 | 0.845 | 0.832 | 0.799 |

low precision. Advanced training pipeline, e.g., PROFIT [21] for STE-based algorithm, is required for better accuracy.

To validate the superiority of NIPQ on the regression application, we apply quantization to the super-resolution task. Table 2 shows the output quality of diverse quantization algorithms on EDSR [46], a representative network for super-resolution. In this experiment, we limit average bit-width and follow the convention of leaving the first and last layers in full precision. NIPQ surpasses PAMS, the best static quantization technique, by a substantial margin and even shows slightly better results than the best dynamic quantization scheme, DDTB. In the case of super-resolution that conducts image restoration, the advantage of dynamic quantization, whose quantization parameters are updated regarding input data, is well demonstrated. Even in this case, NIPQ outmatches DDTB. We think the benefit of NIPQ could be maximized with dynamic quantization; however, we have left the detailed implementation as future work.

Finally, we conduct an experiment to quantize the object detection task, which is known to be difficult to quantize. The difficulty is rapidly increased because we apply quantization to the advanced optimized network, Yolov5-S [53]. Table 3 shows the comparison of existing quantization studies in the same average bit-width. NIPQ exhibits excellent quantization performance in both activation and weight. Yolov5 has a very complex structure based on a number of concatenations for efficient computation. Existing 4-bit solutions are difficult to use in reality due to large accuracy loss, but NIPQ shows practical, reliable quality in 4-bit precision based on bit-width allocation considering layer-wise sensitivity and avoiding the instability of STE-based training. The results of this experiment validate the stability of the NIPQ algorithm regardless of the difficulty of the target task.

8 Conclusion

The accurate training of a quantized neural network maximizes the benefit of low-precision acceleration. In this study, we proposed a novel QAT pipeline based on PQN, called noise injection pseudo quantization (NIPQ). NIPQ improves the accuracy of the quantized network by removing unstable STE approximation and provides fully automated quantization, which enables us to train a low-precision network having the highest accuracy within the given resource constraints. Our extensive experiments verified the excellence of the proposed method in diverse applications, outperforming the existing STE-based or mixed-precision quantization methods. Energy-efficient inference is a crucial topic related to environment-aware computation; we expect that our optimization scheme will enable us to exploit the outstanding deep learning algorithms with minimal overhead in reality.
References

[1] Jongsoo Park, Maxim Naumov, Protonu Basu, Summer Deng, Aravind Kalaiah, Daya Shanker Khudia, James Law, Parth Malani, Andrey Malevich, Nadathur Satish, Juan Miguel Pino, Martin Schatz, Alexander Sidorov, Viswanath Sivakumar, Andrew Tulloch, Xiaodong Wang, Yiming Wu, Hector Yuen, Utku Diril, Dmytro Dzhulgakov, Kim M. Hazelwood, Bill Jia, Yangqing Jia, Lin Qiao, Vijay Rao, Nadav Rotem, Sungjoo Yoo, and Mikhail Smelyanskiy. Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications. CoRR, abs/1811.09886, 2018. URL http://arxiv.org/abs/1811.09886.

[2] Mário Almeida, Stefanos Laskaridis, Abhinav Mehrotra, Lukasz Dudziak, Ilias Leontiadis, and Nicholas D. Lane. Smart at what cost?: characterising mobile deep neural networks in the wild. In Dave Levin, Alan Mislove, Johanna Amann, and Matthew Luckie, editors, IMC '21: ACM Internet Measurement Conference, Virtual Event, USA, November 2-4, 2021, pages 658–672. ACM, 2021. doi: 10.1145/3487552.3487863. URL https://doi.org/10.1145/3487552.3487863.

[3] Hao Wu. NVIDIA Low Precision Inference on GPU. GPU Technology Conference, 2019.

[4] Jinook Song, Yunkyo Cho, Jun-Seok Park, Jun-Woo Jang, Sehwan Lee, Joon-Ho Song, Jae-Gon Lee, and Inyup Kang. 7.1 an 11.5 tops/w 1024-mac butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile soc. International Solid-State Circuits Conference (ISSCC), 2019.

[5] Nvidia. Int4 precision for ai inference. https://devblogs.nvidia.com/int4-for-ai-inference/, 2019. Accessed: 2021-11-16.

[6] Andrew Tulloch and Yangqing Jia. Quantization and training of neural networks for efficient integer-arithmetic-only inference. Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[7] Andrew Tulloch and Yangqing Jia. High performance ultra-low-precision convolutions on mobile devices. arXiv:1712.02427, 2017.

[8] Charles Eckert, Xiaowei Wang, Jingcheng Wang, Arun Subramaniyan, Ravi R. Iyer, Dennis Sylvester, David T. Blaauw, and Reetuparna Das. Neural cache: Bit-serial in-cache acceleration of deep neural networks. In Murali Annavaram, Timothy Mark Pinkston, and Babak Falsafi, editors, 45th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2018, Los Angeles, CA, USA, June 1-6, 2018, pages 383–396. IEEE Computer Society, 2018. doi: 10.1109/ISCA.2018.00040. URL https://doi.org/10.1109/ISCA.2018.00040.

[9] Boyuan Feng, Yuke Wang, Tong Geng, Ang Li, and Yufei Ding. APNN-TC: accelerating arbitrary precision neural networks on ampere GPU tensor cores. In Bronis R. de Supinski, Mary W. Hall, and Todd Gamblin, editors, SC ’21: The International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, Missouri, USA, November 14 - 19, 2021, pages 37:1–37:13. ACM, 2021. doi: 10.1145/3458817.3476157. URL https://doi.org/10.1145/3458817.3476157.

[10] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. CoRR, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

[11] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. CoRR, abs/1602.02830, 2016. URL http://arxiv.org/abs/1602.02830.

[12] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: ImageNet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV, volume 9908 of Lecture Notes in Computer Science, pages 525–542. Springer, 2016. doi: 10.1007/978-3-319-46493-0_32. URL https://doi.org/10.1007/978-3-319-46493-0_32.

[13] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. CoRR, abs/1606.06160, 2016. URL http://arxiv.org/abs/1606.06160.
[14] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: parameterized clipping activation for quantized neural networks. CoRR, abs/1805.06085, 2018. URL http://arxiv.org/abs/1805.06085

[15] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgO66VKDS

[16] Ying Wang, Yadong Lu, and Tijmen Blankevoort. Differentiable joint pruning and quantization for hardware efficiency. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX, volume 12374 of Lecture Notes in Computer Science, pages 259–277. Springer, 2020. doi: 10.1007/978-3-030-58526-6_16. URL https://doi.org/10.1007/978-3-030-58526-6_16

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90

[18] Zechun Liu, Baoyuan Wu, Wenhao Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV, volume 11219 of Lecture Notes in Computer Science, pages 747–763. Springer, 2018. doi: 10.1007/978-3-030-01267-0_44. URL https://doi.org/10.1007/978-3-030-01267-0_44

[19] Ruihao Gong, Xionglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 4851–4860. IEEE, 2019. doi: 10.1109/ICCV.2019.00495. URL https://doi.org/10.1109/ICCV.2019.00495

[20] Sungho Shin, Jinhwan Park, Yoonho Boo, and Wonyong Sung. Hlhlp: Quantized neural networks training for reaching flat minima in loss surface. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 5784–5791. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6035

[21] Eunhyeok Park and Sungjoo Yoo. PROFIT: A novel training method for sub-4-bit mobilenet models. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI, volume 12351 of Lecture Notes in Computer Science, pages 430–446. Springer, 2020. doi: 10.1007/978-3-030-58539-6_26. URL https://doi.org/10.1007/978-3-030-58539-6_26

[22] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00474. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[23] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 1314–1324. IEEE, 2019. doi: 10.1109/ICCV.2019.00140. URL https://doi.org/10.1109/ICCV.2019.00140

[24] Chaim Baskin, Natan Liss, Yoav Chai, Evgenii Zheltonozhskii, Eli Schwartz, Raja Giryes, Avi Mendelson, and Alexander M. Bronstein. NICE: noise injection and clamping estimation for neural network quantization. CoRR, abs/1810.00162, 2018. URL http://arxiv.org/abs/1810.00162
[25] Pierre Stock, Angela Fan, Benjamin Graham, Edouard Grave, Rémi Gréboul, Hervé Jégou, and Armand Joulin. Training with quantization noise for extreme model compression. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021. URL: https://openreview.net/forum?id=dV19Yyi1fS3

[26] Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. Differentiable model compression via pseudo quantization noise. CoRR, abs/2104.09987, 2021. URL: https://arxiv.org/abs/2104.09987

[27] Bernard Widrow, Istvan Kollar, and Ming-Chang Liu. Statistical theory of quantization. IEEE Transactions on instrumentation and measurement, 45(2):353–361, 1996.

[28] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: hardware-aware automated quantization with mixed precision. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 8612–8620. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00881. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Wang_HAQ_Hardware-Aware_Automatic_Quantization_With_Mixed_Precision_CVPR_2019_paper.html

[29] Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemehsadat Miresghallah, Amir Yazdanbakhsh, and Hadi Esmaeilzadeh. Relqe: A reinforcement learning approach for automatic deep quantization of neural networks. IEEE Micro, 40(5):37–45, 2020. doi: 10.1109/MM.2020.3009475. URL: https://doi.org/10.1109/MM.2020.3009475

[30] Jing Liu, Jianfei Cai, and Bohan Zhuang. Sharpness-aware quantization for deep neural networks. CoRR, abs/2111.12273, 2021. URL: https://arxiv.org/abs/2111.12273

[31] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V3: dyadic neural network quantization. In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, volume 139 of Proceedings of Machine Learning Research, pages 11875–11886. PMLR, 2021. URL: http://proceedings.mlr.press/v139/yao21a.html

[32] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso García, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision dnns: All you need is a good parametrization. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. URL: https://openreview.net/forum?id=Hyx0SlrFvH
[37] Moran Shkolnik, Brian Chmiel, Ron Banner, Gil Shomron, Yury Nahshan, Alex M. Bronstein, and Uri C. Weiser. Robust quantization: One model to rule them all. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual. 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3948ead63a9f2944218de038d8934305-Abstract.html.

[38] Milad Alizadeh, Arash Behboodi, Mart van Baalen, Christos Louizos, Tijmen Blankevoort, and Max Welling. Gradient $\ell_1$ regularization for quantization robustness. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. URL https://openreview.net/forum?id=ryxK0JBtPr.

[39] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020. Virtual Event, pages 119 of Proceedings of Machine Learning Research, pages 1554–1565. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20f.html.

[40] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[41] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Christopher J. C. Burges, et al., editors, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, pages 1581–1589. JMLR.org, 2015. URL http://jmlr.org/proceedings/papers/v37/li15.html.

[42] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Dark knowledge. Presented as the keynote in BayLearn, 2(2), 2014.

[43] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 6105–6114. PMLR, 2019. URL http://proceedings.mlr.press/v97/tan19a.html.

[44] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, pages 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

[45] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis., 88(2):303–338, 2010. doi: 10.1007/s11263-009-0275-4. URL https://doi.org/10.1007/s11263-009-0275-4.

[46] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017, pages 1132–1140. IEEE Computer Society, 2017. doi: 10.1109/CVPRW.2017.151. URL https://doi.org/10.1109/CVPRW.2017.151.

[47] Huixia Li, Chenqian Yan, Shaohui Lin, Xiawu Zheng, Baochang Zhang, Fan Yang, and Rongrong Ji. PAMS: quantized super-resolution via parameterized max scale. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXV, volume 12370 of Lecture Notes in Computer Science, pages 564–580. Springer, 2020. doi: 10.1007/978-3-030-58595-2_34. URL https://doi.org/10.1007/978-3-030-58595-2_34.
We provide additional details for better understanding in the supplementary material. Section A presents the ablation result (e.g., the visualization of bit-width allocation and the accuracy of object detection tasks) when training a network with BOPs constraint. Section B introduces the detailed configurations and training hyper parameters of the experiments in the main paper, explained in Sections 4 to 7.

A Ablation Studies

A.1 Sensitivity-aware Layer-wise Mixed-precision Quantization

Figure 6 shows the assigned bit-width of activation and weight when restricting the computation cost (bit-operations) as a 1.5G BOPs, which is equal to the computation cost of the quantized 4-bit model using PACT [14] or LSQ [15]. Unlike Figure 2 in the main paper, the bit-width of activation is slightly misaligned with the sum of the hessian trace. In Figure 2 in the main paper, we aim to optimize the average bit-width of activation and weight independently. Those two configurations are optimized via disjoint target losses, and thereby each precision is assigned proportionally to the
sensitivity of activation and weight separately. However, in the case of bit-operations (BOPs), the computation cost is proportional to the product of bit-width of activation and weight. Thereby, when we restrict the overall computation cost, the activation bit-width is allocated aware of the activation sensitivity as well as the corresponding weight sensitivity, and vice versa. This experimental result indicates that assigning the bit-width based on layer-wise sensitivity naively might not be an optimal policy for computation-aware quantization. Regardless of the optimization target, NIPQ automates the bit-width assignment based on the implicit sensitivity awareness based on noise robustness.

### A.2 Quantization Results with BOPs Constraints

Table 4 shows the quantization results of the YoloV5-S model with limited computation costs (bit-operations). In this experiment, we aim to match the computation cost for end-to-end 5/4/3-bit precision networks. As shown in the table, NIPQ successfully quantizes the network and preserves the quality of output in low precision representation by assigning the layer-wise bit-width automatically. When we restrict the average precision, the bit-width is allocated toward utilizing more computation or higher BOPs. Likewise, when we restrict the computation cost, the network tends to use more bit-width to store the data within the bounded BOPs. As well as the image classification, NIPQ can be applicable for diverse vision applications with different resource limitations, e.g., average precision for memory footprint or bit operations for computation cost. Easy-to-use implementation is one of the most important advantages of NIPQ.

In the case of super-resolution, the quantization results with BOPs constraint show identical results in Table 2 in the main paper. In both cases, all layers have the same bit-width, equal to the target precision. This is mainly because EDSR [46] has identical and repeated structures, and the layer-wise sensitivity is almost indistinguishable across the stacked layer.

### B Experiment Configurations

In this paper, all experiments are performed using GPU servers having 8 x NVIDIA GTX3090 with 24 GB VRAM with 2 x AMD 7313 (16 Core 32 T). The number of GPUs is selected satisfying the validation error to be less than 10% for all models.
Table 5: Fine-tuning configurations of ImageNet classification task.

| Configuration   | Epoch | SGD | Cosine annealing with warmup | λ | λ_w | λ_a | λ_b |
|-----------------|-------|-----|-----------------------------|---|-----|-----|-----|
| ResNet-18       | 25 3  | 0.04 | 1 × 10^{-7} | 3 | 1 × 10^{-3} | 1 | 1 | 1 |
| MobileNet-v2     | 30 5  | 0.04 | 5 × 10^{-5} | 5 | 1 × 10^{-3} | 1 | 1 | 1 |
| MobileNet-v3     | 25 3  | 0.04 | 1 × 10^{-5} | 3 | 1 × 10^{-3} | 1 | 1 | 3 |

Table 6: Fine-tuning configurations of super-resolution task with EDSR.

| Configuration | Epoch | Adam | Cosine annealing | λ | λ_w | λ_a | λ_b |
|--------------|-------|------|------------------|---|-----|-----|-----|
| EDSR 4bit    | 30 10 | 1 × 10^{-4} | 0 | 1 × 10^{-4} | 15 | 15 | 15 |
| EDSR 3bit    | 40 10 | 1 × 10^{-4} | 0 | 1 × 10^{-4} | 15 | 15 | 15 |

Table 7: Fine-tuning configurations of object detection task with YoloV5-S.

| Configuration | Epoch | SGD | Cosine annealing with warmup | λ | λ_w | λ_a | λ_b |
|---------------|-------|-----|-----------------------------|---|-----|-----|-----|
| YoloV5-S      | 30 5  | 0.0032 | 5.6 × 10^{-3} | 5 | 1 × 10^{-3} | 1 | 1 | 0.1 |

minimum requirement of GPU memory for the target task. All of the experiments are implemented based on the PyTorch framework (v1.7.0) [54], and the experiment source code is also provided. The additional details of training configuration, e.g., optimizer type, initial learning rate, decay policy, etc., depend on the characteristics of applications. The details are provided in the following paragraphs.

Table 5 shows the detailed configurations of ImageNet training for NIPQ results. In this experiment, we apply quantization for every convolution and linear layer, including the first and last layers. One exception is that the input of the first convolution layer is fixed as 8-bit. We use SGD with momentum optimizer and cosine annealing with warmup scheduling for learning rate adjustment [55]. η_{min} is the final LR multiplier of cosine annealing, and λ_w, λ_a, and λ_b are the hyper-parameter of resource constraints for the bit-width of weight, bit-width of activation, and BOPs, respectively.

When knowledge distillation is triggered, we use EfficientNet-B0 [43] as a teacher network. We use the conventional dark-knowledge-based distillation [42].

Tables 6 and 7 show the detailed configurations of super-resolution task and object detection task, respectively. In both experiments, we keep the precision of the first and last layers as full-precision and apply low-precision quantization for the rest of the layers. In the super-resolution task, we use ADAM optimizer [56] and cosine annealing scheduling for learning rate adjustment. In the object detection task, we use SGD with momentum optimizer and cosine annealing with warmup scheduling for learning rate adjustment. Like the image classification task, η_{min} is the final LR multiplier of cosine annealing, and λ_w, λ_a, and λ_b represent the hyper-parameter of resource constraints for the bit-width of weight, bit-width of activation, and BOPs, respectively.