Weighted-Entropy-based Quantization for Deep Neural Networks

Eunhyeok Park, Junwhan Ahn[§], and Sungjoo Yoo
canusglow@gmail.com, junwhan@snu.ac.kr, sungjoo.yoo@gmail.com
Seoul National University
Computing and Memory Architecture Laboratory [§]Design Automation Laboratory

Abstract

Quantization is considered as one of the most effective methods to optimize the inference cost of neural network models for their deployment to mobile and embedded systems, which have tight resource constraints. In such approaches, it is critical to provide low-cost quantization under a tight accuracy loss constraint (e.g., 1%). In this paper, we propose a novel method for quantizing weights and activations based on the concept of weighted entropy. Unlike recent work on binary-weight neural networks, our approach is multi-bit quantization, in which weights and activations can be quantized by any number of bits depending on the target accuracy. This facilitates much more flexible exploitation of accuracy-performance trade-off provided by different levels of quantization. Moreover, our scheme provides an automated quantization flow based on conventional training algorithms, which greatly reduces the design-time effort to quantize the network. According to our extensive evaluations based on practical neural network models for image classification (AlexNet, GoogLeNet and ResNet-50/101), object detection (R-FCN with ResNet-50), and language modeling (an LSTM network), our method achieves significant reductions in both the model size and the amount of computation with minimal accuracy loss. Also, compared to existing quantization schemes, ours provides higher accuracy with a similar resource constraint and requires much lower design effort.

1. Introduction

Deep neural networks (DNNs) are becoming more and more popular in mobile and embedded systems [1, 7, 23]. Those systems are characterized by tight resource constraints in terms of performance, energy consumption, and memory capacity. Due to this, today's typical scenario for deploying DNNs to mobile systems is to train such DNNs in servers and perform only the inference in such systems. Therefore, it is imperative to reduce the inference cost of neural networks for widespread application of DNNs to mobile and embedded systems.

One of the most effective methods for reducing the inference cost of neural networks is to reduce the precision of computation. Recent research has demonstrated that inference of DNNs can be accurately done by using 8-bit or even narrower bitwidth representations for weights and activations, rather than conventional 32-/64-bit floating-point numbers [22]. In addition, there have been active studies that aim at further reducing the precision of both computation and values by aggressively quantizing the weights and/or activations for inference [4, 12, 17, 19, 25]. Such aggressive quantization methods are promising in that they can achieve significant reductions in the execution time, energy consumption, and memory capacity requirements of neural networks during the inference by exploiting the benefits of dedicated hardware accelerators, e.g. NVIDIA P40 and P4 [2] which support 8-bit integer arithmetic or *Stripes* [14] which provides execution time and energy consumption proportional to the bitwidth.

However, existing quantization techniques have two limitations that can hinder practical application of such techniques into mobile and embedded systems. First, existing methods lack in supporting flexible trade-off between output quality and inference performance. Mobile and embedded systems often have stringent constraints in both resource and inference accuracy, which requires design space exploration for trade-off between output quality and inference performance. However, some of the existing approaches are not flexible enough to exploit such trade-off relationship. For example, techniques that binarize weights [4, 19, 25] suffer from a significant loss of output quality for deep networks, which cannot be applied if the target system allows a very small accuracy loss, e.g. 1%.

Second, even if existing quantization techniques support such trade-off, they require modifications to the target network to achieve good quantization quality and/or apply quantization to only part of the network. Due to this, such techniques may require significant effort at design time, which may eventually prevent widespread adoption of them. In addition, existing methods such as XNOR-

Net [19] and DoReFa-Net [25] do not apply quantization to the first and the last layer to avoid excessive accuracy loss, which may limit the benefits of reduced precision.

In order to address these two limitations, we propose a new quantization scheme based on the concept of weighted entropy. Our approach addresses both of the aforementioned limitations while quantizing weights and activiations. Our contributions can be summarized as follows:

1. We propose a new multi-bit quantization method for both weights and activations. Unlike binary quantization approaches, our scheme is able to produce quantization results for any number of bits per weight/ activation, thereby realizing much more flexibility for exploiting accuracy-performance trade-off.

2. Our scheme facilitates automated quantization of the entire neural network. It does not require any modifications to the network except for activation quantization, and thus, it can be easily integrated into conventional training algorithms for neural networks.

3. We demonstrate the effectiveness of our method based on various practical neural network designs, including AlexNet [15], GoogLeNet [21], ResNet50/101 [11], R-FCN [5], and an LSTM for language modeling [24].

2. Related Work

In this section, we briefly review previous work on quantization methods for DNN inference. Vanhoucke et al. [22] presented a comparison between 8-bit and 32-bit implementations of neural networks. Miyashita et al. [17] proposed to quantize weights and activations in base-2 logarithm representation (called *LogQuant*) and showed that 4-bit weights and 5-bit activations achieve around 1.7% accuracy loss for AlexNet. LogQuant shows its potential in bit widths narrower than 8 bits, but it significantly degrades the accuracy under 4 bits for AlexNet.

There have been several approaches that try to quantize weights and/or activations into only two (i.e., binary) or three (i.e., ternary) levels. Hwang and Sung [12] showed that ternary weights (i.e., -1, 0, and $+1$) and 3-bit activations can preserve accuracy in character and phoneme recognition tasks. Courbariaux et al. [4] presented a binary-weight network called *BinaryConnect* and demonstrated its good accuracy on small-scale models such as CIFAR-10 and SVHN. Rastegari et al. [19] proposed a binary network (a binary-weight version of XNOR-Net), which does not experience accuracy loss on AlexNet.

Zhou et al. [25] proposed *DoReFa-Net*, which applies linear quantization to normalized weights and bounded activations, and showed 6% top-1 accuracy loss in AlexNet with 1-bit weights and 2-bit activations. They also reported results with 1-bit weights and k-bit activations, which allows us to exploit the trade-off relationship between accuracy loss and performance/energy/model size.

As explained in the previous section, the key benefits of our approach over previously proposed quantization methods are (1) flexibility of exploiting accuracy-performance trade-off via multi-bit quantization and (2) quantization of the full network without modifications to the existing networks. In this regard, most of the binary-/ternary-weight approaches fail to provide even the former. While XNOR-Net and DoReFa-Net provide the former, they still fail to achieve the latter. XNOR-Net requires channel-wise scaling and layer reordering by placing normalization and activation layers in front of a convolution layer. DoReFa-Net adds bounded activation functions to existing networks. Both XNOR-Net and DoReFa-Net do not apply their quantization scheme to the first and the last layers of the network to avoid noticeable accuracy loss. Such limitations induce additional effort to modify the network at design time, high performance/energy overhead of those full-precision layers, and extra hardware cost to support large-scale full-precision hardware units for fast execution of those layers.

3. Motivation

Recent studies have shown that most of the weights in convolutional or fully-connected layers are concentrated near zero, resulting in a bell-shaped distribution [10]. The distribution of activation values are similar, except that activation values are always non-negative due to a ReLU layer. Existing quantization schemes are based on such characteristics to judiciously assign quantization levels. For example, logarithm-based quantization (or LogQuant) exploits denser distribution of weights near zero by assigning more quantization levels to near-zero values.

In addition to the distribution of weight/activation values, we make a key observation that *the impact of each weight/activation value on the final result should also be considered during the quantization*. Since the objective of a quantization method is to minimize the accuracy degradation with the fewest quantization levels, taking the actual impact of quantizing each value into account allows us to develop a new scheme that uses each quantization level more effectively. More specifically, our insight can be summarized as follows:

1. **Near-zero values** dominate the total frequency of values in both weight and activation distribution; however, their impact on the output is small (e.g., errors in a very small weight may not affect much to the result of convolution). Thus, it is desirable to assign fewer quantization levels (in short, *levels* throughout this paper) to near-zero values than in a typical linear or logarithm-based quantization.

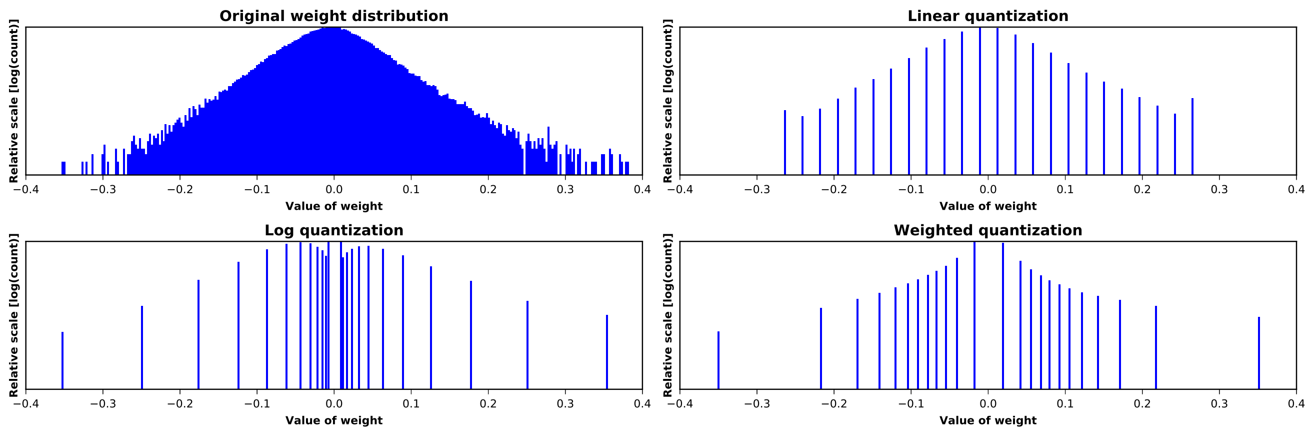

Figure 1. Comparison of various quantization schemes. The weights are extracted from the second 3×3 convolution layer of GoogLeNet [21]. Each quantization scheme is given to assign 24 levels. We use $2^{0.5}$ as the base of LogQuant and optimize both linear quantization and LogQuant towards minimizing the L2 norm of overall activations.

2. **Large weights and activations** have significant impact on the quality of output, but they are infrequent. Thus, it is also desirable to assign a small number of levels to those values in order to maximize the utility of each quantization level.
3. **Values that do not belong to neither of the two aforementioned categories** have a relatively large number of population with noticeable impacts on the output quality. Thus, it makes sense to assign more levels to those values than in conventional quantization methods.

Figure 1 illustrates how existing and proposed methods assign levels to the given weight distribution. While the linear quantization does not consider the weight distribution at all and LogQuant assigns too many levels to near-zero values, our approach shows distribution that is more concentrated on the values that are neither too small nor too large. Through quantitative evaluations, we will show later that this style of quantization achieves higher efficiency than conventional schemes.

4. Quantization based on Weighted Entropy

4.1. Weight Quantization

The high-level idea of our weight quantization approach is to group weights into N clusters in a way to have more clusters for important ranges of weights, assign a representative value to each cluster, and quantize all weights in each cluster into the representative value of the cluster. For this purpose, we have to be able to evaluate the clustering quality and find a set of clusters optimizing such quality metric.

As the first step, we define a quantitative metric for evaluating the importance of a single weight (or the impact of a weight on output quality). Since larger weights have a higher impact on the output quality, we empirically define the importance $i_{(n,m)}$ of m-th weight in n-th cluster, i.e., $w_{(n,m)}$ to be quadratically proportional to the magnitude of the weight, i.e., $i_{(n,m)} = {w_{(n,m)}}^2$.

Based on this importance value of each weight, we derive a metric for evaluating the quality of a clustering result (i.e., quantization result) based on *weighted entropy* [8, 9]. Weighted entropy is originated from the concept of entropy in physics and is designed to take the importance of data into account. For a set of clusters C_0, ..., C_{N-1}, weighted entropy S is defined as

$$S = -\sum_n I_n P_n \log P_n \tag{1}$$

where

$$P_n = \frac{|C_n|}{\sum_k |C_k|} \qquad \text{(relative frequency)} \tag{2}$$

$$I_n = \frac{\sum_m i_{(n,m)}}{|C_n|} \qquad \text{(representative importance)} \tag{3}$$

In this equation, P_n represents how many weights are in the range of values for cluster C_n, while I_n is the average importance of all weights in cluster C_n. Roughly speaking, clusters for large weights will generally have high I_n but low P_n (i.e., high importance but low frequency), while clusters for small weights will have high P_n but low I_n (i.e., high frequency but low importance). According to our experiments, finding a clustering result that maximizes S yields quantization whose levels are assigned sparsely for too small or too large values, just as we showed in Figure 1. Therefore, we define our weight quantization problem as follows:

Problem 1 (Weight Quantization)**.** *Given the training data (i.e., mini-batch input) and the desired* $\log N$*-bit precision (i.e., the number of clusters* N*), our method aims at finding*

Algorithm 1 Weight Quantization

```
1: function OPTSEARCH($N, w$)
2:     for $k$ = 0 to $N_w - 1$ do
3:         $i_k \leftarrow f_i(w_k)$
4:     $s \leftarrow sort([i_0, \cdots, i_{N_w - 1}])$
5:     $c_0, \cdots, c_N \leftarrow$ initial cluster boundary
6:     while $S$ is increased do
7:         for $k$ = 1 to $N - 1$ do
8:             for $c'_k \in [c_{k-1}, c_{k+1}]$ do
9:                 $S' \leftarrow S$ with $c_0, \cdots, c'_k, \cdots, c_N$
10:                if $S' > S$ then
11:                    $c_k \leftarrow c'_k$
12:    for $k$ = 0 to $N - 1$ do
13:        $I_k \leftarrow \sum_{i=c_k}^{c_{k+1}-1} s[i]/(c_{k+1} - c_k)$
14:        $r_k \leftarrow f_i^{-1}(I_k)$
15:        $b_k \leftarrow f_i^{-1}(s[c_k])$
16:    $b_N \leftarrow \infty$
17:    return $[r_0 : r_{N-1}], [b_0 : b_N]$
18: function QUANTIZE($w_n, [r_0 : r_{N-1}], [b_0 : b_N]$)
19:    return $r_k$ for $k$ s.t. $b_k \leq w_n < b_{k+1}$
```

- N: The number of levels
- N_w: The number of weights
- w_n: Value of n-th weight
- i_n: Importance of n-th weight
- f_i: Importance mapping function
- c_i: Cluster boundary index
- S: Overall weighted entropy

N weight clusters that maximize the weighted entropy. The representative value of a cluster corresponds to a level in the weight quantization.

Our solution to this problem is shown in Algorithm 1. Note that the algorithm shows the weighted quantization for non-negative weights only. This is because, due to the limitation of the weighted entropy theory, we cannot obtain a clustering result that has both negative and non-negative representative values. Thus, we separate the weights into two negative and non-negative groups, and apply our algorithm to each group with $N/2$ levels each.

At the beginning of the algorithm, we calculate the importance of each weight (lines 2 and 3). This is done by an importance mapping function f_i, which calculates the importance i_k from weight w_k. In this work, we empirically choose a square function $f_i(w) = w^2$ to compute the importance of each weight. After obtaining the importance values of all weights, they are sorted in the increasing order of their magnitude (line 4).

Based on the sorted importance values, the algorithm initializes cluster boundary indexes c_0 to c_N[1] (line 5) such that (1) each cluster has the same number of weights and (2) weights in C_{i+1} have higher importance than weights in C_i. This is achieved simply by partitioning the sorted array s into N pieces and assign each piece to each cluster. For example, if $s = [1,2,3,4]$ and $N = 2$, we set $c_0 = 0$, $c_1 = 2$, and $c_2 = 4$ so that $C_0 = \{1,2\}$ and $C_1 = \{3,4\}$.

Starting from the initial cluster boundaries, we iteratively perform incremental search on the new cluster boundaries (lines 6–11). At each iteration, for each cluster C_i and its boundaries c_i and c_{i+1}, we sweep c_i from c_{i-1} to c_{i+1} by using bisection method. For each cluster boundary candidate c'_i, we recalculate the weighted entropy of cluster C_{i-1} and C_i, which are the only ones affected by the new boundary, and update the boundary to c'_i only if the new overall weighted entropy S' is higher than the current one.

After obtaining the new cluster boundaries, we calculate the representative importance I_k of each cluster C_k (line 13). We obtain the representative weight value r_k for cluster C_k (line 14). In order to identify which weights belong to which cluster, weight values at cluster boundaries, b_k, are identified as well for weight quantization (line 15), i.e., cluster C_i contains weights w that satisfy $b_k \leq w < b_{k+1}$. Function QUANTIZE implements this quantization method. That is, given a weight w_n, it produces the representative weight value r_k of the associated cluster c_k.

The weighted-entropy-based clustering can provide levels that satisfy our requirements on quantization in Section 3. Maximizing the weighted entropy optimizes the quantization result towards maximizing entropy while considering the importance of data. Thus, our method groups many near-zero values into a large cluster by considering their lower importance. Large, but infrequent values are also grouped into a cluster that covers a wide range of weight values.

4.2. Activation Quantization

Activation quantization needs a different approach from weight quantization. While weights are fixed after the training, activations change at inference time according to the input data. This makes activations less suitable to be quantized by clustering-based approaches, which require a stable distribution of values.

According to our investigation, logarithm-based quantization (LogQuant) can be effective for activation quantization. LogQuant is also beneficial to minimize the cost of implementation (e.g., dedicated hardware accelerators) as it can transform multiplications into inexpensive bitwise shift operations (i.e., $w \times 2^x = w \ll x$). However, the original LogQuant method does not provide an effective search strategy for exploring the best LogQuant parameters (i.e., base and offset) for each layer of the network.

[1]Cluster boundary indexes determine which weights belong to which clusters. Precisely, cluster C_i is defined as containing c_i-th weight to $(c_{i+1} - 1)$-th weight (zero-based indexing) in array s.

Algorithm 2 Activation Quantization

```
function BinaryToLogQuant($a_n$)
    return round$\left(\frac{16\times\log_2 a_n - \text{fsr}}{\text{step}}\right) + 1$

function LogQuantToBinary(index)
    if index $= 0$ then
        return 0
    else
        return $2^{\frac{1}{16}\times(\text{fsr}+\text{step}\cdot(\text{index}-1))}$

function WeightedLogQuantReLU($a_n$)
    if $a_n < 0$ then
        return 0
    level_idx $\leftarrow$ BinaryToLogQuant($a_n$)
    if level_idx $\leq 0$ then
        return 0
    else if level_idx $\geq N-1$ then
        return LogQuantToBinary($N-1$)
    else
        return LogQuantToBinary(level_idx)

function ReprImportance(index)
    return LogQuantToBinary(index)
function RelativeFrequency(index, $a$)
    for $k = 0$ to $N_a - 1$ do
        level_idx$_k$ $\leftarrow$ BinaryToLogQuant($a_n$)
    if index $= 0$ then
        return $|\{a_n \mid \text{level\_idx}_n \leq 0\}|$
    else if index $= N-1$ then
        return $|\{a_n \mid \text{level\_idx}_n \geq N-1\}|$
    else
        return $|\{a_n \mid \text{level\_idx}_n = \text{index}\}|$
```

- N: The number of levels
- N_a: Total number of activations
- a_n: Value of n-th activation
- fsr: Optimal fsr value (integer)
- step: Optimal step value (a multiple of 2)

Our approach to activation quantization consists of two parts: a modified version of LogQuant and a fast search strategy for LogQuant parameters. Algorithm 2 shows key functions used in our modified LogQuant method.

First, we modify the the original LogQuant method to improve overall accuracy and stability. Unlike the conventional LogQuant, we adopt smaller log bases (1/8 and its multiples) and offsets (1/16 and its multiples), which correspond to 'step' and 'fsr' in Algorithm 2, respectively. We assign the first quantization level to zero activation and the other levels to the corresponding log scale. For example, when we perform 3-bit quantization of activations, the first level is assigned to value 0, the second one to $2^{\frac{\text{fsr}}{16}}$, the third one to $2^{\frac{\text{fsr}+\text{step}}{16}}$, and so on. For simplicity, we integrate our activation quantization as part of the rectified linear unit (ReLU) activation function, which is described as Function WeightedLogQuantReLU in Algorithm 2.

Second, we propose a novel parameter search method for our LogQuant variant, which determines the base and the offset in a way to minimize the loss of output quality. Our idea is to take advantage of the concept of weighted entropy maximization in our weight quantization. Algorithm 2 shows functions that calculate the representative importance I (ReprImportance) and the relative frequency P (RelativeFrequency), which are the two ingredients for computing the weighted entropy. During training, in order to maximize the weighted entropy of the given per-layer activations under LogQuant, we apply an exhaustive search for 'fsr' and 'step' since the numbers of possible bases and offsets are usually small (e.g., 16 for bases and around 500 for offsets in our experiments).

4.3. Integrating Weight/Activation Quantization into the Training Algorithm

We integrate the proposed weight/activation quantization into the conventional training algorithm for neural networks. Since weights do not change during each mini-batch, weight quantization can be simply applied by quantizing the weights at the end of each mini-batch after the weight update. Note that we use full-precision weights during the weight update as in other previous work [19, 25].

On the other hand, activation quantization has to be applied to every forward/backward pass as each pass has its own set of activations. For each layer, we first perform the forward pass and apply the ordinary ReLU (without LogQuant). The resulting activations are fed into our algorithm for LogQuant parameter search. The best base/offset combination from the algorithm is then used to quantize the activations by using WeightedLogQuantReLU. The quantized activations are passed to the next layer to perform the same process for the rest of the layers in the network.

Under our training framework, any network can automatically benefit from our quantization schemes without modifications to the network. This makes it much easier to apply aggressive quantization to the entire neural network, which contributes to greatly reducing the inference cost of the network. Existing approaches are less practical in this regard, considering that they require network modification and/or significant manual effort at design time.

5. Experiment

We evaluate our approach in three representative domains of neural network applications: image classification, object detection, and language modeling. We modify Caffe [13] to implement our technique on top of all networks[2], except for language modeling, in which we use TensorFlow [3] to implement an LSTM. We constrain the accuracy loss to 1% and aim at finding the quantization configuration that gives the minimum bitwidth while satisfying the accuracy constraint. For brevity, we introduce a notation (x,y) to represent the bitwidth of weights x and that of activations y in a quantization configuration. In this notation, 'f' represents full precision. For example, (1,f) indicates 1-bit weights and full-precision activations.

5.1. Image Classification: AlexNet, GoogLeNet and ResNet-50/101

For image classification tasks, we evaluate the proposed method by quantizing two widely used CNNs for ImageNet tasks [6]: AlexNet [15] GoogLeNet [21] (both from Caffe framework [13]) and ResNet[3] [11]. In order to apply our quantization scheme into these networks, we perform fine-tuning combined with our weight/activation quantization schemes under the batch size of 256 (for AlexNet), 64 (for GoogLeNet), or 16 (for ResNet-50/101). In the cases of GoogLeNet and ResNet, the batch size is limited due to insufficient GPU memory capacity; this may increase overall accuracy loss. We use ILSVRC2012 data set, which contains 1.28M images for training and 50K images for testing. During the six epochs of fine-tuning, we first set the initial learning rate to 0.001 and decrease it by 10 times every two epochs.

In the following subsections, we present two styles of evaluation results. First, we demonstrate the effectiveness of our approach by quantizing the entire networks (whole network quantization), which was not possible in prior work. Second, we apply our scheme to all layers except the first and the last one (partial network comparison) and compare ours against previous quantization approaches that use the full precision at the first/last layer of a network.

5.1.1 Whole Network Quantization

Figure 2 compares the test accuracy of CNNs quantized by our techniques. As shown in the figure, the quantized CNNs achieve higher accuracy under less restrictive bitwidth constraint.

For AlexNet, the best quantization configurations that use the fewest bits while satisfying the 1% top-5 accuracy loss constraint are (3,6), (4,4), (4,5) and (4,6). For example, (4,4) reduces the bitwidths of both weights and activations by 87.5% ($= 1 - 4/32$) with less than 1% loss of top-5 accuracy. Moreover, our approach provides much lower iso-accuracy bitwidth compared to previous work. For example, LogQuant [17] achieves 75.1% top-5 accuracy with

[2]Modified caffe code is available at `https://github.com/EunhyeokPark/script_for_WQ`

[3]Base models are available at `https://github.com/KaimingHe/deep-residual-networks`

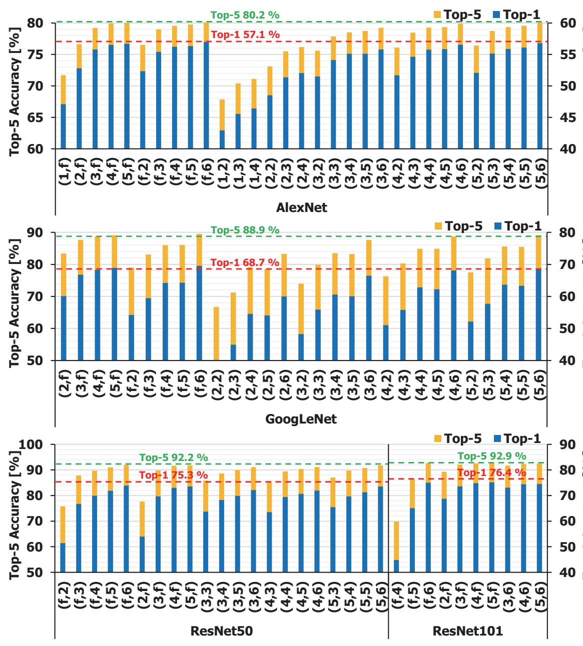

Figure 2. Top-1 and top-5 accuracy of quantized CNNs after fine-tuning. The dashed lines represent the accuracy of the baseline networks, which use full-precision arithmetic.

4-bit weights and 5-bit activations; Qiu et al. [18] used 8-bit weights and activations and showed 76.6% (53.0%) of top-5 (top-1) accuracy. Our approach achieves a similar level of top-5/top-1 accuracy (i.e., 75.49%/51.37%) with only 2-bit weights and 3-bit activations.

For GoogLeNet, under the 1% accuracy loss constraint, our approach can quantize weights and activations down to only 4–5 bits and 6 bits, respectively, as shown in Figure 2. We also observe that GoogLeNet suffers more from accuracy loss than AlexNet under the same level of bitwidth constraint. We believe that this is because the model size of GoogLeNet is more compact than AlexNet, yet the former performs more computation than the latter. In other words, GoogLeNet reuses each weight more frequently during the computation than AlexNet, which makes the impact of reduced weight precision more pronounced in GoogLeNet. Even so, our approach still achieves a significant (more than 5x) reduction in both the model size and the amount of computation (in bits) compared to the full-precision implementation.

For ResNet, to the best of our knowledge, this paper is the first to report the result of quantizing the entire networks whose depth is as much as 50 and 101 layers. Both networks maintain similar levels of accuracy even after aggressive quantization of weights, e.g., 3 bits. However, we observe that the deeper network demands more bits for activations, e.g., 6 bits, possibly because quantization errors of activations get accumulated over deeper layers.

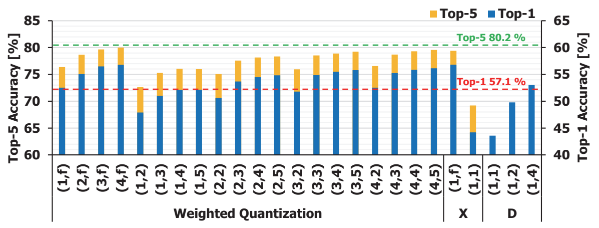

Figure 3. Accuracy comparison of quantization methods applied to AlexNet. 'Weighted Quantization' represents our approach, while 'X' and 'D' are for XNOR-Net and DoReFa-Net, respectively. The dashed lines indicate the accuracy of the baseline full-precision network.

5.1.2 Partial Network Quantization

In this subsection, we compare the performance of our quantization method against two state-of-the-art approaches: XNOR-Net [19] and DoReFa-Net [25]. For fair comparison, we apply our quantization scheme to all layers but the first and last ones, just as in our comparison targets. Note that the comparison is still not apple-to-apple in that (1) the (best available) results from previous work are limited to 1-bit weights and *k*-bit activations, while ours include *k*-bit weights and activations, and (2) both XNOR-Net and DoReFa-Net modify the network, whereas ours does not except ReLU layers (in activation quantization).

Figure 3 shows the comparison of our approach against XNOR-Net and DoReFa-Net. While XNOR-Net with binary weights, i.e., (1,f), shows very small accuracy drop with 1-bit weights, it is limited to binary quantization and full-precision activation, which is not flexible enough to exploit accuracy-performance trade-off under a stringent accuracy loss constraint. XNOR-Net with binary weight and activation quantization, i.e., (1,1), degrades the accuracy too much, whereas our method provides multi-bit quantization meeting the accuracy constraint. DoReFa-Net alleviates some of such limitations by allowing multi-bit quantization of activations. However, under the similar configurations, our scheme with 2-bit weights and 3-bit activations outperforms DoReFa-Net with 1-bit weights and 4-bit activations by 0.69% in terms of top-1 accuracy. In summary, our method facilitates more flexible choice of quantization configurations with smaller iso-accuracy bitwidth than previous work, which is extremely useful for systems that require efficient inference under a tight accuracy constraint.

5.1.3 Compression Analysis

Table 1 compares existing methods and ours in the context of compression. From this, we observe the followings.

First, both XNOR-Net [19] and DoReFa-Net [25] show larger weights than our method (WQ) since they do not quantize the first and the last layers. Moreover, Huffman encoding is not helpful since they use binary and full-precision weights, respectively.

	Weights			Activations	Top-1
	P [%]	**Q** [MB]	**+H**	**Q** [MB]	[%]
WQ(4,4)	-	30.5	18.1	0.47	55.8
WQ(2,3)	-	15.3	12.5	0.35	53.7
XNOR-Net [19]	-	23.7	-	0.72	44.2
DoReFa-Net [25]	-	23.6	-	0.47	53.0
Deep Compression [10]	11	8.9	6.9	3.75	57.2
[10] + WQ(4,6)	11	8.3	6.5	0.70	56.3

Table 1. Memory requirement comparison with AlexNet (P: Pruning ratio, Q: Quantization, H: Huffman encoding).

Second, when WQ is applied on top of pruning [10], it achieves 5.4x smaller activations and slightly smaller weights (8.9 MB vs. 8.3 MB) at an additional accuracy loss of 0.9%. Ours achieves larger bitwidth reductions in activations than in weights because [10] utilizes full-precision activations while ours uses 6-bit activations.

5.1.4 Layer-wise Quantization: A Feasibility Study

In the previous subsections, we use the same bitwidth constraint for all layers in the network. However, according to our observation, different layers have different levels of sensitivity to the quantization bitwidth. Thus, we perform a feasibility study of the potential of *layer-wise* quantization, where different layers may have different bitwidths. In this study, we evaluate the following four styles of per-layer bitwidth assignment based on AlexNet: monotonically decreasing (DEC), monotonically increasing (INC), concave (CONCAVE), and convex (CONVEX). All four schemes are designed to have the same number of bitwidth in total. For example, DEC assigns 6 bits to each weight/activation in the first convolution layer, while it uses only 2 bits for weights/activations in the last fully-connected layer.

	DEC	**INC**	**CONCAVE**	**CONVEX**
Top-1 [%]	53.79	50.35	54.45	54.33
Top-5 [%]	77.59	74.89	76.43	78.20

Table 2. Accuracy comparison of our approach under different styles of layer-wise quantization.

As shown in Table 2, we observe that using less bits in intermediate layers (i.e., CONVEX) achieves the highest accuracy, while assigning fewer bits to near-input layers (i.e., INC) shows the lowest. A similar phenomenon to this was observed by Zhou et al. [25]. We believe that even more aggressive bitwidth optimization could be possible by taking this layer-wise sensitivity to bitwidths into account during quantization. Exhaustive search of all possible combinations of bitwidths is impractical as there are too many of them even for small networks (e.g., AlexNet has

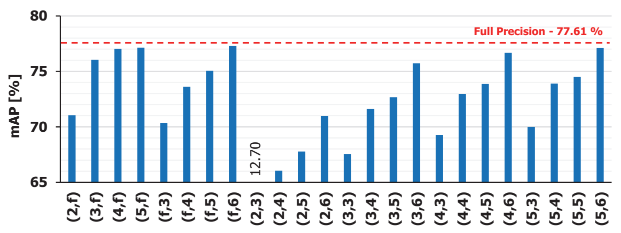

Figure 4. mAP results of R-FCN. The dashed line represents the accuracy of the baseline full-precision network.

$5^{15} \approx 3 \times 10^{10}$ possible configurations that use two to six bits for each layer). Algorithms for fast design space exploration of layer-wise quantization are left for future work.

5.2. Object Detection: R-FCN with ResNet-50

In order to evaluate the effectiveness of our quantization method on more complex vision tasks, we use a state-of-the-art 50-layer R-FCN model for object detection [5]. The R-FCN model combines a residual network (ResNet) [11] (for representation) and Faster R-CNN [20] (for region proposal, object classification, and box localization). On top of the existing full-precision model, we perform fine-tuning with our quantization method.

Even though deep models are known to be difficult to quantize since quantization errors are accumulated over deep layers, our method successfully quantizes the 50-layer model for object detection with very small accuracy loss. Figure 4 shows that the configuration of 5-bit weights and 6-bit activations loses only 0.51% of mAP, while reducing the model size and the amount of computation by more than 5x. We also observe that activations typically require more bits than weights in our quantization method (e.g., 6-bit activations or 4-/5-bit weights are needed for stable and satisfactory results, as shown in Figure 4). We believe that this is because the bounding box regression mechanism of R-FCN is simple (obtaining boxes directly from the region proposal network), and thus, is sensitive to activation accuracy. We will perform further investigation on this in our future work.

In our future work, we will also study the feasibility of our method with deeper models. According to our preliminary study with R-FCN based on ResNet-101, we failed to obtain quantization with reasonable accuracy when fine-tuning the model with the PASCAL VOC data set. We believe that this problem is due to the mixed effects of transfer learning and quantization on a very deep network, which requires further investigation into quantization on the very deep networks.

5.3. Language Modeling: An LSTM

In order to evaluate our scheme on recurrent neural networks, we perform a preliminary analysis by applying our method to an LSTM network for language modeling [24], provided along with the Tensorflow framework [3]. We evaluate three sizes of RNNs, small (200 hidden units and 20 time steps), medium (650 hidden unuts and 35 time steps), and large (1500 hidden units and 35 time steps), all of which have two layers each. We measure the word-level perplexity of these three RNNs before/after quantization with the Penn Tree Bank dataset [16]. We apply only the weight quantization to the LSTM network since our activation quantization is currently incompatible with bounded gates and linear outputs in RNNs.

	Large		**Medium**		**Small**	
	Valid	**Test**	**Valid**	**Test**	**Valid**	**Test**
float	82.77	78.63	87.69	83.54	119.19	114.46
1-bit	92.20	88.48	104.0	100.7	147.19	141.07
2-bit	86.73	82.90	92.49	89.24	137.34	131.15
3-bit	85.59	81.57	86.73	83.50	121.21	117.00
4-bit	81.83	78.09	88.01	83.84	121.84	114.95

Table 3. Impact of quantization on word-level perplexity of an LSTM for language modeling.

Table 3 compares the word-level perplexity of the LSTM network between full-precision (float) and quantization cases. The result shows that 4-bit weights achieve comparable results to the full-precision implementation. Also, our scheme provides options to further reduce the model size and the amount of computation by using fewer bits at a cost of lower output quality (i.e., higher perplexity).

6. Conclusion

In this paper, we proposed a novel weight/activation quantization method based on the concept of weighted entropy. The key benefits of our approach are twofold: (1) flexible multi-bit quantization, which allows us to optimize the neural network design under the tight accuracy loss constraint and (2) automated quantization, which does not require modifications to the input networks. According to our extensive evaluation results based on practical neural networks including AlexNet, GoogLeNet, ResNet-50/101, R-FCN, and an LSTM, our approach achieves 1% accuracy loss (top-5 or mAP) with 4-bit weights/ activations (AlexNet), 4/5-bit weights and 6-bit activations (GoogLeNet, ResNet and R-FCN). Our future work includes investigating the effectiveness of our method on very deep neural network models (e.g., ResNet-152) and devising activation quantization for RNN models.

Acknowledgement

This work was supported by National Research Foundation of Korea (NRF-2016R1A2B3009361) and Samsung Electronics (SAIT and Samsung Research Funding Center SRFC-TC1603-04).

References

[1] Facebook Caffe2Go. https://code.facebook.com/posts/196146247499076/. Accessed: 2016-11-15.

[2] Nvidia P40 and P4. https://devblogs.nvidia.com/parallelforall/mixed-precision-programming-cuda-8/. Accessed: 2016-11-15.

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[4] M. Courbariaux, Y. Bengio, and J.-P. David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131, 2015.

[5] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. *arXiv preprint arXiv:1605.06409*, 2016.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[7] M. Gottmer. Merging reality and virtuality with Microsoft HoloLens, 2015.

[8] S. Guiaşu. Weighted entropy. *Reports on Mathematical Physics*, 2(3):165–179, 1971.

[9] S. Guiasu. Grouping data by using the weighted entropy. *Journal of Statistical Planning and Inference*, 15:63–69, 1986.

[10] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding. *CoRR, abs/1510.00149*, 2, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[12] K. Hwang and W. Sung. Fixed-point feedforward deep neural network design using weights +1, 0, and -1. In *Proceedings of the IEEE Workshop on Signal Processing Systems*, pages 1–6, 2014.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the International Conference on Multimedia*, pages 675–678, 2014.

[14] P. Judd, J. Albericio, and A. Moshovos. Stripes: Bit-serial deep neural network computing. *IEEE Computer Architecture Letters*, 2016.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[16] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

[17] D. Miyashita, E. H. Lee, and B. Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[18] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang. Going deeper with embedded FPGA platform for convolutional neural network. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, pages 26–35, 2016.

[19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. *XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks*, pages 525–542. 2016.

[20] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[22] V. Vanhoucke, A. Senior, and M. Z. Mao. Improving the speed of neural networks on CPUs. In *the Deep Learning and Unsupervised Feature Learning Workshop at NIPS*, 2011.

[23] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, ukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[24] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[25] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.