Layer Importance Estimation with Imprinting for Neural Network Quantization

Hongyang Liu
University of Alberta
hliu11@ualberta.ca

Sara Elkerdawy
University of Alberta
elkerdaw@ualberta.ca

Nilanjan Ray
University of Alberta
nray1@ualberta.ca

Mostafa Elhoushi
Huawei
m.elhoushi@ieee.org

Abstract

Neural network quantization has achieved a high compression rate using fixed low bit-width representation of weights and activations while maintaining the accuracy of the high-precision original network. However, mixed precision (per-layer bit-width precision) quantization requires careful tuning to maintain accuracy while achieving further compression and higher granularity than fixed-precision quantization. We propose an accuracy-aware criterion to quantify the layer's importance rank. Our method applies imprinting per layer which acts as a proxy module for accuracy estimation in an efficient way. We rank the layers based on the accuracy gain from previous modules and iteratively quantize first those with less accuracy gain. Previous mixed-precision methods either rely on expensive search techniques such as reinforcement learning (RL) or end-to-end optimization with a lack of interpretation to the quantization configuration scheme. Our method is a one-shot, efficient, accuracy-aware information estimation and thus draws better interpretability to the selected bit-width configuration.

1. Introduction

Deep neural networks (DNN) models achieved state-of-the-art (SOTA) accuracy in numerous applications. However, DNNs commonly demand large computational costs and memory consumption. Numerous light-weight footprint architectures are designed to tackle the computation bottleneck deployment on embedded and edge devices [13, 15, 14, 32]. Other ways to address this challenge are model pruning [11, 22, 7], low-rank factorization [36, 4, 24], and quantization [19, 35, 23] that have been extensively studied. Related to our work, quantization converts weights and optionally activations from 32-bit floating points to a low precision representation. The quantized model requires less memory storage and computation cost resulting in faster inference. Different layers might require different low-bit representations based on a multitude of factors such as weight distribution, information gain, and sensitivity to quantization.

Quantization stores parameters of a model with a low-bit precision and utilizes the hardware's ability to efficiently process computational operations such as convolution with low-bit operations to speed up the inference, which enables us to run neural networks on edge devices. Conventional quantization methods referred to as fixed-precision quantization in the paper such as [23, 3, 26] ignore the fact that different layers of a network have different levels of redundancy. This observation led to the development of mixed-precision quantization methods [35, 39, 5, 27, 34]. However, the challenge is to determine the bit length per layer and explore the vast design space with minimal expert knowledge. Current mixed-precision methods adopt computationally expensive optimization to search for the configuration with optimal bit-length per layer. Our method is a one-shot search optimization that is followed by a constant fine-tuning step. Computation in other quantization methods grows linear in the compression target budget. That stems from the iterative and progressive joint quantization-finetuning usually adopted in current methods. Figure 1 shows different budget-architecture pairs. Our method achieves comparable accuracy with smaller model sizes while adopting an efficient one-shot search optimization.

In this paper, we propose an efficient one-shot mixed-precision quantization method. We introduce an accuracy-aware criterion using imprinting which is introduced in few-shot learning literature [29] and adopted in pruning [7]. We insert a proxy classifier after each layer and estimate the accuracy gain per layer. The layer with the lowest rank is quantized and imprinting is re-applied again to update the ranks. This process is done iteratively until a model size

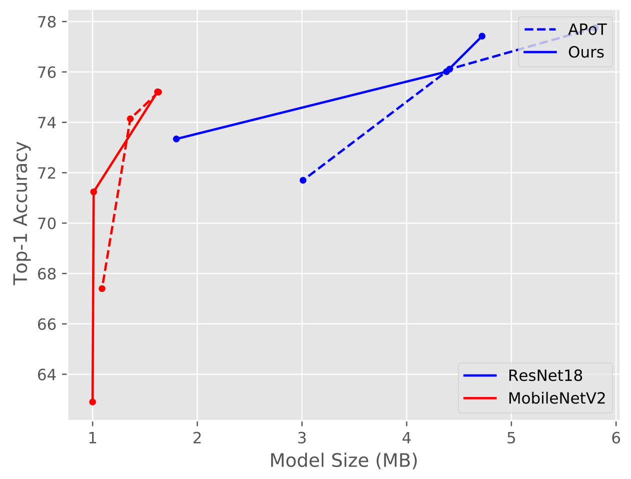


Figure 1: Comparison under different dataset-budget pairs with ResNet20 on CIFAR-100.

budget is reached. Finally, a fine-tunning step is performed using the bit-length configuration. Our contributions can be summarized as follows:

- We introduce the idea of accuracy estimation to bring any fixed-precision quantization to a compressed mixed-precision quantization so that further compression can be reached with higher flexibility.
- We propose an accuracy-aware criterion to weigh the importance of each layer. This will allow us to have a better interpretation of the final bit-length configuration compared to other search methods.
- Our search method using imprinting is a one-shot step, we are able to quickly identify rank the layers resulting in shorter training time.

2. Related Work

In this section, we split quantization methods into two main categories: 1) fixed-precision quantization, 2) mixed-precision quantization.

2.1. Fixed-Precision Quantization

A plethora of network quantization methods has been extensively studied and proposed. Han et al. [10] quantized model using heuristics jointly with pruning. Krishnamoorthi et al. [20] propose two designs of uniform quantization leading to the conclusion that uniform quantization can achieve 4x model size reduction with no accuracy drop but to reach higher compression, non-uniform quantization techniques will be required.

Zhou et al. [37] and Miyashita et al. [26] proposed non-uniform quantization methods using powers-of-two values. These methods give higher resolution for values near the center of a bell-shaped weight/activation distribution [9]. Power-of-two (PoT) method converts floating-point multiplications into powers-of-two multiplications, which can be easily achieved by utilizing efficient bit-shift operations. TQT [18] introduced uniform quantization using trainable interval values for thresholding.

Li et al. [23] improves over these methods to recover from what they called rigid resolution limitation for weight/activation values around zero. Therefore, PoT is unable to increase a model's expressiveness effectively. Li et al. [23] proposed Additive-Power-of-two (APoT) quantization which adopts a finer resolution around zero, and thus, reduces information loss. One issue with APoT is that the model needs to be progressively trained from higher bits to lower bits in order to get the optimal results. This will result in a time-consuming training time for highly compressed models.

Jacob et al. [17] managed to use integer-only arithmetics during inference by quantizing the weights and activations as 8-bit integers and biases as 32-bit integers during training. Quantization-aware training that simulates quantization errors during training was also proposed in [17].

Some recent works focused on binary networks which are essentially networks with 1-bit precision networks. [25] proposed Bi-Real Net which is a variant of the residual structure that preserves the real activation before the sign function. [16] introduced a method that adds Shannon entropy based penalty to convolutional filters to maintain the same level of information capacity throughout the training process for binary networks.

2.2. Mixed-Precision Quantization

Mixed-precision quantization methods focus on optimizing the bit-length per layer. As the number of layers in the neural network increases, the search space of the traditional neural network quantization method will show exponential growth. As a result, it is impossible to achieve the optimal quantization precision per layer by brute force search. Wang et al. [34] introduce the hardware-aware automatic quantization (HAQ) framework based on reinforcement learning (RL). The HAQ framework uses the RL strategy, which can help predict the latency time of specific hardware given a specific bit length. Then the RL proxy is used to determine the bandwidth of each layer, that is, the weight of each layer and the width of the activation bits. However, the training time for HAQ is significant due to the nature of RL.

BitPruning [27], formulates quantization as an optimization problem, which incorporates a regularization loss to penalize high bit-length representations throughout the architecture. The results for ResNet18 [12] on Imagenet [30] show that the quantized model can guarantee at least 90% accuracy with low bit-length. As a result, the new bit pruning strategy can quantify each layer of the neural network better. It greatly reduces memory and the consumption of hardware. However, end-to-end methods require incremental careful joint optimization between task and quantization loss and hard to optimize. In addition, they lack interpretation of the final configuration.

In ZeroQ [2], a generated data obtained from the statistics of batch normalization layers are used in bit configuration search instead of the original data. Pareto frontier optimization is proposed using the synthetic data generated to find the optimal bit-precision configuration. Differential Quantization (DQ)[33] proposed a method to learn both the range and threshold of quantization jointly with model weights, from which the bit width can be inferred.

3. Methodology

Our motivation comes from the fact that different layers have different effects on the final accuracy of a model [8]. Therefore, they should be treated differently during quantization. We propose to use imprinting [28], which is a one-shot pass, to approximate the accuracy up to each layer. We then rank based on the accuracy gain and efficiently single out layers that will harm the final accuracy less after quantization. We refer to this as an important estimation. Our focus in the paper is to show the efficiency of imprinting as a search optimization for the bit-length configuration; hence, we adopt different quantization methods. We experimented with uniform and non-uniform methods proposed in fixed-precision quantization literature. Weights and activations are quantized using any arbitrary quantization method as per our final bit-length configuration after imprinting. Our solution consists of two steps. First, we select the bit configuration using Algorithm 1. Then, we fine-tune the model with the selected bit configuration from the previous step, using various quantization methods. The selection of the optimal bit-width is orthogonal to the different quantization methods proposed in the literature.

3.1. Preliminaries

For quantization, we consider two operations: clipping and projection. Suppose we define the weight of a convolutional layer as W, then the quantized weight will be defined as:

$$\tilde{W} = \prod_{Q(\alpha,b)} \lfloor W, \alpha \rceil \tag{1}$$

where $\lfloor W, \alpha \rceil$ is the clipping function that clips W to the range $[-\alpha, \alpha]$. $Q(\alpha, b)$ is a set of quantization levels whose maximum is α, and b is the bit-width. $\prod$ denotes the projection function which will project the clipped weight onto the quantization levels. For uniform quantization, the quantization levels will be

$$Q_{uni}(\alpha, b) = \alpha \times \{0, \frac{\pm 1}{2^{b-1}-1}, \frac{\pm 2}{2^{b-1}-1}, \frac{\pm 3}{2^{b-1}-1}, ..., \pm 1\} \tag{2}$$

3.2. Bit-width selection

Ideally, to accurately rank layers based on task accuracy, we could insert an untrained task predictor after each layer. That is to create a classifier head with adaptive average pooling, fully-connected layer, and softmax, then train the weights of these classifiers after each layer. This will be time-consuming to train such a large number of classifiers. Instead, we adopt imprinting to approximate the weights of the fully connected layer without training.

We use imprinting to get the approximate weights with only one epoch. The method is adopted from [7], we used quantized convolutional layer instead of what's used in [**?**]. To simplify the imprinting module across all layers, we sample from the input feature maps using adaptive average pooling. The kernel size for adaptive average pooling is automatically calculated such that all layers generate a 1D embedding of a similar length. This stage is named as weights imprinting stage which can be formulated as the following [7]:

$$\begin{aligned} d &= round(\sqrt{\frac{N}{f_i}}) \\ E_i &= AdaptiveAvgPool(F_i, d), \end{aligned} \tag{3}$$

where N is the length of embedding and f_i is the number of filters in the i-th layer. F_i is the i-th layer's feature map and E_i is our re-shaped embedding which is also the input of imprinting.

The layer ranking stage is then formulated to calculate the accuracy using the imprinted weights. Figure 2 shows the whole pipeline for the imprinting step. This setup is motivated by the pruning work [7].

For example, we can see from Figure 5 for ResNet20 on CIFAR-10, some layers have similar or even lower estimated accuracy as the layer before them (e.g. layer conv18 has the same accuracy as layer conv17 in the Figure 5). This implies that these latter layers have less importance gain compared to the early ones and thus can be compressed using a smaller bit length.

We start with all the layers set to the same bit length. Then we iteratively select one layer to be 1 bit lower in both weights and activations based on the difference between the estimated accuracy and that of the previous layer until the target budget on the bit length is reached. This budget constraint could be the total minimum bit length or model size. The effect of this method is shown in Figure 6.

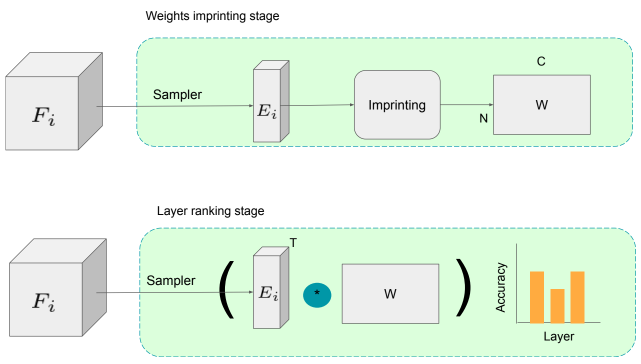


Figure 2: Pipeline illustrating layer ranking by imprinting. An embedding E_i is sampled from input feature maps F_i. Imprinting is then applied on the embedding to estimate the weight matrix W. Accuracy per layer is then calculated by a simple dot product between embedding and imprinted weights.

3.3. Quantization

The bit-width selection method mentioned before can be applied to any fixed-precision quantization method. Quantization consists of clipping and projection, for example, APoT, has these two parts respectively: the reparameterized clipping function and the additive power of two quantization schema.

Reparameterized Clipping Function The clipping threshold α is the value that defines the range of weights in a quantization layer. If α is too small, the clipping function will throw away too many outliers. If α is too large, the weights around the center will not have enough resolution for projection, since the weights are long-tail distributed [9]. Thus, finding an optimal α is a crucial challenge in quantization.

Most of the recent papers use a Straight-Through Estimator (STE) [1] to help jointly train the clipping threshold and weights. STE is used to estimate the gradient of α for back propagation:

$$\frac{\partial \tilde{W}}{\partial \alpha} = \begin{cases} \frac{\partial \lfloor W, \alpha \rceil}{\partial \alpha} = sign(W) & if|W| > \alpha \\ 0 & if|W| \leq \alpha \end{cases} \tag{4}$$

It is clear that the clipped weights are not contributing to the gradient. Thus, the estimation is not accurate here. Therefore, AOPT [23] proposed the following clipping function:

$$\tilde{W} = \alpha \prod{}_{Q(1,b)} \lfloor \frac{W}{\alpha}, 1 \rceil, \tag{5}$$

which scales the weights into range $[-1, 1]$ then scales them back after projection. The estimated gradient is, therefore, changed to

$$\frac{\partial \tilde{W}}{\partial \alpha} = \begin{cases} sign(W) & if|W| > \alpha \\ \prod_{Q(1,b)} \frac{W}{\alpha} - \frac{W}{\alpha} & if|W| \leq \alpha \end{cases} \tag{6}$$

Additive Power of Two Unlike uniform quantization, [23] defines the quantization levels as a set of sums of multiple power-of-two terms, as shown below:

$$\begin{gathered} Q_{APoT}(\alpha, kn) = \gamma \times \{\sum_{i=0}^{n-1} p_i\}, \\ where\ p_i \in \{0, \frac{1}{2^i}, \frac{1}{2^{i+n}}, ..., \frac{1}{2^{i+(2^k-2)n}}\} \end{gathered} \tag{7}$$

where k is a hyper parameter that defines base bit-width, and n is the number of additive terms. If the bit-width is b, n can be calculated as $n = \frac{b}{k}$. Finally, γ is a scaling coefficient to guarantee the maximum level equals α

These quantization levels will project the values non-uniformly and enable more levels near 0 which is the center of the weights after clipping. Since the distribution of the weights of convolutional layers is long-tailed and bell-shaped [9]. This will give us the ability to have more resolution around the center.

4. Experiments & Analysis

We validate our method with ResNet18 [6], ResNet-20, ResNet-56 [12] and MobileNet-V2 [31] on CIFAR-10 and CIFAR-100 [21] datasets. We also evaluate ResNet18 on ImageNet-ILSVRC2012 [30]. For our method, we start our training by setting the weights and activations of all the convolutional layers to a 4 bit-width representation for CIFAR and 5 bit-width for ImageNet. The first and last layer, however, is kept at 8 bit to balance the accuracy drop and hardware overhead as a common practice in quantization papers [**?**, **?**, **?**]. The minimum bit lengths of weights and activations are set to 1 and 2, respectively, following [38] which shows that increasing bit-width of activation from 1-bit to 2-bit leads to a better accuracy-efficiency trade-off. For CIFAR, the number of fine-tuning epochs is 300 with batch size=128. The learning rate is initialized with 0.1 and is divided by 10 at 150 and 225 epoch. For ImageNet, the number of fine-tuning epochs is 120 with batch size=256. The initial learning rate is set to 0.01 and decays by 10 times every 30 epochs. We did not use quantization-aware training [17] but the clipping threshold α is trainable and is learned during the training process.

4.1. CIFAR

Imprinting Evolution We first show the layer rank evolution using imprinting as a criterion. Figure 6 shows sampled iterations from different stages in the search loop to select bit length for ResNet-20 on CIFAR-10. The x-axis

Algorithm 1 Pipeline for bit-selection

Input: initial bit length configuration of all convolutional layers B_{start}, minimum bit length configuration of all convolutional layers B_{end} , number of convolutional layers K
Output: bit configuration for all convolutional layers B

```
 1: Calculate maximum total bit-length $\hat{B}_{max} = \sum_{i=0}^{K} b_i$ , $b_i \in B_{start}$
 2: Calculate minimum total bit-length $\hat{B}_{min} = \sum_{i=0}^{K} b_i$ , $b_i \in B_{end}$
 3: Calculate maximum of iterations needed $N_{max} = \hat{B}_{max} - \hat{B}_{min} + 1$
 4: Initialize the configuration to use $B_{current} = B_{start}$
 5: for iteration $N = 1, 2, \ldots, N_{max}$ do
 6:     With $B_{current}$, use imprinting with quantization to get the estimated accuracy of each convolutional layer, $Acc = \{acc_1, acc_2, \ldots, acc_K\}$
 7:     Find the difference between each layer and its previous layer, $diff = \{\|acc_i - acc_{i-1}\| \,|\forall i \in [2, K-1], acc_i \in Acc\}$
 8:     Find the index of the minimum difference, $idx = argmin(diff))$
 9:     Record the configuration as $B_N$
10:     Record the accuracy of the last layer as $Acc_N$
11:     Update the configuration $B_{current}$ by setting $b_{idx} = b_{idx} - 1$
12: end for
13: Choose the best configuration $B = B_N$ where $Acc_N = max(\{Acc_i | \forall i \in [1, N_{max}]\})$
```

shows the type of the convolutional layers and the y-axis represents the estimated accuracy obtained through imprinting. The first and last layers are set to 8 bits and will not be selected to change their bit length for the entire selection process. Figure 6 shows that the second layer of each block will usually bring the accuracy down which is also observed in [7]. This can be attributed to the fact that the second layer in ResNet block heavily acts as a residual to the identity path so accuracy using its output alone is not meaningful. In the first step, as shown in Figure 6a, the second layer of the 8th block, $conv8.2$, has the smallest difference from the previous layer. Therefore, $conv8.2$ will be selected to be 1 bit smaller, which is 3 bits. After 7 iterations (Figure 6b), $conv8.2$ is selected again, which sets its configuration to 2 bits. Around 20 iterations later, $conv5.1$ will be selected as shown in Figure 6c, leading to a bit length of 2. This process continues until we meet the target average bit length requirement. The final layer's accuracy of each iteration will be recorded as a measurement of the performance of that bit-length configuration. The configuration with the best final-layer accuracy within a certain average bit length range is selected as the final configuration. The overall accuracy from one iteration to the next drops because there is no fine-tuning between the iterations. Adding a small fine-tuning step between the iterations can help to maintain a similar level of accuracy (see Figure 7). However, since the final fine-tuning result after using such bit-selection with fine-tuning does not improve much, we didn't include it in our algorithm.

Different quantization With this selection method, we conduct experiments using different quantization methods to show the applicability of our method to different quantization methods. We apply fixed-precision quantization results for Uniform, Power of Two, Additive Power of Two (APoT) quantization methods, with bit-width set to 4, 3, and 2. Then we apply our method to each of these methods to find the best mixed-precision configuration whose average bit length is in-between 4, 3, and 2 bits. These experiments are evaluated using ResNet-20 on CIFAR-10. The final bit-length configuration is shown in Figure 4. To compare to more state-of-the-art methods, we also included results for PACT and ZeroQ [2] as well as the results reported in [33] for the Differential Quantization (DQ)[33] and Trained Quantization Threshold(TQT)[18] methods. These results are shown in Table 1. PACT[3] uses uniform quantization. Comparing to our method with uniform quantization, PACT is outperformed. ZeroQ [2] achieves better accuracy than our method on a similar model size but the average bit length for the activation is also higher than ours. Comparing to DQ [33] and TQT [18], we are able to maintain a comparable level of accuracy with a similar weight size with ResNet-20 on CIFAR-10. We can also see that as the average bit length gets smaller, the accuracy is not hurt significantly. One huge advantage of our method compared to APoT [23], is that, instead of using the fine-tuned pre-trained higher precision model as a starting point to train a lower bit configuration, our approach does not require such a high amount of fine-tuning epochs to reach a low bit configuration and smaller model size. It is worth mentioning that in some cases the model size can be larger even if the bit length for weights and activations are smaller. The reason is that these bit lengths are average values of all convolutional layers and some layers are much more compressed than in our mixed-precision quantized model than their counterpart fixed-precision quantized models.

We also evaluate our method with APoT quantization on different models. Results of CIFAR-10 and CIFAR-100 are shown in Table2 and 3 respectively. From the table, we can see that we can achieve results similar to APoT[23] on most of the ResNet models, but with less average bit length and model size. We have a larger accuracy drop for MobileNet-V2 compared to APoT[23]. However, we are still able to produce some mixed-precision settings with less model size but higher accuracy than APoT.

Method	Weight	Activation	Size (MB)	Accuracy	Epoch
PACT [3]	4	4	-	91.3	-
APoT [23]	4	4	0.14	92.45	300
PoT	4	4	0.14	91.85	300
Uniform	4	4	0.14	92.86	300
ZeroQ [2]	learned	8	0.13	93.16	-
Ours(APoT)	3.85	3.85	0.13	92.82	300
Ours(Uniform)	3.1	3.3	0.12	92.04	300
Ours(PoT)	3.3	3.4	0.11	91.37	300
PACT [3]	3	3	-	91.1	-
Ours(Uniform)	2.8	3.15	0.11	91.87	300
APoT [23]	3	3	0.1	92.49	600
PoT	3	3	0.1	91.78	600
Uniform	3	3	0.1	92.36	600
Ours(PoT)	2.8	3	0.1	91.29	300
Ours(APoT)	2.3	2.7	0.1	91.65	300
PACT [3]	2	2	-	89.7	-
DQ(Uniform) [33]	learned	4	0.07	91.42	160
DQ(POT) [33]	learned	4	0.07	88.77	160
APoT[23]	2	2	0.07	90.96	900
PoT	2	2	0.07	91.14	900
Uniform	2	2	0.07	91.06	900
Ours(PoT)	1.7	2.4	0.07	90.28	300
TQT (Uniform) [18]	2.3	4	0.065	90.83	160
TQT (POT) [18]	2.3	4	0.065	88.71	160
Ours(APoT)	1.6	2.4	0.06	90.64	300
Ours(Uniform)	1.5	2.3	0.06	90.04	300

Table 1: Comparing different quantization methods with ResNet20 on CIFAR-10

Model	Method	Weights	Activations	Accuracy	Size (MB)	Epoch
	Baseline	32	32	94.97	44.61	300
	APoT	4	4	94.57	5.62	300
	Ours	3.07	4.35	94.11	4.87	300
ResNet18	Ours	2.7	3.11	94.02	4.79	300
	APoT	3	3	94.13	4.23	600
	Ours	1.96	2.55	93.42	2.95	300
	APoT	2	2	93.22	2.84	900
	Baseline	32	32	92.96	1.04	300
	APoT	4	4	92.45	0.14	300
	Ours	3.85	3.85	92.82	0.13	300
ResNet20	APoT	3	3	92.49	0.1	600
	Ours	2.3	2.7	91.65	0.1	300
	APoT	2	2	90.96	0.07	900
	Ours	1.65	2.4	90.64	0.07	300
	Baseline	32	32	94.46	3.31	300
	APoT	4	4	93.93	0.42	300
	Ours	3.37	3.42	93.74	0.32	300
ResNet56	APoT	3	3	93.77	0.32	600
	APoT	2	2	93.05	0.21	900
	Ours	2.37	2.75	92.89	0.2	300
	Ours	1.82	2.42	92.22	0.16	300
	Baseline	32	32	94.24	8.7	300
	APoT	4	4	89.99	1.19	300
	APoT	3	3	83.85	0.92	600
MobileNetV2	Ours	3.32	3.39	84.82	0.87	300
	APoT	2	2	69.79	0.66	900
	Ours	2.48	2.83	74.42	0.63	300
	Ours	1.32	2.14	63.92	0.55	300

Table 2: Comparing different Models (CIFAR-10)

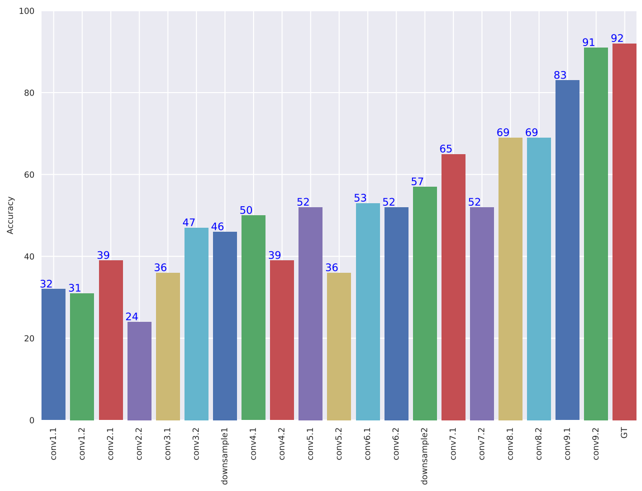


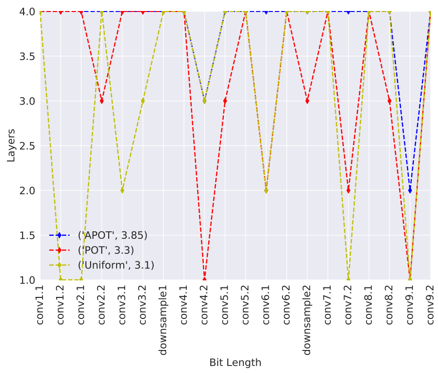
Figure 3: Accuracy approximation using imprinting. The model used is ResNet20 on CIFAR-10

Model	Method	Weights	Activations	Accuracy	Size (MB)	Epoch
	Baseline	32	32	78.07	44.79	300
	APoT	4	4	77.75	5.8	300
	Ours	3.03	3.22	77.42	4.72	300
ResNet18	APoT	3	3	76.11	4.41	600
	Ours	2.67	3.04	76.01	4.38	300
	APoT	2	2	71.7	3.01	900
	Ours	1.29	2.18	73.34	1.8	300
	Baseline	32	32	66.93	1.09	300
	APoT	4	4	66.95	0.16	300
	Ours	3.8	3.8	67.9	0.15	300
ResNet20	APoT	3	3	66.98	0.13	600
	Ours	2.3	2.75	66.42	0.12	300
	APoT	2	2	66.42	0.09	900
	Ours	1.8	2.45	64.25	0.09	300
	Baseline	32	32	94.46	3.27	300
	APoT	4	4	93.93	0.42	300
	Ours	3.37	3.42	93.74	0.32	300
ResNet56	APoT	3	3	93.77	0.32	600
	APoT	2	2	93.05	0.21	900
	Ours	2.37	2.75	92.89	0.2	300
	Ours	1.82	2.42	92.22	0.16	300
	Baseline	32	32	75.58	9.13	300
	APoT	4	4	75.2	1.63	300
	Ours	3.94	3.94	75.21	1.62	300
MobileNetV2	APoT	3	3	74.14	1.36	600
	APoT	2	2	67.4	1.09	900
	Ours	2.19	2.69	71.24	1.01	300
	Ours	1.71	2.37	62.9	1	300

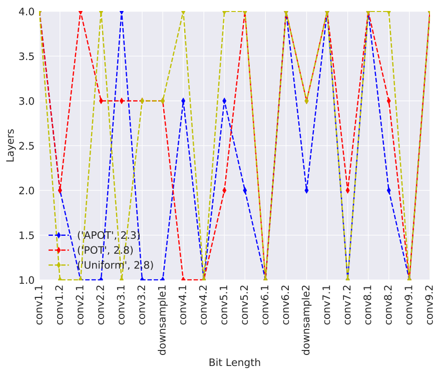
Table 3: Comparing different Models (CIFAR-100)

4.2. ImageNet

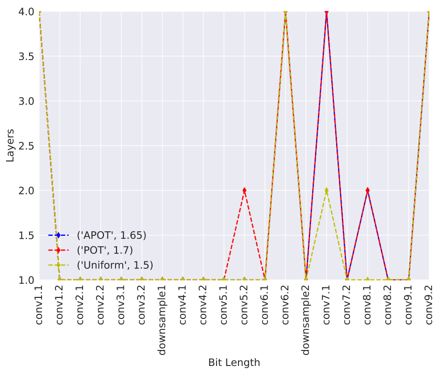
Table 5 shows the result of ResNet-18 on ImangeNet. We compare our result to Pact[3] and BitPruning[27]. Our method outperforms the mixed quantization method BitPruning and the fixed quantization method PACT on a similar configuration. Although we achieve comparable accuracy to APoT, we perform quantization in constant time regardless of the budget constrain. Unlike APoT, which applies an iterative process where each fine-tuned model is used as a starting point for the next bit-width configuration. This means training time grows linearly as a lower budget is targeted.



(a) The final bit-length configuration for models with average bit length $\in [3, 4]$



(b) The final bit-length configuration for models with average bit length $\in [2, 3]$



(c) The final bit-length configuration for models with average bit length $\in [1, 2]$

Figure 4: The final big-length configuration for our ResNet20 models trained on CIFAR-10 with average weight bit-length 3 reported in 1

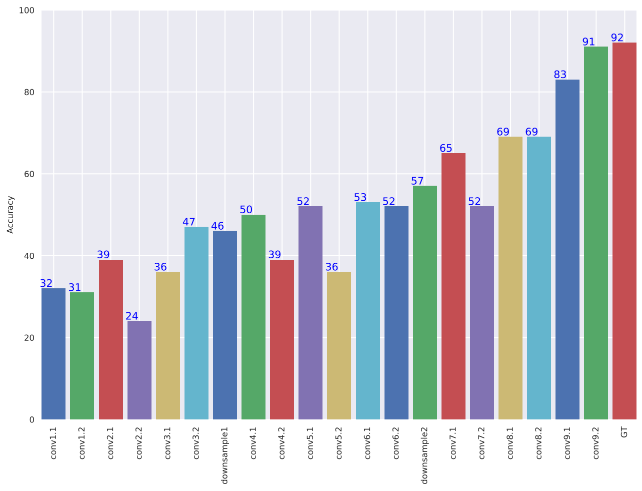


Figure 5: Accuracy approximation using imprinting. The model used is ResNet20 on CIFAR-10

Criterion	Weights	Activations	Accuracy	Size (MB)	Epochs
max_variance	3.9	3.9	92.78	0.14	460
norm	3.9	3.9	92.66	0.14	460
qt_norm	3.8	3.8	92.58	0.14	460
Imprinting	3.85	3.85	92.82	0.13	300
min_variance	3.8	3.8	92.68	0.13	460
qt_norm	2.85	2.85	92.25	0.13	620
max_variance	2.9	2.9	91.77	0.13	620
Imprinting	2.3	2.7	91.65	0.1	300
norm	2.95	2.95	92.16	0.09	620
min_variance	2.9	2.35	91.99	0.07	620
qt_norm	1.85	2.8	90.82	0.09	780
max_variance	1.95	2.6	90.69	0.07	780
Imprinting	1.65	2.4	90.64	0.07	300
norm	1.95	2.2	91.09	0.06	780
min_variance	1.9	2.35	90.73	0.06	780

Table 4: Different bit selection criteria with ResNet-20 on CIFAR-10. The quantization method used is APoT[23]. ”norm”, ”min_variance” and ”max_variance” are the average statistics of full precision weights, whereas ”qt_norm” is the average norm of quantized weights.

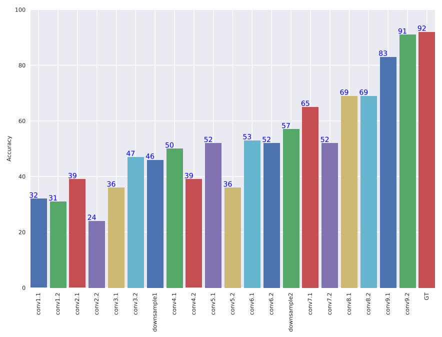
4.3. Ablation Study

4.3.1 Imprinting vs Statistical criteria

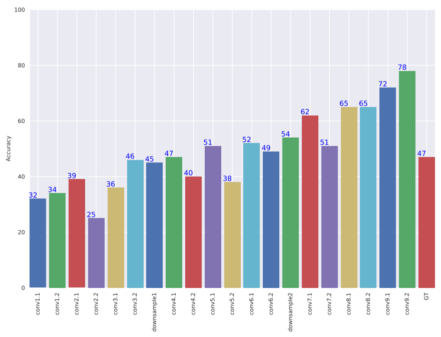
Our bit length selection with the imprinting method is an efficient way to estimate layer importance based on accuracy. We also experimented with other statistical criteria for layer importance estimation such as ℓ_2 norm and variance. The norm of a layer is calculated by taking the mean value of the norm across the last two dimensions of the weights. For each criterion, similar to imprinting, we rank the layers based on that criteria, and then the selected layer is quantized. However, as statistical criteria are model-based, we perform a short-term fine-tuning for these criteria. We observed that fine-tuning for one epoch does not produce an accurate representation of the ranking of the different configurations. To produce the best results of these criteria, we have to fine-tune the model for a couple of epochs in each iteration of the bit-length selection process. For this experiment, we find that 8 epochs of fine-tuning are enough to accurately represent the weights. The model used is ResNet-20 on CIFAR-10. From Table 4, it is shown that the effect of the two criteria is quite similar. However, as the imprinting method is one-shot and the statistical criteria require fine-tuning for a couple of epochs each selection, the imprinting method is more efficient in terms of the overall time.

5. Conclusion

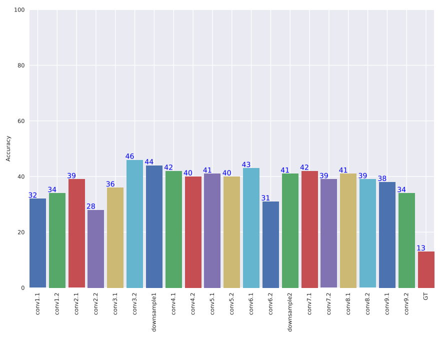
In this paper, we have introduced a bit-length selection method to identify and rank the importance of each convolutional layer by using one-shot imprinting. This method



(a) An iteration from early stage. The second convolutional layer in the 8th block for ResNet20 will reduce its bitwidth by 1

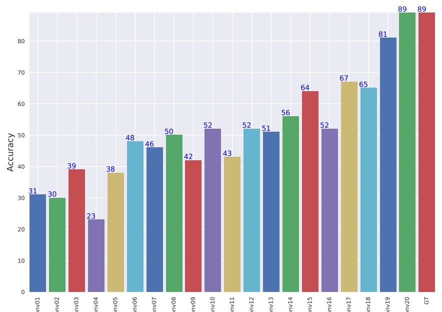


(b) An iteration from middle stage. The second convolutional layer in the 8th block for ResNet20 will reduce its bitwidth by 1

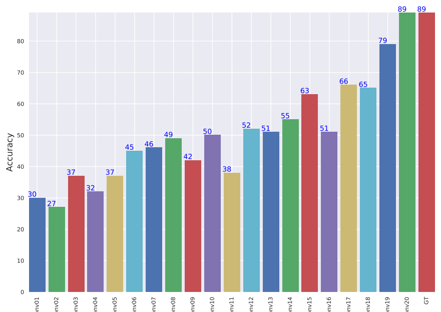


(c) An iteration from late stage. The first convolutional layer in the 5th block for ResNet20 will reduce its bitwidth by 1

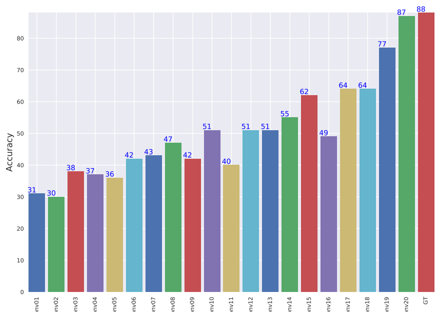
Figure 6: Evolution of layer ranking in iterative imprinting. The model used is ResNet20 on CIFAR-10 dataset.



(a) An iteration from early stage. The second convolutional layer in the 1st block for ResNet20 will reduce its bitwidth by 1



(b) An iteration from middle stage. The first downsample layer for ResNet20 will reduce its bitwidth by 1



(c) An iteration from late stage. The second convolutional layer in the 6th block for ResNet20 will reduce its bitwidth by 1

Figure 7: Evolution of layer ranking in iterative imprinting. We added a fine-tuning of 8 epochs between interations compared to Figure 6. The model used is ResNet20 on CIFAR-10 dataset.

Method	Weights	Activations	Accuracy
PACT [3]	5	5	69.8
APoT	5	5	70.75
Ours(APoT)	4.38	4.38	70.59
PACT [3]	4	4	69.2
APoT	4	4	70.74
Ours(APoT)	3.55	3.55	70.12
BitPruning [27]	3.38	4.14	69.19
PACT [3]	3	3	68.1
APoT	3	3	69.79
Ours(APoT)	2.72	2.72	69.84
APoT	2	2	66.46

Table 5: Results for ResNet18 on Imagenet. We compared our results with the fixed-precision methods PACT[3] and APoT[23], as well as mixed-precision method BitPruning[27].

gives us the ability to convert any fixed-precision quantization method into mixed-precision, which usually produces neural network models with smaller model sizes. The use of imprinting also reduces the training epochs required to reach a relatively low average bit length. We have acquired comparable results on CIFAR-10, CIFAR-100, and ImageNet, compared to APoT. In future work, we want to investigate the use of different search methods in comparison to the proposed greedy approach.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[2] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. *CoRR*, abs/2001.00281, 2020.

[3] 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.

[4] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *CoRR*, abs/1404.0736, 2014.

[5] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: hessian aware quantization of neural networks with mixed-precision. *CoRR*, abs/1905.03696, 2019.

[6] Mostafa Elhoushi, Farhan Shafiq, Ye Henry Tian, Joey Yiwei Li, and Zihao Chen. Deepshift: Towards multiplication-less neural networks. *CoRR*, abs/1905.13298, 2019.

[7] S. Elkerdawy, M. Elhoushi, A. Singh, H. Zhang, and N. Ray. One-shot layer-wise accuracy approximation for layer pruning. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2940–2944, 2020.

[8] Sara Elkerdawy, Mostafa Elhoushi, Abhineet Singh, Hong Zhang, and Nilanjan Ray. To filter prune, or to layer prune, that is the question. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[9] Song Han, Huizi Mao, and William Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *arXiv preprint arXiv:1510.00149*, 10 2016.

[10] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[11] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[15] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[16] Dmitry Ignatov and Andrey Ignatov. Controlling information capacity of binary neural network. *Pattern Recognition Letters*, 138:276–281, Oct 2020.

[17] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CoRR*, abs/1712.05877, 2017.

[18] Sambhav R. Jain, Albert Gural, Michael Wu, and Chris Dick. Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware. *CoRR*, abs/1903.08066, 2019.

[19] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342, 2018.

[20] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342, 2018.

[21] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016.

[23] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: A non-uniform discretization for neural networks. *CoRR*, abs/1909.13144, 2019.

[24] Yawei Li, Shuhang Gu, Luc Van Gool, and Radu Timofte. Learning filter basis for convolutional neural network compression, 2019.

[25] Zechun Liu, Baoyuan Wu, Wenhan 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. *CoRR*, abs/1808.00278, 2018.

[26] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *CoRR*, abs/1603.01025, 2016.

[27] Miloš Nikolić, Ghouthi Boukli Hacene, Ciaran Bannon, Alberto Delmas Lascorz, Matthieu Courbariaux, Yoshua Bengio, Vincent Gripon, and Andreas Moshovos. Bitpruning: Learning bitlengths for aggressive and accurate quantization, 2020.

[28] Hang Qi, Matthew Brown, and David G. Lowe. Learning with imprinted weights. *CoRR*, abs/1712.07136, 2017.

[29] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[31] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[32] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[33] Stefan Uhlich, Lukas Mauch, Kazuki Yoshiyama, Fabien Cardinaux, Javier Alonso García, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Differentiable quantization of deep neural networks. *CoRR*, abs/1905.11452, 2019.

[34] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[35] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *CoRR*, abs/1812.00090, 2018.

[36] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[37] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *CoRR*, abs/1702.03044, 2017.

[38] 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.

[39] Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. *CoRR*, abs/1712.01048, 2017.