SPIQ: Data-Free Per-Channel Static Input Quantization

Edouard Yvinec[1,2] , Arnaud Dapogny[2] , Matthieu Cord[1] , Kevin Bailly[1,2]
Sorbonne Université[1], CNRS, ISIR, f-75005, 4 Place Jussieu 75005 Paris, France
Datakalab[2], 114 boulevard Malesherbes, 75017 Paris, France
ey@datakalab.com

Abstract

Computationally expensive neural networks are ubiquitous in computer vision and solutions for efficient inference have drawn a growing attention in the machine learning community. Examples of such solutions comprise quantization, i.e. converting the processing values (weights and inputs) from floating point into integers e.g. int8 or int4. Concurrently, the rise of privacy concerns motivated the study of less invasive acceleration methods, such as data-free quantization of pre-trained models weights and activations. Previous approaches either exploit statistical information to deduce scalar ranges and scaling factors for the activations in a static manner, or dynamically adapt this range on-the-fly for each input of each layer (also referred to as activations): the latter generally being more accurate at the expense of significantly slower inference. In this work, we argue that static input quantization can reach the accuracy levels of dynamic methods by means of a per-channel input quantization scheme that allows one to more finely preserve cross-channel dynamics. We show through a thorough empirical evaluation on multiple computer vision problems (e.g. ImageNet classification, Pascal VOC object detection as well as CityScapes semantic segmentation) that the proposed method, dubbed SPIQ, achieves accuracies rivalling dynamic approaches with static-level inference speed, significantly outperforming state-of-the-art quantization methods on every benchmark.

1. Introduction

Deployment of State-of-the-art deep neural networks (DNNs) on edge devices has become increasingly difficult. Although edge computing has recently drawn more attention, motivated by privacy [30] and environmental sustainability concerns [22], DNNs have grown more computationally expensive. However, several techniques exist that aim at reducing this burden, among which quantization.

As defined in [14], quantization consists in mapping a set of continuous variables to a finite set of values, e.g. int8, int4 or ternary, in order to compress the bit-wise representation. Quantization trends can be distinguished by data-usage and range calibration.

First, the approximation introduced by quantization often requires adjustments in order to preserve the original accuracy of the model. This can be performed using real training data and is called data-driven quantization [17, 23, 32, 16]. Although such methods can afford lower bit-wise representations, they are both computationally expensive and less convenient to use. On the other hand, when quantization is performed without re-training, it is often referred to as post-training quantization (PTQ) or data-free quantization [3, 12, 37, 27, 6]. Such methods are convenient for applications where privacy and security are mandatory. This work aims at reducing the gap between data-free and data-driven quantization by rethinking input quantization.

Second, in order to quantize inputs, the range of their distribution has to be estimated. In data-free quantization, inputs of each layer are quantized based on statistics determined either from the already trained parameters (static quantization) [27] or based on statistics computed on-the-fly based on each sample at inference (dynamic quantization) [31]. The latter usually offers significantly higher accuracy at the expense of a slower inference, more-so on low bit-wise representations.

While most research on data-free quantization [2, 3, 5, 12, 13, 37, 27] don't thoroughly study the specificities of activation quantization. As such the naive per-channel activation quantization from [2] can't be applied in practice in addition to per-channel weight quantization. This is due to the requirement of rescaling each terms in the summation in the matrix products. We detail and tackle this issue in order to achieve true per-channel quantization for both weights and activations. As compared to standard static quantization, the proposed method offers strong benefits in terms of accuracy as illustrated in Fig 1. Furthermore, we show that per-channel input range estimation allows tighter modelling of the full-precision distribution as compared to a per-example, dynamic approach. We call this approach SPIQ, standing for **S**tatic **P**er-channel **I**nput **Q**uantization.

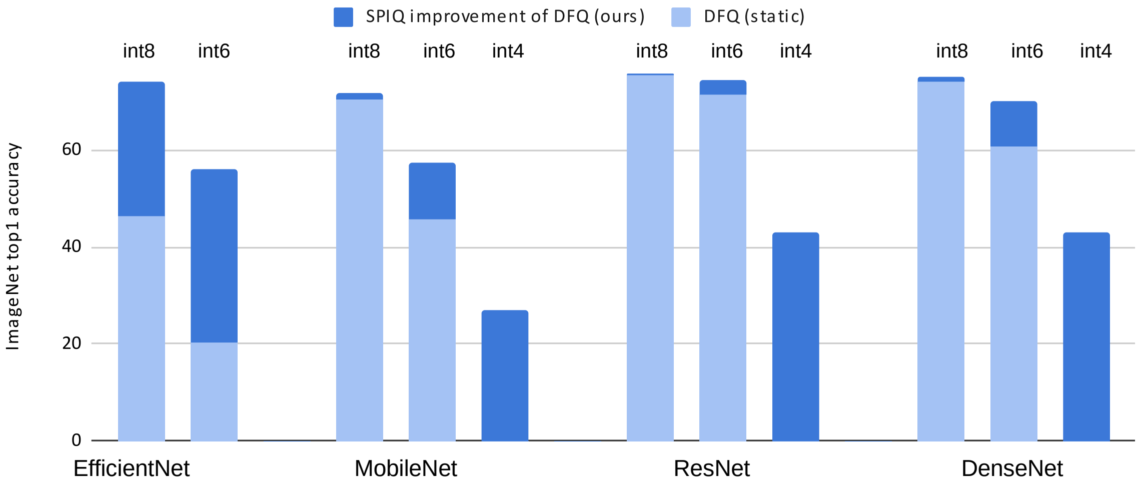

Figure 1. Illustration of the accuracy drop attributable to the input and activation quantization. We perform input quantization as defined in [27] as well as SPIQ (ours) but keep the DFQ quantized weight values, *i.e.* per-tensor weight quantization is applied. The results show that input quantization is paramount to the network accuracy preservation, most notably on already compact designs (e.g. MobileNet and EfficientNet). On all tested configurations, SPIQ significantly improves DFQ [27].

In practice, we show that SPIQ significantly improves over both the static and dynamic approaches. It also outperforms current state-of-the-art data-free quantization techniques on a variety of benchmarks, including image classification, object detection and semantic segmentation at several bit widths.

2. Related Work

2.1. Quantization

As pointed out in [15] by Gray and Neuhoff, quantization as a compression method transforming continuous values to discreet ones, has a long history. Rounding and truncating are the most common examples. As discussed in [14], quantization methods are classified as either data-driven [17, 20, 23, 32, 8, 16] or data-free [2, 3, 5, 12, 13, 37, 27, 6]. Data-driven methods have been shown to work remarkably well despite a coarse approximation of the continuous optimisation problem. However, the cost of retraining the model has limited the use of such solutions in inference engine and in general for Machine Learning as a Service (MLaaS) [28]. Furthermore, with the rise of privacy concerns especially in health services [35], data-free methods are becoming of most importance. However such methods often come at the cost of a lower accuracy.

2.2. Data-Free Quantization

The importance of data-free quantization is discussed in more details, in the most recent survey on the matter [14]. Most data-free methods focus on mitigating the accuracy drop resulting from the quantization process. For instance, DFQ [27] proposes to balance-out weight distribution across layers in order to reduce the bias induced by quantization. They also propose the first static input quantization method based on learned statistics stored in batch-normalization layers parameters. In SQuant [6], authors propose to further improve weight quantization by changing the implicit objective function. More formally, rounding scalar weights minimizes the mean squared error between the scalar quantized weights and the original weights. SQuant minimizes the absolute sum of errors between tensor instead of scalar values. Similarly most data-free methods [2, 3, 5, 12, 13, 37, 27] don’t tackle the specificities of activation quantization.

2.3. Input Quantization

In this work, we put the emphasis on the importance of input quantization, especially for the already compact architectures such as MobileNet [29] and EfficientNet [33] as previously shown in Fig 1. The standard method for input quantization, introduced in [27] and latter used in [6, 36], consists in statically estimate coarse variation ranges for each layer. The parameters for this static method are computed once and for all during quantization then fixed during inference: this provides the best inference speed but for at the cost of a lower accuracy due to coarse modelization of the input ranges. Other work, such as [2], apply the same per-channel quantization to both weights and activations. Unfortunately, this leads to unpractical inference. Consequently, they require a floating point scaling factor for each combination of weights and inputs which leads to as many floating point multiplications as quantized multiplications and defeats the purpose of quantization. This is why such approaches are not considered here nor in practice. Still, in this work, we argue that a significant fraction of the accuracy loss comes from input (and activation) quantization. Following these observations, we propose SPIQ, a method that reaches the accuracy level of the dynamic method by implementing a per-channel input quantization scheme, that don't introduce floating operations in the quantized model.

3. Methodology

Let $F : \mathcal{D} \mapsto \mathbb{R}^{n_o}$ be a feed forward neural network defined over a domain $\mathcal{D} \subset \mathbb{R}^{n_i}$ and output space $\mathbb{R}^{n_o}$. The operation performed by a layer f_l, for $l \in \{1, \dots, L\}$, is defined by the corresponding weight tensor $W_l \in \mathcal{A}^{n_{l-1} \times n_l}$ where $\mathcal{A}$ is simply $\mathbb{R}$ in the case of fully-connected layers and $\mathbb{R}^{k \times k}$ in the case of a $k \times k$ convolutional layer. We note I_l the input of a fully-connected layer f_l. Let's consider a quantization operator $Q : \mathbb{R} \rightarrow [-\beta; \beta] \cap \mathbb{N}$ which maps real values to a bounded set of integer values where $\beta = 2^{b-1} - 1$ and b defines the bit-width of the target representation. The standard quantization operator is defined as $Q : x \mapsto \lfloor x/s_x \rceil$ where $\lfloor \cdot \rceil$ is the rounding operation and s_x is a scaling factor. Then, the quantized layer f_l^q is defined as

$$f_l^q : I_l \mapsto Q^{-1}\left(Q(I_l) \times Q(W_l)\right) = s_{I_l} \odot s_{W_l} \odot \left(\left\lfloor \frac{I_l}{s_{I_l}} \right\rceil \times \left\lfloor \frac{W_l}{s_{W_l}} \right\rceil \right) \tag{1}$$

where $\odot$ is the element-wise product. The values of s_{I_l} and s_{W_l} depend on the information available on I_l and W_l respectively. In the case of weight tensor W_l during the quantization process, all the information is available. Consequently, the value of s_{W_l} is derived from W_l in order to scale the scalar weight values distribution to $[-\beta; \beta]$. There are two quantization options. First, per output-channel weight quantization, in this case $s_{W_l} \in \mathbb{R}_+^{n_l}$ is a n^l-dimensional vector and each output channel (or neuron) is scaled independently. Second, per-layer (or per-tensor) quantization, where $s_{W_l} \in \mathbb{R}_+$ is a scalar value that scales the whole weight tensor W_l. Formally, if the note W_l^{channel} the per-channel quantized tensor and W_l^{layer} the per-layer quantized tensor,

$$\begin{cases} W_l^{\text{channel}} & = \left\lfloor (2^{b-1} - 1) \left(\frac{W_l^n}{\max_{w \in W_l^n}\{|w|\}} \right)_{n \in \{1,\dots,n_l\}} \right\rceil \\ W_l^{\text{layer}} & = \left\lfloor (2^{b-1} - 1) \frac{W_l}{\max_{w \in W_l}\{|w|\}} \right\rceil \end{cases} \tag{2}$$

where W_l^n is the n^{th} column of W_l corresponding to the n^{th} neuron of layer f_l.

3.1. Static and Dynamic Input Quantization

The definition of s_{I_l} from equation 1 induces a dimensionality constraint. We need to apply s_{I_l} to both I_l (which has n_{l-1} channels) and $\left\lfloor \frac{I_l}{s_{I_l}} \right\rceil \times \left\lfloor \frac{W_l}{s_{W_l}} \right\rceil$, *i.e.* if s_{I_l} is a vector then s_{I_l} needs to be of dimension n_{l-1} and n_l in order to be applied to I_l and $\left\lfloor \frac{I_l}{s_{I_l}} \right\rceil \times \left\lfloor \frac{W_l}{s_{W_l}} \right\rceil$ respectively. Therefore, s_{I_l} has to be a single, scalar value that scales the whole input tensor.

Similarly to the weight scaling factor s_{W_l}, the input scale s_{I_l} is computed based on the support of the distribution to scale. However, in the case of data-free quantization, we don't have access to the statistical properties of the input domain $\mathcal{D}$ of F. In order to circumvent this limitation we can apply either a static or dynamic activation quantization scheme.

Static Input Quantization: The goal is to compute $s_{I_l}^{\text{static}} \in \mathbb{R}$ based on an estimation of the maximum of I_l over the domain $\mathcal{D}$. Assume a BN layer precedes f_l, we can assert that

$$\mathbb{E}[I_l]^n = \beta^n \quad \text{and} \quad \mathbb{V}[I_l]^n = \gamma^n \tag{3}$$

where $\beta \in \mathbb{R}^{n_{l-1}}$ and $\gamma \in \mathbb{R}^{n_{l-1}}$ are the centering and scaling vector parameters of the BN layer respectively. Consequently, the maximum value of I_l over the domain $\mathcal{D}$, can be derived by searching for the maximum over the output channels and we get,

$$\frac{\max_{i \in I_l \text{ from } \mathcal{D}}\{|i|\}}{2^{b-1}-1} \approx \frac{\max_n\{\beta^n + \lambda \times \sqrt{\gamma^n}\}}{2^{b-1}-1} = s_{I_l}^{\text{static}} \in \mathbb{R} \tag{4}$$

where λ is a sensitivity parameter. This quantization method requires no additional computations at inference but only introduces a very coarse, per-layer scaling factor $s_{I_l}^{\text{static}}$.

Dynamic Input Quantization: The goal is to compute $s_{I_l}^{\text{dynamic}} \in \mathbb{R}$ based on the inferred input I_l at the cost of overhead computations at inference. Consequently

$$\frac{\max_{i \in I_l}\{|i|\}}{2^{b-1} - 1} = s_{I_l}^{\text{dynamic}} \in \mathbb{R} \tag{5}$$

The computation of $\max i \in I_l\{|i|\}$ is performed at each inference which adds a significant computational overhead (see section 4.3). However, the scaling factor $\max i \in I_l\{|i|\}$ is necessarily tighter than in the static case, hence a lower quantization error. Nevertheless, we argue that it is possible to design a tighter static input quantization scheme thanks to per-channel rescaling.

3.2. Per-Channel Static Input Quantization

We define the scaling vector $s_{I_l}^{\text{channel}} \in \mathbb{R}^{n_{l-1}}$ using the BN layers. Formally,

$$\frac{\max_{i \in I_l^n \text{ from } \mathcal{D}}\{|i|\}}{2^{b-1} - 1} \approx \frac{\beta^n + \lambda \times \sqrt{\gamma^n}}{2^{b-1} - 1} = \left(s_{I_l}^{\text{channel}}\right)^n \tag{6}$$

with $s_{I_l}^{\text{channel}} \in \mathbb{R}^{n_{l-1}}$. However, we are no longer able to perform the de-quantization as described in equation 1 because of dimensionality issues. Formally, the scaling vector $s_{I_l}^{\text{channel}}$ can be applied to I_l but not to the activation

$\left\lfloor \frac{I_l}{s_{I_l}} \right\rceil \times \left\lfloor \frac{W_l}{s_{W_l}} \right\rceil$. To tackle this limitation, we propose to decompose the quantization in two steps. First, we update W_l such that it applies both the inverse of the rescaling s_{I_l} to the inputs I_l and the operation originally defined by W_l. Then we note,

$$W_l^{\text{channel}} = \text{diag}(s_{I_l}^{\text{channel}}) \times W_l \tag{7}$$

where diag is the transformation of a vector in a diagonal matrix. Second, we scale the new value W_l^{channel} as a single weight tensor. Consequently, equation 1 becomes:

$$f_l^q : I_l \mapsto s_{W_l^{\text{channel}}} \odot \left(\left\lfloor \frac{I_l}{s_{I_l}^{\text{channel}}} \right\rceil \times \left\lfloor \frac{W_l^{\text{channel}}}{s_{W_l^{\text{channel}}}} \right\rceil \right) \tag{8}$$

In other words, the per-channel input ranges and scaling factor $s_{I_l}^{\text{channel}}$ are computed and folded within W_l (equation 7). This allows us to re-scale the input I_l prior to quantization only, thus circumventing the dimensionality constraint introduced in section 3.1. Moreover, this allows us to reduce the error as compared to the quantization as each output channel (or neuron) becomes:

$$(f_l^q(I_l))^n = \textstyle\sum_{m=1}^{n_{l-1}} s_W^n \left\lfloor \frac{I_l^m}{s_{I_l}^m} \right\rceil \times \left\lfloor (2^{b-1}-1) \frac{W_l^{n,m}/s_{I_l}^m}{\max_m\{|W_l^{n,m}|/s_{I_l}^m\}} \right\rceil \tag{9}$$

where $s_{I_l}^m$ is the m^{th} value of $s_{I_l}^{\text{channel}}$ and $W_l^{n,m}$ is value of coordinate n, m. Furthermore, we deduce from equation 6 that,

$$\left\| I_l - s_{I_l}^m \left\lfloor \frac{I_l^m}{s_{I_l}^m} \right\rceil \right\| \leq \left\| I_l - s_{I_l}^{\text{static}} \left\lfloor \frac{I_l^m}{s_{I_l}^{\text{static}}} \right\rceil \right\| \tag{10}$$

in other words, the quantization error on the input is lower with the per-channel method. However, this method also changes the weight quantization by folding the input scales in the weight tensor W_l. The difference between the static and per-channel static methods lies in the denominator $\max_m\{|W_l^{n,m}|/s_{I_l}^m\}$ from equation 9. By definition, we have $s_{I_l}^m \leq s_{I_l}^{\text{static}}$ and scalar values $W_l^{n,m}$ of W_l are likely to be cancelled if and only if both the $W_l^{n,m}$ and corresponding I_l^m have near zero ranges. We deduce that the proposed method results in a lower quantization error on average, *i.e.*

$$\begin{aligned} &\mathbb{E}_{I \in \mathcal{D}} \left[\left\| IW - s_{W s_I^{\text{channel}}} \left\lfloor \frac{I}{s_I^{\text{channel}}} \right\rceil \left\lfloor \frac{W s_I^{\text{channel}}}{s_{W s_I^{\text{channel}}}} \right\rceil \right\| \right] \\ &\leq \mathbb{E}_{I \in \mathcal{D}} \left[\left\| IW - s_I^{\text{static}} s_W \left\lfloor \frac{I}{s_I^{\text{static}}} \right\rceil \left\lfloor \frac{W}{s_W} \right\rceil \right\| \right] \end{aligned} \tag{11}$$

This provides an intuition on the superior performance of the proposed SPIQ quantization scheme over the reference static approach. In what follows, we show that SPIQ also empirically outperforms the dynamic approach, which in turn allows to significantly improve over current state-of-the-art methods.

4. Experiments

4.1. Datasets and Implementation Details

We validate the proposed method on three challenging computer vision tasks. First, on image classification, we consider ImageNet [9]. Second, on object detection, we conduct the experiments on Pascal VOC 2012 [11]. Third, on image segmentation, we use the CityScapes dataset [7].

In our experiments we tackle the challenging compression of MobileNets [29], ResNets [18], EfficientNets [33] and DenseNets [19] on ImageNet. For Pascal VOC object detection challenge we use an SSD [24] architecture. On CityScapes we use DeepLab V3+ [4].

ResNet, DenseNet, MobileNet and EfficientNet for ImageNet come from Tensorflow model zoo [1]. In object detection, we tested the SSD model with a MobileNet backbone from [25]. Finally, in image semantic segmentation, the DeepLab V3+ model came from [10]. The networks pre-trained weights provide standard baseline accuracies on each task. SPIQ and quantization baselines are implemented using Numpy. The results were obtained using an Intel Core i9-9900K CPU and RTX 3090 GPU.

We performed hyper parameter settings as well as comparisons using the standard quantization operator over weight values from [21] (same as as OCS [37] and SQNR [26]). For our comparison with state-of-the-art approaches in data-free quantization, we applied the more complex quantization operator from SQuant [6] using our own implementation which was carefully implemented so as to match the results for the original paper.

4.2. Hyper-Parameter Setting

The proposed method only requires one hyper-parameter λ which sets the number of standard deviations in the scaling value of the inputs, as defined in equation 6. In DFQ [27], authors recommend setting $\lambda = 6$ for the static input quantization, based on a Gaussian prior and the objective to keep over 99.99% of the input values not clipped. Intuitively, the value of λ determines the support of the expected input distribution. In other words, a large value λ induces almost no outliers but many small values will be quantized in a very coarse manner. On the other hand, a small value λ induces many input outliers that will be clipped but a fine-grained quantization of smaller inputs. We empirically validate the best value for λ and report our results in Fig 2. We observe that the bit-width (int4,...) has more importance than the neural network architecture on the value of λ: the smaller the representation the lower the optimal value for λ. This is a consequence of the fact that smaller bit-width can represent less values while still needing to finely quantize small input values. For the sake of simplicity, we use a common value of λ for all architectures and define $\lambda = b$, e.g. in int8 we use $\lambda = 8$ and in int4 we use $\lambda = 4$.

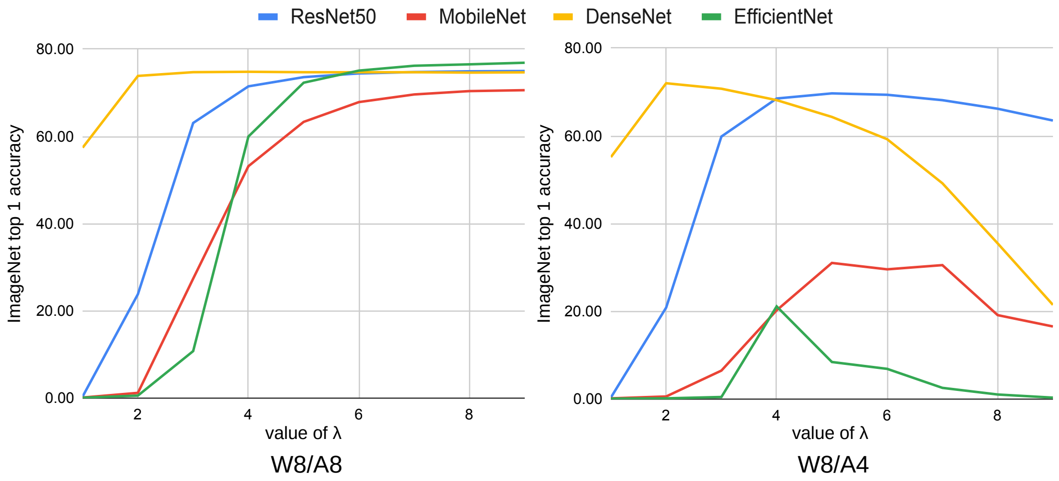

Figure 2. Influence of hyper-parameter λ on top1 accuracy for weights quantized in int8 using the naive per-channel quantization and inputs quantized either in int8 or int4 our protocol for input quantization, on ResNet 50, MobileNet V2, DenseNet 121 and EfficientNet B0 for classification on ImageNet.

Table 1. Comparison of the inference time on the ImageNet validation set for different architectures quantized with the static (same runtime as SPIQ) and the dynamic methods. We report the boost induced by using the proposed static method.

Method	ResNet	MobNet V2	DenseNet	EffNet B0
dynamic	79s	50s	93s	59s
SPIQ	63s	41s	77s	51s
boost	20.2%	18.0%	17.2%	13.6%

Table 2. Comparison between state-of-the-art, data-free, post training quantization techniques with ResNet 50 on ImageNet. We distinguish methods requiring data generation (No DG). In SPIQ the weight quantization method is SQuant.

	Method	No DG	W-bit	A-bit	Accuracy
ResNet 50	Baseline	-	32	32	76.15
	DFQ [27]	✓	8	8	75.45
	ZeroQ [3]	✗	8	8	75.89
	DSG [36]	✗	8	8	75.87
	GDFQ [34]	✗	8	8	75.71
	SQuant [6]	✓	8	8	76.04
	SPIQ + SQuant	✓	8	8	**76.15**
	DFQ [27]	✓	4	4	0.10
	ZeroQ [3]	✗	4	4	7.75
	DSG [36]	✗	4	4	23.10
	GDFQ [34]	✗	4	4	55.65
	SQuant [6]	✓	4	4	68.60
	SPIQ + SQuant	✓	4	4	**69.70**

4.3. Comparison with Static and Dynamic baselines

Fig 3 presents the comparison between the static, dynamic and SPIQ method in terms of accuracy with respect to the bit-width of the inputs and activations. Given weights quantized with [21] in int8, we observe the accuracy improvement offered by the dynamic approach over the static one. For instance, on DenseNet 121 in W8/A3 (int 8 weights and int3 activations), we observe an improvement of 15.38 points. This is due to the adaptive scaling to each input from the dynamic method. Nonetheless, the proposed per-channel manages to further improve the accuracy over the dynamic method. On the same example, SPIQ adds $46,39$ and $31,01$ points over the static and dynamic baselines respectively. The only architecture on which the dynamic and SPIQ methods achieve similar results is MobileNet V2 while on EfficientNet B0 quantized in W8/A6, SPIQ outperforms the dynamic approach by 30.35 points. These results are a consequence of a tighter quantization for each specific channel with SPIQ.

Furthermore, in terms of inference speed, as measured in Table 1, the SPIQ method systemically outperforms the dynamic approach. For instance, on MobileNet V2 the proposed method achieves a 18% faster inference. This corresponds to the cost of tuning the scaling parameters for each inputs during inference. Consequently, SPIQ offers the inference speed of the static approach and with an accuracy on par or greater than the dynamic method.

In the following section, we comapre the SPIQ performance to other data-free quantization algorithm.

4.4. Comparison with State-Of-The-Art

Table 2 lists the performances of several data-free quantization methods on different quantization configurations of ResNet 50 on ImageNet. We classify methods by their usage of data generation (DG). Such requirement is time consuming as compared to the proposed method which takes less than a second to quantize the model while several back-propagation passes take a few minutes and fine-tuning a few hours. Nonetheless, we demonstrate that the proposed input quantization allows us to achieve superior results than other

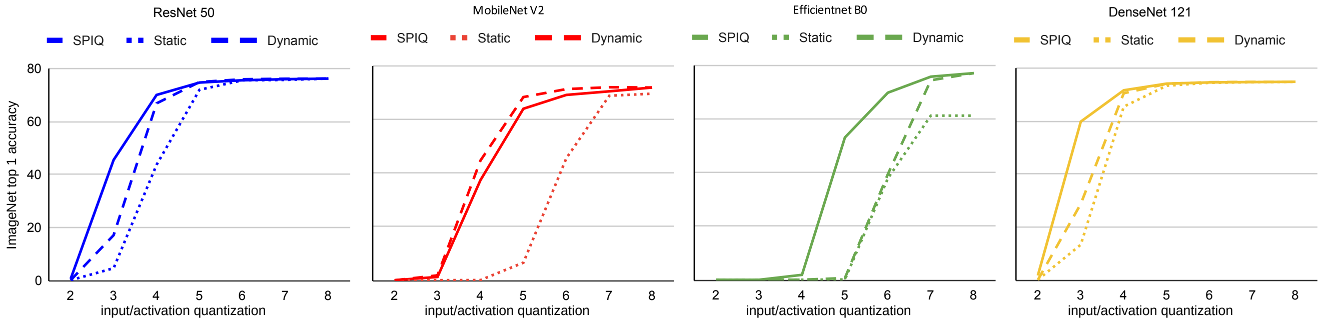

Figure 3. Comparison between SPIQ and static and dynamic inputs quantization. The weight quantization is fixed to 8 bits and we vary the input bit range from int2 (ternary quantization) to int8. We report the top1 accuracy on over ImageNet for ResNet 50, MobileNet V2, EfficientNet B0 and DenseNet 121.

Table 3. Comparison between state-of-the-art, data-free, post training quantization techniques with MobileNet V2, DenseNet 121 and EfficientNet B0 on ImageNet. We focused on data-free post training quantization methods that don't involve back-propagation. In SPIQ the weight quantization method is SQuant.

	Method	No BP	W-bit	A-bit	Accuracy
MobileNet V2	Baseline	-	32	32	71.80
	DFQ [27]	✓	8	8	70.92
	SQuant [6]	✓	8	8	71.68
	SPIQ + SQuant	✓	8	8	**71.79**
	DFQ [27]	✓	6	6	45.84
	SQuant [6]	✓	6	6	55.38
	SPIQ + SQuant	✓	6	6	**63.24**
DenseNet 121	Baseline	-	32	32	75.00
	DFQ [27]	✓	8	8	74.75
	OCS [37]	✓	8	8	74.10
	SQuant [6]	✓	8	8	74.70
	SPIQ + SQuant	✓	8	8	**75.00**
	DFQ [27]	✓	4	4	0.10
	OCS [37]	✓	4	4	0.10
	SQuant [6]	✓	4	4	47.14
	SPIQ + SQuant	✓	4	4	**51.83**
EfficientNet B0	Baseline	-	32	32	77.10
	DFQ [27]	✓	8	8	46.43
	SQuant [6]	✓	8	8	76.93
	SPIQ + SQuant	✓	8	8	**77.02**
	DFQ [27]	✓	6	6	20.29
	SQuant [6]	✓	6	6	54.51
	SPIQ + SQuant	✓	6	6	**74.67**

data-free quantization protocols by a large margin. Specifically, in int8 the accuracy almost reaches the full precision (float 32) accuracy while in int4, we reduce the accuracy drop by 14.56% as compared to SQuant alone and by 68.5% as compared to GDFQ [34]. This confirms that the application of the input scaling to the weights before quantization (equation 8 in section 3.2) does not harm the weight quantization even in low precision. Overall, the proposed method achieves remarkable accuracy in this benchmark.

Table 4. Performance (mIoU) on semantic segmentation on CityScapes dataset.

	method	W4/A4	W6/A6	W8/A8	-
DeepLab V3+	baseline	-	-	-	70.71
	DFQ + static	6.51	45.71	70.11	-
	DFQ + dynamic	7.51	66.65	70.22	-
	SQuant + static	7.69	66.77	70.21	-
	SQuant + dynamic	28.87	66.98	70.42	-
	SQuant + SPIQ	**36.14**	**68.69**	**70.66**	-

Table 5. Performance (mAP) on object detection on Pascal VOC 2012 dataset with SSD MobileNet.

	method	W4/A4	W6/A6	W8/A8	-
SSD MobileNet	baseline	-	-	-	68.56
	DFQ + static	3.94	53.52	67.91	-
	DFQ + dynamic	15.95	62.31	67.52	-
	SQuant + static	14.98	61.29	68.43	-
	SQuant + dynamic	35.47	66.72	**68.56**	-
	SQuant + SPIQ	**37.88**	**68.01**	**68.56**	-

To further validate the efficiency of SPIQ, in Table 3, we report results on DenseNet 121, EfficientNet and MobileNet V2. The considered architectures, especially MobileNet V2 and EfficientNet, are even more challenging than ResNet to quantize without accuracy drop even in relatively large representations such as int6. We only focused on the state-of-the-art approaches (without data generation) OCS [37], DFQ [27] and SQuant [6]. We observe the large benefits of a stronger input quantization method as SPIQ improves by 7.86% the accuracy of SQuant and 17.4% over DFQ on MobileNet V2 in int6. The results are even more impressive on EfficientNet B0 in int6, as SPIQ improves the accuracy by 20.16% of SQuant and 31.59% over DFQ. As compared to OCS, on DenseNet 121, the proposed method boosts the accuracy by 8.74%. Still, data-free quantization has room for improvement in int4 quantization, on already efficient architectures such as MobileNet V2 and EfficientNet B0. In the following section, we propose to generalize these remarkable results to other challenging tasks.

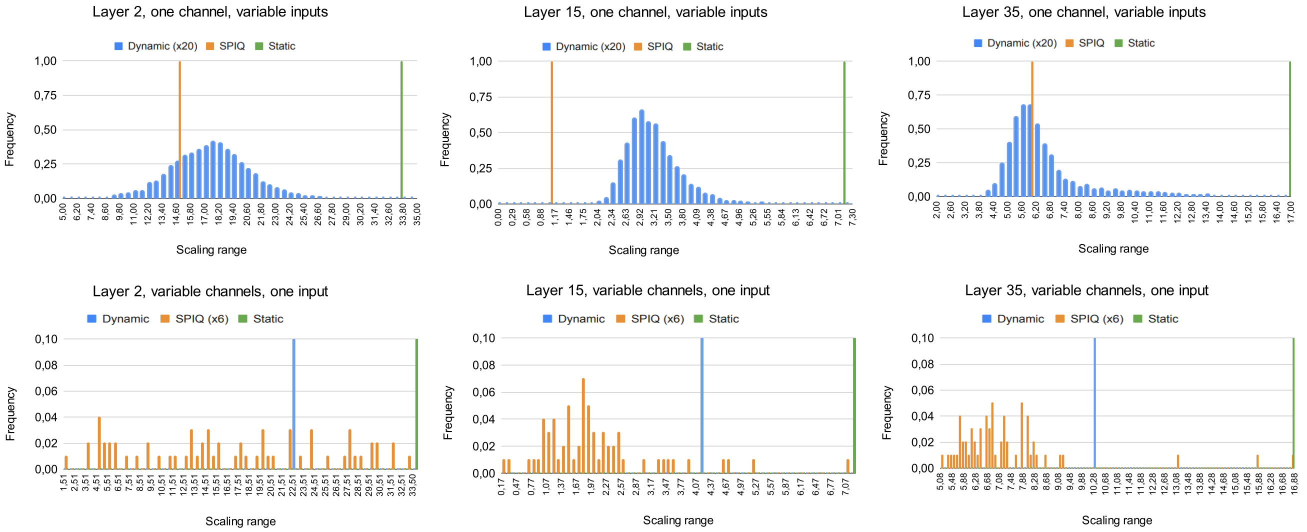

Figure 4. Distribution of the quantization ranges outputted by SPIQ, the dynamic and static baselines on the inputs of 3 different layers of a ResNet 50. The static baseline is constant, while dynamic and SPIQ vary depending on the input samples and channels respectively. The lower the computed ranges to 0 (the closer to the left of the each subplot), the better. SPIQ generally allows tighter adaptation to the original input distribution, as compared with both the static and dynamic methods.

4.5. Other Applications

Semantic Segmentation: In table 4, we report the performance of SPIQ method on image semantic segmentation task of CityScapes dataset. The dynamic approaches still provides more accuracy than the static baseline due to its adaptive scaling of each input regardless of the weight quantization process. Still, due to a finer quantization of the inputs for each channel, SPIQ manages to further improve the accuracy over the dynamic method reaching outstanding results such as 68.69 mIoU in W6A6. This confirms the two previous main results: first, SPIQ offers the highest accuracy while preserving the inference-time benefits of static input quantization. Second, when used in combination with a strong weight quantization protocol, SPIQ achieves state-of-the-art performances and significantly improve the accuracy in low-bit representation (int4). More precisely, we improve by 29.63% the mIoU of a DeepLab V3+.

Object Detection: In Table 5, we report the performance of SPIQ method on object detection of Pascal VOC 2012 dataset. Dynamic input quantization outperforms the static baseline in terms of accuracy at the expense of runtime. Nonetheless, SPIQ manages to further improving the mAP by 2.41 points. This is a consequence of the fine-grained quantization suited for each input channel of each layer of the network, in all bit-width configurations. These results confirm our two main results: SPIQ offers the highest mean average precision (mAP) in all quantization configurations as compared to static and dynamic methods, from int8 to low-bit int4. Furthermore, the SPIQ method achieves higher mAP than other state-of-the-art quantization schemes that focus only on improving weight quantization. These results conclude our empirical validation of SPIQ.

5. Discussion

Empirical Intuition: Fig 4 shows a comparison of sample scaling ranges calculated with the static and dynamic approaches as well as SPIQ. It stems from the definition of these methods that the closer the range is to 0 (on the left on Fig 4 subplots), the tighter the quantized inputs to the original inputs. Furthermore, while the static range is the same across all examples and channels, the dynamic method as well as SPIQ respectively vary upon those two factors. We observe that the static approach is not very tight to the input distribution in all cases. The dynamic approach, allows tighter adaptation in every scenario. Furthermore, depending on the input example (first row of Fig 4), SPIQ is generally tighter than the dynamic approach (most notably on e.g. layer 15 where the range computed with SPIQ is far lower, and to a lesser extent on e.g. layer 2). Furthermore, varying the input channels with one fixed example (second row of Fig 4) shows that the ranges computed with SPIQ are generally tighter than those computed with the dynamic approach. Fig 5 also illustrates how, on certain channels (e.g. channel 32 of layer 2), the dynamic approach struggles to leverage the full quantized range of values. By contrast, SPIQ qualitatively allows to better preserve feature map details, which in turn improves the accuracy. Hence, we argue that, if one had to chose between per-example and per-channel quantization, the latter would be more relevant. However, why wouldn't we do both?

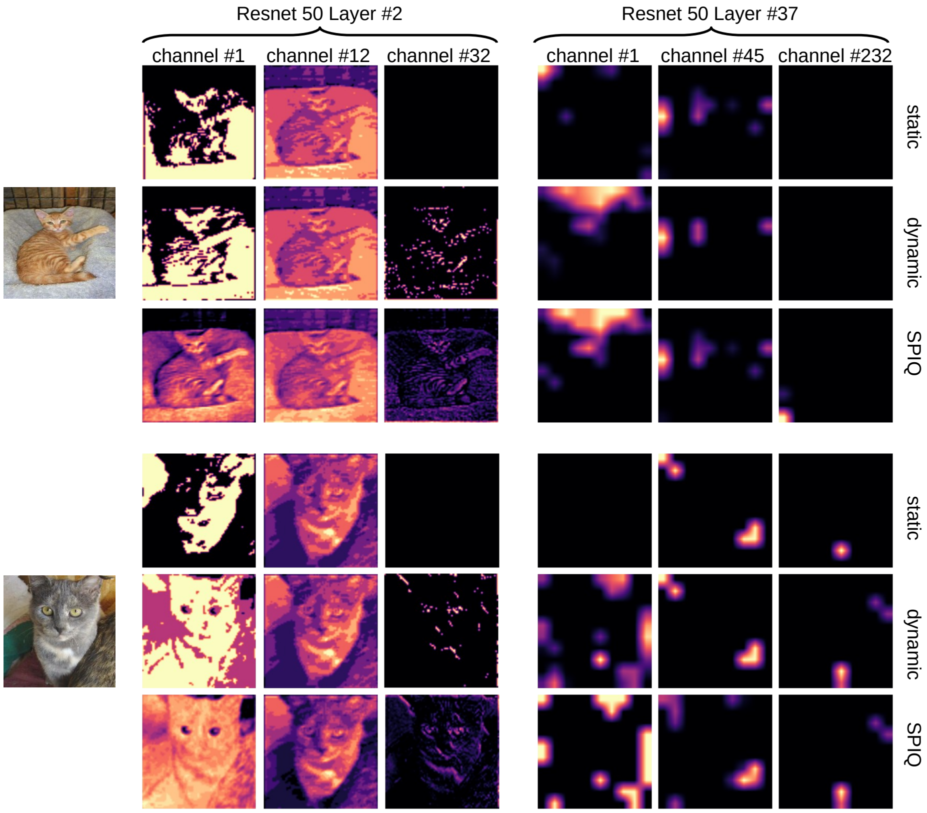

Figure 5. Illustration of different feature map channels of a quantized (static, dynamic and SPIQ) ResNet 50.

On the possibility to design a Per-Channel Dynamic Quantization: Per-channel dynamic quantization could mathematically be performed by simply combining equation 5 and equation 6. However, in practice this would require to perform weight quantization in addition to activations quantization at each inference step. This would be extremely time-consuming, especially when dealing with fully-connected layers that have larger weight tensors than input tensors. Furthermore, this would require to store weight values in full precision instead of low-bit precision which removes one of the benefits of quantization, *i.e.* memory foot-print reduction. Consequently, while per-channel dynamic quantization is theoretically feasible, in practice one has to choose between per-example and per-channel modelling as combining the two is highly impractical. We show that per-channel leads to better performance.

6. Conclusion

In this work, we highlighted a current limitation of post-training quantization methods, arguing that quantizing the inputs of each layer is of paramount importance to successful PTQ, that is often neglected in the literature. Furthermore, we showed that per-channel range estimation allows tighter modelling of the full-precision distribution e.g. as compared to per-example, dynamic approaches. Thus, we proposed SPIQ, a novel static input quantization approach which leverages per-channel quantization of the inputs in a data-free manner. We empirically showed that SPIQ achieved better speed vs. accuracy trade-offs than both the static and dynamic input methods, in addition to significantly improving existing state-of-the-art methods across a wide range of applications and neural network architectures without bells and whistles.

Limitations and Future Work: Very low-bit representation remains an extremely challenging task for data-free acceleration. In cases such as binary or ternary quantization, the proposed method would greatly benefit from fine-tuning. Generated data, obtained with similar methods as [36, 34], may provide better insight on input distributions and improve scale estimation for input quantization.

Acknowledgments

This work has been supported by the french National Association for Research and Technology (ANRT), the company Datakalab (CIFRE convention C20/1396) and by the French National Agency (ANR) (FacIL, project ANR-17-CE33-0002). This work was granted access to the HPC resources of IDRIS under the allocation 2022-AD011013384 made by GENCI.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *NeurIPS*, pages 7950–7958, 2019.

[3] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *CVPR*, pages 13169–13178, 2020.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 801–818, 2018.

[5] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, pages 3009–3018, 2019.

[6] Guo Cong et al. Squant: On-the-fly data-free quantization via diagonal hessian approximation. *ICLR*, 2022.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016.

[8] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, pages 3123–3131, 2015.

[9] J. Deng, W. Dong, et al. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[10] Zakirov Emil. Mobilenet-ssd-keras. https://github.com/bonlime/keras-deeplab-v3-plus, 2018.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012.

[12] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph H Hassoun. Post-training piecewise linear quantization for deep neural networks. In *ECCV*, pages 69–86. Springer, 2020.

[13] Sahaj Garg, Anirudh Jain, Joe Lou, and Mitchell Nahmias. Confounding tradeoffs for neural network quantization. *arXiv preprint arXiv:2102.06366*, 2021.

[14] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[15] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383, 1998.

[16] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi. Hardware-oriented approximation of convolutional neural networks. *ICLR workshop*, 2016.

[17] Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 29(11):5784–5789, 2018.

[18] Kaiming He, Xiangyu Zhang, et al. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[20] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *NeurIPS*, 29, 2016.

[21] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[22] Meng Li, F Richard Yu, Pengbo Si, and Yanhua Zhang. Green machine-to-machine communications with mobile edge computing and wireless network virtualization. *IEEE Communications Magazine*, 56(5):148–154, 2018.

[23] Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009*, 2015.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016.

[25] Soni Manish. Mobilenet-ssd-keras. https://github.com/ManishSoni1908/Mobilenet-ssd-keras, 2019.

[26] Eldad Meller, Alexander Finkelstein, Uri Almog, and Mark Grobman. Same, same but different: Recovering neural network quantization error through weight factorization. In *ICML*, pages 4486–4495, 2019.

[27] Markus Nagel, Mart van Baalen, et al. Data-free quantization through weight equalization and bias correction. In *ICCV*, pages 1325–1334, 2019.

[28] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *ICMLA*, pages 896–902. IEEE, 2015.

[29] Mark Sandler, Andrew Howard, et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[30] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[31] Ximeng Sun, Rameswar Panda, Chun-Fu Richard Chen, Aude Oliva, Rogerio Feris, and Kate Saenko. Dynamic network quantization for efficient video inference. In *ICCV*, pages 7375–7385, 2021.

[32] Shyam A Tailor, Javier Fernandez-Marques, and Nicholas D Lane. Degree-quant: Quantization-aware training for graph neural networks. *ICLR*, 2021.

[33] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, pages 6105–6114, 2019.

[34] Shoukai Xu, Haokun Li, Bohan Zhuang, Jing Liu, Jiezhang Cao, Chuangrun Liang, and Mingkui Tan. Generative low-

bitwidth data free quantization. In *ECCV*, pages 1–17. Springer, 2020.

[35] Xing Zhang et al. Health information privacy concerns, antecedents, and information disclosure intention in online health communities. *Information & Management*, 55(4):482–493, 2018.

[36] Xiangguo Zhang, Haotong Qin, Yifu Ding, Ruihao Gong, Qinghua Yan, Renshuai Tao, Yuhang Li, Fengwei Yu, and Xianglong Liu. Diversifying sample generation for accurate data-free quantization. In *CVPR*, pages 15658–15667, 2021.

[37] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *ICML*, pages 7543–7552, 2019.