QPP: Real-Time Quantization Parameter Prediction for Deep Neural Networks

Vladimir Kryzhanovskiy[1,*], Gleb Balitskiy[2,*], Nikolay Kozyrskiy[2,*], Aleksandr Zuruev[3]
[1]Huawei Noah's Ark Lab, [2]Skolkovo Institute of Science and Technology,
[3]Novosibirsk State University
kryzhanovskiy.vladimir@huawei.com, {gleb.balitskiy,nikolay.kozyrskiy}@skoltech.ru,
alexander.zuruev@yandex.ru

Abstract

Modern deep neural networks (DNNs) cannot be effectively used in mobile and embedded devices due to strict requirements for computational complexity, memory, and power consumption. The quantization of weights and feature maps (activations) is a popular approach to solve this problem. Training-aware quantization often shows excellent results but requires a full dataset, which is not always available. Post-training quantization methods, in turn, are applied without fine-tuning but still work well for many classes of tasks like classification, segmentation, and so on. However, they either imply a big overhead for quantization parameters (QPs) calculation at runtime (dynamic methods) or lead to an accuracy drop if pre-computed static QPs are used (static methods). Moreover, most inference frameworks don't support dynamic quantization. Thus we propose a novel quantization approach called QPP: quantization parameter prediction. With a small subset of a training dataset or unlabeled data from the same domain, we find the predictor that can accurately estimate QPs of activations given only the NN's input data. Such a predictor allows us to avoid complex calculation of precise values of QPs while maintaining the quality of the model. To illustrate our method's efficiency, we added QPP into two dynamic approaches: 1) Dense+Sparse quantization, where the predetermined percentage of activations are not quantized, 2) standard quantization with equal quantization steps. We provide experiments on a wide set of tasks including super-resolution, facial landmark, segmentation, and classification.

1. Introduction

Deep Neural Networks (DNNs) have been actively developing in recent years. They have already proven to be powerful tools for solving a wide variety of complex tasks such as classification, segmentation, image enhancement, etc. But modern architectures are usually quite bulky and do not meet the industry requirements for energy and memory efficiency. Therefore, additional methods have to be used to compress a neural network in order to decrease the model's inference time while preserving its quality. One of such approaches is the quantization of weights and feature maps (activations). The efficiency of quantization is based on the redundant representation of parameters in a single-precision floating-point format. The values in low-precision (e.g., 4, 8 bit-width) format require less memory, and fixed-point arithmetic operations take less time with hardware support than floating-point ones.

To date, many different quantization approaches have been proposed, so we give a classification of these methods and highlight their scope below. We also indicate the place of our work in this classification.

The following systematization is built upon the two basic criteria: the availability of a training dataset and the inference time.

1. **Quatization-Aware Training.** These methods are characterized by the presence of the fine-tuning procedure, which requires a labeled training dataset. The quantization parameters are fixed after the quantized neural network converges. Usually, such methods have better performance compared to other schemes. For example, the method described in [25] aims to train CNNs with low bit-width weights and activations using low bit-width parameter gradients. In [5] authors propose to find the proper activation clipping parameter during training, which helps to find to the optimal quantization scale. The further improved quantization-aware scheme based on the trainable quantization step size is introduced in the paper [8]. Esser et al. describe how they adjust the task loss gradients in order to learn the quantization step size and the weights of NN simultaneously. In [3] authors added the offset, a new trainable parameter, in addition to trainable quantization step size, and achieved good results compared

*Equally-contributed first authors

to other methods. However, the main disadvantages of quantization-aware training methods are privacy and copyright considerations in the industry. It means that training datasets may be either unavailable or partially available, so quantization-aware training schemes cannot be used in such cases.

2. **Post-Training Dynamic Quantization (Dynamic).** In such methods the quantization is applied to a well-trained NN's weights and its activations without further fine-tuning procedure. The quantization parameters (QP) like scale factors are computed at runtime using exact values of feature maps. The main advantages of this approach are that it doesn't require a training dataset and is stable to changes in data between samples. So it is easy to convert any full-precision model to a quantized one and get acceleration. Such approach is not used so often because it imposes a computational overhead on QPs estimation. It also restricts the ability to fuse quantized convolution and followed quantization in order to produce quantized output data directly avoiding additional scanning of tensor of activations. Even the computation of min and max values (used for the scale factors estimation) of an input feature maps is a developer's headache (2-3% of the complexity of 448x448x64@64 convolution), so using more complex statistics is impossible blue(for example, the overhead for α-quantile for above convolution equals to 10%). The method ACIQ, described in [1], is an example of a post-training dynamic method. Banner et al. proposed a method for bias correction and methods for on-line estimation of the optimal clipping value and per-channel bit allocation with prior assumptions on the input distribution. The combination of these methods showed good results for the 4-bit quantization of weights and activations.

3. **Post-Training Static Quantization (Static).** In contrast to dynamic quantization, static methods estimate QPs using a calibration dataset, e.g., a subset of a training dataset and use predefined QPs for data quantization during the inference stage without an opportunity to adjust them to particular samples. These methods are computationally more efficient because, firstly, there is no overhead on QPs estimation during the inference stage and, secondly, quantization can be fused with a convolutional layer. But static methods have typically higher quality degradation compared to dynamic ones and therefore are not always applicable. This happens when the standard deviation of feature maps varies significantly among samples in the task domain, and it is impossible to choose static quantization bounds to satisfy all data, i.e., they will be either too large (data doesn't cover all quantization range) or too small (data does not fit the boundaries and saturates). In [14], Migacz proposed the quantization scheme, which finds a tradeoff between the saturation and internal quantization errors mentioned above. The idea of this method is based on computing the optimal clipping values for each layer off-line by minimizing the Kullback-Leibler divergence with the help of a time-consuming iterative procedure. In [12] the author proposes a simple post-training quantization scheme based on the minimum and maximum values for weights and a moving average of the minimum and maximum values across batches for activations. Another works, [16] and [17], develop methods based on combining floating-point arithmetic and fixed-point one. The main idea is to apply the low-bit quantization to the majority of activations of small values while keeping the rest (up to 5% of data) in the full-precision format. This approach is compelling because, on the one hand, it saves precise high values, which have a strong influence on the NN output, and on the other hand, the quantization range becomes significantly smaller (due to feature maps distribution usually has long tails). In our work we call such an approach as Dense + Sparse (D+S) quantization and will focus on it in Section 5. Looking ahead, QPs for the D+S approach should be estimated carefully to get a certain amount of data in the sparse part. Otherwise, an accuracy drop or an increase of inference time can happen. That is why the static QPs for D+S is not a good solution.

We propose a new post-training quantization approach that eliminates the computational overhead of dynamic quantization and simultaneously allows one to maintain stability to changes between samples. Contributions of our work can be summarized more precisely as follows:

1. The detailed description of the novel method for the fast estimation of QPs before the inference stage for every input sample called Quantization Parameter Predictor (QPP, see Section 3). The small set of samples from the same domain as the supposed input data of the neural network is used to build an estimator of QPs. Before the inference stage one applies this estimator on the new input data and then run the inference using obtained estimations of QPs to quantize activations. Thereby this approach combines advantages of static (efficient inference) and dynamic (adjusting to a sample) quantization schemes. We also describe the examples of proposed QPP.

2. Theoretical analysis with simple assumptions to clarify the motivation and possibility of building such an estimator described above (see Section 4). We also provide a toy example on the MNIST dataset to confirm

the validity of this analysis and illustrate the correlation between NN's input and corresponding QPs.

3. We describe the implementation of QPP for the following dynamic methods: 1) standard dynamic quantization with equal quantization steps (see supplementary materials); 2) Dense+Sparse approach, where the predetermined percentage of activations are not quantized (see Section 5). We compare QPP with the static approach and underline the advantages of the first one on the wide set of tasks, including super-resolution, facial landmark, segmentation, and classification.

The 8-bit (both for weights and activations) post-training quantization works agreeably for the selected tasks. To highlight the advantages of the proposed method we decrease the activations bit-width to 4 bits. The current work doesn't cover the efficiency of 4-bit activations in comparison to 8-bit. The low-bits strengthen the disadvantages of the standard methods and therefore the improvements become easier to measure.

We deliberately don't use the pipeline of methods aimed to recover accuracy degradation after quantization like AdaRound [15], AdaQuant [11], batch norm tuning [11], bias correction [1] and so on. We use the QPP approach alone proving its efficiency.

We used QPP to predict the following QPs: 1) α-quantile for both standard and D+S quantization methods; 2) standard deviations used in the toy example in the section 4. We believe that QPP can predict more complex QPs obtained by some other dynamic methods. So, we can safely state that the suggested idea can be applied to any dynamic quantization method.

2. Static Quantization with Averaging

Until now, the only way to implement the dynamic method statically was to use the average value of the QPs calculated on the calibration dataset. More precisely, this approach can be described as follows. Denote $\mathbf{X}_m^l$ as the activations at l-th NN's layer of m-th input sample of the subset of the initial training dataset. Q_m^l as corresponding target QPs at the l-th layer computed dynamically, $l \in \overline{1, L}$, $m \in \overline{1, M}$. For example, Q_m^l is the maximum of absolute value of a feature maps for the symmetric quantization. Then, static QPs Q_{static}^l can be expressed as follows:

$$Q_{static}^l = \frac{1}{M}\sum_{i=1}^{M} Q_i^l \qquad (1)$$

Further, these QPs are used on the inference stage and never changed.

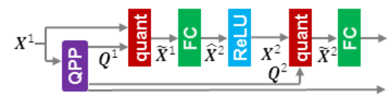

Figure 1. QPP Inference Scheme

3. QPP: Quantization Parameter Predictor

The static methods, assuming that the QPs are calculated on the subset of the training dataset in advance and then are fixed, are more computationally efficient than dynamic ones. Still, in general, they are more inaccurate. Instead of averaging the suboptimal QPs found for each sample from the training subset, one can build an estimator that evaluates them using the information from the NN input data. Our proposed method has the following steps:

1. **Estimator Construction.** Denote $\mathbf{X}_m^1$ as the m-th input sample ("1" denotes that it is the input of the first layer, i.e. the NN input) of the subset of the initial training dataset, $Q_m^l = f^l(\mathbf{X}_m^1)$ as corresponding target QPs at the l-th layer, $l \in \overline{1, L}$, $m \in \overline{1, M}$. Also let $\mathcal{F}$ be the class of considered estimators. Given the subset $\{\mathbf{X}_m^1\}$, the task of the estimator $\hat{f}^l \in \mathcal{F}$ is to minimize the MSE between optimal QPs and obtained estimations of QPs. The precise optimization problem is formalized as follows:

$$\hat{f}^l = arg \min_{f^l \in \mathcal{F}} \sum_{m=1}^{M} \frac{1}{M} \|f^l(\mathbf{X}_m^1) - Q_m^l\|_2, \forall l \in \overline{1, L}. \qquad (2)$$

   After the estimator is found, one can proceed to the inference stage.

2. **Inference Stage.** Assume $\mathbf{X}^l$ is the tensor of activations at l-th layer and $quant(\cdot, \cdot)$ is a quatization function at l-th layer. At first one obtains estimations of QPs $\{Q_l = f^l(\mathbf{X}^1)\}_{l=1}^L$ for all quantized layers. Then the quantization function with obtained estimations at each quantized layer $\tilde{\mathbf{X}}^l = quant(\mathbf{X}^{l+1}, Q^l)$ is applied during the forward-pass. The schematic representation of the inference scheme is given in Figure 1.

QPP is applicable only if there is a dependency between the input sample and the QPs. In the next section we present the proof of the existence of such dependencies and in Section 8 confirm it experimentally for all considered tasks.

4. Theoretical Analysis

In this section we demonstrate the motivation for using QPP on a toy example. First, we show a linear relationship between the activations variance of the deep feedforward neural network on different layers with simple assumptions. Next, we will consider the NN consisting of several linear

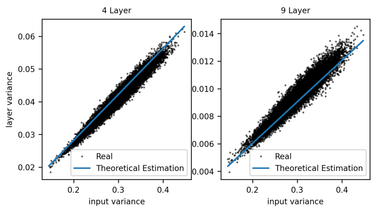

Figure 2. Variance dependence (untrained NN)

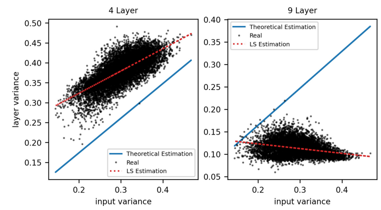

Figure 3. Variance dependence (trained NN)

layers and ReLU activation functions located between them (see Figure 1). We demonstrate the validity of the obtained result in the assumptions used on the denoising task for the MNIST dataset [13]. Then we show that for the trained NN the linear dependence remains but becomes worse. Let $\mathbf{X}^l \in \mathbb{R}^{N_l}$ be the activations of the l-th linear layer (w/o bias), $\mathbf{W}^l \in \mathbb{R}^{N_{l+1} \times N_l}$ be its weights and $\hat{\mathbf{X}}^{l+1} \in \mathbb{R}^{N_{l+1}}$ be the output of this layer. In this section we assume that the weights are sampled from gaussian distribution by the following way: $\mathbf{W}^l \sim \mathcal{N}(0, \sigma^2_{w_l})$. The activations $\mathbf{X}^l$ are sampled from arbitrary distribution with the following expectation $E[\mathbf{X}^l] = \mu_{x_l}$ and variance $D[\mathbf{X}^l] = \sigma^2_{x_l}$. Also, we assume independence between elements of $\mathbf{W}^l$ and elements of $\mathbf{X}^l$. Now, consider linear transformation:

$$\hat{\mathbf{X}}^{l+1} = \mathbf{W}^l \mathbf{X}^l. \tag{3}$$

Then we assume that the i-th item of $\hat{\mathbf{X}}^{l+1}$ has the gaussian distribution. This assumption is justified by the fact that layers of NNs have a sufficiently large dimension and the law of large numbers is fulfilled. With the following expectation and variance (taking into account that $\mu_{w_l} \equiv 0$):

$$E[\hat{x}_i^{l+1}] = E\left[\sum_{j=1}^{N_l} x_j^l w_{i,j}^l\right] = N_l \mu_{x_l} \mu_{w_l} = 0; \tag{4}$$

$$D[\hat{x}_i^{l+1}] = N_l \sigma^2_{w_l} (\sigma^2_{x_l} + \mu^2_{x_l}). \tag{5}$$

Here we used an assumption of independence of random variables. To finalize the analysis one needs to calculate the expectation and the variance of the output of ReLU layer:

$$\mathbf{X}^{l+1} = ReLU(\hat{\mathbf{X}}^{l+1}), \tag{6}$$

Then probability density function can be expressed as follows:

$$\rho_{\mathbf{X}^{l+1}}(x) = \begin{cases} 0 & \text{if } x < 0, \\ \frac{1}{2} & \text{if } x = 0, \\ \frac{1}{\sqrt{2\pi D[\hat{x}_i^{l+1}]}} e^{\frac{-x^2}{2D[\hat{x}_i^{l+1}]}} & \text{otherwise.} \end{cases} \tag{7}$$

Further, $\mu_{x_{l+1}}$ and $\sigma^2_{x_{l+1}}$ calculating is straightforward, so we give only the exact result:

$$\mu^2_{x_{l+1}} = \frac{1}{2\pi} D[\hat{x}_i^{l+1}] = \frac{1}{2\pi} N_l \sigma^2_{w_l} (\sigma^2_{x_l} + \mu^2_{x_l}); \tag{8}$$

$$\sigma^2_{x_{l+1}} = \frac{1}{2}(1 - \frac{1}{\pi}) N_l \sigma^2_{w_l} (\sigma^2_{x_l} + \mu^2_{x_l}). \tag{9}$$

The linear dependence between variance of activations of l-th layer and variance of NN's input can be expressed as follows:

$$\sigma^2_{x_{l+1}} = \frac{1}{2^l}(1 - \frac{1}{\pi}) N_1 \dots N_l \sigma^2_{w_1} \dots \sigma^2_{w_l} (\mu^2_{x_1} + \sigma^2_{x_1}). \tag{10}$$

To verify this result we consider the NN consisting of $L = 10$ linear layers with the following sizes $N^l = 3088 - 256(l-1)$ of hidden layers. We also used xavier initialization [9] for weights. As input of NN we consider MNIST dataset. In Figures 2, 3 one can see the dependence of the variance at 4-th and 9-th layers on the variance of the input sample for the trained and untrained NNs. Expression 10 is very accurate for an untrained network because the assumptions of the independence of weights and activations and the zero expectation of weights are met.

For the trained network the assumptions are violated, and the dependence becomes weaker and can no longer be described by 10 (Figure 3, blue line). We estimated variance with the least-squares method (red dotted line) to take into account the internal correlations. As one can see, there is a strong linear dependence at the shallow layers. However, we would like to note that the linear dependence vanishes at the deep layers. This problem can be solved by applying additional estimators at the intermediate layers of NNs. We will describe this approach in Section 7 more precisely.

The mathematical derivation and conclusions above can be repeated for CNN, but we will lose the simplicity of the toy example in this case.

Section conclusions. The formula 10 proves the existence of a linear dependency of the variance of an internal activation on the input statistics. Of course, the formula cannot be used in practice directly due to the correlations. Thus we train QPP to take into account all internal dependencies of the model.

5. Dense+Sparse Quantization

5.1. Definition

As noted in Section 1, selecting too small quantization step leads to large changes in outlier values, which in turn leads to quality degradation. To cope with this effect the authors of [17] propose to compute a fixed number of outliers in full-precision (FP) format without applying quantization on them. For this case the quantization function can be described formally as follows:

$$quant_{DS}(x, Q^l) = \begin{cases} \lfloor \frac{x}{s^l} \rceil \cdot s^l & \text{if } x \in [a^l,\ b^l], \\ x & \text{otherwise;} \end{cases} \quad (11)$$

where $\lfloor \cdot \rceil$ denotes the round operation. We choose quantization bounds $Q^l = (a^l, b^l)$ as QPs at each l-th layer (a^l and b^l represent bottom and upper thresholds). We assume equal quantization steps s^l with bit-width k of quantized values defined as follows:

$$s^l = \frac{b^l - a^l}{2^k - 1}. \quad (12)$$

The amount of outliers computed in the precise mode is defined by the density. In the original formulation, a^l and b^l are chosen as dynamically computed α-quantiles to preserve fixed density. But in static implementation α-quantile can only be estimated. Thus we suppose a^l and b^l are thresholds, which have values close to the target quantile. In the next section, we will describe the problem associated with the inaccurate estimation of QPs a^l and b^l, which leads to a degradation in the model's quality or an increase of inference time.

5.2. Stability Problem

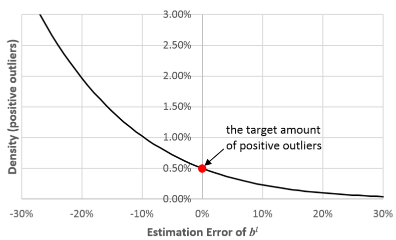

Figure 4. Density as a function of the estimation error b^l. Target density is 1%.

In the case of D+S quantization the problem of accurate choice a^l and b^l is critical. The Figure 4 shows the exponential dependence of the density on the estimation error (the plot was made under the assumption that the distribution of activations is Gaussian). Even rather small deviations from optimal values can cause either a drop in quality (if $b^l > b^l_{opt}$) or an increase in inference time (if $b^l < b^l_{opt}$).

Drop in quality. To illustrate this effect we show the relation between model performance and density and summarize the results for ResNet34 in Figure 5. One can see that the accuracy increases very fast with density growing up and reaching the full-precision model's accuracy at 10%. However, even 1% is enough to recover the accuracy loss after quantization. This can be explained as follows. Firstly, one reduces the quantization range Q^l. As a result,

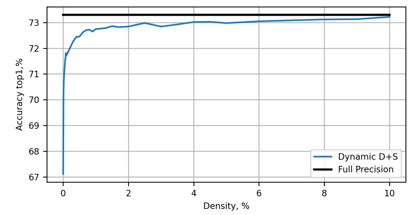

Figure 5. Accuracy vs Density (ResNet34. w:8, a:4)

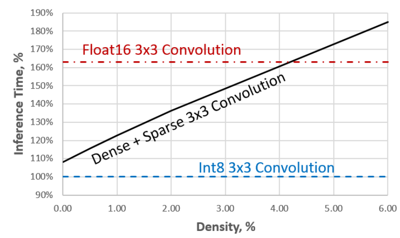

Figure 6. Normalized Inference Time vs Density (data presented in the Table 1).

the quantization step (12) shrinks and the quantization error decreases. Secondly, a small number of outliers, which play an important role for NN, are processed in FP32.

Increase in inference time. The D+S inference time linearly depends on the density (see Figure 6), which, in turn, has exponential dependence on QPs $Q^l = (a^l, b^l)$ represented as α - quantiles. For example, if an actual value of a^l or b^l is decreased by 30% by any estimation procedure, then the density can rise from desired 1% up to 6%. As a result, the inference time of D+S exceeds the full-precision models. This can lag or even hang the application, which is not acceptable for real-time applications.

Since this approach requires finding a fixed number of outliers (α-quantiles) for every tensor of activations, it can only be applied dynamically in the original formulation. But as noted, the dynamic searching of the α-quantiles value of activations imposes high computational costs (see the detailed discussion of the results presented in Table 1 in the Section 8). Most importantly, it restricts the opportunity to fuse convolution and quantization operations. Therefore, Park et al. use static clipping values (see Section 2) - the average value of the 0.97-quantile (3% density) on the training subset. However they can face the stability issue described above.

Thus we propose using QPP instead of static QPs to reduce these effects.

6. QPP based on Linear Regression

The QPs and the quantization function depend on the chosen basic dynamic method. To finish the method proposed above one needs to define the class of estimators $\mathcal{F}$ suitable for it. Since NN's input has typically large dimen-

Table 1. Inference time of different implementations of convolutions: float16 - full-precision convolution via BOLT[2](w32a32); Int8 - quantized convolution via BOLT (w8a8); D+S - the Dense (w8a8) + Sparse (w32a32) convolution (w - weights, a - activations). Input size (HxWxC): 56x56x64. Convolution size (KxKxF): 3x3x64. CPU: Kirin 980 (Cortex A76, 1 thread).

Conv Type	Density, %	Inference time, ms (%)					
		Max*/ Quantile**	Quantization	Sparese (float16)	Dense (int8)	D+S	Total
float16	-						3.23 (163%)
Dynamic Int8*	-	0.03 (2%)	0.19 (10%)	-	1.76 (89%)	-	1.98 (100%)
Dynamic D+S**	0	0.22 (11%)	0.23 (12%)	0 (0%)	1.7 (86%)	1.69 (86%)	2.14 (108%)
	0.5		0.25 (13%)	0.13 (6%)		1.82 (92%)	2.29 (116%)
	1.0		0.28 (14%)	0.24 (12%)		1.94 (98%)	2.43 (123%)
	2.0		0.32 (16%)	0.47 (24%)		2.17 (109%)	2.70 (136%)
	6.0		0.39 (20%)	1.36 (68%)		3.06 (155%)	3.67 (185%)

sions, the information has to be compressed to a small set of several variables. For Desnse+Sparse approach the QPs values are α-quantiles. Thus we consider linear regressors constructed in the space of α-quantiles as the base estimators. More precisely, given the input $\mathbf{X}$ of the NN, we define a corresponding feature space $\mathbf{S} \in \mathrm{R}^{M \times d}$, where $\mathbf{S}_{ij}$ defined as follows:

$$\mathbf{S}_{ij} = F(\mathbf{X}_i, \alpha_j), \tag{13}$$

and where $F(\mathbf{X}, \alpha_j)$ is an α_j-quantile of $\mathbf{X}$. Denote $\boldsymbol{\beta}^l$ as weight vectors of linear regressors for l-th layer for upper thresholds b^l (equations for the bottom thresholds a^l are omitted to save the space). Then the optimization task 2 can be formulated as follows:

$$\hat{\boldsymbol{\beta}}^l = arg \min_{\boldsymbol{\beta}^l} \frac{1}{M} \sum_{m=1}^{M} \left(\boldsymbol{\beta}^l \mathbf{S}_m - b_m^l\right)^2, \forall l \in \overline{1, L}. \tag{14}$$

The solution to this problem is well-known and can be expressed as follows:

$$\hat{\boldsymbol{\beta}}^l = (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{b}^l. \tag{15}$$

7. Prediction Quality and Extra QPP

We have introduced the optimization problem 2, where the goal of QPP is minimizing MSE between predictions and dynamically calculated QPs. In this section, we illustrate how much the QPP allows to reduce MSE compared to the static approach with averaging. Figure 7 demonstrates the MSE of predictions at each quantized layer of ResNet34. Set up of the experiment can be found in Section 8. For example, for a static approach (brown line) we can see that the first and 29-th layers have, on average, about 4% error of estimation of QPs, and the error standard deviation is huge - it is about 6%. With only one predictor (dotted line) predictions have significantly smaller MSE at the first half of NN compared to ordinary averaged static values (brown line). Furthermore, QPP is much more stable than a static scheme: predictors have a considerably smaller variance of their predictions. We have introduced a problem of vanishing correlation between feature space and QPs which can be clearly seen in Figure 7: QPP (dotted line) predicts the same values as the static method (brown line). A natural extension of the proposed design is to add QPP to the intermediate layers of the CNN. To maintain the low MSE in deep layers, we added another predictor at 17-th layer (blue line). One can see that this simple approach allows one to use the advantages of the predictor also at the deep layers of the CNN. But we would like to note that setting another QPPs increases the model's inference time since one needs to build a feature space several times.

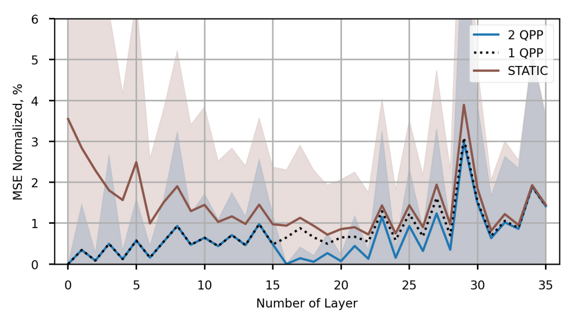

Figure 7. Comparison of 2 QPP, 1 QPP and Static approaches

8. Experiments

Let us clarify what we expect from the quantized model before embarking on analysis of the experimental results. The post-training quantization doesn't imply a fine-tuning, so it's only fair to expect that both pure static and static with QPP approaches can't overcome the dynamic quantization methods QPs. Here we expect that QPs of all these three methods are based on the same statistics, like maximum or

[2]https://github.com/huawei-noah/bolt

Table 2. Performance of QPP based D+S Quantization Scheme with 1% density (weights: 8 bit, activations: 4 bit)

4*Scheme	Classification Acc Top 1, %		Segmentation mIoU, %	Facial Landmark NME, %			Super-Resolution PSNR, dB		
	1 QPP (first ReLU)		4 QPP (first ReLU, Transition layer 1,2,3)				1 QPP (input of NN)		
	Imagenet		CityScapes	COFW	WFLW	300W	Vimeo	Set5	Set14
	ResNet18	ResNet34	HRNet	HRNet			ESPCN		
FP32	69.76	73.30	70.26	3.45	4.60	3.85	34.06	30.74	27.06
Full Dynamic	66.85	70.34	66.14	3.7	4.95	4.01	33.45	30.39	26.84
D+S Dynamic	69.04	72.72	69.84	3.56	4.70	3.95	34.01	30.69	27.02
D+S Static	69.08	72.60	69.99	3.52	4.68	3.92	34.0	30.70	27.04
D+S QPP	69.14	72.67	70.0	3.55	4.69	3.93	34.01	30.69	27.02

Table 3. Stability of D+S vs Static (weights: 8 bit, activations: 4 bit)

3*Scheme	Classification FR(1), % / FR(0.5), %		Segmentation FR(1), % / FR(0.5), %	Facial Landmark FR(1), % / FR(0.5), %			Super-Resolution FR(1), % / FR(0.5), %		
	Imagenet		CityScapes	COFW	WFLW	300W	Vimeo	Set5	Set14
	ResNet18	ResNet34	HRNet	HRNet			ESPCN		
D+S Static	3.4 / 16.1	4.5 / 23.9	0.7 / 13.9	3.5 / 64.2	8.5 / 93.4	2.1 / 34.2	26.5 / 59.5	60.0 / 60.0	62.9 / 80.0
D+S QPP	0.78 / 7.7	0.5 / 7.1	0.06 / 2.2	0.2 / 56.7	1.4 / 64.4	0.3 / 13.9	3.8 / 20.1	0 / 28.0	0 / 11.4

α-quantile. In an optimal case, they tend to dynamic behavior. So, if we note that in an experiment, then it means that QPP works perfectly.

The type of statistics is responsible for the proximity to the behavior of a full-precision model. Both the development of a new quantization method or finding the optimal statistic for QPs are out of the paper's scope. The purpose of the section is experimental proof that the QPP can predict different statistics, and therefore it can be used in various quantization methods.

We examined QPP's performance for 4 tasks: **classification** (ResNet18 and ResNet34 models [10] on ImageNet [7] dataset), **segmentation** (HRNetV2-W18 [3] model [20, 21] on CityScapes dataset [6]), **facial landmark** (HRNetV2-W18 model on COFW [4], WFLW [22], 300W [18] datasets), and **super-resolution** (ESPCN [19] on Vimeo [23], Set5 [2] and Set14 [24] datasets).

The pipeline for every experiment is the same and can be described as follows. In our experiments, we considered α-quantiles of activations as clipping values, i.e., QPs. Then the regressor was trained to predict them. We used 0.9-quantile, 0.99-quantile, 0.999-quantile, the mean, and maximum value of NN's input as the regressor's features.

In the presented experiments for weights quantization we had used 8-bit ($t = 8$) asymmetric quantization method:

$$\tilde{w}^l = \left\lfloor \frac{w^l}{s^l_w} \right\rceil s^l_w; \;\; s^l_w = \frac{\max(W^l) - \min(W^l)}{2^t - 1}. \tag{16}$$

The 8-bit (both for weights and activations) post-training quantization works agreeably for the selected tasks. To highlight the QPP's advantages, we decrease the activations bit-width to 4 bits. The current work doesn't cover the efficiency of 4-bit activations in comparison to 8-bit. The low-bits strengthen the disadvantages of the standard methods, and improvements become easier to measure.

In all models, except ESPCN, we didn't quantize the first and last layer of NN as it was accepted in the literature. Therefore, in this case, QPP was placed after the first ReLU layer to increase QP's predictions' accuracy. A position of QPPs in a model and their number are indicated in Table 2 (the second row in the head). For example, we embed four QPPs to HRNet model because it contains a huge amount of layers. The QPP positions are depicted in the supplementary materials.

In this section we presented only results for D+S approach combined with QPP and summarized them in Tables 2 and 3. The results for standard quantization can be found in supplementary materials.

Firstly, we would like to note that the D+S approach with 1% density outperforms standard dynamic quantization significantly. For example, for ResNet34 the increase in accuracy is 2,19% due to the lack of quantization in just one percent of the activations (please, compare "Full Dynamic" and "D+S Dynamic" in Table 2). This strongly motivates us to draw attention to the D+S approach.

One can observe that the performance of models is comparable with dynamic D+S, and at first glance, the static D+S works well so that there are no reasons to use QPP (see Table 2). However, there is a significant issue with the computational cost stability of the static D+S on the sparse part

[3] https://github.com/HRNet

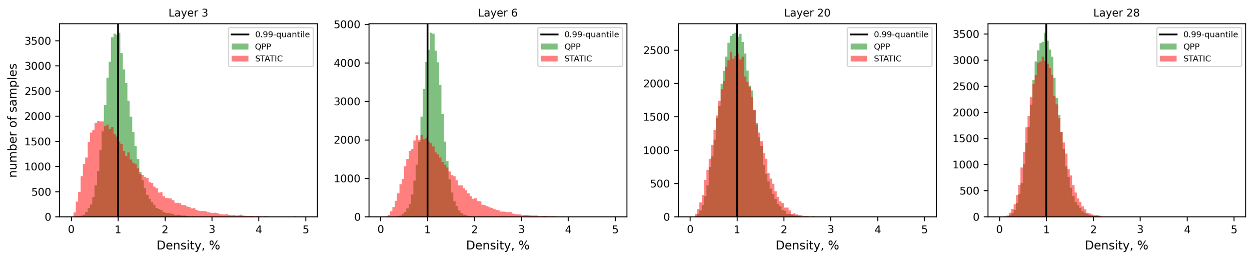

Figure 8. Distribution of density on validation dataset with 1 QPP (ResNet34). Black lines denote the desired density.

(see Section 5.2). In case of non-dynamic D+S approach the density is different for every input sample. Thus, some input data will be processed with a quality drop (see Fig 5) or with computational overhead (see Fig 6).

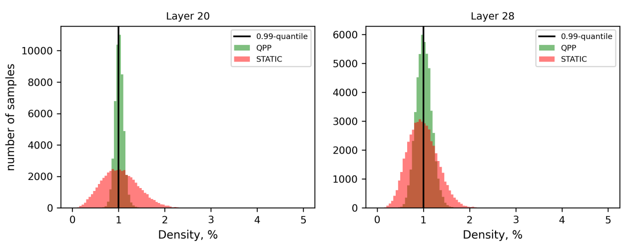

Figure 9. Distribution of density on validation dataset with 2 QPPs (ResNet34)

Also, we show the distribution of density for different input samples from the validation dataset for static and QPP quantization with one predictor for average density 1% in Figure 8 (black lines denote the desired density) or in Figure 9 for approach with two predictors. By correlating the obtained curves, one can see that some samples are at risk of getting into a zone with very low density and, therefore, suffer from quantization quality degradation, or in the zone of big density, which leads to big computational overhead. But as noted, QPP can significantly reduce these risks, making the Dense+Sparse approach more attractive for industrial use. The good evidence of this approval is contained in Table 3. We show the Failure Rate (FR) metric for 1% and 0.5% deviations from the desired 1% of density, i.e., the percentage of elements that lie outside $[0; 2]$ and $[0.5; 1.5]$ intervals correspondingly. Both FR(1) and FR(0.5) values of QPP are smaller than the static approach values in all our experiments. For example, the number of samples with a density of more than 2% is reduced by more than 10 times for a segmentation task. And each such sample leads to a slowdown in inference by at least 13% (see Table 1. Thus we can conclude that models quantized with QPP based D+S approach are much more stable.

We implemented dynamic D+S for the CPU Kirin 980 (Cortex A76, 1 thread) with ARM architecture and performed low-level optimization. We compared our results with the quantization presented in the BOLT framework, which uses dynamic quantization with the max/min value as QPs. The inference time of these approaches is shown in Table 1. One can observe that D+S with 1% density imposes 23% overhead on inference time but almost eliminates the drop in quality. But quantile calculation can be dropped by QPP, and inference time can be decreased by 11%. Moreover, the implementation of convolution with fused quantization also allows one to reduce computational complexity. Unfortunately, we don't have the implementation of fused operation, so we can't present the exact numbers for static D+S. In comparison with the FP model, D+S+QPP allows one to achieve x1.46 (BOLT int8: x1.63) acceleration with a negligible drop in quality.

9. Conclusions

In our work we presented a new post-training quantization method QPP that allows one to effectively use dynamic quantization statically with small computation overhead on CNN's inference. Unlike the only previously known static approach with QP's averaging, QPP adjusts QPs for every input sample, leading to less quality degradation and more stable quantization. For a more detailed study of the method we conducted experiments on a wide range of problems. Our analysis showed greater stability of QPP based schemes compared to the static approach with averaging. We mainly discussed the Dense+Sparse schemes, where the stability problem affects not only the quality of the model but also the inference time. For them we demonstrated that QPP also reduced the influence of static QPs on the model's performance.

In this paper we considered a specific linear regressor to estimate QPs, but we don't claim that this scheme is an optimal solution to the optimization problem (2). So by further analysis of feature maps distribution one can try to find a more suitable class of estimators $\mathcal{F}$ to solve (2). Also we would like to note that even for dynamic quantization the choice of optimal QPs is still an open problem. Since their values lie in the range of upper α-quantiles, we expect that QPP can be applied instead of the arbitrary dynamic quantization method.

References

[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7950–7958. Curran Associates, Inc., 2019.

[2] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie line Alberi Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proceedings of the British Machine Vision Conference*, pages 135.1–135.10. BMVA Press, 2012.

[3] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization, 2020.

[4] X.P. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *ICCV*, 2013.

[5] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks, 2018.

[6] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[8] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization, 2019.

[9] Xavier Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[11] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving post training neural quantization: Layer-wise calibration and integer programming, 06 2020.

[12] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper, 2018.

[13] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[14] Szymon Migacz. 8-bit inference with tensorrt. In *GPU Technology Conference*, 2017.

[15] M. Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. *ArXiv*, abs/2004.10568, 2020.

[16] E. Park, D. Kim, and S. Yoo. Energy-efficient neural network accelerator based on outlier-aware low-precision computation. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 688–698, 2018.

[17] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. Value-aware quantization for training and inference of neural networks, 2018.

[18] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: database and results. *Image and vision computing*, 47:3–18, 3 2016.

[19] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

[21] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2019.

[22] Wayne Wu, Chen Qian, Shuo Yang, Quan Wang, Yici Cai, and Qiang Zhou. Look at boundary: A boundary-aware face alignment algorithm. In *CVPR*, 2018.

[23] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, Feb 2019.

[24] Roman Zeyde, Michael Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, 2010.

[25] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2016.