JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

HUAWEI

Toward Accurate Post-Training Quantization for Image Super Resolution

No. 967

Zhijun Tu, Jie Hu, Hanting Chen, Yunhe Wang

Huawei Noah’s Ark Lab

Background & Motivation

- Model quantization is a crucial step for deploying super resolution (SR) networks on mobile devices.
- Existing works focus on quantization-aware training (QAT), which requires complete dataset and expensive computational overhead.
- On the contrast, post-training quantization (PTQ) only requires a few unlabeled calibration images without training, which enables fast deployment on various devices within minutes.

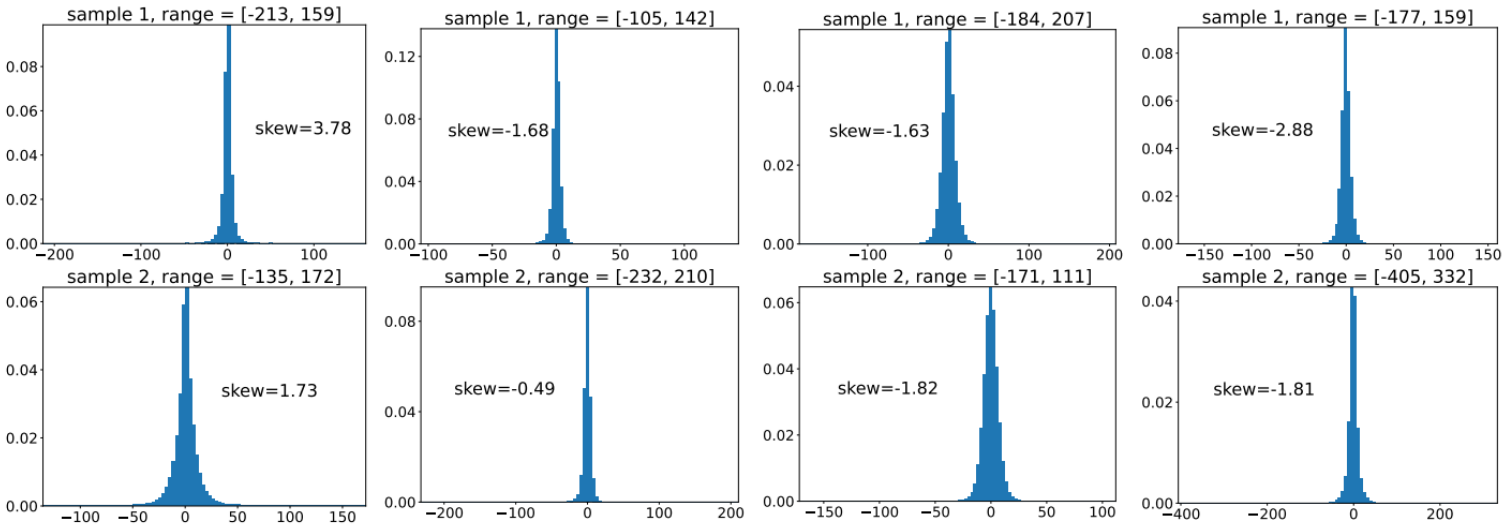
Method	Type	Data	Gt	Bs	Iters	Run time
EDSR [28]	FP	800	✓	16	15,000	240×
PAMS [25]	QAT	800	✓	16	1,500	24×
FQSR [39]	QAT	800	✓	16	15,000	120×
CADyQ [14]	QAT	800	✓	8	30,000	240×
DAQ [15]	QAT	800	✓	4	300,000	1200×
DDTB [49]	QAT	800	✓	16	3,000	48×
Ours	PTQ	100	×	2	500	1×

Background & Motivation

- Different from the image classification, super resolution requires accurate prediction for each pixel of the output images, which is much sensitive to low-bit compression for feature maps.
- We observe three properties of their distributions that are much unfriendly to quantization:

■ Long-tailed ■ Asymmetric ■ Highly-dynamic

W/A	32/32	6/32	32/6	6/6
PSNR/SSIM	26.646/0.804	26.587/0.802	25.894/0.773	25.890/0.773

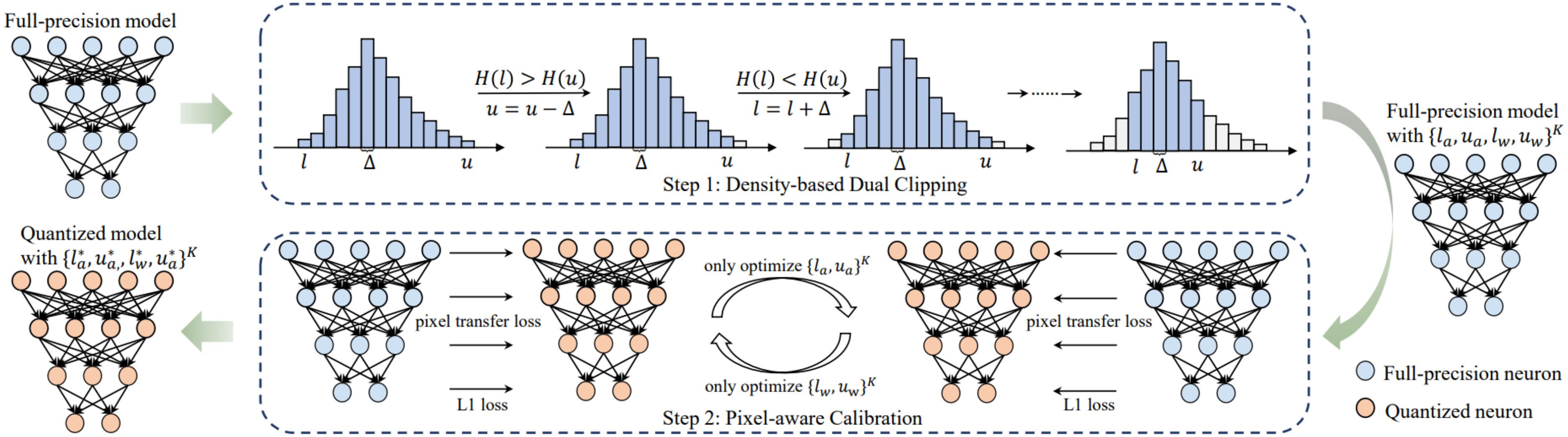


(a) body.0.conv1 (b) body.11.conv1 (c) body.21.conv1 (d) body.31.conv1

Building Accurate Post-training quantization for SR

we propose a coarse-to-fine method to get the accurate quantized SR model with post-training quantization.

- We first introduce the density-based dual clipping (DBDC) to cut off most of the outliers for narrowing the distribution to a valid range
- Then utilize pixel-aware calibration (PaC) to help the quantized network fit the highly dynamic activations for different samples



What is the Model Quantization

There are three steps in tensor quantization:

(1) Truncate the tensor x into range $[l, u]$,

(2) Map the floating-point tensor of the range $[l, u]$ to the integer tensor of the range $[0, 2^n - 1]$,

(3) Reconstruct the floating point tensor from the integer tensor.

$$x_c = \mathrm{Clamp}(x, l, u),$$

$$x_{int} = \mathrm{Round}(\frac{x_c - l}{u - l} \times (2^N - 1)),$$

$$x_q = x_{int} \times \frac{(u - l)}{2^N - 1} + l,$$

Density-based Dual Clipping

- DBDC Aims to cut off outliers of activations, help narrow the distribution to a valid range.
  1. We first divide the original activation x into the N equal interval based on its min-max value

$$\Delta = (\max(x) - \min(x))/N,$$
$$H(p) = \sum_{i \in x} \mathbb{I}(i{>}p \,\&\, i{<}p + \Delta),$$

  2. By comparing the density values between the position of l_a and u_a iteratively, we make the clipping position with smaller density closer to the middle

$$l_a^t, u_a^t = \begin{cases} l_a^{t-1} + \Delta, u_a^{t-1}, & H(l_a^{t-1}) < H(u_a^{t-1}) \\ l_a^{t-1}, u_a^{t-1} - \Delta, & H(l_a^{t-1}) \geq H(u_a^{t-1}) \end{cases}$$

  3. The global bounds l_a and u_a are updated by the exponential moving average (EMA) method

$$l_a = \beta \cdot l_a + (1 - \beta) \cdot l_a^T,$$
$$u_a = \beta \cdot u_a + (1 - \beta) \cdot l_a^T,$$

Pixel-aware Calibration

- PaC finetunes these clipping parameters for fitting the highly-dynamic feature maps.

1. With the unlabeled calibration images and full-precision pretrained model, we can get outputs and the middle feature maps for different layers to build a dataset for finetuning

$$(\text{input}, \text{label})^i = (x^i, (F_1^i, F_2^i, ..., F_N^i, O^i))$$

2. Then we construct the PaC loss between the output and feature maps of middle layers for full-precision model and quantized model

$$L_{PaC} = L_o + \lambda L_{pt}$$

$$L_o = \frac{1}{H_o \cdot W_o \cdot C_o} ||O - O_q||_1, \qquad L_{pt} = \frac{1}{B} \sum_i^N \frac{1}{H_i \cdot W_i \cdot C_i} ||\hat{F}_i - \hat{F}_{q_i}||_2,$$

3. To stabilize the finetune process, we further propose to iteratively optimize the clipping parameters of weights and activations instead of finetuning them together.

Experimental Results on EDSR

Method	Bit	Set5 (×4)	Set14 (×4)	BSD100 (×4)	Urban100 (×4)	Set5 (×2)	Set14 (×2)	BSD100 (×2)	Urban100 (×2)
Baseline	32	32.485/0.899	28.815/0.788	27.721/0.742	26.646/0.804	38.193/0.961	33.948/0.920	32.352/0.902	32.967/0.936
Bicubic	32	28.420/0.810	26.000/0.703	25.960/0.668	23.140/0.658	33.660/0.930	30.24/0.869	29.560/0.843	26.880/0.840
OpenVINO [11]	8	32.148/0.892	28.629/0.782	27.572/0.735	26.454/0.796	32.148/0.892	28.629/0.782	27.572/0.735	26.454/0.796
TensorRT [38]	8	32.329/0.895	28.711/0.784	27.639/0.738	26.548/0.799	37.880/0.958	33.774/0.917	32.217/0.899	32.764/0.933
SNPE [18]	8	32.329/0.896	28.707/0.786	27.646/0.740	26.551/0.800	37.786/0.957	33.751/0.917	32.189/0.898	32.733/0.932
MSE [4]	8	32.191/0.897	28.524/0.785	27.539/0.740	26.341/0.799	37.781/0.960	33.349/0.919	32.114/0.901	32.237/0.934
Percentile [26]	8	32.306/0.897	28.642/0.785	27.630/0.739	26.310/0.796	38.041/0.960	33.686/0.910	32.256/0.901	32.690/0.934
MinMax [20]	8	32.350/0.896	28.730/0.785	27.654/0.740	26.560/0.800	37.983/0.959	33.832/0.918	32.260/0.900	32.719/0.934
Ours	8	**32.460/0.898**	**28.763/0.787**	**27.695/0.741**	**26.567/0.802**	**38.120/0.960**	**33.850/0.920**	**32.313/0.901**	**32.810/0.935**
OpenVINO [11]	6	30.283/0.843	27.426/0.735	26.592/0.687	25.214/0.740	34.337/0.907	31.436/0.860	30.236/0.833	30.172/0.878
TensorRT [38]	6	30.696/0.851	27.719/0.744	26.765/0.694	25.459/0.749	34.735/0.913	31.778/0.867	30.472/0.841	30.582/0.887
SNPE [18]	6	30.493/0.839	27.599/0.735	26.664/0.685	25.386/0.742	34.305/0.903	31.499/0.858	30.249/0.831	30.336/0.877
MSE [4]	6	30.648/0.879	27.593/0.771	26.881/0.725	25.256/0.773	35.746/0.950	32.163/0.909	31.231/0.909	30.302/0.917
Percentile [26]	6	31.496/0.875	28.188/0.768	27.213/0.720	25.890/0.773	36.610/0.944	32.890/0.904	31.599/0.885	31.666/0.917
MinMax [20]	6	31.073/0.863	27.986/0.760	27.011/0.713	25.643/0.713	36.037/0.936	32.544/0.897	31.286/0.878	31.208/0.908
Ours	6	**32.300/0.894**	**28.653/0.784**	**27.627/0.738**	**26.382/0.797**	**37.896/0.958**	**33.675/0.918**	**32.186/0.899**	**32.452/0.932**
OpenVINO [11]	4	20.526/0.542	18.949/0.475	18.636/0.439	18.418/0.467	24.157/0.606	22.642/0.559	22.346/0.543	22.083/0.589
TensorRT [38]	4	21.343/0.512	19.809/0.461	19.495/0.423	19.100/0.450	23.897/0.608	22.325/0.571	22.208/0.553	22.068/0.600
SNPE [18]	4	21.417/0.472	20.035/0.413	19.925/0.392	19.320/0.406	23.284/0.548	22.086/0.522	22.215/0.517	21.873/0.555
MSE [4]	4	24.600/0.737	24.365/0.668	24.343/0.635	22.183/0.649	28.813/0.855	27.898/0.827	27.706/0.813	25.714/0.826
Percentile [26]	4	26.570/0.696	24.834/0.620	24.173/0.576	22.871/0.608	29.803/0.788	27.992/0.758	27.187/0.736	26.514/0.766
MinMax [20]	4	23.132/0.635	21.208/0.569	23.266/0.508	20.220/0.554	28.005/0.744	25.960/0.703	24.684/0.682	24.717/0.725
Ours	4	**31.203/0.867**	**27.977/0.760**	**27.085/0.714**	**25.556/0.764**	**36.327/0.942**	**32.753/0.904**	**31.477/0.884**	**30.900/0.913**

Experimental Results on SRResNet

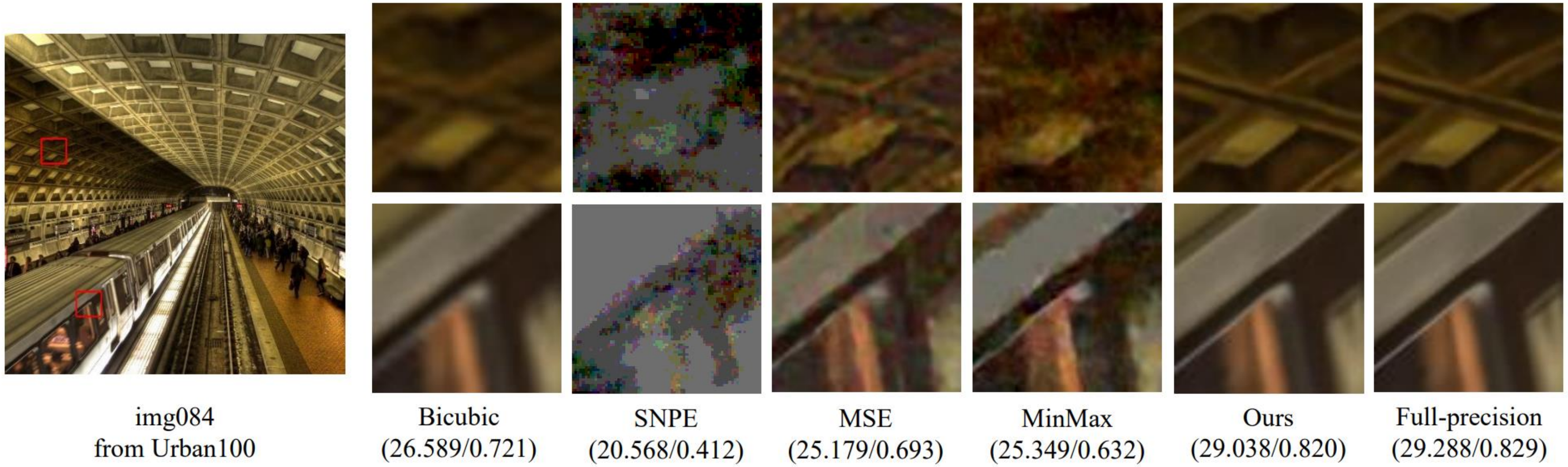
Method	Bit	Set5 (×4)	Set14 (×4)	BSD100 (×4)	Urban100 (×4)	Set5 (×2)	Set14 (×2)	BSD100 (×2)	Urban100 (×2)
Baseline	32	32.234/0.896	28.656/0.784	27.630/0.738	26.229/0.791	38.091/0.961	33.752/0.919	32.241/0.900	32.367/0.931
Bicubic	32	28.420/0.810	26.000/0.703	25.960/0.668	23.140/0.658	33.660/0.930	30.240/0.869	29.560/0.843	26.880/0.840
OpenVINO [11]	8	32.003/0.890	28.505/0.778	27.509/0.732	26.039/0.783	37.451/0.955	33.350/0.912	31.978/0.895	31.978/0.924
TensorRT [38]	8	32.013/0.891	28.507/0.779	27.508/0.733	26.069/0.785	37.506/0.956	33.428/0.913	31.984/0.895	32.026/0.925
SNPE [18]	8	32.120/0.893	28.556/0.781	27.562/0.736	26.111/0.788	37.734/0.957	33.529/0.915	32.085/0.896	32.100/0.927
MSE [4]	8	32.006/0.892	28.387/0.779	27.469/0.734	25.910/0.784	37.737/0.958	33.247/0.915	31.972/0.897	31.665/0.926
Percentile [26]	8	32.092/0.893	28.492/0.780	27.525/0.735	26.046/0.786	37.739/0.958	33.414/0.916	32.058/0.897	31.965/0.927
MinMax [20]	8	31.984/0.891	28.495/0.779	27.503/0.733	26.057/0.785	37.539/0.956	33.413/0.913	31.992/0.895	32.020/0.925
Ours	8	**32.207/0.895**	**28.619/0.783**	**27.618/0.738**	**26.191/0.790**	**38.032/0.960**	**33.648/0.919**	**32.212/0.900**	**32.210/0.930**
OpenVINO [11]	6	30.080/0.835	27.348/0.727	26.665/0.683	24.861/0.721	33.539/0.884	31.007/0.849	30.050/0.827	29.505/0.857
TensorRT [38]	6	29.990/0.828	27.277/0.724	26.553/0.681	24.782/0.719	33.634/0.885	30.923/0.846	30.011/0.827	29.270/0.854
SNPE [18]	6	29.650/0.814	27.112/0.714	26.449/0.671	24.690/0.710	33.120/0.874	30.501/0.834	29.654/0.813	28.895/0.842
MSE [4]	6	30.822/0.872	27.642/0.760	27.002/0.718	25.003/0.752	36.010/0.944	32.099/0.898	31.174/0.881	29.935/0.904
Percentile [26]	6	30.970/0.869	27.874/0.760	27.085/0.715	25.340/0.756	35.826/0.936	32.314/0.893	31.192/0.874	30.707/0.902
MinMax [20]	6	30.725/0.859	27.784/0.750	26.987/0.704	25.233/0.744	34.964/0.919	31.895/0.877	30.755/0.856	30.286/0.886
Ours	6	**32.089/0.892**	**28.504/0.779**	**27.561/0.733**	**26.011/0.783**	**37.811/0.959**	**33.295/0.916**	**32.068/0.898**	**31.719/0.926**
OpenVINO [11]	4	24.316/0.573	23.201/0.519	23.276/0.500	21.614/0.528	24.415/0.535	23.570/0.508	23.551/0.502	22.942/0.556
TensorRT [38]	4	23.729/0.461	22.648/0.402	22.808/0.389	21.089/0.399	24.769/0.535	23.753/0.502	23.733/0.491	22.753/0.526
SNPE [18]	4	23.130/0.413	22.317/0.376	22.404/0.358	20.793/0.371	24.111/0.505	23.297/0.477	23.195/0.464	22.452/0.511
MSE [4]	4	27.979/0.784	25.828/0.680	25.704/0.641	23.042/0.639	31.239/0.870	29.106/0.828	28.470/0.801	26.376/0.804
Percentile [26]	4	27.283/0.699	25.411/0.625	25.329/0.603	22.990/0.605	27.369/0.703	26.477/0.689	26.180/0.668	24.866/0.686
MinMax [20]	4	26.639/0.654	25.122/0.599	25.107/0.577	22.746/0.573	25.824/0.603	25.302/0.602	25.191/0.584	23.914/0.606
Ours	4	**31.146/0.878**	**27.889/0.763**	**27.152/0.718**	**25.133/0.753**	**36.487/0.951**	**32.404/0.904**	**31.357/0.885**	**29.896/0.904**

Combined with QAT

Method	Scale	Bit	FQ	QAT	Set5	Set14	BSD100	Urban100
PAMS [25]	×4	32			32.095/0.894	28.576/0.781	27.562/0.736	26.035/0.785
		4	✗	✓	31.591/0.885	28.199/0.773	27.322/0.728	25.321/0.762
	×2	32			37.985/0.960	33.568/0.918	32.155/0.899	31.977/0.927
		4	✗	✓	37.665/0.959	33.196/0.915	31.936/0.897	31.100/0.919
FQSR [39]	×4	32			32.007/0.892	28.486/0.778	27.528/0.731	25.934/0.781
		4	✓	✓	30.928/0.870	27.816/0.761	27.073/0.715	24.927/0.744
	×2	32			37.885/0.958	33.425/0.915	32.106/0.897	31.777/0.924
		4	✓	✓	37.038/0.951	32.835/0.908	31.668/0.889	30.646/0.911
Ours	×4	32			32.485/0.899	28.815/0.788	27.721/0.742	26.646/0.804
		4	✗	✗	32.105/0.891	28.563/0.781	27.553/0.714	26.051/0.787
		4	✗	✓	32.295/0.895	28.576/0.784	27.558/0.738	26.232/0.794
		4	✓	✗	31.203/0.867	27.977/0.760	27.085/0.714	25.556/0.764
		4	✓	✓	31.641/0.881	28.217/0.772	27.332/0.727	25.748/0.777
	×2	32			38.193/0.961	33.948/0.920	32.352/0.902	32.967/0.936
		4	✗	✗	37.837/0.958	33.662/0.917	32.146/0.898	32.335/0.930
		4	✗	✓	37.992/0.960	33.838/0.919	32.205/0.900	32.545/0.933
		4	✓	✗	36.327/0.942	32.753/0.904	31.477/0.884	30.900/0.913
		4	✓	✓	37.561/0.955	33.442/0.915	31.992/0.896	31.725/0.924

Ablation study & Visualization

DBDC	PaC	Set5	Set14	BSD100	Urban100
		26.570/0.696	24.834/0.620	24.173/0.576	22.871/0.608
✓		30.406/0.838	27.510/0.735	26.633/0.687	25.312/0.736
	✓	28.000/0.775	26.002/0.681	25.406/0.630	24.116/0.669
✓	✓	31.203/0.867	27.977/0.760	27.085/0.714	25.556/0.764



Thank you!

PyTorch

MindSpore