Sensitivity-Aware Post-Training Quantization for Deep Neural Networks

Zekang Zheng[1], Haokun Li[1], Yaofo Chen(✉)[1], Mingkui Tan[1,2,3], Qing Du[1]

South China University of Technology[1] Pazhou Laboratory[2] Key Laboratory of Big Data and Intelligent Robot, Ministry of Education[3]

PRCV 2025 智汇沪上，海纳视界

Background

- **Problem**: The widespread application of deep learning models has led to a continual increase in their **scale** and **computational complexity**, posing significant challenges for deploying these models on **resource-constrained devices**.
- **Limitations of Existing Methods**: Prior quantization approaches, being inherently an approximation, inevitably **leads to performance degradation**.

How to achieve **efficient and rapid quantization while preserving model accuracy** has become a prominent research focus of model compression.

Related Work

Model quantization methods can be broadly divided into two categories:

- **Quantization-Aware Training (QAT):** Quantization operations are simulated during the training or fine-tuning phase to align model parameters with the quantized representation.
  - *Advantages:* Achieves **high accuracy**; effectively mitigates quantization-induced errors.
  - *Disadvantages:* **Requires retraining** with full datasets; incurs **high computational cost and long training time**, unsuitable for rapid deployment.
- **Post-Training Quantization (PTQ):** Applies quantization directly to a pre-trained full-precision model without retraining, usually with the aid of a small calibration dataset.
  - *Advantages:* **Simple, flexible, and fast**; enables efficient deployment.
  - *Disadvantages:* May cause **non-negligible accuracy degradation**, especially under low-bit settings; sensitive to calibration data quality.

Mainstream PTQ Methods. Representative Post-Training Quantization (PTQ) methods include **AdaRound** and **OBS/OBQ**, effectively preserve accuracy under high compression ratios, they suffer from **high computational complexity, sequential parameter updates, and long quantization times**, making them unsuitable for real-time or resource-constrained deployment scenarios.

Contributions

- **Quantitation Framework**. We designed a **sensitivity-guided model parameter quantization strategy**, which effectively mitigates accuracy loss during the quantization process.
- **Parallel Quantitation Algorithm**. We proposed a **parallel parameter quantization algorithm** along the row dimension. By establishing a shared inverse Hessian update mechanism, parameter quantization operations are executed in parallel along the row dimension, significantly reducing computational overhead and **substantially decreasing quantization time**.
- **Great Efficiency**. Extensive experimental results show that our method significantly **reduces quantization time and memory footprint** across various models, achieving nearly **lossless accuracy** compared to current SoTA methods.

Second-Order Quantization Background (OBS/OBQ)

We formulate **post-training quantization as a second-order optimization problem**. By applying a Taylor expansion of the loss function around a local minimum, the error increase from weight perturbation is approximated as:

$$\Delta E \approx \tfrac{1}{2}\,\Delta w^\top H\,\Delta w,$$

where H is the Hessian matrix. The **Optimal Brain Surgeon (OBS)** method minimizes this error under the constraint that a target weight is fixed (*e.g.*, pruned or quantized), yielding the optimal compensation

$$\Delta w = -\frac{w_q}{[H^{-1}]_{qq}}\,H^{-1}e_q, \quad L_q = \frac{w_q^2}{2[H^{-1}]_{qq}},$$

where L_q measures parameter sensitivity. Extending this to quantization gives **OBQ**, which uses H^{-1} to propagate quantization error and update remaining weights optimally.

Advantages: Accurately models quantization error; theoretically minimizes accuracy loss. **Limitations:** Requires **computing and updating large Hessian matrices**; updates are sequential and computationally expensive, limiting its practicality for large-scale or real-time quantization.

Sensitivity-Guided Efficient Quantization (FastOBQ)

We propose **FastOBQ**, an efficient post-training quantization method that improves both accuracy and speed over OBQ through two core designs.

(1) Sensitivity-Guided Quantization Order. Traditional PTQ methods quantize weights in **ascending sensitivity order—starting from low-sensitivity parameters**. However, this causes error accumulation: when high-sensitivity weights are quantized later, most other parameters have already been fixed, leaving little room for compensation. Our FastOBQ reverses this process by quantizing in **descending sensitivity order**. High-sensitivity weights are processed first, allowing the remaining low-sensitivity weights to compensate for larger quantization errors.

(2) Row-Parallel Column-Synchronized Quantization. OBQ performs independent row-wise updates and **repeatedly recomputes the inverse Hessian matrix**, resulting in high computational and memory cost. Our FastOBQ observes that parameter sensitivity tends to **cluster along columns, enabling column-wise synchronization**. It aggregates sensitivity per column

$$S_j = \sum_i L_q(w_{ij}),$$

then quantizes columns in descending S_j order while maintaining a single shared inverse Hessian H^{-1}. All weights in one column are quantized simultaneously with a single global Hessian update.

Complexity Comparisons and Analysis

Computational Complexity of OBQ: The standard OBQ performs independent Hessian updates for each row, leading to a total computational cost of $\mathcal{O}(d_{\text{row}}\, d_{\text{col}}^3)$.
Computational Complexity of Our FastOBQ: FastOBQ quantizes all parameters column by column while sharing a inverse Hessian matrix, thereby reducing redundant updates. Its total complexity becomes $\mathcal{O}(d_{\text{col}}^3) + \mathcal{O}(d_{\text{row}}\, d_{\text{col}}^2)$, representing an **order-of-magnitude reduction compared to OBQ**.

Comparisons with SoTA Methods

- Comparisons of different methods under 4-bit weight quantization. "Time" denotes quantization times. "Mem." denotes peak memory usage.

Model	Bit Width	Method	Layer-wise Quant.	Accuracy	Time (s)	Mem. (MB)
ResNet-18	FP32	-	-	69.76	-	-
	W4A32	BRECQ		70.94	1789	5391
		Bias Correction	✓	53.76	-	-
		AdaRound	✓	68.52	1225	3834
		AdaQuant	✓	67.01	341	4789
		Bit-split	✓	69.11	3191	10803
		OBQ	✓	69.33	7784	6502
		FastOBQ (Ours)	✓	**69.37**	**58**	**3069**
ResNet-50	FP32	-	-	76.13	-	-
	W4A32	BRECQ		76.463	5558	10295
		Bias Correction	✓	63.52	-	-
		AdaRound	✓	75.26	3766	4517
		AdaQuant	✓	75.22	1127	7760
		Bit-split	✓	75.58	5032	10856
		OBQ	✓	75.71	9287	6859
		FastOBQ (Ours)	✓	**75.77**	**80**	**3683**

- Comparisons of different methods under YOLOv5 model quantization. "Time" denotes quantization times. "Mem." denotes peak memory usage.

Model	Bit Width	Method	Layer-wise	mAP@0.5	mAP@0.5:0.95	Time (s)	Mem. (MB)
YOLOv5s	FP32	-	-	37.46	56.73	-	-
	W6A32	Bias Correction	✓	26.52	45.69	-	-
		OBQ	✓	**37.01**	**56.46**	611	1966
		FastOBQ (Ours)	✓	36.60	56.25	**17**	**1966**
	FP32	-	-	37.46	56.73	-	-
	W8A32	Bias Correction	✓	26.95	45.89	-	-
		OBQ	✓	**37.41**	**56.74**	766	1966
		FastOBQ (Ours)	✓	37.36	56.70	**22**	**1966**
YOLOv5m	FP32	-	-	45.16	63.88	-	-
	W6A32	Bias Correction	✓	35.33	53.96	-	-
		OBQ	✓	**44.83**	**63.81**	3289	4350
		FastOBQ (Ours)	✓	44.37	63.44	**36**	**3007**
	FP32	-	-	45.16	63.88	-	-
	W8A32	Bias Correction	✓	35.56	53.99	-	-
		OBQ	✓	**45.09**	63.87	3282	4350
		FastOBQ (ours)	✓	45.08	**63.88**	**39**	**3007**

Contact Information

- Email: chenyaofo@scut.edu.cn
- Personal Page: https://chenyaofo.com

PDF

Personal Page