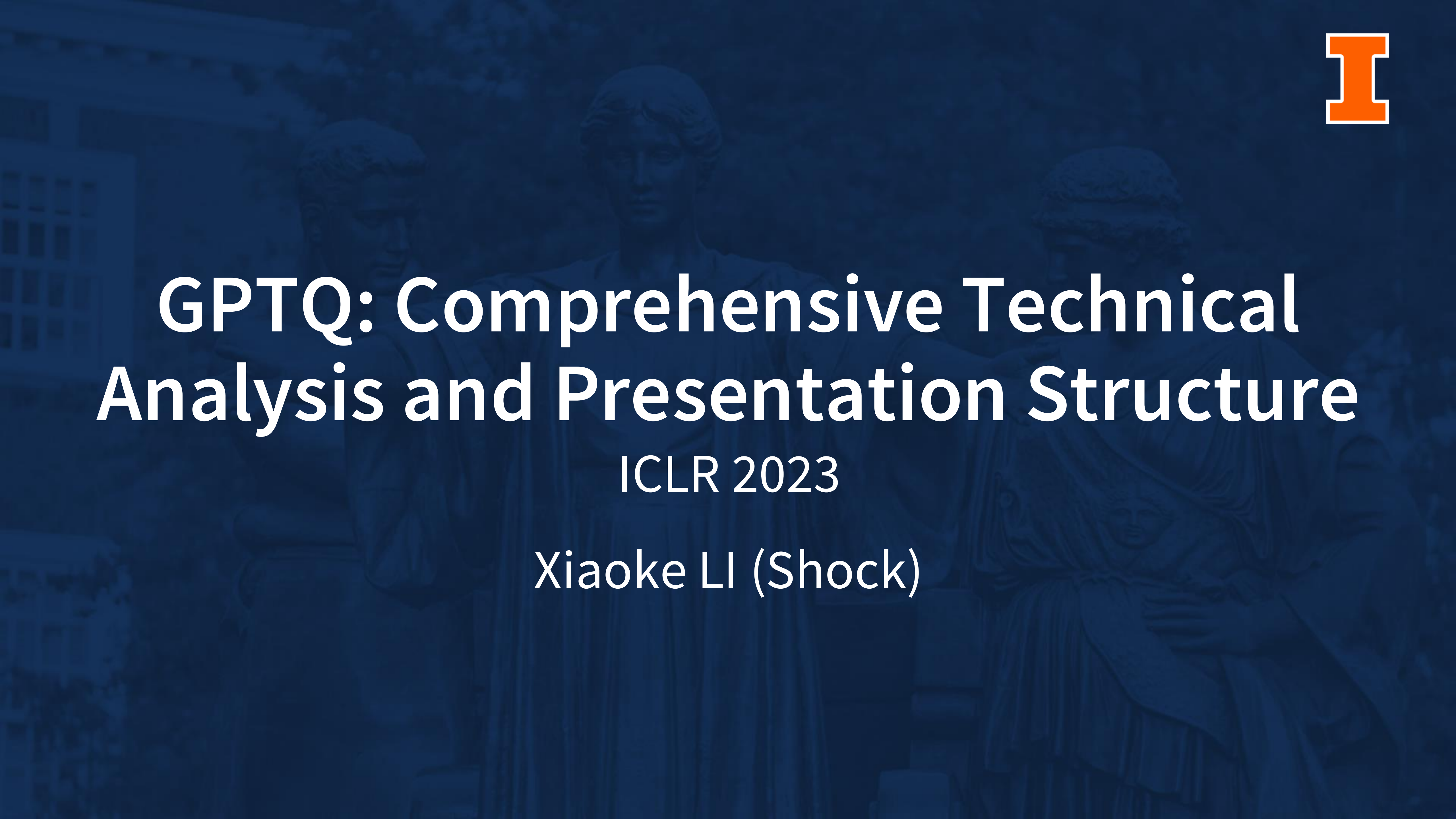


GPTQ: Comprehensive Technical Analysis and Presentation Structure

ICLR 2023

Xiaoke LI (Shock)

Background and Motivation

Problem Scope

- GPT-3 and other LLMs: Breakthrough performance in language tasks
- **Massive computational costs**: Hundreds of GPU years for training
- **Memory and inference challenges**: 175 billion parameters in GPT-3 require over 326GB of memory (float16)
- **Deployment difficulties**: Requires multiple GPUs, high latency, and resource-intensive setups
- The need for **efficient model compression** without sacrificing performance

Quantization Landscape

- **Pruning**: Removes redundant weights (e.g., OBD) but requires retraining
- **Knowledge Distillation**: Smaller models mimic larger ones, but expensive retraining
- **Quantization**: Reduces bit width of weights and activations
    - **Quantization-Aware Training (QAT)**:
        - Higher accuracy but requires additional training
        - Incorporates quantization into the training process
    - **Post-Training Quantization (PTQ)**:
        - Fast, no retraining, but accuracy drop if poorly implemented

Quantization Landscape

Post-Training Quantization (PTQ):

- **Post-Training Dynamic Quantization**
  - does not use calibration datasets,
  - directly converting each layer through quantization formulas
  - **QLoRA**
- **Post-Training Calibration Quantization**
  - requires input of a representative dataset
  - adjusting quantized weights based on each layer's input and output
  - **GPTQ**

Previous Limitations

RTN (Round-to-Nearest): Applied to GPT models, works for 8-bit weights, but fails at higher compression rates

Higher compression rates (e.g., 3-bit or 2-bit) lead to **significant accuracy loss**

Old methods are **not scalable** for models with hundreds of billions of parameters (e.g., W8A8)

High latency: The high communication overhead between GPUs during inference creates **IO bottleneck**

GPTQ Intro

Efficient Compression:

• GPTQ compresses models with **175 billion parameters in approximately 4 GPU hours**, reducing bitwidth to 3 or 4 bits per weight with **negligible accuracy loss**

• More than **doubles compression gains** compared to prior one-shot quantization methods

• For the first time, GPTQ enables **175-billion-parameter models** to be executed on a **single GPU** (NVIDIA A100) for generative inference

GPTQ Intro

Key ideas:

- **Minimizes Squared Error**: Uses an **approximate second-order information** to minimize the squared error
- **Arbitrary Order** Quantization: Apply quantization in any fixed order
- **Lazy Batch-Updates**: C2C ratio (Compression-to-Computing) low, update all parameters
- **Cholesky Reformulation**: Handles numerical inaccuracies

GPTQ Intro

Mathematical Rigor:

- GPTQ is based on a rigorous mathematical framework, originating from the **Optimal Brain Damage (OBD)** algorithm proposed by Yann LeCun in 1990.
- Over time, this was improved through **Optimal Brain Surgeon (OBS)** , **OBC**(Optimal Brain Compression) and **OBQ**(Optimal Brain Quantization) methods.
- GPTQ represents an accelerated and optimized version of OBQ, making it more suitable for large-scale models like GPT-3.
- OBD -> OBS -> OBC/OBQ -> GPTQ

Existing Work and Key Concepts

OBD (Optimal Brain Damage)

The specific algorithmic process is

1. Build the neural network
2. Train the neural network until the loss function converges
3. Calculate the second order derivatives $\mathrm{h_{kk}}$ of each parameter
4. Calculate the significance $s_k = h_{kk}\frac{\mu_k^2}{2}$ of each parameter
5. Sort the parameters by significance and remove some low significance parameters. Deleting parameters can be thought of as setting them to 0 and freezing them during training.
6. Repeat from step 2

OBD (Optimal Brain Damage)

$$\mathrm{L}(\mathrm{w} + \Delta w) = L(w) + g^T \Delta w + \tfrac{1}{2}\Delta w^T H \Delta w + O(\|\Delta w\|^3)$$
$$\Delta L = g^T \Delta w + \tfrac{1}{2}\Delta w^T H \Delta w + O(\|\Delta w\|^3)$$
$$\Delta E = \textstyle\sum_i g_i \delta u_i + \tfrac{1}{2}\sum_{i,j} h_{ij}\delta u_i \delta u_j + O(\|\delta u\|^3)$$

simplify

$$\Delta E = \tfrac{1}{2}\textstyle\sum_{i,i} h_{ii}\delta u_i \delta u_i$$

g_i: the gradient of the loss function to the parameter, first-order information.
δu_i: parameter perturbation.
h_{ij}: elements of the Hessian matrix, second-order information.

OBD (Optimal Brain Damage)

Formula Simplification

- The objective function is second-order, so do not consider higher-order terms **O(Δw3)**
- Assume that the model training has **converged sufficiently** so that the first-order partial derivatives of all parameters are 0
- Assume that after deleting any one parameter, the effect of the other parameters on the objective function remains unchanged. That is, the effect of each parameter on the objective function is **independent**. Thus we can also disregard the **cross terms**: hij=0,∀i,j,i≠j

OBS (Optimal Brain Surgeon)

OBD crudely considers only the **diagonal elements** of the Hessian matrix.

OBS considers **independence** between parameters does not hold, we still have to consider the **cross terms**

The main process

Find the parameter that has the **least impact** on the model and setting it to zero

Update **all** the model parameters to compensate for the impact of the zeroed parameter.

The process does **not** require retraining of the model.

$$\Delta E = \frac{1}{2}\sum_i h_{ii}\Delta w_i^2 + \frac{1}{2}\sum_{i\neq j} h_{ij}\Delta w_i \Delta w_j$$

Using vector/matrix notation, this can be simplified as:

$$\Delta E = \frac{1}{2}\Delta \mathbf{w}^T \mathbf{H} \Delta \mathbf{w}$$

The q-th dimension of $\Delta\mathbf{w}$, constrained by $-w_q$, is expressed as:

$$\mathbf{e}_q^T \cdot \Delta\mathbf{w} + w_q = 0$$

$\mathbf{e}_q$ is a one-hot vector with a value of 1 in the q-th position, and 0 elsewhere. This can be formulated as an optimization problem:

$$\min_{\mathbf{w}_q} \frac{1}{2}\Delta \mathbf{w}^T \mathbf{H} \Delta \mathbf{w} \quad \text{subject to} \quad \mathbf{e}_q^T \cdot \Delta\mathbf{w} + w_q = 0$$

Using the **Lagrange multiplier method** to solve this (unconstrained optimization):

$$L = \frac{1}{2}\Delta \mathbf{w}^T \mathbf{H} \Delta \mathbf{w} + \lambda(\mathbf{e}_q^T \cdot \Delta\mathbf{w} + w_q)$$

We derive:

$$\Delta\mathbf{w} = -\frac{w_q}{[\mathbf{H}^{-1}]_{qq}}\mathbf{H}^{-1}\cdot\mathbf{e}_q = -\frac{w_q}{[\mathbf{H}^{-1}]_{qq}}\mathbf{H}^{-1}_{:,q} and \quad L = \frac{1}{2}\frac{w_q^2}{[\mathbf{H}^{-1}]_{qq}}$$

We compute the impact of each parameterw_q on the objective, which is $\frac{1}{2}\frac{w_q^2}{[\mathbf{H}^{-1}]_{qq}}$, and rank them accordingly for pruning. After pruning, the remaining parameters are updated based $\Delta\mathbf{w}$.

This concept is applied in OBQ and GPTQ: parameters in a **block** are quantized sequentially, with unquantized parameters adjusted after each step to minimize precision loss.

OBC（Optimal Brain Compression）

OBS Complexity O(d ^ 4) = d * d ^3 (Hessian matrix inverse）

Recalculate the inverse Hessian matrix

Impossible applied in LLMs

OBC （ExactOBS）

Assume parameters in same row are correlated, different rows are not correlated

Avoid use entire Hessian matrix, d * d size, d = row parameters located

Rewrite the cost function:

$$||WX - \hat{W}X||_2^2 \approx \sum_{i=1}^{d_{row}} ||W_i x - \hat{W}_i x||_2^2$$

OBC（Optimal Brain Compression）

least squares minimization

$\min_{\hat{W}} \arg\min ||W_i x - \hat{W}_i x||_2^2$

where the least squares degree constraint matrix is

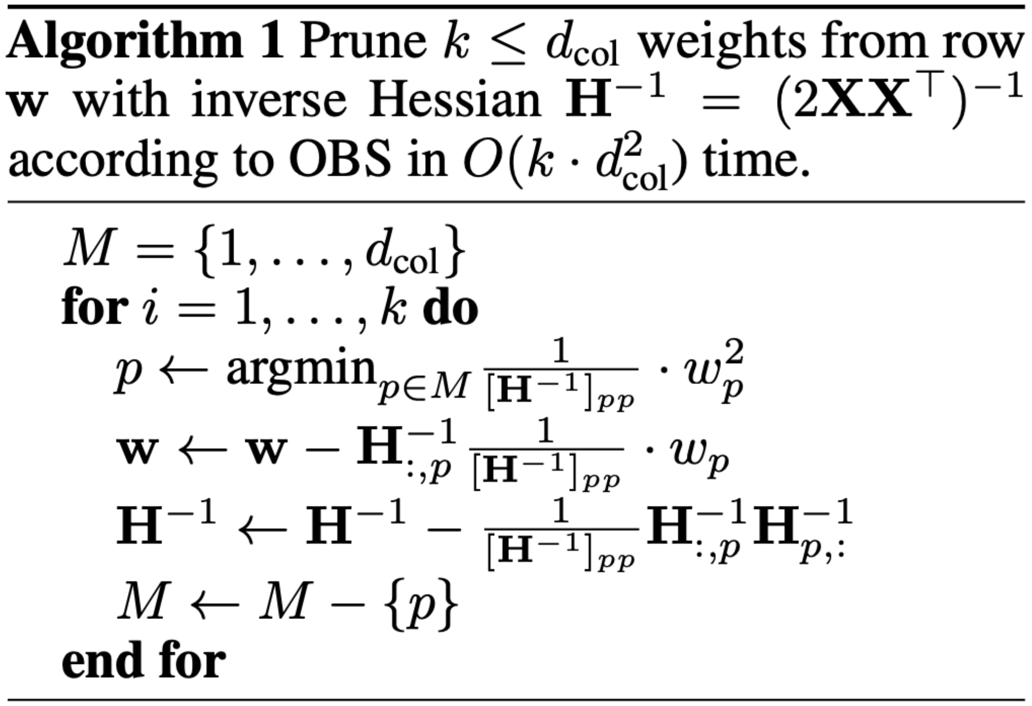
$H = 2XX^T$.

Hessian matrix no need to be recalculated

Update H^{-1}

when pruning, we remove the corresponding row and column of H^{-1}

update the remaining elements using Gaussian elimination

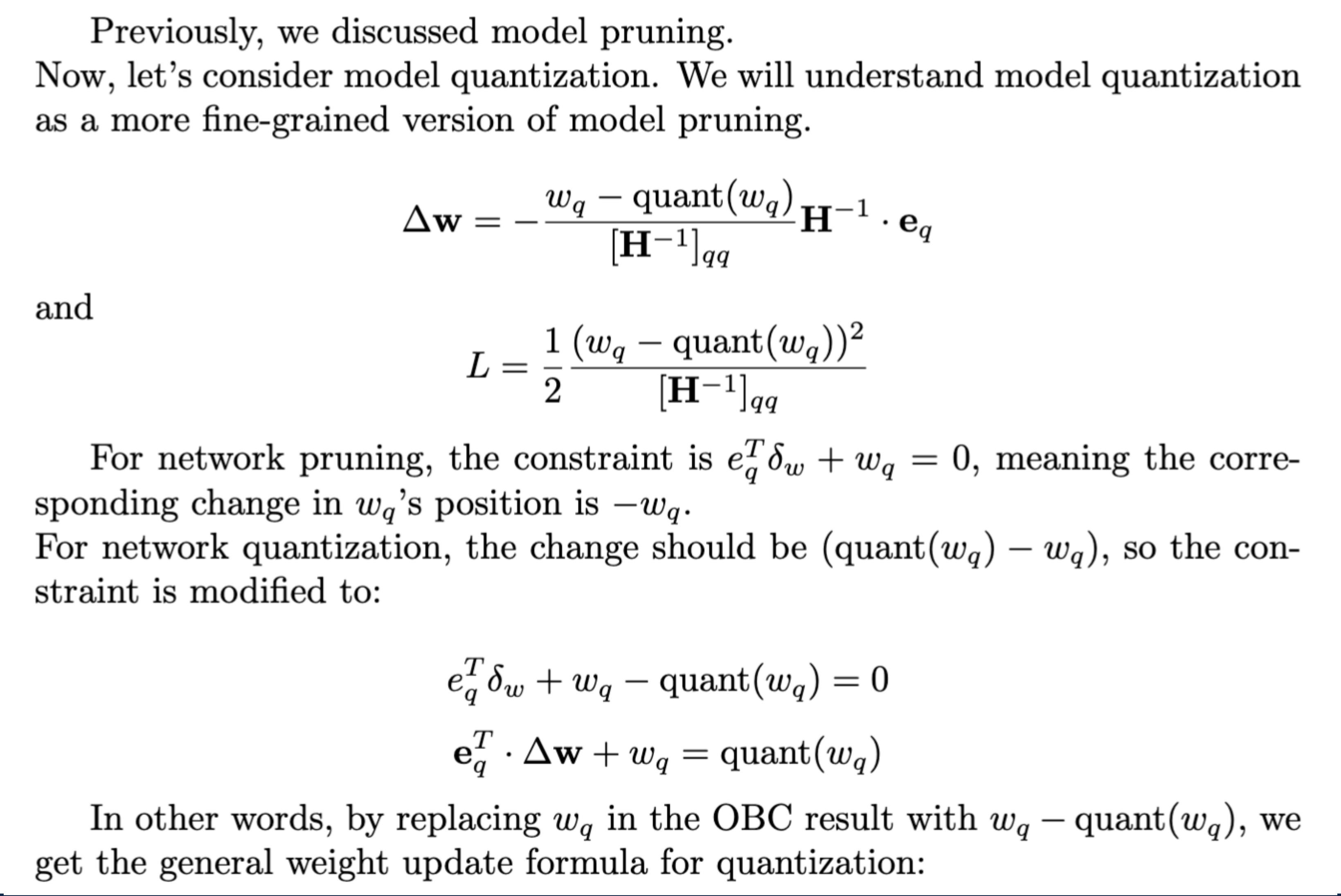


Algorithm 1 Prune $k \leq d_{\text{col}}$ weights from row $\mathbf{w}$ with inverse Hessian $\mathbf{H}^{-1} = (2\mathbf{X}\mathbf{X}^\top)^{-1}$ according to OBS in $O(k \cdot d_{\text{col}}^2)$ time.

```
$M = \{1, \dots, d_{\text{col}}\}$
for $i = 1, \dots, k$ do
    $p \leftarrow \text{argmin}_{p \in M} \frac{1}{[\mathbf{H}^{-1}]_{pp}} \cdot w_p^2$
    $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{H}^{-1}_{:,p} \frac{1}{[\mathbf{H}^{-1}]_{pp}} \cdot w_p$
    $\mathbf{H}^{-1} \leftarrow \mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{pp}} \mathbf{H}^{-1}_{:,p} \mathbf{H}^{-1}_{p,:}$
    $M \leftarrow M - \{p\}$
end for
```

OBQ（Optimal Brain Quantizer ）

Follow OBC in same paper



Previously, we discussed model pruning.
Now, let's consider model quantization. We will understand model quantization as a more fine-grained version of model pruning.

$$\Delta \mathbf{w} = -\frac{w_q - \text{quant}(w_q)}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}^{-1} \cdot \mathbf{e}_q$$

and

$$L = \frac{1}{2} \frac{(w_q - \text{quant}(w_q))^2}{[\mathbf{H}^{-1}]_{qq}}$$

For network pruning, the constraint is $e_q^T \delta_w + w_q = 0$, meaning the corresponding change in w_q's position is $-w_q$.
For network quantization, the change should be $(\text{quant}(w_q) - w_q)$, so the constraint is modified to:

$$e_q^T \delta_w + w_q - \text{quant}(w_q) = 0$$

$$\mathbf{e}_q^T \cdot \Delta \mathbf{w} + w_q = \text{quant}(w_q)$$

In other words, by replacing w_q in the OBC result with $w_q - \text{quant}(w_q)$, we get the general weight update formula for quantization:

GPTQ Design

GPTQ

same author as OBC

same algorithm

- Abritrary order
- Lazy Batch-Updates
- Cholesky Reformulation

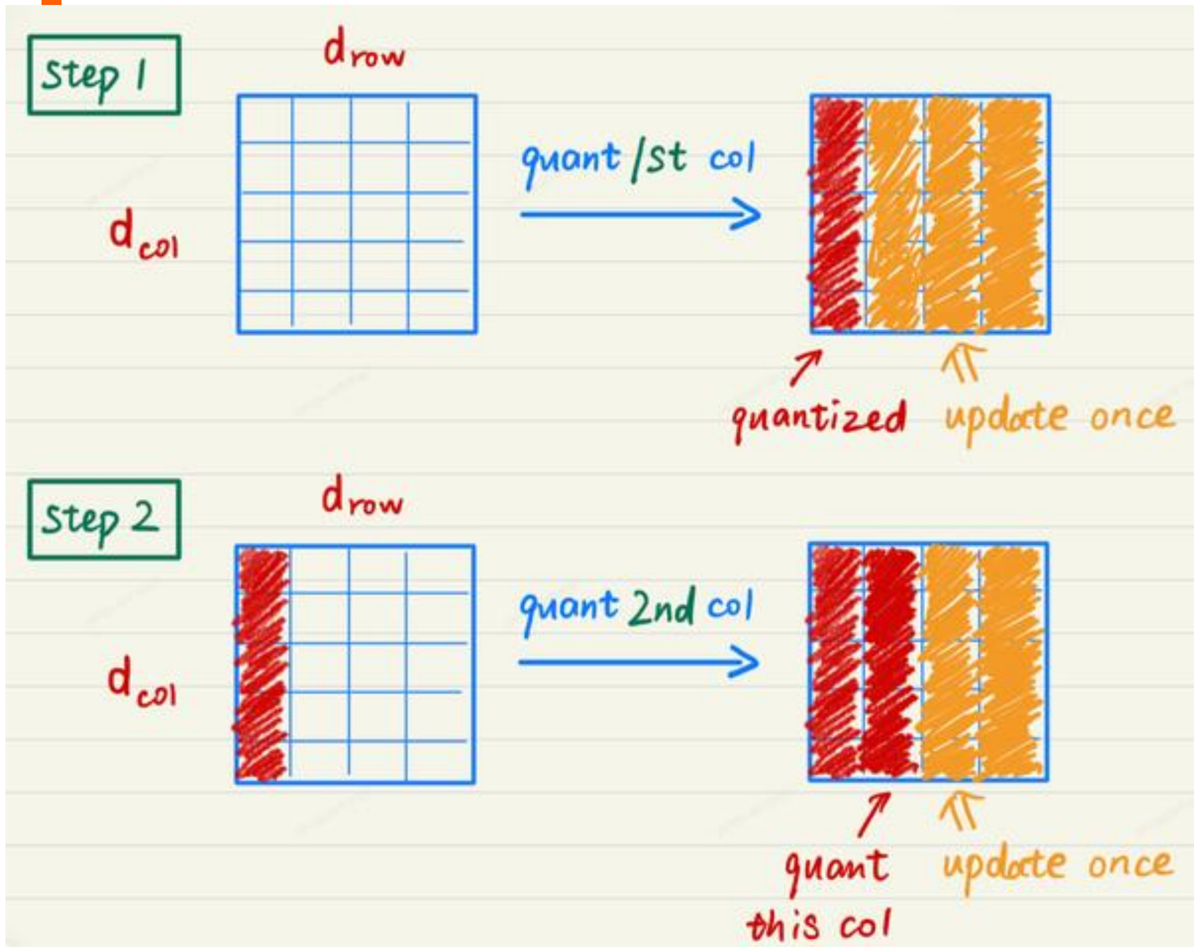
GPTQ

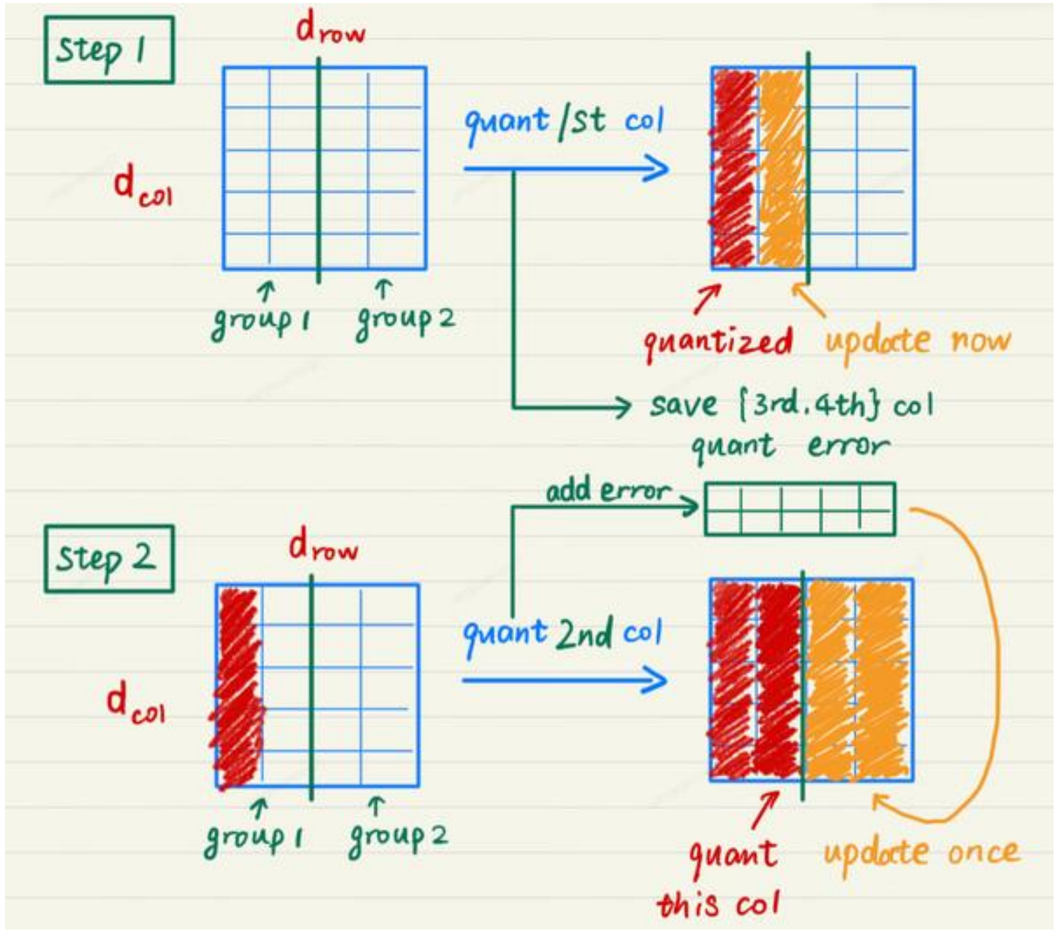
- Abritrary order
    - not significant result bewteen strict order or abritrary order
    - Eventually minimal
    - unquantized weights available to adjust and compensate for the error
- Lazy Batch-Updates
- Cholesky Reformulation

Content

- Abritrary order
- Lazy Batch-Updates
  - read + write the parameter matrix once per update
  - The quantization result of the parameter in column i is affected by the quantization of the previous i-1 columns, **but the quantization result of column i does not affect the quantization of the previous columns**.
  - divided into groups of 128 columns
  - When a column is quantized, its parameters are updated **immediately**, while later columns **record** the update and apply it later. Once the group is fully quantized, all subsequent parameters are updated together.
- Cholesky Reformulation

GPTQ


Step 1
d_{row}
d_{col}
quant 1st col
quantized
update once
Step 2
d_{row}
d_{col}
quant 2nd col
quant this col
update once

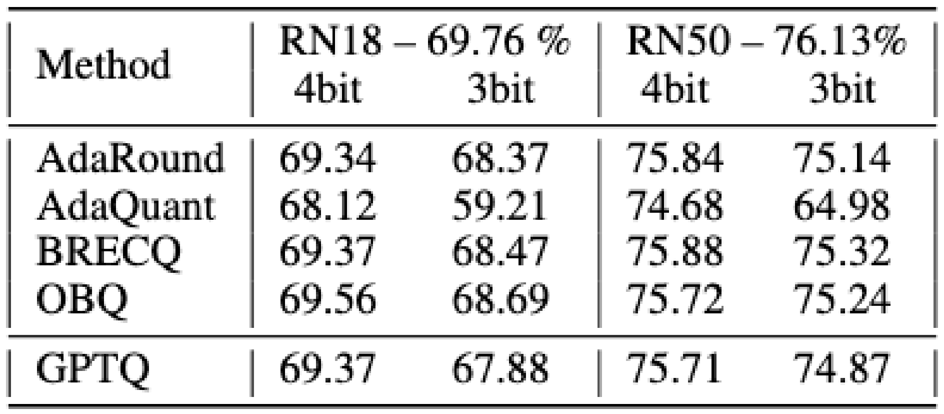

Step 1
d_{row}
d_{col}
group 1
group 2
quant 1st col
quantized
update now
save {3rd.4th} col quant error
add error
Step 2
d_{row}
d_{col}
group 1
group 2
quant 2nd col
quant this col
update once

GPTQ

- Abritrary order
- Lazy Batch-Updates
- Cholesky Reformulation
    - block updates can cause H^{-1} indefinite, leading to incorrect weight updates.
    - Add a small constant λ (1% of the average diagonal value) to the diagonal of H
    - Precompute necessary rows of $\mathrm{H}_{\mathbf{Fq}}^{-1}$ using Cholesky kernels

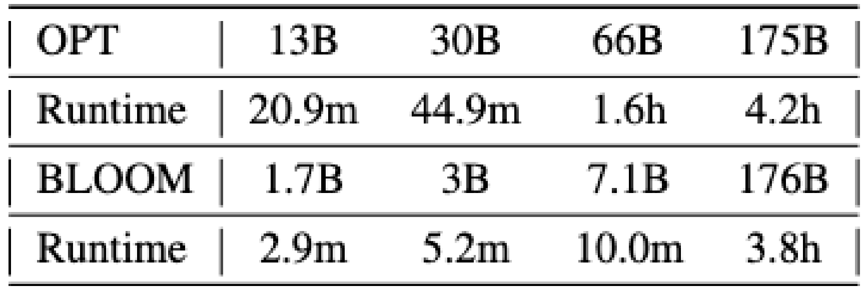
Evaluation and Results

Content



Method	RN18 – 69.76 %		RN50 – 76.13%	
	4bit	3bit	4bit	3bit
AdaRound	69.34	68.37	75.84	75.14
AdaQuant	68.12	59.21	74.68	64.98
BRECQ	69.37	68.47	75.88	75.32
OBQ	69.56	68.69	75.72	75.24
GPTQ	69.37	67.88	75.71	74.87

Table 1: Comparison with state-of-the-art post-training methods for vision models.



OPT	13B	30B	66B	175B
Runtime	20.9m	44.9m	1.6h	4.2h
BLOOM	1.7B	3B	7.1B	176B
Runtime	2.9m	5.2m	10.0m	3.8h

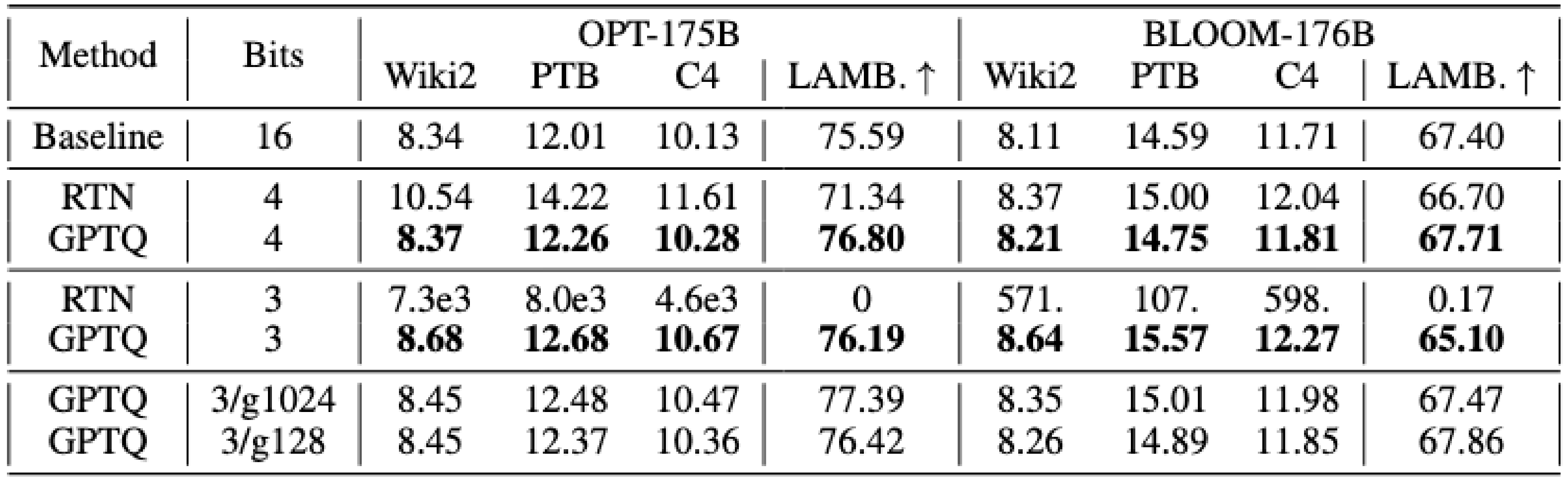
Table 2: GPTQ runtime for full quantization of the 4 largest OPT and BLOOM models.

At 4-bit good performance

At 3-bit competitive performance

GPTQ quantizes models ranging from 1.7B to 175B parameters in a matter of minutes to hours.

Content



Method	Bits	OPT-175B				BLOOM-176B			
		Wiki2	PTB	C4	LAMB. ↑	Wiki2	PTB	C4	LAMB. ↑
Baseline	16	8.34	12.01	10.13	75.59	8.11	14.59	11.71	67.40
RTN	4	10.54	14.22	11.61	71.34	8.37	15.00	12.04	66.70
GPTQ	4	**8.37**	**12.26**	**10.28**	**76.80**	**8.21**	**14.75**	**11.81**	**67.71**
RTN	3	7.3e3	8.0e3	4.6e3	0	571.	107.	598.	0.17
GPTQ	3	**8.68**	**12.68**	**10.67**	**76.19**	**8.64**	**15.57**	**12.27**	**65.10**
GPTQ	3/g1024	8.45	12.48	10.47	77.39	8.35	15.01	11.98	67.47
GPTQ	3/g128	8.45	12.37	10.36	76.42	8.26	14.89	11.85	67.86

Table 5: Results summary for OPT-175B and BLOOM-176B. "g1024" and "g128" denote results with groupings of size 1024 and 128, respectively.

The baseline model is the unquantized version with **full 16-bit precision**.

GPTQ is highly effective at both 4-bit and 3-bit precision, particularly with fine-grained grouping in perplexity

Even at 3-bit, GPTQ with grouping (especially g128) achieves results very close to the baseline.

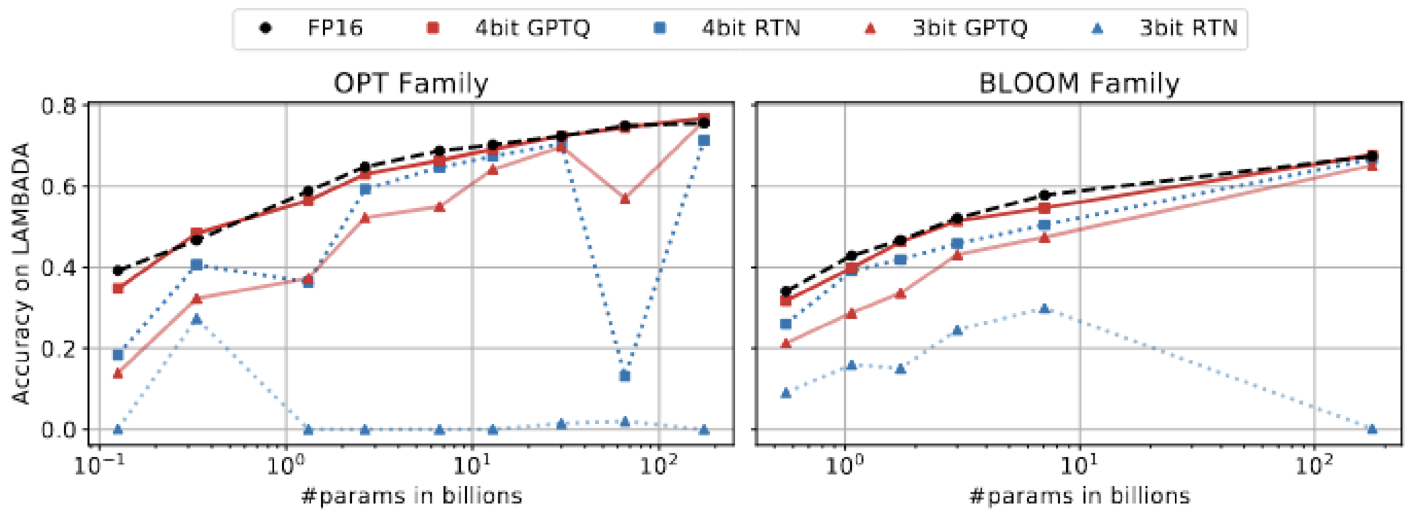
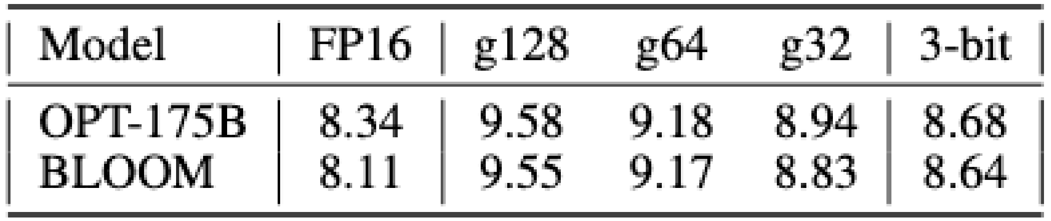


Figure 3: The accuracy of OPT and BLOOM models post-GPTQ, measured on LAMBADA.



Model	FP16	g128	g64	g32	3-bit
OPT-175B	8.34	9.58	9.18	8.94	8.68
BLOOM	8.11	9.55	9.17	8.83	8.64

Table 7: 2-bit GPTQ quantization results with varying group-sizes; perplexity on WikiText2.

Figure3

GPTQ performs close to the FP16 baseline, even at 4-bit precision, showing its effectiveness.

Table7

Group Sizes (g128, g64, g32): As the group size decreases (more granular grouping), perplexity also decreases. This means better model performance at the cost of smaller group sizes.

Analysis and Future Directions

Strengths

- **Efficiency & Scalability**:
  Fast quantization for large models (e.g., 175B parameters) within minutes to hours.
- **Competitive Accuracy**:
  Maintains near-baseline performance at 4-bit and strong results at 3-bit, outperforming RTN.
  The use of **grouping** (e.g., g128, g1024) further improves accuracy
- **Versatility**:
  Works well for both vision and language models, compatible with different quantization grids.
- **Low Memory Overhead**:
  Enables fitting large models on a single GPU, significantly reducing memory usage.
  Allows very large models (e.g., OPT-175B) to fit on a single GPU

Limitations

- **Accuracy Loss at Lower Bits**:

  Accuracy drops at 2-bit precision without fine-grained grouping
- **Sensitivity to Group Size**:

  Smaller group sizes improve accuracy but increase computation time.
- **No Activation Quantization**:

  Currently only quantizes weights, missing further compression from activations.

  SmoothQuantization

Future Work

Advanced Grouping Strategies:

1. Investigating **adaptive or dynamic grouping** based on model layers or weights' importance
2. For instance, different parts of the model could benefit from varying group sizes, rather than applying a fixed size across all layers.

Improved Handling of Numerical Instabilities:

1. Developing more robust techniques to mitigate numerical inaccuracies during block-wise updates.
2. Especially for extremely large models, could improve the method's accuracy when applied to 2-bit or lower precision quantization

Application to a Broader Range of Tasks:

1. Explore its effectiveness on a wider range of **tasks and architectures**
2. e.g., transformers in reinforcement learning or generative models in image synthesis

Hardware-Specific Optimizations:

GPTQ could be optimized for **custom hardware** (e.g., FPGAs or specialized AI accelerators) to maximize its benefits in terms of speed and efficiency

ILLINOIS