UC San Diego

GPTQ

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers. ICLR 2023

Presenter - Shivam Yadav

Contents

Why Quantization?

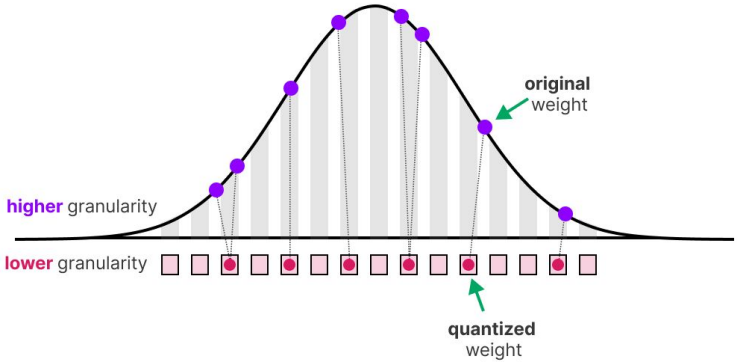
- The Deployment Problem: GPT-3's 175 billion parameters demand **≈ 326 GB of FP16 memory**, requiring multi-GPU clusters just to run inference.
- Can we achieve **high compression and high accuracy** simultaneously, without any retraining?
- Quantization Aware Training (QAT): Quantization **during** training/fine-tuning.
- Post-Training Quantization (PTQ): Quantization **after** training.

Why Quantization?

- The Deployment Problem: GPT-3's 175 billion parameters demand **≈ 326 GB of FP16 memory**, requiring multi-GPU clusters just to run inference.
- Can we achieve **high compression and high accuracy** simultaneously, without any retraining?
- Quantization Aware Training (QAT): Quantization **during** training/fine-tuning.
- Post-Training Quantization (PTQ): Quantization **after** training.
- Why PTQ?
- Training is much more sensitive than inference.
  - Inference: y = WX
  - Training: Backprop, Gradient accumulation, Optimizer state updates, all of which are sensitive.
  - Training: Use a mix quantization, usually FP32 for gradients, and FP16 for weights etc, e.g., Deepseek-v3.

What is Quantization?

- Convert weights: FP16 / FP32 → INT4 / INT8
- Benefits: Lower memory, Faster inference
- Tradeoff: Accuracy vs Compression



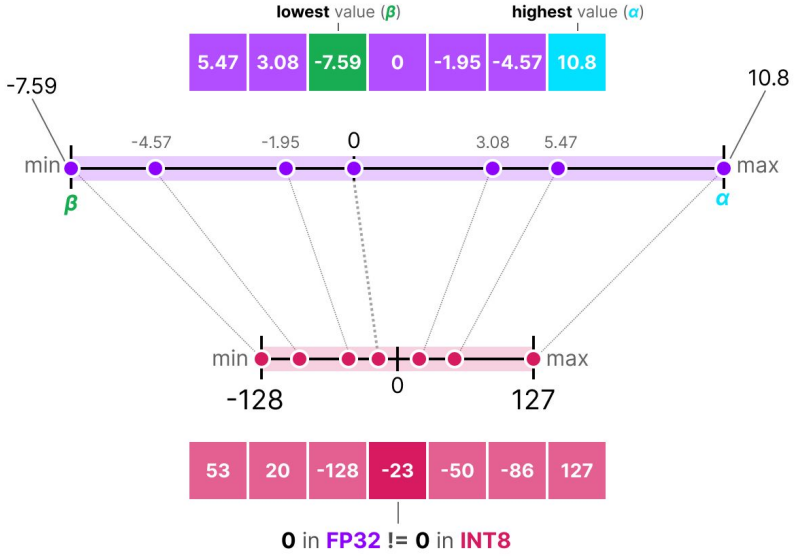


Image Source: Maarten Grootendorst, Demystifying the Compression of Large Language Models

GPTQ: Method Overview

- Layer-Wise Quantization: $argmin_{\hat{W}} = ||WX - \hat{W}X||_2^2$

  X: Layer Inputs/Activations

  s: scaling factor

$$\hat{W} \approx s.W_q$$

- Preserve layer outputs, not weights
- No retraining
- Few calibration samples
- Ultra-low bitwidth (4 bits)

Why Second-Order Info?

During this layer-wise quantization process, it first converts the layer's weights into the **inverse-Hessian**.

- **Gradient** → tells the *direction of steepest increase (slope)*
- **Hessian** → tells *how that direction itself is changing (curvature)*
- **Why?**

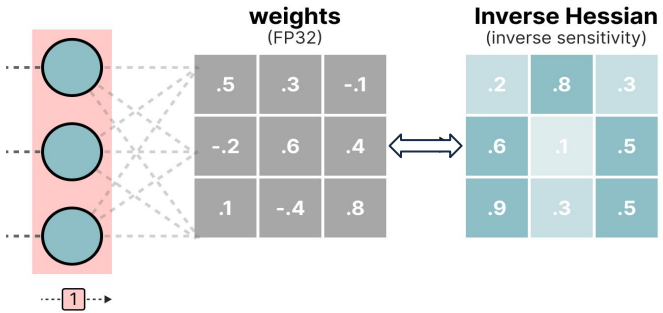
Why Second-Order Info?

During this layer-wise quantization process, it first converts the layer's weights into the inverse-**Hessian**.

- **Gradient** → the *direction of steepest increase (slope)*
- **Hessian** → *how that direction itself is changing (curvature)*
- **Why?** *The Hessian captures how sensitive the output is to each individual weight, enabling far more accurate rounding decisions.* It essentially demonstrates the (*inverse*) **importance of each weight** in a layer.

$$\text{argmin}_{\widehat{\mathbf{W}}} ||\mathbf{WX} - \widehat{\mathbf{W}}\mathbf{X}||_2^2.$$

$$H = 2XX^T$$



$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

GPTQ: Method Overview

Optimal Brain Quantization:

Greedy Sequential Quantisation

Per-row operations

- For each weight w:
  - Select the weight that results in minimum relative error.
  - Quantize the current weight to its nearest grid point.
  - Computes the induced output error.
  - Corrects remaining weights via the inverse Hessian.
  - Update Inverse Hessian.

$$H = 2X_F X_F^T$$

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

$$\boldsymbol{\delta}_F = -\frac{w_q - \text{quant}(w_q)}{[\mathbf{H}_F^{-1}]_{qq}} \cdot (\mathbf{H}_F^{-1})_{:,q}.$$

$$\mathbf{H}_{-q}^{-1} = \left(\mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}_{:,q}^{-1} \mathbf{H}_{q,:}^{-1}\right)_{-p}$$

Computational Complexity OBQ

Step	Per-Step Cost	Per-Row Cost	Full Matrix
Compute Hessian	$O\left(c^2 n\right)$	—	$O\left(c^2 n\right)$
Compute Inverse Hessian	$O\left(c^3\right)$	—	$O\left(c^3\right)$
Quantization scores	$O\left(c\right)$	$O\left(c^2\right)$	$O\left(rc^2\right)$
Quantize one weight	$O\left(1\right)$	$O\left(c\right)$	$O\left(rc\right)$
Compensation update	$O\left(c\right)$	$O\left(c^2\right)$	$O\left(rc^2\right)$
Inverse Hessian update	$O\left(c^2\right)$	$O\left(c^3\right)$	$O\left(rc^3\right)$

$$\mathbf{H}_{-p}^{-1} = \left(\mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{pp}} \mathbf{H}_{:,p}^{-1} \mathbf{H}_{p,:}^{-1}\right)_{-p}$$

$$O\left(rc^3\right)$$

GPTQ: Method Overview

Optimal Brain Quantization:

- Even if each row is quantized in parallel:
  - there exist different orders in which the weights are quantized for each row.
  - different updated Hessian matrices (of free weights) for each row.
  - Too Slow.
  - $O\left(rc^3\right)$

$$H = 2X_F X_F^T$$

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

$$\boldsymbol{\delta}_F = -\frac{w_q - \text{quant}(w_q)}{[\mathbf{H}_F^{-1}]_{qq}} \cdot (\mathbf{H}_F^{-1})_{:,q}.$$

$$\mathbf{H}_{-q}^{-1} = \left(\mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{qq}} \mathbf{H}_{:,q}^{-1} \mathbf{H}_{q,:}^{-1}\right)_{-p}$$

GPTQ: Arbitrary Order Insight

- Quantize in **arbitrary** order.
- You might expect:
  - The order of quantization to be extremely important.
- But in practice, especially for **bigger LLMs**:
  - Even random or simple orders work well.
  - Quantize all rows sequentially.

- **Why?**

GPTQ: Arbitrary Order Insight

- Quantize in **arbitrary** order.
- You might expect:
  - The order of quantization to be extremely important
- But in practice, especially for **bigger LLMs**:
  - Even random or simple orders work well
  - Thus, quantize sequentially

- **Why?**
- **Most likely**, in any arbitrary ordering:
  - Some high-error weights will occur early => their error gets compensated
  - Some high-error weights will occur late => their error stays local

- **Error propagation effects balance out.**

GPTQ: Arbitrary Order Insight

- Quantize in **arbitrary** order.
- You might expect:
  - The order of quantization to be extremely important
- But in practice, especially for **bigger LLMs**:
  - Even random or simple orders work well
  - Thus, quantize sequentially

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

$$\mathbf{H}_{-p}^{-1} = \left(\mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{pp}} \mathbf{H}_{:,p}^{-1} \mathbf{H}_{p,:}^{-1}\right)_{-p}$$

- **Why?**
- **Most likely**, in any arbitrary ordering:
  - Some high-error weights will occur early => their error gets compensated
  - Some high-error weights will occur late => their error stays local

$$O\left(rc^3\right)$$

$$\Downarrow$$

$$O\left(\max\left\{c^3,\ rc^2\right\}\right)$$

- **Error propagation effects balance out.**

Computational Complexity OBQ->GPTQ

UC San Diego

Step	Per-Step Cost	Per-Row Cost	Full Matrix
Compute Hessian	$O\left(c^2 n\right)$	—	$O\left(c^2 n\right)$
Compute Inverse Hessian	$O\left(c^3\right)$	—	$O\left(c^3\right)$
Quantization scores	$O\left(c\right)$	$O\left(c^2\right)$	$O\left(rc^2\right)$
Quantize one weight	$O\left(1\right)$	$O\left(c\right)$	$O\left(rc\right)$
Compensation update	$O\left(c\right)$	$O\left(c^2\right)$	$O\left(rc^2\right)$
Inverse Hessian update	$O\left(c^2\right)$	$O\left(c^3\right)$	~~$O\left(rc^3\right)$~~ $O\left(c^3\right)$

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

$$\mathbf{H}_{-p}^{-1} = \left(\mathbf{H}^{-1} - \frac{1}{[\mathbf{H}^{-1}]_{pp}} \mathbf{H}_{:,p}^{-1} \mathbf{H}_{p,:}^{-1}\right)_{-p}$$

$$O\left(rc^3\right)$$

$$\Downarrow$$

$$O\left(\max\left\{c^3,\ rc^2\right\}\right)$$

GPTQ: Practical Optimisations

- **Lazy Batch-Updates:** Process weights in blocks (e.g. 128 columns) to maximise GPU parallelism.
- **Dampening:** Regularise the Hessian as $\mathbf{H} \leftarrow \mathbf{H} + \lambda \mathbf{I}$ to handle near-singular matrices.
  - λ**:** 1% of the average diagonal value.
- **Cholesky Decomposition:** Replace explicit inversion with a Cholesky factorisation of H for improved numerical stability.
  - As a bonus, Cholesky kernel also yields further speedup.
  - No change in theoretical complexity.

GPTQ: Full Algorithm

Algorithm 1 Quantize $\mathbf{W}$ given inverse Hessian $\mathbf{H}^{-1} = (2\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}$ and blocksize B.

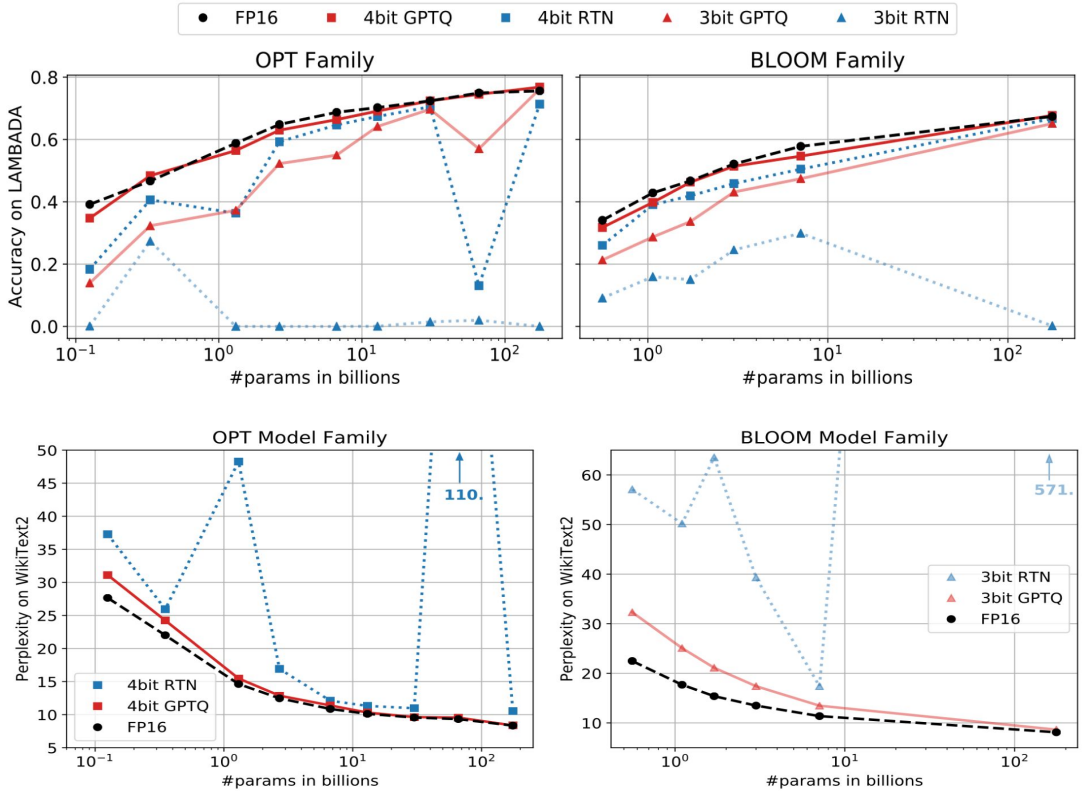
$\mathbf{Q} \leftarrow \mathbf{0}_{d_{\text{row}} \times d_{\text{col}}}$ *// quantized output*
$\mathbf{E} \leftarrow \mathbf{0}_{d_{\text{row}} \times B}$ *// block quantization errors*
$\mathbf{H}^{-1} \leftarrow \text{Cholesky}(\mathbf{H}^{-1})^\top$ *// Hessian inverse information*
for $i = 0, B, 2B, \ldots$ **do**
  for $j = i, \ldots, i + B - 1$ **do**
    $\mathbf{Q}_{:,j} \leftarrow \text{quant}(\mathbf{W}_{:,j})$ *// quantize column*
    $\mathbf{E}_{:,j-i} \leftarrow (\mathbf{W}_{:,j} - \mathbf{Q}_{:,j}) \,/\, [\mathbf{H}^{-1}]_{jj}$ *// quantization error*
    $\mathbf{W}_{:,j:(i+B)} \leftarrow \mathbf{W}_{:,j:(i+B)} - \mathbf{E}_{:,j-i} \cdot \mathbf{H}^{-1}_{j,j:(i+B)}$ *// update weights in block*
  end for
  $\mathbf{W}_{:,(i+B):} \leftarrow \mathbf{W}_{:,(i+B):} - \mathbf{E} \cdot \mathbf{H}^{-1}_{i:(i+B),(i+B):}$ *// update all remaining weights*
end for

Experiments & Results

Calibration data:

- 128 random 2048 token segments.
- Represents generic text data.

Format	Estimated Size	Hardware Needs
FP16 (Original)	~350 GB	8x A100 (80GB)
4-bit GPTQ	~90 GB	2x A6000 (48GB) or 2x A100
3-bit GPTQ	~63 GB	1x A100 (80GB)

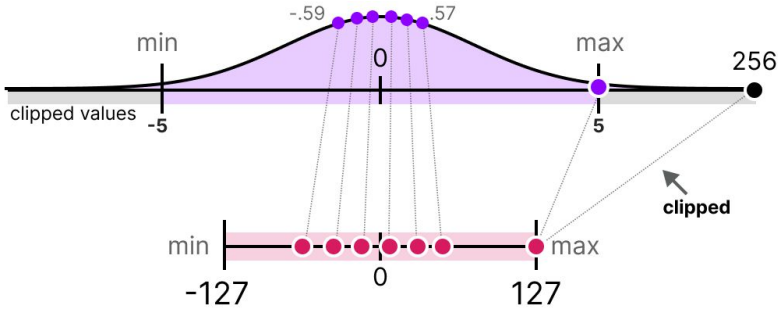


Limitations & Broader Landscape

- Quantization of trillion-parameter models.
- Activations remain in FP16, outliers in activations still cause significant error => AWQ
  - *Lin, J., Tang, J., Tang, H., Yang, S., Chen, W. M., Wang, W. C., & Han, S. (2024). AWQ: Activation-aware weight quantization for on-device llm compression and acceleration. Proceedings of machine learning and systems, 6, 87-100.*
- Quantisation quality/sensitive w.r.t calibration data size, domain, and sequence length. What is the minimum sufficient statistic?
  - *Kurtic, E., Marques, A.N., Pandit, S., Kurtz, M. and Alistarh, D., 2025, July. "Give Me BF16 or Give Me Death"? Accuracy-Performance Trade-Offs in LLM Quantization. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 26872-26886).*
- KV Cache: Extend quantization via error correction to KV cache quantisation.
  - *Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., & Gholami, A. (2025). Kvquant: Towards 10 million context length llm inference with kv cache quantization. Advances in Neural Information Processing Systems, 37, 1270-1303.*

Appendix: Quantization

- Convert weights: FP16 / FP32 → INT4 / INT8
- Benefits: Lower memory, Faster inference
- Tradeoff: Accuracy vs Compression



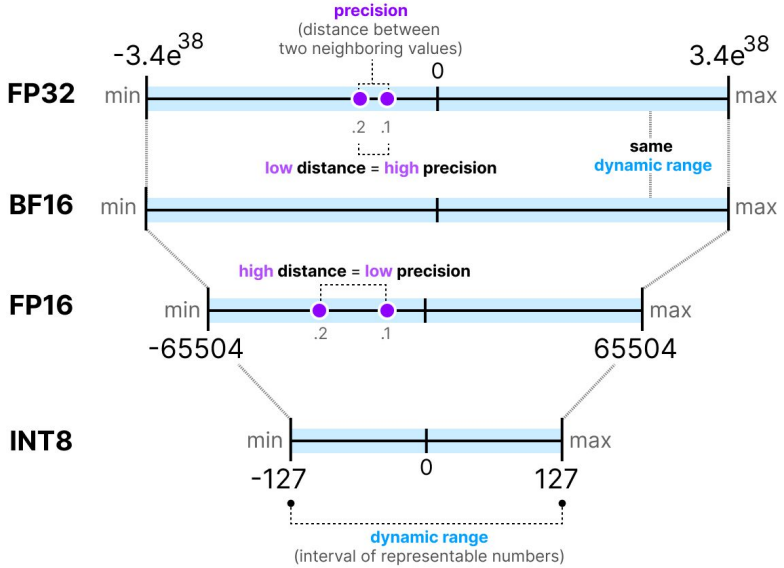


Image Source: Maarten Grootendorst, Demystifying the Compression of Large Language Models

Appendix: Quantization

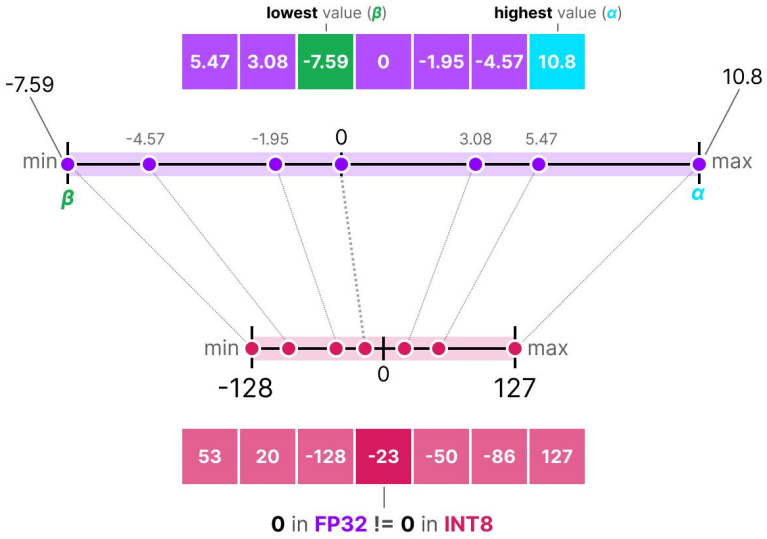
- s: scaling factor
- z: zeropoint
- x_q: quantized x
- x_hat: de-quantized x

$$s = \frac{q_{max} - q_{min}}{\alpha - \beta}$$

$$z = round(-s.\beta) + q_{min}$$

$$x_q = round(s.x + z)$$

$$\hat{x} = \frac{x_q - z}{s}$$



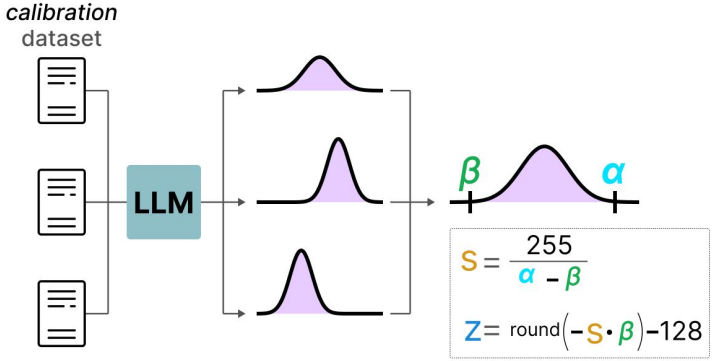
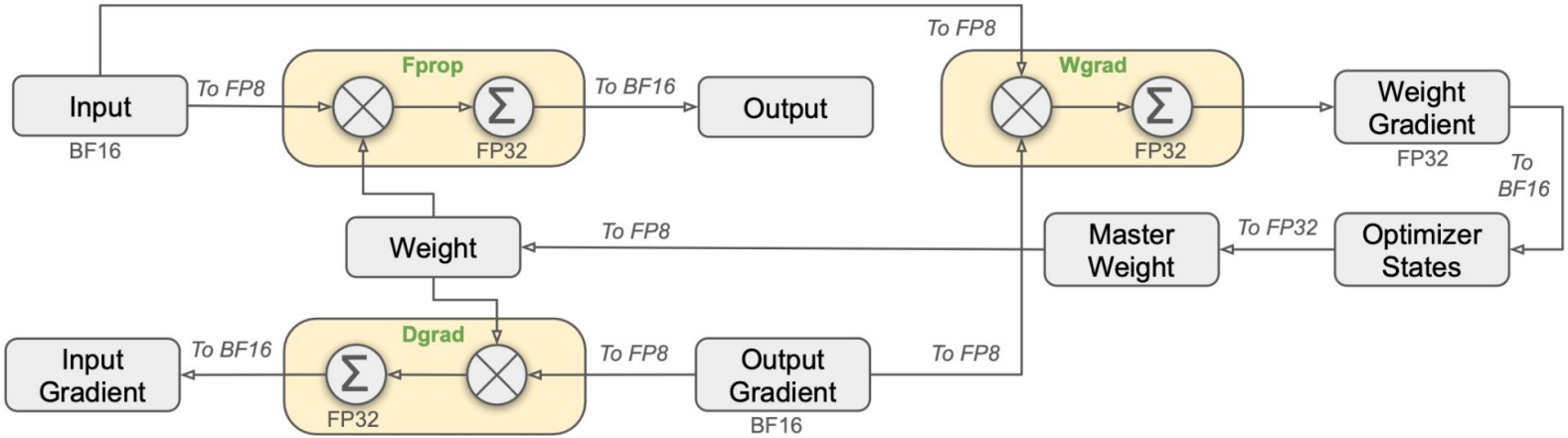


Image Source: Maarten Grootendorst, Demystifying the Compression of Large Language Models

Appendix: Quantization Aware Training (QAT)

Deepseek-v3



Liu, Aixin, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao et al. "Deepseek-v3 Technical Report." *arXiv preprint arXiv:2412.19437* (2024).

UC San Diego

Thank you!