### **Main Ideas**

GPTQ didn’t emerge from nowhere; its principles stem from another quantization method, OBQ, which can be seen as an accelerated version of it. OBQ itself is actually a modification of OBS (a classic pruning method), and OBS originates from OBD (a pruning method proposed by LeCun in 1990). It's a method with a long history that has evolved over time.

### **Layer Quantization**

GPTQ focuses on a single-layer perspective, aiming to find a quantized weight www that minimizes the difference in output between the new and original weights.



### **OBQ (Optimal Brain Quantization)**

The core principle of GPTQ comes from OBQ, whose idea mainly derives from OBS (Optimal Brain Surgeon). In OBS, the authors sought a method to remove a weight, denoted as wq w\_q​, with minimal increase in overall error. They also calculated a compensation δq \delta\_q to apply to the remaining weights to minimize the impact of this removal.

### **Key Concepts**

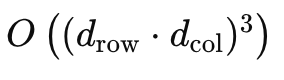
* **Quantization**: Model quantization involves converting floating-point numbers to fixed-point, reducing memory access to lower model size, inference latency, and improving performance with minimal error.
* **Calibration Set**: A small set of training data used to calibrate and evaluate quantization errors during the quantization process.
* **PTQ**: Post-Training Quantization, where no weight training is involved during quantization.
* **QAT**: Quantization-Aware Training, where gradients are backpropagated to update weights during quantization.
* **OBD & OBS**: Model pruning methods that use the second derivative of the loss to determine which weights to remove while compensating for the pruned weights to achieve better pruning results.
* **OBQ**: Extends OBS to quantization and introduces row-wise operations.
* **GPTQ**: Uses a similar order with row-parallel computation, batch updates, and grouped quantization to accelerate the quantization process.

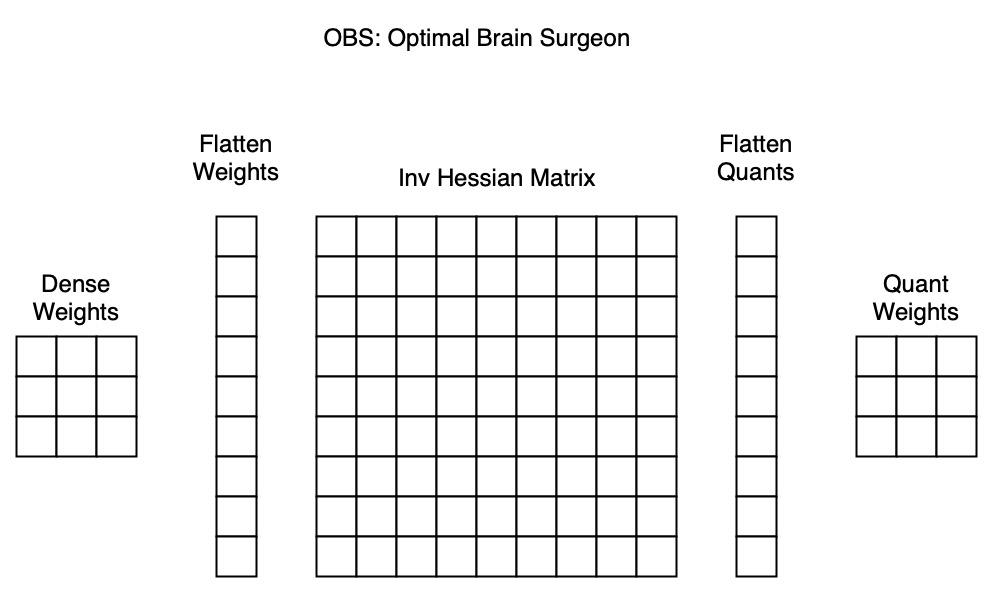
### **Industrial Application**

Unlike the academic pursuit of achieving SOTA (state-of-the-art), the industry prioritizes ROI. An important topic is how to transfer large research models to industrial deployment with minimal effort while maintaining accuracy and maximizing resource efficiency.

### **OBS**

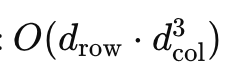
Directly unroll the weights and calculate the corresponding Hessian matrix, then quantize in order. The time complexity is: O\left((d\_{\text{row}} \cdot d\_{\text{col}})^3\right)

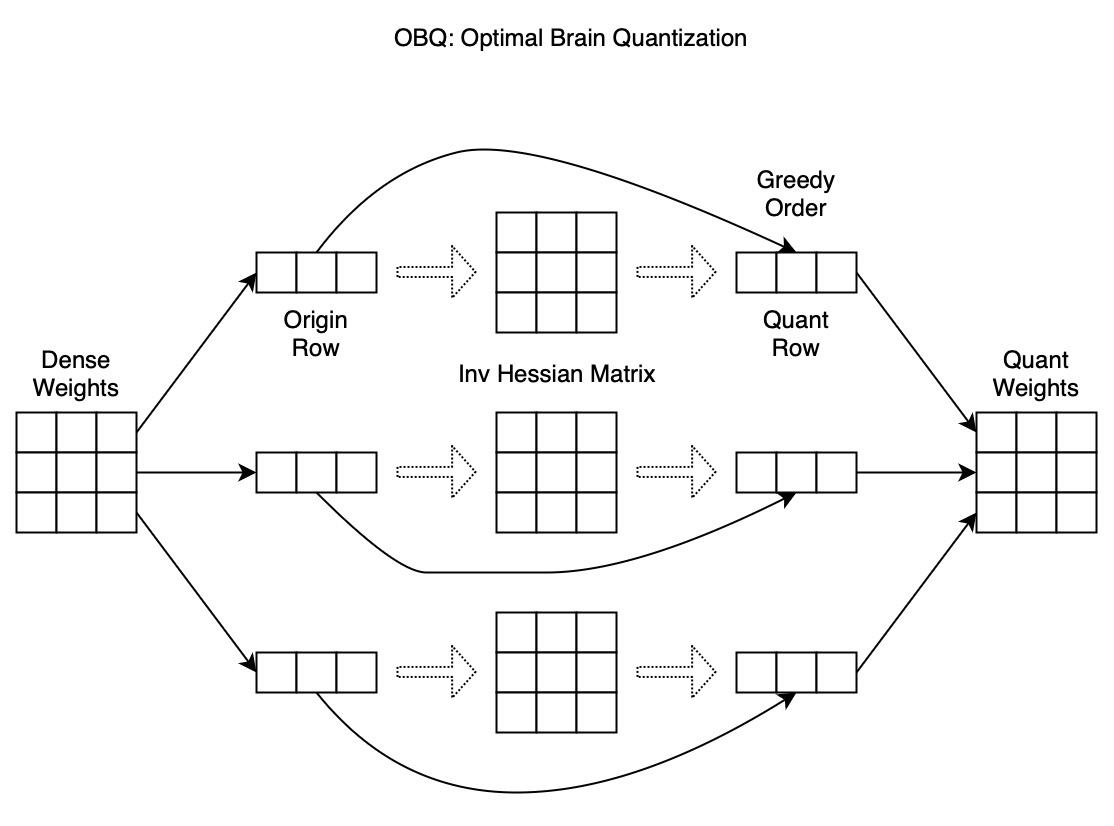




### **OBQ**

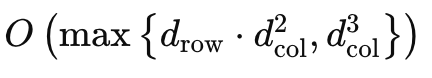
Weights are calculated row by row with a greedy algorithm, quantizing each time to minimize quantization error. The time complexity is: O(d\_{\text{row}} \cdot d\_{\text{col}}^3)O(drow​⋅dcol3​)



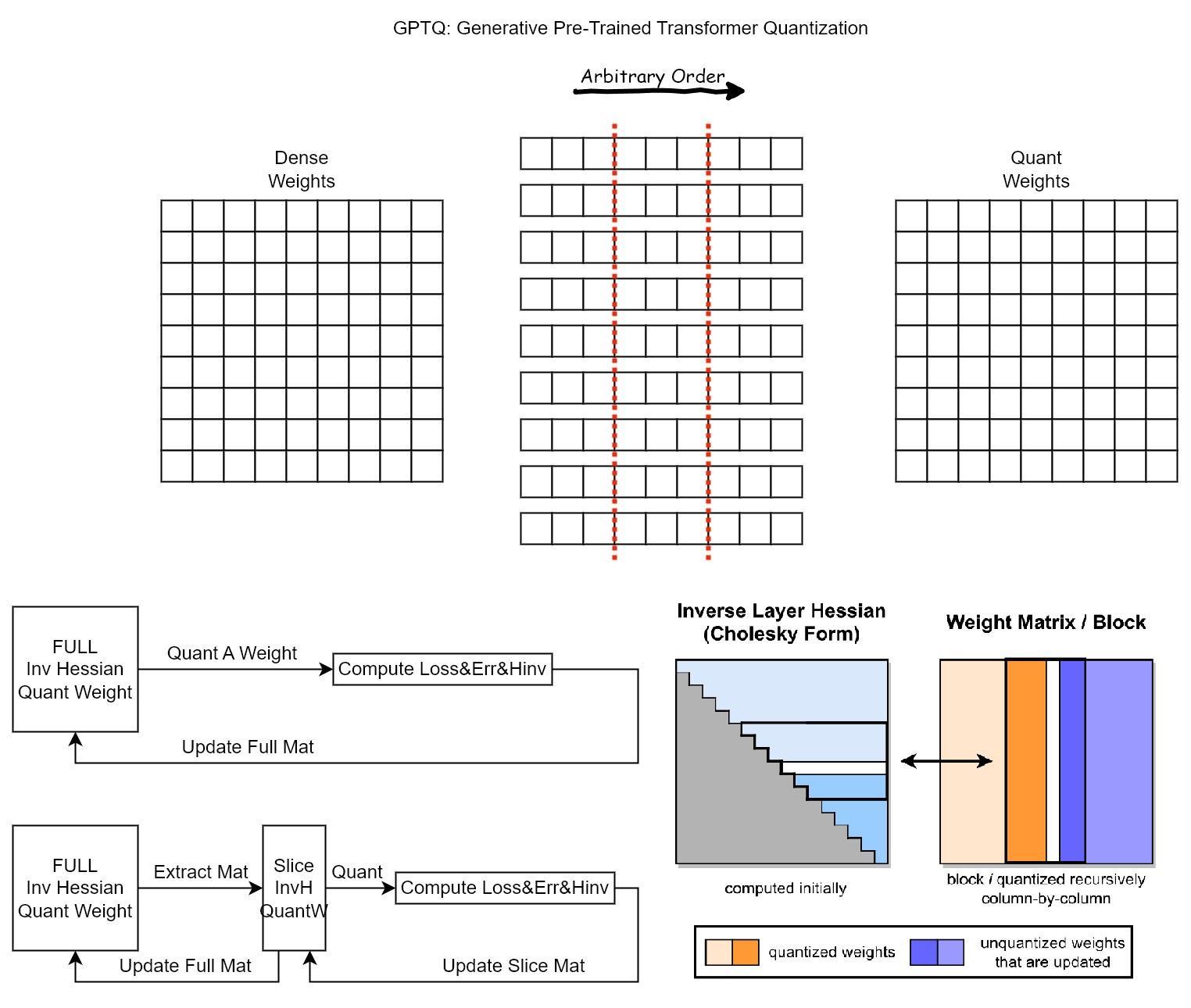


### **GPTQ**

Uses the same order, with row-wise parallel computation, batch updates (BatchUpdate), and grouped quantization. The time complexity is: O\left(\max \left\{d\_{\text{row}} \cdot d\_{\text{col}}^2, d\_{\text{col}}^3\right\}\right)O(max{drow​⋅dcol2​,dcol3​})



* **Fixed Order**: Using a greedy algorithm to quantize weights with the smallest error each time performs well, but in large models, fixed ordering may perform better than this approach. Therefore, we use fixed order quantization.
* **Parallel Computation**: Since quantized weights do not interfere across rows, we can quantize multiple rows in parallel to accelerate computation.
* **Batch Update**: Updating weights every time a parameter is quantized results in most of the time spent on memory access rather than fully utilizing the GPU’s computational power. To address this, we perform updates in batches, updating the global matrix only after batch completion.
* **Numerical Stability**: Using Cholesky reformulation to enhance numerical stability in calculating the inverse of the Hessian matrix.
* **Grouped Quantization**: Instead of quantizing the entire matrix with the same scale and zero-point, using a smaller group size to calculate specific quantization parameters within groups allows handling global outliers better, yielding improved quantization results.



The models we currently use likely have significant room for optimization in both size and performance. GPTQ approaches this from the angle of quantization, but combining multiple techniques could be very effective. Currently, while 170-billion-parameter models are large, our methods might already fit them on consumer-grade GPUs like the RTX 4080 if we could find the right engineering combination. If we effectively integrate techniques like distillation, pruning, and quantization, this could be achievable.

GPTQ Code Flowchart:

