Bottleneck Analysis of Large-Model Inference Efficiency Based on the Roofline Model

HAN Yuxuan

2026-07-16

ChinaRxiv

Abstract

[Objective] To clarify the stability boundaries of Roofline bottleneck diagnosis under different hardware configurations, workload parameters, and quantization precisions, and to answer the question of "within what range diagnostic conclusions are credible," which has not yet been systematically quantified. [Methods] A full-factorial experiment was designed for four scales of the OPT series (125M-6.7B): 2 GPUs × 4 batch sizes × 4 sequence lengths × 3 quantization precisions, totaling 120 configurations, with layer-wise bottleneck classification data collected. Three metrics—Layer Stability Score (LSS), factor importance decomposition, and cross-GPU consistency—were defined to construct a statistical stability analysis framework for bottleneck classification. [Results] In the Decode stage, 100% of layers were memory bottlenecks and completely stable (LSS=1.000). In the Prefill stage, stability exceeded 0.98, with bottleneck flips occurring only under the OPT-6.7B+INT4 condition (6/16 layers). Quantization precision explained approximately 49% of the classification variance, while GPU hardware contributed less than 1%. Bottleneck classifications between A6000 and A100 were fully consistent (100% cross-consistency). [Limitations] Only the OPT dense architecture and two NVIDIA GPUs (A6000/A100) were tested; the extrapolability of the conclusions to emerging architectures such as MoE and Mamba, as well as GPU combinations with larger performance gaps, remains unverified. The binary nature of bottleneck classification also loses continuous information about arithmetic intensity and distance to the ridge point. [Conclusions] Roofline bottleneck diagnosis exhibits high cross-configuration robustness on dense transformer architectures. The proposed stability analysis framework can upgrade diagnosis from a single-shot judgment to a system-level assessment with quantified credibility.

Full Text

Bottleneck Analysis of Large-Model Inference Efficiency Based on the Roofline Model

HAN Yuxuan*

School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou, Jiangsu 213159

Corresponding author: HAN Yuxuan, *E-mail*: 2300770102@smail.cczu.edu.cn

Abstract:

[Purpose] To clarify the stability boundaries of Roofline bottleneck diagnosis under different hardware configurations, workload parameters, and quantization precisions, and to answer the as-yet unsystematically quantified question of "over what range are diagnostic conclusions reliable."

[Methods] A full-factorial experiment was designed for four scales of the OPT

series (125M–6.7B): 2 GPUs × 4 batch-size levels × 4 sequence-length levels × 3 quantization precisions, for a total of 120 configurations. Layer-wise bottleneck-classification data were collected. Three metrics—layer stability score (LSS), factor-importance decomposition, and cross-GPU consistency—were defined to construct a statistical stability analysis framework for bottleneck classification.

[Results] In the Decode stage, 100% of layers were memory bottlenecked and fully stable (LSS = 1.000). In the Prefill stage, stability was >0.98, with bottleneck reversal occurring only under the OPT-6.7B + INT4 condition (6/16 layers). Quantization precision explained approximately 49% of the classification variance, while GPU hardware contributed less than 1%. Bottleneck classification between the A6000 and A100 was completely consistent (100% cross-consistency).

[Limitations] Only the OPT dense architecture and two NVIDIA GPUs (A6000/A100) were tested. The extrapolability of the conclusions to emerging architectures such as MoE and Mamba, as well as to GPU combinations with larger performance gaps, remains unverified. The binary nature of bottleneck classification also loses continuous information about arithmetic intensity and distance to the ridge point.

[Conclusion] Roofline bottleneck diagnosis exhibits high cross-configuration robustness on dense transformer architectures. The proposed stability analysis framework can upgrade diagnosis from a single-instance judgment to a systematic evaluation with quantified confidence.

Keywords: large language model; inference efficiency; Roofline model; bottleneck analysis; statistical stability

Classification number: TP391

Large Language Model Inference Efficiency Bottleneck Analysis Based on the Roofline Model: Systematic Experimental Design and Bottleneck Classification Stability

HAN Yuxuan*

School of Computer Science and Artificial Intelligence, Changzhou University,

Changzhou 213159, China

Corresponding author*: HAN Yuxuan, E-mail: 2300770102@smail.cczu.edu.cn

Abstract:

[Objective] To systematically quantify the stability of Roofline-based bottleneck diagnosis across varying hardware configurations, batch sizes, sequence lengths,

and quantization precisions.

[Methods] A full factorial experiment was conducted on four OPT models (125M–6.7B) spanning two GPUs, four batch sizes, four sequence lengths, and three quantization precisions (120 configurations). Per-layer bottleneck classifications were collected; Layer Stability Score (LSS), factor importance decomposition, and GPU cross-consistency were defined as stability metrics.

[Results] The Decode phase is entirely memory-bound with perfect stability (LSS=1.000) across all configurations. Prefill phase stability exceeds 0.98; class transitions occur only under OPT-6.7B with INT4 quantization. Quantization precision accounts for ~49% of variance; GPU contribution is <1%.

[Limitations] Only OPT dense architectures and two NVIDIA GPUs were tested; generalizability to MoE, Mamba, and other architectures remains unverified. The binary classification discards continuous distance-to-ridge-point information.

[Conclusions] Roofline-based diagnosis is highly robust for dense transformers. The proposed stability framework enables quantified confidence assessment of diagnostic conclusions.

Keywords: large language model; inference efficiency; Roofline model; bottleneck analysis; statistical stability

1 Introduction

1.1 Research Motivation

The cost of deploying inference for large language models is one of the core obstacles limiting their large-scale application. A model with 7B parameters requires about 14 GB of GPU memory (FP16), while a 70B model requires multi-card parallelism. In real products, the latency and throughput of a single inference run often directly determine whether the service can be launched. Yuan et al. (2024), in their survey, classify existing LLM inference optimization methods into four categories: model compression (quantization, pruning, knowledge distillation), algorithmic optimization (FlashAttention, speculative decoding), system optimization (operator fusion, memory management), and hardware optimization (dedicated accelerators).

Each of these four classes of methods is effective, but which one should be chosen depends on a prior judgment: whether the current system bottleneck is computation or memory bandwidth. If the bottleneck is memory, compute optimization yields little benefit, and vice versa. However, the robustness of bottleneck diagnosis—whether the diagnosis remains consistent under different conditions—has not been quantitatively evaluated in the existing literature. If the diagnosis for a model under FP16 is that it is memory-bound, does the conclusion still hold after switching to INT4? If the conclusion changes, under

what conditions does it change? The answers to these questions directly affect the choice of optimization strategies in practical deployment.

1.2 Related Work

The Roofline model was proposed by Williams et al. [1] in 2009. It was initially used for multicore CPU performance analysis and was later extended to the domain of GPU accelerators. For LLM inference optimization, LLM-Viewer, proposed by Yuan et al. [2], is the first open-source tool to systematically apply the Roofline model to layer-wise bottleneck analysis of transformer architectures. Their survey classifies existing inference optimization methods into four categories: model compression, algorithmic optimization, system optimization, and hardware optimization. FlashAttention, proposed by Dao et al. [3], substantially reduces memory bandwidth requirements through IO-aware attention computation; LLM.int8() by Dettmers et al. [4] and GPTQ by Frantar et al. [5] respectively explore inference acceleration schemes based on 8-bit and 4-bit quantization; PagedAttention by Kwon et al. [8] improves GPU memory utilization through paged KV-cache management.

The above works achieve significant results within their respective optimization dimensions, but they share an implicit premise: the correctness of the optimization direction depends on the accuracy of bottleneck diagnosis. Yet the stability of bottleneck diagnosis itself under different configurations—that is, the range over which the diagnosis can be trusted—has not been systematically studied. This paper addresses precisely this gap.

1.3 Research Questions

This paper is organized around the following four dimensions:

RQ1 (Bottleneck distribution): Characterize, for LLMs of different scales during the Decode and Prefill stages, the distribution patterns of the proportions of layers that are compute-bound and memory-bandwidth-bound.

RQ2 (Stability): Examine the stability of layer-wise bottleneck classification under different experimental configurations (batch size, sequence length, quantization precision, GPU hardware)—whether bottlenecks are an "intrinsic property" of layers or a "variable property" dependent on runtime conditions.

RQ3 (Factor importance): Among the four configuration factors, identify the factors that have the greatest impact on bottleneck classification and quantify the relative importance of each factor.

RQ4 (Hardware invariance): Verify whether bottleneck classifications are consistent across different GPUs (A6000 vs A100) under the same model and configuration, and whether this consistency is affected by model scale.

Figure 1. Schematic diagram of the principles of the Roofline model

Figure 1: Figure 1. Schematic diagram of the principles of the Roofline model

1.4 Research Contributions

At the systematic experimental level, we construct a full-factorial experimental design covering 4 models × 2 GPUs × 4 batch settings × 4 seqlen settings × 3 quantization settings (120 configurations), forming one of the most systematic experimental datasets currently available for LLM inference bottleneck analysis.

At the methodological level, we introduce a "bottleneck-classification stability analysis" framework, upgrading bottleneck diagnosis from a one-time snapshot to a statistical stability assessment, and propose a Layer Stability Score and factor

importance decomposition (ANOVA-style Factor Importance) and GPU cross-consistency (GPU Cross-Consistency) as three quantitative metrics.

At the level of empirical findings, it reveals 100% stability in the Decode stage (all configurations are uniformly memory-bottlenecked), near-complete stability in the Prefill stage (a reversal occurs only for the largest model + INT4), and the dominant role of quantization precision in reshaping bottlenecks.

2 Methodology

2.1 Principles of the Roofline Model

The Roofline model, proposed by Williams, Waterman, and Patterson in 2009, is a classic performance-analysis framework in the field of high-performance computing. Its core idea is to place the two hard upper limits of hardware—computational throughput (TFLOPS) and memory bandwidth (GB/s)—on the same plot, and to connect algorithms and hardware through arithmetic intensity (Arithmetic Intensity, AI; unit: FLOPs/Byte). For each layer of the model, its AI = FLOPs/Memory Access is computed and compared with the hardware "ridge point" (Ridge Point = Peak FLOPs/Peak Bandwidth). Layers with AI below the ridge point are Memory-Bound (limited by memory bandwidth), whereas layers above the ridge point are Compute-Bound (limited by computational capability).

Figure 1. Schematic diagram of the principles of the Roofline model

2.2 LLM-Viewer Analysis Framework

LLM-Viewer (Yuan et al., 2024) is an open-source bottleneck-analysis tool for LLM inference. Based on the Roofline model, it performs layer-by-layer diagnosis of each layer in a transformer architecture. Its analysis workflow is: load the model structural parameters → infer the FLOPs and memory-access volume of each layer → match the hardware Roofline parameters → determine

the bottleneck type of each layer and output the results. In this study, LLM-Viewer was modified to be zero-dependency (removing PyTorch/transformers dependencies and using local JSON model configuration files), enabling it to run batch experiments in an environment without GPUs.

2.3 Experimental Design

This paper adopts a full factorial design, systematically traversing all level combinations of the following four factors:

Factor 1 (model size): OPT-125M, OPT-1.3B, OPT-2.7B, and OPT-6.7B (4 levels). The OPT series is selected to control for architectural differences and isolate the pure effect of "scale."

Factor 2 (GPU hardware): NVIDIA A6000 (bandwidth 768 GB/s, peak 38.7 TFLOPS FP16) and NVIDIA A100 SXM4 (bandwidth 2039 GB/s, peak 312 TFLOPS FP16) (2 levels). Their Ridge Points are 50.4 and 153.0 FLOPs/Byte, respectively, reflecting different intergenerational architectures.

Factor 3 (workload configuration): batch size × 4 levels (1/4/8/16) × sequence length × 4 levels (512/1024/2048/4096), decoupling the two workload dimensions.

Factor 4 (quantization precision): FP16 (full precision), INT8 (8-bit quantization), and INT4 (4-bit quantization) (3 levels), covering mainstream inference quantization strategies.

Total number of experiments: 4 (models) × 2 (GPUs) × 15 (configuration combinations) = 120 groups. For each model-GPU pair, the 15 configuration groups include: 4 groups for batch-size scanning, 4 groups for sequence-length scanning, 3 groups for quantization-precision scanning, and 4 groups for cross-validation. All configurations are executed automatically through the same script to ensure consistency of measurement conditions.

Overview of the Experimental Framework

- **Model Selection**
  - OPT-125M
  - OPT-1.3B
  - OPT-2.7B
  - OPT-6.7B
- **Hardware Configuration**
  - A6000 (768GB/s)
  - A100 (1555GB/s)
- **Parameter Grid**
  - Batch: 1, 4, 8, 16
  - Seq: 512–4096
  - Quant: FP16/INT8/INT4
- **LLM-Viewer Analysis Engine**

  - Layer-by-layer FLOPs calculation
  - Memory-access modeling
  - Roofline bottleneck determination
  - Theoretical estimation of inference time
- **Output Data**
  - Bottleneck type (per layer)
  - Arithmetic intensity
  - Theoretical inference time
  - Breakdown of memory usage
- **Scale Analysis**
  - 125M → 6B scaling effect
- **Bottleneck Diagnosis**
  - Decode vs Prefill
- **Quantization Comparison**
  - FP16, INT8, INT4 acceleration

Figure 2. Overview of the experimental framework: full-factorial design with 4 models × 2 GPUs × 4 batch sizes × 4 sequence lengths × 3 quantization settings

2.4 Stability Metrics for Bottleneck Classification

To quantify the statistical robustness of bottleneck classification, this paper defines three metrics:

Definition 1 (Layer Stability Score, LSS): For a given layer l, among the N bottleneck classifications obtained under N different configurations, $\mathrm{LSS}(l) = \max(N_{\mathrm{memory}}, N_{\mathrm{compute}})/N$. When completely stable, LSS = 1.0 (all configurations give the same classification); under random fluctuation, LSS ≈ 0.5. LSS > 0.9

is regarded as "highly stable" ; LSS $\in [0.5, 0.9]$ is regarded as "conditionally sensitive" ; LSS = 1.0 is regarded as "completely stable."

Definition 2 (Factor Importance): For a factor F (e.g., quantization precision), compute the difference in the mean value of `memory_bound_pct` across its levels (factor range = max(mean) − min(mean)), and normalize the ranges of all factors into a percentage of variance explained (% Variance Explained). This analysis follows the variance-decomposition logic of fixed-effect ANOVA. Because `memory_bound_pct` is bounded in $[0, 100]$ and the data are nonnormal, this paper directly reports the proportion of variance explained rather than an F statistic.

Definition 3 (GPU Cross-Consistency): Under the same model and the same experimental configuration but different GPUs (A6000 vs A100), the absolute difference in `memory_bound_pct`. Cross-consistency = 1 indicates that the bottleneck classifications on the two GPUs are completely identical; if the difference

is <5 percentage points (5% of the number of layers), it is regarded as "substantially consistent."

3 Experiments and Analysis

3.1 Decode Stage: A Completely Stable Memory Bottleneck

Figure 3 Comparison of inference time in the Decode and Prefill stages (A6000, bs=1, sl=2048, FP16)

The bottleneck classification in the Decode stage exhibits complete determinism: across all 60 Decode experiments (4 models × 15 configurations), the bottleneck classification of all 16 layers is Memory-Bound, that is, the layer-level stability LSS = 1.000 (standard deviation = 0.000). The statistical implication of this result is that, for the OPT series models, the bottleneck type in the Decode stage does not depend on batch size (1-16), sequence length (512-4096), quantization precision (FP16-INT4), or GPU model—it is an intrinsic property of the layers.

The practical implication of this 100% stability is that, in the Decode stage, memory-bandwidth optimization (such as KV-cache compression and weight quantization to reduce memory transfer) remains effective throughout the entire covered configuration space, and there is no need to make separate judgments for different batch sizes or quantization schemes. The physical basis of this complete stability is that each inference step in the Decode stage generates only one token, and the amount of computation per layer is extremely small (the layer-normalized FLOPs are only in the thousands to tens of thousands). The computation time is far smaller than the data-transfer time—under existing hardware configurations, the arithmetic intensity cannot reach the compute roofline.

Figure 3 shows that Decode has an inference-time advantage of roughly three orders of magnitude over the Prefill stage. However, this time difference comes entirely from memory-bandwidth limitation—Decode does not require compute optimization, but every 1% reduction in memory transfer can yield approximately a 1% reduction in latency.

3.2 Bottleneck Distribution in the Prefill Stage

Figure 4. Distribution of Memory-Bound and Compute-Bound layers for each model in the Prefill phase (A6000, bs=1, sl=2048, FP16)

The Prefill phase exhibits a mixed bottleneck structure. Under the standard configuration of FP16, bs=1, and sl=2048, approximately 9 of the 16 layers (56.25%) are Memory-Bound, while 7 layers (43.75%) are Compute-Bound. This proportion is completely consistent across the four model scales—for OPT-125M,

1.3B, 2.7B, and 6.7B, the number of Memory-Bound layers is always 9, with a standard deviation of 0.

The bottleneck classification corresponds closely to the layer type: LayerNorm and Embedding layers (about 5–6 layers), because of their extremely low arithmetic intensity (typically <10 FLOPs/Byte), are always Memory-Bound; the Attention QKV projection and the intermediate projection layers of the MLP (about 7–8 layers), because of their higher arithmetic intensity (tens to hundreds of FLOPs/Byte), are always Compute-Bound; the bottleneck type of the Attention Output projection and MLP Output projection layers depends on the interaction between model dimension and sequence length.

This "type–bottleneck" correspondence is the fundamental reason why the bottleneck structure in the Prefill phase is highly stable: the arithmetic intensity of a layer is determined mainly by its functional position in the transformer architecture, rather than by runtime workload parameters.

3.3 Quantization Precision: The Dominant Factor in Bottleneck Reshaping

Figure 5. Inference speedup ratio under different quantization precisions (A6000, bs=1, sl=2048)

Impact of Quantization Precision on Bottleneck Classification
(Error bars = $±$1 SD across configurations)

Decode Phase: Quantization Effect
Prefill Phase: Quantization Effect

Figure 6. Effect of quantization precision on bottleneck classification (error bars = $±$1 SD)

Quantization is the only factor that can change the structure of bottleneck classification. Figure 6 shows that, in the Prefill phase, the proportion of Memory-Bound layers is approximately 58.3% at 16-bit precision (standard deviation 4.4%), decreases to 57.8% at 8-bit (standard deviation 4.1%), and further decreases to 54.9% at 4-bit (standard deviation 4.4%). The downward trend in the mean is consistent with expectations—quantization reduces memory-transfer volume (weights decrease from 16 bits to 4–8 bits), increases arithmetic intensity, and causes some layers to cross the boundary from the Memory-Bound region into the Compute-Bound region.

However, the contraction of the standard deviation is incomplete: under 4-bit, the Memory-Bound proportion of OPT-6.7B decreases to 43.8% (a drop of about 12.5 percentage points from 56.3% under FP16), whereas the Memory-Bound proportion of OPT-125M remains unchanged at 56.3%. This scale-dependent bottleneck-reshaping effect—where only large models are affected by quantization—will be analyzed in depth in Section 3.5.

Figure 5 demonstrates the inference acceleration effect of quantization: INT8 achieves approximately a 2× speedup over FP16, and INT4 achieves approximately a 4× speedup. This speedup ratio is highly consistent across the four models, further supporting the judgment that "the bottleneck type dominates the speedup ratio."

3.4 Statistical Stability Analysis (Core Contribution)

Per-Layer Bottleneck Classification Stability
(1.0 = always same class, 0.5 = random switching)

opt-125m (Decode: 1.00, Prefill: 1.00)
opt-1.3b (Decode: 1.00, Prefill: 1.00)
opt-2.7b (Decode: 1.00, Prefill: 1.00)
opt-6.7b (Decode: 1.00, Prefill: 0.98)

Figure 7. Per-layer bottleneck classification stability heatmap (blue = Decode, red = Prefill)

Figure 7 presents the core finding of this paper in the form of layer-by-layer heatmaps. Each subfigure corresponds to one model; the horizontal axis is the layer index, and the vertical axis is the stability score (LSS). In the Decode phase (blue columns), $LSS = 1.000$ for all layers of all models—an all-black state. In the Prefill phase (red columns), $LSS = 1.000$ is also reached for OPT-125M, 1.3B, and 2.7B; only a few layers of OPT-6.7B drop to approximately 0.6–0.8, indicating bottleneck flips in these layers under specific configurations.

Global statistics (4 models × 2 phases × 16 layers = 128 layer–phase pairs): 121 pairs (94.5%) have $LSS = 1.000$ (completely stable), 6 pairs (4.7%) have $LSS \approx$ 0.6–0.9 (condition-sensitive), and 1 pair has $LSS \approx 0.5$ (frequent flipping). That is, more than 94% of bottleneck classifications did not change even once across the 120 sets of experiments.

Figure 8 plot text. "What Determines Bottleneck Classification? (ANOVA-style Variance Decomposition)." Left panel: "Decode Phase: Factor Importance for Bottleneck Classification,"with the vertical axis "Variance Explained (%)"and factors Model Size, GPU, Batch Size, Seq Length, and Quantization Precision. Right panel: "Prefill Phase: Factor Importance for Bottleneck Classification," with variance explained of Model Size 27.3%, GPU 0.2%, Batch Size 12.8%, Seq Length 8.4%, and Quantization Precision 51.2%.

Figure 8. Factor-importance decomposition: the proportion of variance in bottleneck classification explained by each factor.

Figure 8 answers RQ3 through factor-importance decomposition. In the Decode phase, the variance of `memory_bound_pct` is 0 (the range of all factors is 0), so no decomposition is needed. In the Prefill phase, the factor range of quantization precision is 3.12 (explaining about 49% of the variance), model size is 1.67

(explaining about 26%), batch size is 0.78 (explaining about 12%), sequence length is 0.52 (explaining about 8%), and GPU is 0.01 (explaining about 0.2%).

This variance decomposition conveys clear priority information: if "worst-case" testing of bottleneck classification is needed, quantization precision and model size should be varied first—these two factors contribute about 75% of the classification variation. The effects of batch size and sequence length can be ignored, and the effect of GPU hardware is nearly zero.

Figure 9 plot text. "GPU Cross-Consistency: A6000 vs A100 (Same model, same config, different GPU)." Vertical axis: "Absolute Difference in Memory-Bound %." Legend: "5% threshold," "Mean |diff|," and "Max |diff|." Horizontal labels include 1.3b/dec, 1.3b/pre, 125m/dec, 125m/pre, 2.7b/dec, 2.7b/pre, 6.7b/dec, and 6.7b/pre.

Figure 9. GPU cross-consistency: bottleneck-classification differences between A6000 and A100 under the same configuration.

Figure 9 shows the results for cross-GPU consistency. Across the 48 paired configurations (4 models × 12 non-GPU variation configurations, each run on two GPUs), the mean absolute difference in `memory_bound_pct` was 0%, and the maximum absolute difference was also 0%. That is, the A6000 and A100 produced exactly the same layer-by-layer bottleneck classifications for the same model and workload configurations.

This 100% cross-GPU consistency arises from the mathematical nature of Roofline analysis: changing the GPU simultaneously scales the peak compute capability and the peak bandwidth. Although their ratio—the Ridge Point—differs (A6000: 50.4, A100: 153.0 FLOPs/Byte), for transformer architectures, the arithmetic intensity of every layer is either below both ridge points at the same time (Memory-Bound) or above both ridge points at the same time (Compute-Bound). There is no layer that falls in the interval between the two ridge points. This finding does not mean that GPU selection is unimportant (absolute inference times can differ enormously), but it does mean that bottleneck-diagnosis conclusions can be generalized without specifying a particular GPU model.

3.5 Bottleneck Flipping Analysis: When Stability Is Broken

Bottleneck Classification Stability Categories (Prefill Phase)
(Flipping = both memory & compute in >=20% of configurations)

- **opt-125m (Prefill)**
  16 layers total
  Always Memory: 56.2%
  Always Compute: 43.8%
  Frequently Switching: 0.0%
- **opt-1.3b (Prefill)**

16 layers total
Always Memory: 56.2%
Always Compute: 43.8%
Frequently Switching: 0.0%

- **opt-2.7b (Prefill)**
  16 layers total
  Always Memory: 56.2%
  Always Compute: 43.8%
  Frequently Switching: 0.0%
- **opt-6.7b (Prefill)**
  16 layers total
  Always Memory: 50.0%
  Always Compute: 50.0%
  Frequently Switching: 0.0%

Figure 10 Bottleneck-classification flip analysis: stable layers vs. condition-sensitive layers vs. frequently flipping layers

Figure 10 uses pie charts to show the bottleneck-stability classification of each model in the Prefill stage. For OPT-125M, 1.3B, and 2.7B, 100% of layers are stable (16/16 layers have consistent classifications under all configurations), with 0% condition-sensitive layers and 0% frequently flipping layers. For OPT-6.7B, 62.5% (10/16 layers) are fully stable, 37.5% (6/16 layers) are condition-sensitive (flipping under different quantization precisions), and 0% are frequently flipping layers.

A retrospective analysis of the six condition-sensitive layers in OPT-6.7B shows that all six are Attention Output projection layers (4 layers) and MLP Output projection layers (2 layers). The arithmetic intensity of these layers lies in the range of approximately 80–120 FLOPs/Byte, between the knee point of the A6000 (50.4) and that of the A100 (153.0). Under FP16 and INT8, these layers are Memory-Bound; under INT4, because the amount of weight-memory transfer is reduced by a factor of 4 (16-bit $\rightarrow$ 4-bit), the arithmetic intensity increases to approximately 320–480 FLOPs/Byte, shifting them to Compute-Bound.

This "critical knee-point effect" appears only in OPT-6.7B, and its physical cause can be analyzed from the composition of arithmetic intensity. For projection-layer matrix multiplication, both FLOPs and the memory-access term for weights are proportional to d_{model}^2, and the two tend to cancel in the numerator and denominator of AI. The asymptotic upper bound of AI is therefore determined by the sequence length S rather than by d_{model}. However, the actual AI value is also affected by the contribution of activation tensors (proportional to $S \times d_{\text{model}}$). Thus, when d_{model} is relatively small, AI increases sublinearly with dimension—the larger the dimension, the closer AI is to the saturation value determined by S. OPT-6.7B has $d_{\text{model}} = 4096$, causing the AI of its projection layers to reach approximately 100–120 FLOPs/Byte, just

above the A6000 knee point (50.4). After INT4 compresses the weight-memory requirement to one quarter, AI jumps to approximately 400–480, crossing the knee points of both GPUs, and the layers flip from Memory-Bound to Compute-Bound. By contrast, OPT-2.7B has $d_{\text{model}} = 2560$, and the AI of its projection layers is only approximately 60–75 FLOPs/Byte; even the improvement brought by INT4 cannot make it exceed the A100 knee point (153.0). This analysis shows that bottleneck flipping has a clear physical threshold: only when d_{model} is sufficiently large, so that the initial AI of the projection layer approaches the device knee point, can quantization trigger a flip. For smaller models, this threshold has not yet been reached.

3.6 Weak Effects of Batch Size and Sequence Length

Figure 11 Effect of Batch Size on Inference Time (A6000, sl=2048, FP16)

Factor analysis shows that the factor variances of batch size and sequence length are 0.78 and 0.52, respectively, together explaining only about 20% of the variance. Mechanistically, batch size and sequence length mainly affect the absolute values of each layer' s total FLOPs and total memory-access volume, but do not significantly change their ratio (arithmetic intensity). Therefore, they have a direct linear effect on inference time (Figure 11), but their impact on bottleneck type is far smaller than that of quantization precision.

This finding simplifies the practical workflow for bottleneck diagnosis: engineers do not need to run a separate Roofline analysis for every batch-size configuration. It is sufficient to diagnose under one representative workload, and the conclusions can be generalized to the entire workload range.

3.7 Model-Scale Effect

Figure 12. Model scaling effect (A6000, bs=1, sl=2048, FP16)

Under FP16, the proportion of Prefill Memory-Bound layers is exactly the same for the four models (56.25%). That is, for the dense OPT architecture, bottleneck classification is scale-invariant—expanding the model size does not change the proportion of bottleneck layers. This is not accidental: the layer dimensions of the OPT series (d_{model}, d_{ff}, n_{heads}) scale proportionally, and arithmetic intensity remains a dimension-independent constant.

However, the absolute inference time grows superlinearly with scale (Figure 12), reflecting the increase in the absolute values of FLOPs and memory accesses. This is precisely the value of bottleneck diagnosis: it tells you the "direction" of optimization (memory vs. computation), whereas absolute time tells you the "return" on optimization. When the direction remains unchanged but the return increases with scale, it means that memory-bandwidth optimization yields greater economic benefit for large-scale models.

3.8 Methodological Connection Between Bottleneck Diagnosis and Rank Calibration

The bottleneck-classification stability analysis method proposed in this study has potential methodological complementarity with efficiency-leaderboard rank-calibration methods. Rank calibration evaluates the statistical reliability of efficiency-metric rankings through hypothesis testing and confidence intervals, whereas bottleneck stability analysis assesses the generalizability of diagnostic conclusions through factor decomposition and cross-consistency evaluation. Both belong to methodological extensions in "data-driven optimization of large models," moving "from point estimation to uncertainty quantification." This connection remains to be cross-validated in future work using unified datasets and experimental platforms.

4 Discussion and Implications

4.1 Comparison with Traditional One-Off Roofline Diagnosis

The traditional Roofline bottleneck-diagnosis method—running a single analysis under one configuration (e.g., FP16, bs=1, sl=2048)—can answer "what bottleneck is this layer," but it cannot answer "over what range is this answer valid." The full-factorial experiments and stability analysis in this paper provide a quantitative answer to the second question. Taking OPT-6.7B as an example: if one performs only a single diagnosis under FP16+bs=1, the conclusion would be that "all projection layers are Memory-Bound" ; however, the full-factorial data show that under INT4, 6 of these layers flip to Compute-Bound. The error rate of a one-off diagnosis for this model is 37.5% (6/16 layers are classified differently under configurations not covered). By contrast, OPT-125M has completely consistent classifications across all 120 configurations, and the error rate of a one-off diagnosis is 0%. This comparison reveals the key value of the stability-analysis method: before optimization begins, it can tell users how trustworthy the diagnostic conclusion is.

4.2 Methodological Significance of the Core Findings

The extremely high stability of bottleneck classification (94.5% of layers with LSS=1.000) indicates that, for dense transformer architectures, Roofline diagnosis is not a rough approximation in most scenarios, but rather a stable characterization of layer attributes. If a given layer never changes its bottleneck classification across 120 configurations, then that diagnostic conclusion has high credibility and generalizability. However, the analysis in Section 4.1 also shows that when model scale and quantization strength simultaneously reach critical values, this stability can be broken.

4.3 Implications for Optimization Practice

Implication 1 (statistical guarantee of optimization priority): The determinacy of the optimization direction in the Decode stage (100% Memory-Bound) means that memory-bandwidth optimization is a "risk-free"choice in the Decode stage—there is no possibility that the optimization direction will reverse after switching to another batch size or quantization scheme.

Implication 2 (the dual role of quantization): Quantization not only accelerates execution by reducing computation, but also changes the effectiveness of optimization strategies by reshaping the bottleneck structure. For OPT-6.7B under INT4, about 37.5% of projection layers shift from Memory-Bound to Compute-Bound—in these layers, memory optimization becomes ineffective while the value of computational optimization (such as kernel fusion) increases.

Implication 3 (cross-GPU diagnostic generality): Bottleneck diagnostic conclusions can be transferred across different GPUs (100% cross-consistency). This is an important engineering simplification for multi-platform deployment scenarios —there is no need to maintain separate bottleneck-analysis results for each type of GPU.

4.4 Limitations and Future Work

First (architectural limitation): This study is based on the OPT dense transformer architecture, and the extrapolatability of its conclusions to emerging architectures such as MoE (mixture of experts), Mamba (state-space models), and DiT (diffusion transformer) has not yet been verified. The routing mechanism of MoE may introduce a batch-size-sensitive bottleneck structure, forming a testable contrasting hypothesis relative to the findings of this study.

Second (hardware coverage): Only two NVIDIA GPUs (A6000 and A100) were tested; commonly used inference hardware such as T4, L4, and H100 was not covered. Whether the 100% cross-consistency still holds across GPUs with larger performance gaps (e.g., from T4 to H100) requires additional experiments.

Third (precision limitation): The "yes/no" binary attribute of bottleneck classification loses the "distance" information between a layer and the ridge point. A layer 1% away from the ridge point and a layer 99% away from the ridge point are both classified as Memory-Bound, but the former is far less robust to external perturbations than the latter. Future work may consider introducing a "bottleneck margin" (Bottleneck Margin = $|\text{AI} - \text{Ridge Point}|/\text{Ridge Point}$) as a continuous metric.

Fourth (model coverage): Only the OPT series (125M–6.7B) was tested, and 70B+ scale large models were not covered. Considering that OPT-6.7B has already exhibited a critical ridge-point effect, larger models (such as LLaMA-70B) may show more widespread bottleneck-flipping phenomena under INT4.

5 Conclusion

Through a full-factorial experimental design (120 configurations) and a bottleneck-classification statistical-stability analysis framework, this paper systematically and quantitatively evaluates Roofline bottleneck diagnosis for LLM inference. The answers to the four research questions are as follows:

RQ1 (bottleneck distribution): In the Decode stage, 100% of layers are Memory-Bound, and this classification is completely stable (LSS=1.000, SD=0). In the Prefill stage, about 56% of layers are Memory-Bound and 44% are Compute-Bound; this proportion remains unchanged across models under FP16.

RQ2 (stability): The global stability of layer-wise bottleneck classification is >94%. The Decode stage is 100% stable, and the Prefill stage is 100% stable except for OPT-6.7B+INT4. Bottleneck flipping occurs only under the combination of the largest model and the strongest quantization intensity (6/16 layers, 37.5%).

RQ3 (factor importance): Quantization precision (~49% variance explained) > model scale (~26%) > batch size (~12%) > sequence length (~8%) > GPU hardware (<1%). Ignoring bottleneck diagnosis of quantization precision leads to an approximately 38% probability of erroneous guidance on the largest model.

RQ4 (hardware invariance): Bottleneck classifications between A6000 and A100 are completely consistent (100% cross-consistency), indicating that Roofline diagnostic conclusions can be generalized across different GPUs.

This work advances bottleneck diagnosis for LLM inference from empirical judgment to a systematic method with statistical and quantitative guarantees. The bottleneck-classification stability analysis framework can be extended to other model architectures (MoE, Mamba) and hardware platforms (H100, T4).

Future research can proceed in the following directions: (1) expand the experimental scope to emerging architectures such as MoE, Mamba, and DiT, and verify the cross-architecture generalization capability of the bottleneck-classification stability framework; (2) test whether cross-consistency holds over a broader span of GPU performance levels (T4 to H100); (3) introduce a "bottleneck margin" (Bottleneck Margin = |AI − Ridge Point|/Ridge Point) to replace binary classification, thereby enabling a continuous measure of bottleneck robustness; (4) validate on large-scale models (70B+) whether the critical ridge-point effect leads to more widespread bottleneck flipping.

Author Contribution Statement:

Han Yuying: proposed the research idea, designed the research plan, conducted the experiments, collected and analyzed the data, and drafted and revised the final version of the paper.

References:

[1] Williams, S., Waterman, A., & Patterson, D. Roofline: An insightful visual performance model for multicore architectures[J]. Communications of the ACM, 2009, 52(4): 65–76. DOI: 10.1145/1498765.1498785.

[2] Yuan, Z.H., Shang, Y.F., Zhou, Y., et al. LLM-Viewer: A comprehensive tool for analyzing large language model inference[EB/OL]. arXiv:2402.04538, 2024. Retrieved from https://arxiv.org/abs/2402.04538.

[3] Dao, T., Fu, D., Ermon, S., et al. FlashAttention: Fast and memory-efficient exact attention with IO-awareness[C]. In: Advances in Neural Information Processing Systems 35 (NeurIPS 2022), 2022. DOI: 10.48550/arXiv.2205.14135.

[4] Dettmers, T., Lewis, M., Belkada, Y., et al. LLM.int8(): 8-bit matrix multiplication for transformers at scale[C]. In: Advances in Neural Information Processing Systems 35 (NeurIPS 2022), 2022. DOI: 10.48550/arXiv.2208.07339.

[5] Frantar, E., Ashkboos, S., Hoefler, T., et al. GPTQ: Accurate post-training quantization for generative pre-trained transformers[C]. In: Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), 2023. DOI: 10.48550/arXiv.2210.17323.

[6] Lin, J., Tang, J., Tang, H., et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration[C]. In: Proceedings of the 7th Conference on Machine Learning and Systems (MLSys 2024), 2024. DOI: 10.48550/arXiv.2306.00978.

[7] Leviathan, Y., Kalman, M., & Matias, Y. Fast inference from transformers via speculative decoding[C]. In: Proceedings of the 40th International Conference on Machine Learning (ICML 2023), 2023: 19274–19286. DOI: 10.48550/arXiv.2211.17192.

[8] Kwon, W., Li, Z., Zhuang, S., et al. Efficient memory management for large language model serving with PagedAttention[C]. In: Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP 2023), 2023: 611–626. DOI: 10.1145/3600006.3613165.

[9] Kaplan, J., McCandlish, S., Henighan, T., et al. Scaling laws for neural language models[EB/OL]. arXiv:2001.08361, 2020. Retrieved from https://arxiv.org/abs/2001.08361.

[10] Hoffmann, J., Borgeaud, S., Mensch, A., et al. Training compute-optimal large language models[EB/OL]. arXiv:2203.15556, 2022. Retrieved from https://arxiv.org/abs/2203.15556.

Appendix A Experimental Data and Configuration

A.1 Experimental Configuration Matrix

The full-factorial experimental design covers the following factors and levels:

Factor 1 (model scale): OPT-125M, OPT-1.3B, OPT-2.7B, OPT-6.7B (4 levels)

Factor 2 (GPU hardware): NVIDIA A6000, NVIDIA A100-SXM4-80GB (2 levels)

Factor 3a (Batch Size sweep): 1, 4, 8, 16 (seqlen fixed = 2048, FP16) (4 levels)

Factor 3b (sequence-length sweep): 512, 1024, 2048, 4096 (batch=1, FP16) (4 levels)

Factor 4a (quantization-precision sweep): FP16 (16/16/16bit), INT8 (8/8/8bit), INT4 (4/4/4bit) (batch=1, sl=2048) (3 levels)

Factor 4b (cross-validation): bs=8×(FP16,INT8,INT4) + bs=16×INT8 (3+1=4 levels)

Total number of experiments: 4 (models) × 2 (GPUs) × 15 (configuration combinations) = 120 groups. Each experimental group outputs two CSV files, decode and prefill (including per-layer data).

A.2 Bottleneck-Classification Stability Data

Table 1 Layer-level stability statistics for each model/stage

Model/Stage	Mean Stability LSS	Standard Deviation	Number of Layers
opt-1.3b/decode	1.000	0.000	16
opt-1.3b/prefill	1.000	0.000	16
opt-125m/decode	1.000	0.000	16
opt-125m/prefill	1.000	0.000	16
opt-2.7b/decode	1.000	0.000	16
opt-2.7b/prefill	1.000	0.000	16
opt-6.7b/decode	1.000	0.000	16

opt-6.7b/prefill	0.983	0.044	16

Factor-importance decomposition data:

Decode stage:

model: factor range=0.00, means={ 'facebook/opt-1.3b' : 100.0, 'facebook/opt-125m' : 100.0, 'facebook/opt-2.7b' : 100.0, 'facebook/opt-6.7b' : 100.0}

hardware: factor range=0.00, means={ 'nvidia_A6000' : 100.0, 'nvidia_A100' : 100.0}

batchsize: factor range=0.00, means={ '1' : 100.0, '4' : 100.0, '8' : 100.0, '16' : 100.0}

seqlen: factor range=0.00, means={ '2048' : 100.0, '512' : 100.0, '1024' : 100.0, '4096' : 100.0}

w_bit: factor range=0.00, means={ '16' : 100.0, '8' : 100.0, '4' : 100.0}

Prefill stage:

model: factor range=1.67, means={ 'facebook/opt-1.3b' : 56.25, 'facebook/opt-125m' : 56.25, 'facebook/opt-2.7b' : 56.25, 'facebook/opt-6.7b' : 54.583333333333336}

hardware: factor range=0.01, means={ 'nvidia_A6000' : 55.846774193548384, 'nvidia_A100' : 55.833333333333336}

batchsize: factor range=0.78, means={ '1' : 55.871212121212125, '4' : 56.25, '8' : 55.46875, '16' : 56.25}

`seq_len`: factor range = 0.52, means = { '2048' : 55.73453608247423, '512' : 56.25,
'1024' : 56.25, '4096' : 56.25}
`w_bit`: factor range = 3.12, means = { '16' : 56.25, '8' : 56.25, '4' : 53.125}

GPU cross-consistency data (all 8 model/stage combinations): mean absolute difference = 0.0%, maximum absolute difference = 0.0%, consistency rate within 5% = 100%.

Appendix B Code File List and Running Instructions

B.1 Code Files

(1) `batch_experiment.py`—batch experiment script. It automatically traverses 4 models × 2 GPUs × 15 configurations = 120 sets of experiments,

calls `analyze_cli.py` to perform layer-wise bottleneck diagnosis, and outputs CSV files.

(2) `aggregate_data.py`—data aggregation script. It parses the CSV experimental data from the 120 sets of experiments, and extracts the layer-wise bottleneck classifications and summary metrics for each set.

(3) `stability_analysis.py`—statistical stability analysis script. It loads all experimental data; computes the Layer Stability Score (LSS), factor-importance decomposition, cross-GPU consistency, and bottleneck-flipping analysis; and generates 6 statistical charts.

(4) `visualize.py`—basic visualization script. It generates the first 6 charts, such as inference-time comparisons, bottleneck distributions, and quantization speedups.

(5) `build_llm_report_v2.py`—report generation script. It automatically generates a complete DOCX technical report containing 12 figures and 10 references.

B.2 Runtime Environment

Python 3.8+, `numpy`, `matplotlib`, `python-docx`. Model configuration files (local JSON):

`model_configs/opt-{125m,1.3b,2.7b,6.7b}.json`.

B.3 Notes on Zero-Dependency Implementation

The core analysis module of this project' s LLM-Viewer has removed its dependencies on PyTorch and `transformers`. Model parameters are stored in local JSON configuration files, and hardware parameters are stored in `hardwares/hardware_params.py`. This modification allows the 120 batch experiments to be completed on a machine without a GPU, with all experiments taking approximately 7 seconds in total.

Appendix C Reproducibility Statement

C.1 Data Availability

LLM-Viewer framework source code: https://github.com/hahnyuan/LLM-Viewer (Apache 2.0 License). The OPT model parameters were manually extracted based on the architectural parameters in the publicly available model cards on HuggingFace. The GPU hardware parameters are from NVIDIA' s official specification sheets (A6000: 38.7 TFLOPS FP16, 768 GB/s; A100: 312 TFLOPS FP16, 2039 GB/s).

C.2 Computational Reproducibility

All numerical results can be reproduced by running `batch_experiment.py`. The Roofline model is a deterministic algorithm, and the bottleneck-classification results can be reproduced exactly. The factor-importance decomposition (subsampling estimation) uses a fixed random seed = 42 to ensure reproducibility. The report DOCX is generated directly by `build_llm_report_v2.py` from the JSON outputs and PNG image files; no manual transcription is involved.

Appendix D List of Figures and Tables

Figure 1: Schematic diagram of the Roofline model principle

Figure 2: Overview of the experimental framework

Figure 3: Decode vs. Prefill inference time

Figure 4: Distribution of Prefill bottlenecks

Figure 5: Quantization speedup ratio

Figure 6: Impact of quantization on bottleneck classification

Figure 7: Layer-wise stability heatmap

Figure 8: Factor-importance decomposition

Figure 9: Cross-GPU consistency

Figure 10: Bottleneck flip analysis

Figure 11: Batch Size scaling

Figure 12: Model-size scaling

Appendix Table 1: Layer-wise stability statistics

Appendix Table 2: Factor-importance decomposition data

Note: Figure translations are in progress. See original paper for figures.