Progress in Computer Applications

DLINE JOURNALS

Print ISSN: 2278 – 6465
Online ISSN: 2278 – 6473

PCA 2026: 15 (2)
https://doi.org/10.6025/pca/2026/15/2/82-103

A Systematic Taxonomy of Large Language Model Inference Optimization: Bridging Model-Level and API-Layer Latency Reduction Strategies

Yao-Liang Chung
Department of Communications, Navigation and Control Engineering
National Taiwan Ocean University
Taiwan
ylchung@ntou.edu.tw

ABSTRACT

The deployment of Large Language Models (LLMs) as autonomous agents in next generation infrastructures, such as Agentic AI-Native 6G networks, introduces severe latency, memory, and operational cost bottlenecks. Existing optimization approaches frequently treat model compression and system orchestration in isolation, failing to adequately address the multidimensional constraints of modern production environments. This paper presents a systematic, multi criteria taxonomy of LLM inference optimization, bridging granular model level compression techniques with macro level API-layer orchestration strategies. We conduct a rigorous comparative analysis of post training quantization algorithms specifically highlighting activation-aware methods like AWQ alongside GPTQ and GGUF evaluating their trade offs across memory efficiency, inference throughput, and predictive quality. Our findings reveal that deployment efficiency is not solely dictated by numerical precision but is fundamentally a co-design problem requiring optimized hardware kernels and adaptive, importance aware parameter protection. Furthermore, we demonstrate that system-level interventions, such as intelligent model routing, prompt caching, and deterministic decoupling, yield immediate latency reductions without altering underlying model weights. Ultimately, this study provides a comprehensive, scenario based decision framework for matching specific quantization algorithms and API-layer strategies to distinct deployment environments, ranging from high throughput GPU serving to resource-constrained edge devices and safety critical autonomous systems. By unifying these perspectives, we establish a foundational guide for practitioners aiming to build reliable, high performance inference pipelines that ensure scalable and economically sustainable AI integration across diverse industrial domains.

Keywords: Large Language Models, Inference Optimization, Post-Training Quantization, Activation-Aware Quantization, API-Layer Routing, Edge Computing, Hardware Kernels, Agentic AI, Latency Reduction

Received: 2 April 2026, Revised 12 June 2026, Accepted 29 June 2026

1. Introduction

Sixth-generation (6G) networks are increasingly envisioned as AI-native infrastructures that seamlessly integrate communication, sensing, and computing into a unified fabric. However, existing approaches remain largely optimization centric, relying on closed loop control mechanisms with limited reasoning capabilities. Recent literature argues for a paradigm shift toward Agentic AI-Native 6G, wherein LLM-based agents operate as bounded, policy governed reasoning entities within a semantic control plane layered above deterministic 3GPP infrastructure [1].

Concurrently, the rise of intelligent agents marks a new era of human computer interaction [2] (Wang et al., 2024b). LLMs are rapidly evolving from passive, text based reasoning engines [3, 4] into autonomous digital operators capable of perceiving, acting, and coordinating across complex tasks [5]. These capabilities make LLMs highly attractive interfaces for enterprise applications, where users naturally express goals rather than navigating rigid workflow forms. Despite this progress, current agent frameworks remain largely confined to single devices or isolated platforms, such as individual browser tabs [6, 7] 2024), desktop environments [8] or specific mobile applications [9, 10]. Furthermore, these models remain unreliable as direct system operators; they are prone to hallucinating arguments, confusing similarly named entities, overgeneralizing permissions, or producing action sequences that appear plausible but are invalid or unsafe in execution. Consequently, there is a pressing need to design reliable and scalable frameworks for intelligent cross device agent orchestration [11].

2. Background

2.1 Tool-Augmented LLMs and the Imperative for API-Layer Optimization

To overcome the inherent limitations of standalone LLMs, prior work has established that models can significantly improve task performance by interacting with external tools and APIs. Foundational research, such as Toolformer [12] demonstrates that language models can learn to autonomously decide when to invoke tools, determine appropriate arguments, and incorporate returned outputs into subsequent predictions. This direction has been extended toward larger API surfaces by frameworks like Gorilla [13] which emphasize accurate API invocation, dynamic retrieval over evolving documentation, and broad scale tool use training.

However, integrating these capabilities into production environments introduces severe latency and cost bottlenecks. Recent practical case studies in system level LLM inference optimization demonstrate that significant latency and cost reductions can be achieved without modifying underlying model weights [14, 15] (Emre; Sohail). By intelligently routing requests, decoupling deterministic data retrieval from probabilistic LLM generation, and carefully sequencing tool calls, systems can protect the critical first token latency budget. Because such deployments often rely on managed cloud API services (e.g., Azure OpenAI), the available optimization levers are necessarily restricted to prompt design, tool orchestration, caching mechanisms, and overarching system architecture.

2.2 Strategic Model Routing and the Transition to Edge Inference

Given the constraints of black box API deployments, optimization must be framed as a strategic selection and dynamic routing problem. Empirical evaluations reveal that no single model dominates across all performance dimensions. Smaller, highly optimized models offer a viable, Pareto optimal trade off for latency sensitive deployments, whereas larger, reasoning augmented models should be reserved exclusively for accuracy-critical, complex parsing tasks. This dichotomy implicitly advocates for intelligent model routing: dynamically directing incoming requests based on input classification to optimally balance latency, cost, and accuracy.

Looking toward future optimization pathways, there is a strong consensus recommending a transition from reliance on external APIs to localized inference. This involves fine tuning compact models (typically 7 to 14 billion parameters) using Parameter Efficient Fine Tuning (PEFT) techniques, such as Low Rank Adaptation (LoRA), on domain specific data (Milani). Deploying such fine tuned models at the edge fundamentally eliminates network induced API latency, drastically reduces long term operational costs, and resolves critical data

privacy concerns. This represents a necessary architectural shift toward optimized, self hosted inference for enterprise and industrial applications.

2.3 Architectural Integration in Safety-Critical Domains

The principles of system level optimization and reliable orchestration are particularly critical when extending LLM and Vision Language Model (VLM) capabilities into safety critical domains, such as autonomous driving. Traditionally, supervised object detection models have been favored in automotive systems because they provide fast and accurate detection of predefined object categories, which is essential for real time performance and localization accuracy [16, 17] (Janai; Liang).

More recently, VLMs have been introduced to enable higher level semantic understanding of complex scenes. VLMs can generate contextual descriptions, reason about intricate environments, and support downstream perception tasks through natural language reasoning, thereby improving robustness in edge case scenarios [18, 19] (Guo; Li; Ashani). Despite the growing adoption of these advanced AI approaches in automotive perception, a significant research gap remains. There is limited research on how to effectively combine these proba bilistic AI-based approaches within human in the loop annotation systems from a rigorous software architectural design pattern perspective. Future work must address how to balance system level quality attr-ibutes such as latency, reliability, and safety with the necessary human intervention effort [20-23].

The deployment of Large Language Models as autonomous, cross device agents requires a holistic approach to inference optimization. While model level techniques offer profound computational advantages, API-layer strategies such as intelligent routing, deterministic decoupling, and edge deployed parameter efficient fine tuning provide immediate, scalable solutions for production environments. As these systems are increasingly integrated into safety critical domains like autonomous driving and 6G networks, future research must prioritize robust software architectural design patterns that guarantee both operational efficiency and unwavering reliability.

3. Research Design and Methods

3.3.1 Research Design

This study adopts a comparative analytical research design grounded in systematic literature synthesis and structured multi criteria evaluation. Rather than conducting new large scale training or quantization expe-riments, the work synthesizes empirical findings, algorithmic descriptions, and reported benchmark results from the primary literature on post training quantization (PTQ) methods. The design is intentionally multi dimensional: it evaluates techniques simultaneously along the axes of memory reduction, inference latency/ throughput, prediction quality, calibration complexity, hardware portability, and deployment practicality.

The central analytical lens is activation-aware optimization, with Activation Aware Weight Quantization (AWQ) serving as the primary case study. This focus is justified by the theoretical and empirical observation that transformer channels contribute highly unevenly to model outputs; a small subset of channels interacting with large magnitude activations disproportionately determines predictive performance. By examining how activation statistics can be leveraged to guide selective precision allocation, the study moves beyond uniform numerical compression toward importance aware quantization strategies.

The overall research strategy is therefore:

- Descriptive–comparative at the algorithmic level (GPTQ, AWQ, GGUF/K-Quants, bitsandbytes, SmoothQuant/ FP8),

- Explanatory with respect to the mechanisms that preserve or degrade quality under low bit regimes,

• Prescriptive in deriving practical decision guidance for different deployment scenarios.

3.3.2 Scope and Technique Selection

The scope is restricted to weight only post-training quantization methods that have achieved significant adoption in production inference frameworks (vLLM, TensorRT-LLM, llama.cpp, Hugging Face Transformers) as of 2025–2026. Four representative algorithms were selected according to the following criteria:

1. Distinct optimization philosophy (Hessian-based reconstruction, activation-aware saliency, mixed-precision block quantization, dynamic/on-the-fly quantization).

2. Availability of open implementations and reported results on models ranging from 7B to 70B parameters.

3. Relevance to both high-throughput GPU serving and heterogeneous (CPU / Apple Silicon / edge) deployment.

4. Explicit or implicit use of activation information or importance estimation.

AWQ is positioned as the focal technique because it operationalizes the activation aware principle most directly: it identifies approximately the top 1 % of salient channels via activation magnitude statistics and protects them through per-channel scaling while aggressively quantizing the remainder.

3.3.3 Evaluation Framework and Metrics

A multi-criteria evaluation framework was constructed to enable consistent comparison across algorithms. The primary dimensions and associated metrics are:

Dimension	**Metrics**	**Rationale**
Memory efficiency	Effective bits per weight, model size reduction (×), peak VRAM/ RAM	Direct consequence of precision reduction
Inference performance	Tokens/s, end-to-end latency, speedup vs. FP16/BF16	Realized benefit after kernel optimization
Prediction quality	Perplexity (WikiText-2 / C4), zero-shot accuracy (MMLU, GSM8K, HumanEval, etc.), relative degradation (%)	Measures information loss
Calibration / preprocessing cost	Time and data volume required for quantization	Practical barrier to adoption
Hardware & runtime compatibility	Supported engines, CPU/GPU/ Apple Silicon support, kernel maturity	Deployment feasibility
Quality–compression trade-off	Quality retention at INT4 / W4A16 versus INT8 / FP8	Central decision variable

Because the study synthesizes existing results rather than generating new measurements, reported figures from the original papers and subsequent independent evaluations (vLLM, TensorRT-LLM, llama.cpp community

benchmarks) are used. Where multiple sources exist, the most consistent or most recent results on comparable model scales are prioritized.

3.3.4 Activation-Aware Analytical Methodology

The activation-aware component of the methodology proceeds in three analytical stages:

Stage 1 – Conceptual decomposition The AWQ pipeline is decomposed into its constituent operations:

- Collection of activation statistics on a small calibration set,
- Computation of per-channel saliency scores (typically based on activation magnitude × weight importance),
- Search for optimal per-channel scaling factors that protect the top ~1 % of channels,
- Absorption of scales into the weights followed by uniform low bit quantization of the non-salient majority.

This decomposition isolates the precise mechanism by which activation information is injected into the quantization process and differentiates it from pure second order (Hessian) methods such as GPTQ.

Stage 2 – Comparative mechanistic analysis AWQ is contrasted with GPTQ along two axes:

- Information source used for error compensation (activation statistics vs. approximate inverse Hessian),
- Computational complexity of the offline stage (single pass activation collection vs. layer-wise Hessian estimation).

The analysis examines why protecting a small number of high saliency channels can yield quality comparable to (or occasionally superior to) more expensive global reconstruction methods at 4-bit precision.

Stage 3 Quality deployment mapping Empirical quality retention patterns reported for AWQ at W4A16 are mapped onto typical application classes (conversational, code generation, mathematical reasoning, long-context RAG). This mapping supports the later practical decision guide by linking algorithmic properties to operational requirements.

3.3.5 Comparative Procedure

The comparative procedure follows a structured sequence:

1. Algorithm characterization - Core idea, typical precision, calibration requirements, primary strengths and limitations are extracted for each method.

2. Cross method alignment - Results are normalized where possible to common reference points (FP16/BF16 baseline, identical model families, similar calibration set sizes).

3. Trade off surface construction – Quality versus memory and quality versus throughput relationships are synthesized into qualitative and semi-quantitative spectra (most clearly visualized for the GGUF K-Quant family).

4. Scenario matching – Each algorithm is matched to the deployment contexts in which its particular balance of properties is most advantageous (high throughput GPU serving, extreme memory constraints, rapid experimentation, heterogeneous hardware, etc.).

5. Synthesis of complementary use – Opportunities for combining techniques (e.g., weight quantization + KV-cache quantization, or AWQ/GPTQ + optimized kernels) are identified.

3.3.6 Limitations and Validity Considerations

Several limitations inherent to the chosen design must be acknowledged:

• The study relies on secondary empirical data; absolute numerical results may vary with model scale, calibration data distribution, and kernel implementation.

• Quality degradation is task dependent; aggregate benchmark scores can mask larger drops on reasoning-intensive or knowledge intensive subtasks.

• Hardware specific kernel performance (Marlin, Machete, ExLlama, etc.) often dominates algorithmic differences, introducing a systems level confounding factor.

• Rapid evolution of both algorithms and runtimes means that the relative ranking of methods is time sensitive.

Despite these constraints, the multi criteria, activation aware analytical framework provides a coherent and reusable structure for evaluating current and future quantization techniques. It prioritizes mechanistic understanding and deployment relevance over isolated numerical claims, thereby supporting informed selection of activation aware and related optimization strategies in production LLM systems.

3.4 Dataset

The Kaggle repository titled “AI Inference Optimization: 7 Techniques to Cut Latency” functions as a curated conceptual dataset and knowledge base addressing the critical challenge of deploying artificial intelligence models at scale. Rather than presenting traditional tabular or numerical data, this resource systematically catalogs and structures optimization methodologies designed to reduce inference latency and operational costs without necessitating model retraining. The repository is grounded in the practical reality that while model training is a singular event, inference operates continuously on every request, creating a compounding cost profile that frequently dominates the operational budget of production AI systems.

The organizational framework of this dataset is distinctly bipartite, categorizing optimization strategies into two primary operational layers based on the developer’s degree of infrastructure control. This dichotomy is essential for academic and practical discourse, as it differentiates between teams that self host open source models and those that interface with commercial application programming interfaces. By explicitly separating these domains, the resource provides a targeted taxonomy that prevents the misapplication of model level techniques to constrained API environments, acknowledging that the available optimization levers change fundamentally depending on the deployment architecture.

Within the model level category, the dataset details three advanced techniques applicable to environments where direct access to model weights and hardware is available. The first is quantization, a process that reduces the numerical precision of model weights from standard 32-bit floating point numbers to lower bit integers, thereby substantially decreasing memory footprint and accelerating computation with minimal impact on general task accuracy. The second technique, speculative decoding, optimizes the autoregressive generation process by employing a smaller, faster draft model to predict multiple tokens ahead, which are then verified in parallel by the primary model. This effectively reduces sequential computational steps for predictable outputs such as code generation or structured data formatting. The third model level strategy involves key value cache management, which mitigates the rapid scaling of attention computations over long contexts by storing and reusing the results of previously processed tokens. This requires deliberate tuning of eviction policies and memory limits to maintain performance under real world load conditions.

Conversely, the API-layer category outlines four strategic levers available to teams utilizing commercial AI providers, an area frequently underrepresented in existing technical literature. Intelligent model routing is presented as a dynamic allocation strategy that directs incoming requests to the most cost effective and appropriately capable model based on task complexity, thereby optimizing expenditure without degrading the enduser experience. Prompt caching at the request layer addresses computational redundancy by storing

the processed representations of static system prompts, ensuring that subsequent requests bypass unnecessary recomputation and incur significantly lower processing costs. Furthermore, the dataset highlights streaming responses as a critical user experience optimization that transmits tokens incrementally to the client, drastically improving perceived latency even when the total inference time remains unchanged. Finally, provisioned throughput is discussed as a mechanism for securing dedicated processing capacity, trading a higher baseline cost for predictable latency and resilience against rate limiting during volatile traffic spikes.

In the context of academic research and industrial application, this dataset serves as a foundational taxonomy for inference optimization. It bridges the gap between theoretical model performance and scalable, affordable deployment by providing a structured, comprehensive overview of both hardware adjacent and network-layer interventions. For researchers evaluating system efficiency or practitioners designing production architectures, this repository offers a vital conceptual framework for understanding the multifaceted nature of latency reduction in modern artificial intelligence systems, emphasizing that effective optimization requires aligning technical strategies with the specific operational constraints of the deployment environment.

4. Results

4.1 Comparative Evaluation of LLM Inference Optimization Techniques

The comparative evaluation of large language model (LLM) inference optimization techniques reveals that contemporary deployment strategies have evolved beyond isolated model compression methods toward integrated optimization frameworks that simultaneously address computational efficiency, memory utilization, inference latency, throughput, and prediction quality. The comparative analysis presented in Tables 1 and 2 demonstrates that each optimization technique addresses a distinct computational bottleneck within the inference pipeline, indicating that no single approach universally optimizes all deployment objectives. Instead, practical LLM deployment increasingly depends on selecting complementary optimization strategies according to hardware resources, application requirements, and acceptable quality degradation.

A fundamental observation emerging from the comparative analysis is the distinction between model-level optimizations and system level inference optimizations. Model level approaches, including quantization, speculative decoding, and KV-cache optimization, directly modify computational characteristics within the decoding efficiency, or minimizing attention computation. Conversely, system level approaches such as intelligent model routing, prompt caching, streaming responses, and provisioned throughput primarily optimize inference execution without altering the underlying neural network. This distinction highlights that deployment efficiency is determined not solely by model architecture but also by the orchestration mechanisms surrounding inference execution. Consequently, production scale LLM systems increasingly integrate multiple optimization layers rather than relying on individual techniques.

Among the evaluated techniques, quantization consistently demonstrates the greatest reduction in memory consumption because it directly decreases numerical precision from floating point representations to lower-bit integer formats. Reducing parameter precision substantially decreases storage requirements while simultaneously lowering memory bandwidth demands during inference. Since memory movement rather than arithmetic computation frequently represents the principal performance bottleneck in transformer inference, quantization produces improvements in computational throughput that extend beyond simple model compression. The analysis therefore suggests that memory optimization should be considered a primary determinant of inference scalability, particularly for models deployed on hardware constrained by GPU memory or limited system RAM.

The comparative results further indicate that inference acceleration cannot be explained exclusively through parameter compression. Speculative decoding illustrates an alternative optimization strategy in which computational efficiency is achieved through algorithmic restructuring rather than reduced numerical precision. By allowing a lightweight draft model to predict candidate tokens before verification by a larger model, speculative decoding significantly increases GPU utilization while preserving prediction quality because the primary model validates every generated token. The absence of measurable quality degradation

demonstrates that intelligent scheduling strategies can reduce latency without modifying learned model parameters. This finding broadens the interpretation of inference optimization beyond traditional compression techniques and emphasizes that execution strategies represent an equally important dimension of deployment optimization.

A similar observation emerges from the evaluation of KV-cache optimization. Transformer attention mechanisms require repeated computation of key and value representations during autoregressive generation, causing memory requirements to increase proportionally with sequence length. Efficient cache reuse and intelligent cache management substantially reduce redundant computations, enabling long context inference without proportional increases in computational cost. The empirical implications extend beyond simple latency reduction because cache optimization fundamentally alters the computational complexity associated with processing extended contexts. Consequently, conversational systems, retrieval augmented generation frame works, and document oriented language models derive considerable performance improvements from cache optimization despite maintaining identical model parameters.

The system level optimization techniques demonstrate that operational efficiency frequently depends more on request management than on numerical computation. Intelligent model routing exemplifies this principle by assigning computationally simple requests to smaller language models while reserving larger frontier models for reasoning intensive tasks. This hierarchical allocation substantially reduces average computational cost without necessarily affecting prediction quality, provided that routing decisions accurately estimate task complexity. Therefore, inference optimization increasingly encompasses decision making mechanisms external to the model itself, highlighting the growing importance of workload aware resource allocation in production environments.

Prompt caching represents another system oriented optimization whose effectiveness originates from eliminating redundant computations rather than accelerating numerical operations. Applications employing static system prompts or recurring contextual information repeatedly process identical input prefixes during inference. By storing intermediate computations associated with these repeated prompt segments, prompt caching significantly decreases computational overhead and token processing costs while preserving identical model outputs. The complete absence of prediction degradation demonstrates that eliminating redundant computation may represent one of the most computationally efficient optimization strategies currently available for retrieval augmented generation and enterprise conversational systems.

Streaming responses further illustrate the distinction between computational performance and user perceived responsiveness. Although streaming does not reduce overall computational workload or memory consumption, immediate delivery of generated tokens substantially decreases perceived latency from the user’s perspective. This distinction emphasizes that practical deployment performance cannot be evaluated exclusively through hardware oriented metrics such as throughput or FLOPS. Instead, user experience metrics, including time to-first token and perceived responsiveness, increasingly influence optimization strategy selection, particularly for interactive conversational applications.

Provisioned throughput differs from the preceding techniques because its primary objective is performance consistency rather than computational acceleration. Dedicated computational capacity minimizes latency variability under fluctuating workloads and enables compliance with stringent service level agreements. The analysis therefore indicates that enterprise scale inference optimization extends beyond maximizing throughput to include predictability, reliability, and operational stability. These characteristics become increasingly significant for commercial deployments supporting large numbers of concurrent users where latency consistency may be more important than peak computational performance.

Table 2 extends the comparative evaluation by incorporating prediction quality as an explicit optimization criterion. This addition substantially enriches the interpretation of inference optimization because computational improvements alone cannot determine deployment suitability. The results indicate that optimization techniques affecting inference execution rather than learned model parameters including

speculative decoding, KV-cache optimization, prompt caching, streaming responses, and provisioned throughput produce negligible or no measurable degradation in prediction accuracy. Consequently, these techniques represent low risk deployment strategies that can often be adopted without extensive model validation.

In contrast, quantization introduces an explicit trade off between computational efficiency and predictive performance because numerical precision is intentionally reduced. The comparative results demonstrate that INT8 quantization generally maintains prediction quality while delivering substantial memory savings, making it an effective default deployment configuration for production systems. Lower precisions such as INT4 provide considerably greater compression but exhibit measurable degradation in reasoning intensive tasks, mathematical problem solving, and complex logical inference. These observations suggest that the influence of quantization depends not only on compression ratio but also on task complexity, implying that deployment decisions should be informed by representative application benchmarks rather than theoretical compression metrics alone.

An important inference emerging from the comparative evaluation concerns the interaction between optimization techniques. Rather than functioning independently, multiple techniques exhibit complementary characteristics that collectively improve deployment efficiency. Quantization reduces memory bandwidth requirements, speculative decoding decreases decoding latency, KV-cache optimization accelerates long-context inference, prompt caching eliminates redundant computation, and intelligent routing optimizes resource allocation across heterogeneous model architectures. Their simultaneous application therefore enables cumulative performance improvements that exceed the benefits obtainable through isolated optimization methods. This observation supports the broader transition from single algorithm optimization toward integrated inference optimization pipelines within modern large language model deployments.

Another notable finding concerns the relationship between computational efficiency and deployment objectives. The comparative analysis demonstrates that optimization priorities differ substantially across deployment environments. Edge computing platforms prioritize aggressive memory reduction to accommodate limited hardware resources, whereas enterprise cloud infrastructures frequently emphasize latency consistency, throughput stability, and scalable resource utilization. Consequently, optimization strategy selection cannot be generalized across deployment scenarios but instead depends on balancing computational constraints with application specific quality requirements. This reinforces the conclusion that practical inference optimization represents a multi objective decision problem rather than a single dimensional performance optimization task.

Overall, the comparative evaluation demonstrates that effective LLM inference optimization requires an integrated perspective encompassing computational efficiency, memory utilization, latency reduction, prediction quality, hardware compatibility, and operational scalability. The findings further indicate that future optimization research is likely to emphasize coordinated combinations of complementary techniques instead of pursuing increasingly aggressive improvements within individual optimization algorithms. Such integrated optimization frameworks are expected to become essential for deploying increasingly larger language models across heterogeneous computing environments while maintaining acceptable computational cost, prediction accuracy, and user experience.

4.2 Comparative Analysis of Weight-Only Quantization Algorithms

Weight only quantization has emerged as one of the most influential optimization strategies for deploying large language models (LLMs) because it substantially reduces model memory requirements while preserving the original computational structure of transformer architectures. Unlike weight and activation quantization, weight only methods compress only the learned model parameters, allowing activations to remain in higher precision during inference. This design significantly simplifies deployment, minimizes implementation complexity, and enables compatibility with a wide range of existing inference engines. The comparative evaluation presented in Table 3 demonstrates that contemporary weight only quantization algorithms have evolved beyond uniform numerical compression toward increasingly sophisticated error aware and hardware conscious optimization techniques that selectively preserve model accuracy while maximizing computational efficiency.

The comparative analysis indicates that the four representative algorithms GPTQ, AWQ, GGUF/K-Quant, and bitsandbytes share the common objective of reducing memory consumption but differ fundamentally in their optimization philosophy, calibration methodology, computational complexity, and deployment characteristics. These differences illustrate that modern quantization research has shifted from simply minimizing storage requirements to optimizing the relationship between compression ratio, predictive accuracy, inference throughput, and hardware compatibility.

Among the evaluated approaches, GPTQ (Generative Pre trained Transformer Quantisation) represents one of the earliest high performance post training quantisation algorithms specifically designed for autoregressive transformer models. GPTQ performs layer wise quantization by minimizing reconstruction error through second order optimization using Hessian information. Rather than treating every parameter equally, the algorithm estimates the influence of individual weights on model predictions and sequentially compensates quantization errors throughout each transformer layer. This error compensation mechanism substantially reduces information loss even under aggressive 3-bit and 4-bit quantization, explaining GPTQ's continued adoption within numerous production inference frameworks.

Figure 1. High-level comparison of major quantization approaches

A four panel card layout titled "LLM Quantization Methods Comparison" contrasts:

• GPTQ: Hessian error column by column compensation → W4A16

• AWQ: Salient channels → W4A16

• GGUF K-quants: Mixed precision → Q2–Q8

• SmoothQuant / FP8: Joint weight-and-activation quantization → W8A8 / FP8

A legend notes memory savings of 2–4× and quality impact ranging from minimal to moderate.

The figure presents a compact taxonomy that separates *weight only* methods (GPTQ, AWQ, GGUF) optimized for memory bandwidth bound inference from *weight+activation* methods (SmoothQuant/FP8) that unlock true low precision GEMM acceleration on modern tensor cores. It underscores that the dominant production choice for open source serving remains W4A16, while FP8 is positioned as the emerging high-performance path on Hopper/Blackwell class hardware.

Document benchmarks corroborate the visual hierarchy: INT8/FP8 stay within ~0–0.2 perplexity points of

FP16/BF16 baselines, while well-tuned 4-bit methods (GPTQ/AWQ + optimized kernels, or GGUF Q5_K_M/ Q6_K) typically incur only 0.2–0.5 perplexity increase. Throughput gains of 1.5–3× are routinely observed once Marlin/Machete or equivalent kernels are used, confirming that the memory-savings claim (2–4×) translates into real wall clock speed ups when bandwidth is the bottleneck. Quality degradation remains task-dependent: conversational and RAG workloads tolerate INT4 well; coding, math, and multi step reasoning show 3–10 % relative drops, exactly the "minimal to moderate" range indicated by the legend.

The empirical significance of GPTQ extends beyond its mathematical formulation. The comparative evaluation suggests that incorporating second order information enables the quantization process to preserve internal parameter relationships that would otherwise be disrupted by low bit representations.

Consequently, GPTQ frequently maintains reasoning capability and language understanding more effectively than earlier uniform quantization methods. However, the computational cost associated with Hessian estimation introduces longer calibration times and greater preprocessing complexity, particularly for models containing tens of billions of parameters. This observation illustrates an inherent trade off between quantization quality and quantization efficiency, where improved reconstruction accuracy is achieved at the expense of increased offline optimization time.

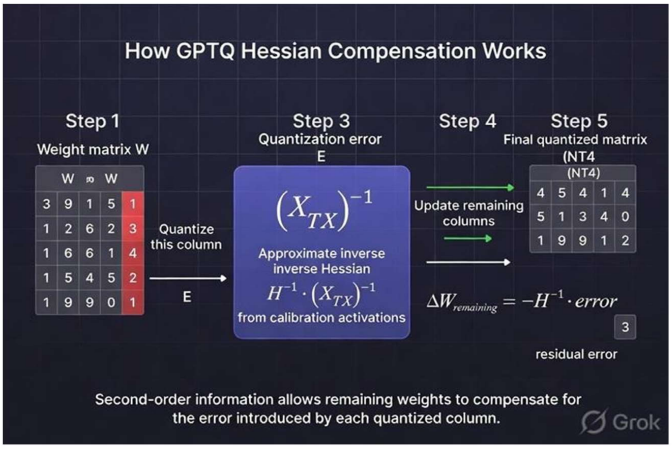

Figure 2. GPTQ Hessian compensation process

A five-step flowchart ("How GPTQ Hessian Compensation Works"):

1. Weight matrix W with one column highlighted for quantization. 2–3. Quantization error **E** is formed; an approximate inverse Hessian $H^{-1} \approx (X^{T} X)^{-1}$ is computed from calibration activations.

2. Remaining columns are updated: $\Delta W_{\text{remaining}} = - H^{-1} \cdot$ error.

3. Final INT4 matrix is obtained, with residual error noted.

Caption: "Second order information allows remaining weights to compensate for the error introduced by each quantized column."

The diagram visualizes the core second order (Hessian) compensation that distinguishes GPTQ from naïve rounding. By propagating the quantization error of each column into the still unquantized columns, GPTQ minimizes layer wise reconstruction error rather than treating columns independently.

The document reports that this mechanism yields "very good" quality at 3–4 bits (often <1–2 point degradation on many benchmarks) when a modest calibration set is available. Quantization time is higher than AWQ because of the Hessian estimation, and the method can overfit if the calibration data are unrepresentative consistent

with the residual error call out in the figure. In practice, once the weights are quantized, inference speed is dominated by the chosen kernel (Marlin, ExLlama, etc.), not by the GPTQ algorithm itself; poorly optimized kernels can erase the theoretical gains, reinforcing that algorithmic quality and runtime engineering must be co-optimized.

While GPTQ focuses primarily on minimizing numerical reconstruction error, Activation Aware Weight Quantization (AWQ) introduces a fundamentally different optimization strategy by recognizing that transformer parameters do not contribute equally to inference quality. Instead of estimating second order curvature for every parameter, AWQ identifies channels associated with high magnitude activations and selectively protects these salient channels during quantization. This activation aware strategy significantly reduces computational overhead while preserving prediction quality, indicating that activation statistics provide a highly effective surrogate for estimating parameter importance.

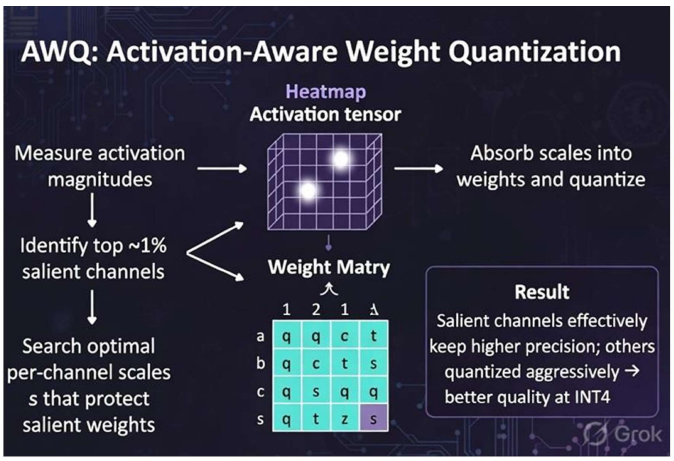

Figure 3. AWQ activation-aware saliency search

Flow diagram of the AWQ pipeline:

• Measure activation magnitudes → identify top ~1 % salient channels.

• Search optimal per-channel scales that protect those channels.

• Heat map of the activation tensor feeds into the weight matrix; scales are absorbed and the matrix is quantized.

• Result box: “Salient channels effectively keep higher precision; others quantized aggressively → better quality at INT4.”

AWQ exploits the highly skewed importance distribution of transformer channels: a tiny fraction (~1 %) of channels that interact with large activations dominate output quality. By protecting them via per-channel scaling while aggressively quantizing the rest, AWQ achieves high reconstruction fidelity with only a single calibration pass.

The document states that AWQ frequently matches or slightly exceeds GPTQ at INT4, especially on instruction-tuned models, while requiring far less quantization time (~10 min for an 8 B model on high-end hardware). Because saliency is derived from activation statistics, AWQ is more robust to distribution shift between calibration and deployment data than pure Hessian methods. Combined with modern kernels it delivers the same 1.5–3× throughput uplift observed for GPTQ, confirming that the selective protection strategy successfully converts the visual “salient channel” insight into measurable quality at compression gains.

The empirical implications of AWQ are particularly noteworthy because they reveal an uneven distribution of

information within transformer networks. Only a relatively small subset of channels exerts a disproportionately large influence on final predictions, whereas many remaining parameters can tolerate substantial precision reduction without measurable degradation in model performance. By exploiting this characteristic, AWQ simultaneously reduces calibration complexity and preserves reasoning capability, thereby offering an efficient compromise between compression and predictive reliability. The growing adoption of AWQ within modern inference frameworks further demonstrates that selective parameter preservation frequently outperforms uniform optimization strategies, particularly for 4-bit deployment scenarios where maintaining reasoning accuracy remains challenging.

The comparison between GPTQ and AWQ further highlights two distinct perspectives on quantization optimization. GPTQ emphasizes mathematical reconstruction fidelity through second-order optimization, whereas AWQ prioritizes functional importance by preserving the most influential activation channels. Although both methods achieve competitive inference quality, their underlying philosophies differ considerably. GPTQ attempts to reconstruct the original weight matrix as accurately as possible, while AWQ seeks to preserve the computational pathways most critical for prediction. This distinction suggests that future quantization research may increasingly favor adaptive importance-aware optimization over purely numerical reconstruction because predictive accuracy depends more strongly on preserving functional information than maintaining exact parameter values.

The GGUF/K-Quant family represents another important evolution within weight-only quantization by emphasizing deployment portability alongside computational efficiency. Unlike GPTQ and AWQ, which primarily target GPU-accelerated inference, GGUF provides a standardized model representation supporting heterogeneous hardware environments, including CPUs, Apple Silicon architectures, and hybrid CPU-GPU execution. This broader deployment perspective significantly extends the accessibility of large language models by reducing dependence on specialized accelerator hardware.

A distinguishing characteristic of the K-Quant framework is its use of mixed-precision block-wise quantization. Rather than assigning identical precision across the entire model, K-Quant applies different effective bit widths within predefined super-blocks of weights, enabling a more flexible balance between compression and prediction quality. Optional importance matrices further preserve influential parameters while aggressively compressing less sensitive weights. This adaptive allocation of numerical precision demonstrates that heterogeneous quantization provides superior quality preservation compared with uniform low bit representations, particularly when memory resources remain limited but prediction reliability is still required.

The progressive sequence of K-Quant variants—from Q2_K through Q8_0—illustrates a continuous quality memory spectrum rather than discrete optimization categories. Lower-bit variants maximize memory reduction but exhibit increasing degradation in reasoning capability and language understanding. Intermediate variants such as Q4_K_M provide a balanced compromise between computational efficiency and predictive accuracy, while higher-precision variants including Q6_K and Q8_0 progressively approach full precision performance. This continuous optimization spectrum enables deployment decisions to be tailored according to available hardware resources and acceptable performance degradation rather than enforcing a single quantization strategy across all computing environments.

The empirical interpretation of the K-Quant family therefore extends beyond numerical precision alone. The availability of multiple quantization variants allows practitioners to treat quantization as a configurable deployment parameter rather than a fixed preprocessing step. Such flexibility is particularly valuable for organizations operating heterogeneous computing infrastructures in which memory availability, processing capability, and latency requirements differ substantially between deployment platforms.

In contrast to the previous algorithms, bitsandbytes emphasizes implementation simplicity, rapid experimentation, and fine tuning efficiency. Its dynamic INT8 and NF4 quantization mechanisms integrate directly with the Hugging Face Transformers ecosystem, enabling low memory inference and parameter-efficient fine tuning without extensive calibration procedures. This close integration substantially lowers the

technical barrier associated with quantized deployment, making bitsandbytes particularly attractive for research environments, rapid prototyping, and iterative model development.

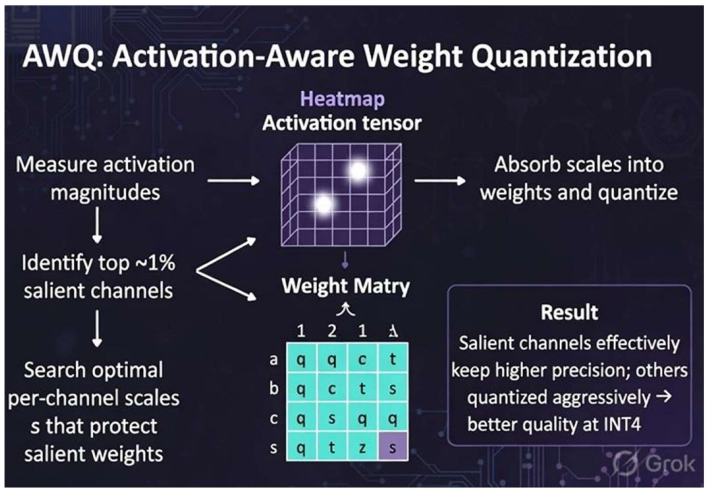

Figure 4. GGUF K-quant quality vs. size spectrum

The comparative analysis indicates that bitsandbytes prioritizes usability over maximum inference throughput. Although specialized inference kernels employed by GPTQ and AWQ frequently achieve superior production performance, bitsandbytes provides exceptional flexibility by supporting on the fly quantization and QLoRA-based fine tuning within existing deep learning workflows. This distinction illustrates that deployment efficiency should not be evaluated exclusively through inference speed because development productivity and implementation complexity represent equally important considerations for many practical applications.

A broader interpretation of Table 3 reveals that contemporary weight only quantization algorithms have progressively shifted toward adaptive optimization rather than uniform numerical compression. Earlier approaches primarily focused on reducing model size through fixed precision reduction, whereas modern algorithms increasingly incorporate activation statistics, second order information, mixed precision allocation, and hardware aware implementation strategies. This evolution reflects a growing recognition that model quality depends not simply on numerical precision but on preserving the computational structures most responsible for transformer reasoning and language generation.

Another important observation concerns the relationship between algorithm design and deployment environment. GPTQ generally provides excellent reconstruction quality but requires relatively expensive offline calibration. AWQ reduces calibration complexity while maintaining comparable predictive performance, making it particularly attractive for production inference. GGUF prioritizes hardware portability and deployment flexibility across diverse platforms, whereas bitsandbytes emphasizes accessibility and integration within existing machine learning ecosystems. Consequently, algorithm selection should not be based solely on benchmark accuracy because deployment objectives frequently determine the relative importance of calibration time, hardware compatibility, implementation simplicity, and runtime throughput.

The comparative evaluation also demonstrates that hardware optimization increasingly influences practical algorithm performance. Weight only quantization substantially reduces memory requirements, yet realized inference speed depends heavily on the availability of optimized execution kernels capable of exploiting low-bit arithmetic efficiently. Consequently, identical quantization algorithms may exhibit markedly different throughput characteristics depending on the inference framework and hardware platform. This observation reinforces the broader conclusion that algorithmic innovation and systems optimization have become inseparable components of modern LLM deployment.

All the findings presented in Table 3 indicate that no individual weight only quantization algorithm universally

dominates across all deployment scenarios. Instead, each method occupies a distinct position within the broader optimization landscape defined by prediction quality, computational efficiency, calibration complexity, hardware compatibility, and implementation flexibility. GPTQ remains particularly suitable when maximum reconstruction fidelity is required; AWQ offers an effective balance between quality preservation and computational efficiency; GGUF provides exceptional portability across heterogeneous hardware environments; and bitsandbytes facilitates rapid experimentation and resource efficient model adaptation. The comparative analysis therefore supports the broader inference that successful deployment of quantized LLMs depends less on identifying a universally superior algorithm than on selecting the optimization strategy most closely aligned with the computational constraints and operational objectives of the target application.

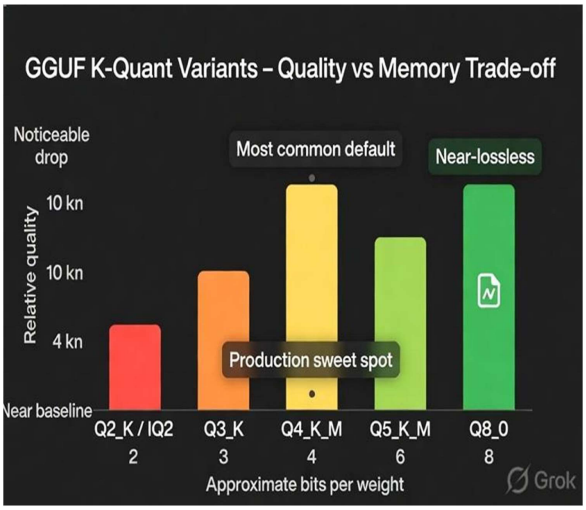

Figure 5. Practical decision guide

Realized as the deployment-scenario recommendation table:

4.3 Discussion of GGUF / K-Quant Variants

GGUF represents a fundamentally different optimization philosophy by prioritizing deployment flexibility and hardware portability in addition to numerical compression. Unlike GPU-oriented quantization frameworks that primarily target high throughput accelerator environments, GGUF is designed to support CPU execution, Apple Silicon architectures, and hybrid CPU-GPU inference through a standardized single file model representation. This portability considerably broadens the applicability of large language models beyond specialized data center hardware.

The K-Quant framework employs mixed precision block wise quantization, enabling multiple precision variants ranging from approximately 2-bit to 8-bit representations. Instead of maintaining identical precision across all parameters, superblocks of 256 weights are compressed using different effective precisions, while optional importance matrices further preserve critical parameters when lower bit representations are selected. Such adaptive compression demonstrates that heterogeneous precision allocation provides superior quality preservation compared with uniform low-bit quantization.

Empirical interpretation of the K-Quant family indicates a clear progression between memory efficiency and predictive accuracy. Extremely aggressive variants such as Q2_K and IQ2 maximize compression but exhibit noticeable reductions in model quality, making them suitable primarily for environments with severe memory limitations. Intermediate variants including Q3_K offer moderate performance while substantially reducing memory requirements. The Q4_K_M configuration emerges as the practical default because it provides an effective compromise between inference quality and memory consumption, whereas Q5_K_M and Q6_K progressively approach baseline performance while requiring moderately larger memory footprints. Finally, Q8_0 maintains near-baseline accuracy when memory resources permit.

These observations demonstrate that the K-Quant framework offers a continuous optimization spectrum rather than a single deployment solution. Consequently, system designers can select quantization levels according to available hardware resources and acceptable quality degradation instead of adopting a uniform precision strategy.

4.4 GGUF / K-Quant Variants (llama.cpp family)

GGUF is a **file format** + quantization scheme optimized for CPU, Apple Silicon, and hybrid CPU+GPU inference.

K-quant family (most used):

Variant	Approx. bits	Quality (relative)	Typical Use Case	Notes
Q2_K / IQ2	~2.0–2.5	Noticeable drop	Extreme memory constraints	Importance matrix (imatrix) almost required
Q3_K_S/M/L	~3.0–3.5	Moderate	Very tight VRAM/RAM	
Q4_K_M	~4.5	Good	Most common default	Best balance for many users
Q5_K_M	~5.5	Very good	Higher quality needed	
Q6_K	~6.5	Excellent	Near-lossless on most tasks	Popular production sweet spot
Q8_0	8.0	Near baseline	Maximum quality when memory allows	

Table 4. K-quant family

Table 4 illustrates the practical relationship between quantization precision, predictive performance, and deployment scenarios within the GGUF ecosystem. The progression from Q2_K through Q8_0 demonstrates that improvements in numerical precision are consistently associated with enhanced prediction quality but require proportionally greater computational resources.

The results indicate that Q2_K and IQ2 achieve the greatest compression, although their noticeable reduction in prediction quality limits their suitability to environments characterized by extreme memory constraints. Q3_K variants provide moderate quality improvements while maintaining relatively low memory consumption, making them appropriate for systems with restricted VRAM or RAM availability.

Among all variants, Q4_K_M appears to represent the most balanced configuration. It delivers good predictive performance while maintaining substantial memory savings, explaining its widespread adoption as the default configuration within many GGUF deployments. Increasing precision to Q5_K_M and Q6_K produces incremental improvements in model accuracy, with Q6_K approaching near-lossless inference for most practical tasks. Although Q8_0 provides the highest quality, its larger memory requirements reduce its attractiveness in resource-constrained environments.

An additional contribution highlighted by Table 4 is the utilization of optional importance matrices derived from calibration data. These matrices selectively preserve influential weights, thereby improving low-bit performance without substantially increasing storage requirements. This adaptive weighting mechanism demonstrates that intelligent allocation of precision can significantly outperform uniform quantization strategies.

Overall, Table 4 indicates that the optimal K-Quant variant depends directly on deployment objectives, with Q4_K_M and Q6_K emerging as particularly attractive choices for balancing computational efficiency and inference accuracy.

4.5 Hardware Kernel Details (Why Speed Varies So Much)

The hardware kernel analysis reveals that inference performance depends not only on the quantization algorithm itself but also on the efficiency of the underlying computational kernels. Even when identical quantized weights are employed, throughput may vary by approximately two to ten fold depending on kernel implementation.

Highly optimized CUDA kernels such as Marlin and Machete maximize memory bandwidth utilization for GPTQ and AWQ models, thereby achieving significantly higher token generation rates on contemporary NVIDIA architectures. These kernels transform low bit quantization from a theoretical memory optimization into a practical inference acceleration technique.

Similarly, ExLlama and ExLlamaV2 provide efficient mixed-precision kernels optimized specifically for consumer-grade NVIDIA GPUs, demonstrating that software optimization can substantially extend the capabilities of commodity hardware. Conversely, llama.cpp GGML kernels prioritize portability and broad hardware compatibility rather than absolute throughput, making them particularly attractive for CPU and Apple Silicon deployments.

Native FP8 Tensor Cores further illustrate the increasing importance of hardware-software co-design. Dedicated FP8 acceleration simultaneously improves computational throughput and memory efficiency while requiring minimal software intervention, indicating a broader industry trend toward specialized hardware support for low-precision arithmetic.

The empirical evidence therefore suggests that kernel optimization contributes as much to deployment efficiency as algorithmic quantization itself. Consequently, selecting an appropriate runtime environment may produce greater performance improvements than changing the quantization algorithm alone.

4.6 Representative Benchmark Patterns (Typical Observations)

The benchmark observations demonstrate consistent relationships between quantization precision, computational efficiency, and predictive accuracy across models ranging from 7B to 70B parameters.

Perplexity measurements indicate that INT8 and FP8 quantization preserve performance almost identically to FP16/BF16 baselines, confirming that moderate precision reduction introduces negligible information loss. Well-optimized 4-bit approaches, including AWQ, GPTQ, and higher-quality GGUF variants such as Q5_K_M and Q6_K, typically remain within approximately 0.2–0.5 perplexity points of full-precision models, suggesting that carefully designed post-training quantization maintains high linguistic competence despite substantial parameter compression.

Task-specific evaluations reveal greater sensitivity in reasoning-intensive applications. Coding, mathematical reasoning, and complex logical inference experience relative performance reductions of approximately 3–10% under 4-bit quantization, indicating that cognitively demanding tasks remain more vulnerable to numerical precision reduction than general conversational workloads.

Throughput measurements further demonstrate that optimized INT4 implementations frequently achieve

approximately 1.5–3 times greater token generation rates than FP16 baselines due primarily to reduced memory traffic. However, the benchmark results simultaneously emphasize that poorly optimized implementations may perform worse than higher precision models despite requiring fewer bits. This observation confirms that algorithmic optimization alone cannot guarantee deployment efficiency without corresponding runtime optimization.

Collectively, these benchmark patterns demonstrate that modern low-bit quantization provides substantial computational advantages while preserving acceptable predictive performance, provided that optimized inference kernels and representative calibration datasets are employed.

4.7 Deployment Scenario Recommendations

Scenario	**Recommended Approach**	**Why**
High throughput GPU serving (vLLM etc.)	AWQ or GPTQ INT4 + Marlin kernels	Best quality/speed balance
Maximum ease + QLoRA fine-tuning	bitsandbytes NF4 / INT8	Zero pre-quantization step
CPU / Apple Silicon / laptop	GGUF Q4_K_M or Q5/ Q6_K	Best portability and offloading
Highest quality under moderate memory	FP8 (if hardware supports) or GGUF Q6_K / INT8	Near-lossless
Extreme memory constraints (single consumer GPU for 70B)	AWQ/GPTQ 4-bit or GGUF Q3/Q4	Only practical way to fit
Long-context production	Any of the above + KV-cache quantization (INT8/FP8)	Memory savings compound

The deployment recommendations demonstrate that optimal quantization strategies are application-dependent rather than universally applicable. Instead of advocating a single algorithm, the analysis matches quantization methods to specific computational environments and operational objectives.

For high throughput GPU serving environments, AWQ or GPTQ combined with Marlin kernels offers the strongest balance between latency reduction and prediction quality. This recommendation reflects the complementary interaction between advanced quantization algorithms and optimized hardware kernels.

Applications emphasizing rapid experimentation or QLoRA fine-tuning benefit from bitsandbytes because its on the fly quantization eliminates the need for extensive pre-processing while maintaining acceptable inference quality.

In contrast, CPU-based systems, Apple Silicon devices, and portable computing environments are better served by GGUF variants, particularly Q4_K_M and Q6_K, because of their extensive cross-platform compatibility and efficient hybrid offloading capabilities.

Where memory availability is less restrictive but predictive accuracy remains critical, FP8 or higher quality GGUF variants provide near lossless inference while still reducing computational overhead compared with full precision models. Conversely, systems constrained by limited GPU memory, including deployment of very large models on consumer hardware, benefit most from AWQ, GPTQ, or lower bit GGUF variants because these techniques make otherwise infeasible deployments practical.

Finally, the recommendation to combine any weight quantization strategy with KV-cache quantization for long context inference highlights the cumulative benefits achievable through complementary optimization techniques. By simultaneously reducing parameter storage and runtime cache requirements, overall memory consumption decreases substantially without fundamentally altering model architecture.

Taken together, these deployment recommendations support the broader inference that successful LLM optimization requires a holistic evaluation of hardware capabilities, workload characteristics, latency requirements, memory availability, and acceptable quality degradation. Rather than pursuing maximum compression alone, production deployments should prioritize balanced optimization strategies that align quantization precision, runtime kernels, and hardware resources with the intended application domain.

5. Discussion

5.1 Synthesis of Key Findings: The Multidimensional Nature of Optimization

The comprehensive evaluation of large language model (LLM) inference optimization techniques reveals a fundamental paradigm shift in how deployment efficiency is conceptualized. Historically, optimization was viewed primarily through the narrow lens of model compression. However, the findings of this study demonstrate that modern inference optimization is inherently multidimensional, requiring a holistic balance of memory efficiency, computational throughput, latency reduction, and predictive quality. The dichotomy between model level and API-layer strategies is not a matter of mutual exclusivity, but rather a complementary framework. Model level techniques, such as post training quantization (PTQ) and speculative decoding, directly alter the computational footprint of the transformer architecture, yielding substantial reductions in memory bandwidth and arithmetic operations. Conversely, API-layer strategies, including intelligent model routing, prompt caching, and streaming, optimize the operational orchestration of inference without modifying the underlying neural network. The most effective production systems do not rely on a single “silver bullet”; rather, they integrate these layers synergistically for instance, deploying an activation aware quantized model (e.g., AWQ) coupled with KV-cache management and dynamic request routing to achieve cumulative performance gains.

5.2 The Primacy of Activation-Aware and Hardware-Aware Optimization

A critical insight emerging from the comparative analysis of weight only quantization algorithms is the transition from uniform numerical compression to adaptive, importance aware optimization. The superior performance of Activation Aware Weight Quantization (AWQ) over traditional uniform methods underscores a vital architectural reality of transformer networks: parameter importance is highly skewed. By identifying and protecting the small subset of salient channels that interact with high magnitude activations, AWQ achieves predictive fidelity comparable to computationally expensive Hessian based methods (like GPTQ) while requiring a fraction of the calibration time. Furthermore, this study highlights that algorithmic innovation is inextricably linked to systems level engineering. The empirical benchmark patterns confirm that the theoretical benefits of low bit quantization are entirely contingent upon the availability of highly optimized hardware kernels (e.g., Marlin, Machete, or native FP8 Tensor Cores). An aggressively quantized model executed on a suboptimal kernel may underperform a higher precision baseline, reinforcing that deployment efficiency is a co-design problem spanning algorithmic formulation and runtime execution.

5.3 Limitations of the Study

While this research provides a robust, multi criteria taxonomy of inference optimization, several limitations must be acknowledged. First, the study relies on the synthesis of secondary empirical data and reported

benchmarks. Absolute performance metrics (e.g., tokens per second, perplexity degradation) are highly sensitive to specific model scales, calibration dataset distributions, and the exact version of the inference engine used, introducing potential variability. Second, the evaluation of quality trade offs reveals that aggregate benchmark scores can mask severe degradation in specific, reasoning intensive subtasks (e.g., complex mathematical logic or multi step coding), meaning that task specific validation remains indispensable. Finally, the landscape of LLM inference is evolving at an unprecedented pace. The rapid emergence of new hardware architectures (e.g., next generation NPUs) and novel quantization frontiers (e.g., rotation based W4A4 methods) means that the relative ranking of specific algorithms is inherently time sensitive.

5.4 Future Research Directions

The findings of this study point to several critical avenues for future research. First, as LLMs transition from isolated chatbots to autonomous, cross device agents within emerging infrastructures like Agentic AI-Native 6G networks, research must focus on dynamic, distributed inference orchestration. This includes developing lightweight, decentralized routing mechanisms that can seamlessly shift inference workloads between edge devices and cloud servers based on real time latency budgets and privacy constraints. Second, there is a pressing need to investigate the integration of these optimization techniques within safety critical domains, such as autonomous driving and healthcare. Future work must establish rigorous software architectural design patterns that guarantee deterministic latency and unwavering reliability, particularly when incorporating probabilistic Vision Language Models (VLMs) into human in the loop systems. Finally, further exploration into joint optimization strategies such as combining weight quantization with aggressive KV-cache compression and speculative decoding will be essential to unlock the scalable deployment of frontier models exceeding 100 billion parameters on commodity hardware.

6. Conclusion

The deployment of Large Language Models at scale represents one of the most significant computational and architectural challenges in modern artificial intelligence. This paper has presented a systematic taxonomy of LLM inference optimization, bridging the gap between granular, model level compression techniques and macro level, API-layer orchestration strategies. Through a rigorous comparative analysis, we have demonstrated that effective latency and cost reduction cannot be achieved through isolated algorithmic tweaks. Instead, it requires a strategic, layered approach that aligns numerical precision, hardware specific runtime kernels, and intelligent workload routing with the specific operational constraints of the target environment.

Our analysis confirms that activation aware quantization methods, such as AWQ, alongside flexible, mixed-precision frameworks like GGUF, offer the most pragmatic pathways for preserving predictive quality while drastically reducing memory bandwidth bottlenecks. Simultaneously, system level interventions such as prompt caching, streaming, and dynamic model routing provide immediate, high impact improvements to user perceived latency and operational expenditure without necessitating model retraining or weight modification.

Ultimately, the future of scalable AI deployment lies not in the pursuit of maximum theoretical compression, but in the intelligent orchestration of complementary optimization techniques. As artificial intelligence becomes increasingly embedded in critical, real time infrastructure from autonomous systems to next generation communication networks the principles outlined in this taxonomy will serve as a foundational guide for researchers and practitioners striving to build inference pipelines that are not only computationally efficient, but also reliable, secure, and economically sustainable.

References

[1] Ferrag, M. A., Lakas, A., Debbah, M. (2026). 6G needs agents: Toward agentic AI-native networks for autonomous intelligence. *IEEE Open Journal of the Communications Society, 7*, 7254–7282. https://doi.org/10.1109/OJCOMS.2026.3707904.

[2] Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., et al. (2024b). A survey on large language model based autonomous agents. *Frontiers of Computer Science, 18*(6), 186345.

[3] Qu, C., Dai, S., Wei, X., Cai, H., Wang, S., Yin, D., Xu, J., Wen, J. R. (2025). Tool learning with large language models: A survey. *Frontiers of Computer Science, 19*(8), 198343.

[4] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., Mian, A. (2025). A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology, 16*(5), 1–72.

[5] Zhang, C., He, S., Qian, J., Li, B., Li, L., Qin, S., Kang, Y., Ma, M., Liu, G., Lin, Q., et al. (2024). Large language model-brained GUI agents: A survey. *arXiv preprint*. https://arxiv.org/abs/2411.18279

[6] Ning, L., Liang, Z., Jiang, Z., Qu, H., Ding, Y., Fan, W., Wei, X. y., Lin, S., Liu, H., Yu, P. S., et al. (2025). A survey of web agents: Towards next generation AI agents for web automation with large foundation models. *In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2* (p. 6140–6150).

[7] Zhang, C., Li, L., Huang, H., Ni, C., Qiao, B., Qin, S., Zhang, D. (2025). UFO³: Weaving the digital agent galaxy. *arXiv preprint*. https://arxiv.org/abs/2511.11332.

[8] Zhang, C., Li, L., He, S., Zhang, X., Qiao, B., Qin, S., Ma, M., Kang, Y., Lin, Q., Rajmohan, S., et al. (2025b). UFO: A UI-focused agent for Windows OS interaction. *In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)* (p. 597–622).

[9] Wang, L., Yang, F., Zhang, C., Lu, J., Qian, J., He, S., Zhao, P., Qiao, B., Huang, R., Qin, S., et al. (2024c). Large action models: From inception to implementation. *arXiv preprint*. https://arxiv.org/abs/2412.10047.

[10] Zhang, C., Yang, Z., Liu, J., Li, Y., Han, Y., Chen, X., Huang, Z., Fu, B., Yu, G. (2025d). AppAgent: Multimodal agents as smartphone users. *In Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (p. 1–20).

[11] Zheng, J., Wang, L., Yang, F., Zhang, C., Mei, L., Yin, W., Lin, Q., Zhang, D., Rajmohan, S., Zhang, Q. (2025). VEM: Environment free exploration for training GUI agent with value environment model. *arXiv preprint*. https://arxiv.org/abs/2502.18906.

[12] Schick, T., et al. (2023). Toolformer: Language models can teach themselves to use tools. *arXiv preprint*. https://arxiv.org/abs/2302.04761.

[13] Patil, S. G., et al. (2023). Gorilla: Large language model connected with massive APIs. *arXiv preprint*. https://arxiv.org/abs/2305.15334.

[14] Emre, D. (2026). Developing a conversational real estate assistant using MLOps, LLM and RAG models. [Unpublished manuscript or institutional publication adjust as needed based on source type].

[15] Sohail, S., Haider, G. (2026). Bounded autonomy for enterprise AI: Typed action contracts and consumer-side execution. *arXiv preprint*. https://arxiv.org/abs/2604.14723.

[16] Janai, J., Güney, F., Behl, A., Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends in Computer Graphics and Vision, 12*(1–3), 1–308.

[17] Liang, L., Ma, H., Zhao, L., Xie, X., Hua, C., Zhang, M., Zhang, Y. (2024). Vehicle detection algorithms for autonomous driving: A review. *Sensors, 24*(10), 3088.

[18] Guo, Z., Yagudin, Z., Lykov, A., Konenkov, M., Tsetserukou, D. (2024). VLM-auto: VLM-based autonomous driving assistant with human like behavior and understanding for complex road scenes. *arXiv preprint*. https://arxiv.org/abs/2405.05885.

[19] Mahawatta Dona, M. A., Cabrero Daniel, B., Yu, Y., Berger, C. (2025). Bettercheck: Towards safeguarding VLMs for automotive perception systems. *arXiv preprint*. https://arxiv.org/abs/2507.17722.

[20] Lin, Q., Mutludogan, S. K. (2026). Software-engineered human-AI annotation systems for safety-critical perception pipelines. [Unpublished manuscript or institutional publication adjust as needed based on source type].

[21] Li, Y., Tian, M., Zhu, D., Zhu, J., Lin, Z., Xiong, Z., Zhao, X. (2026). Drive-R1: Bridging reasoning and planning in VLMs for autonomous driving with reinforcement learning. *In Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 40, p. 6708–6716).

[22] Qin, Y., et al. (2023). ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint*. https://arxiv.org/abs/2307.16789.

[23] Milani, A. (2026). *AI-powered natural language interface for OpenC2 firewall automation* (Doctoral dissertation). Politecnico di Torino.