TECHNICAL PAPER — SHABBE LAB, INC.

Comparative Evaluation of LLM Quantization Formats for Japanese Business Document Tasks: Speed, Quality, and Memory Efficiency of GGUF Q4_K_M vs AWQ INT4

Author Kou Ohashi **Organization** Shabbe Lab, Inc. (株式会社喋ラボ) **Published** June 2026

ABSTRACT

We evaluate two LLM quantization formats — GGUF Q4_K_M and AWQ INT4 — on Japanese business document tasks (meeting summarization and structured JSON action-item extraction) using a constrained consumer GPU environment: NVIDIA RTX 4060 Laptop with 8 GB VRAM. Experiments are conducted across three context lengths (8K, 16K, 32K tokens) with two open-weight models: Qwen3-8B and Gemma4-E4B. Key findings: (1) GGUF Q4_K_M achieves 100% JSON validity across all 24 trials; (2) AWQ INT4 via vLLM fails to launch at even the minimum 8K context due to KV cache exhaustion — model weights alone occupy 5.71 GiB, leaving at most 0.73 GiB for KV cache against a required 1.12 GiB; (3) extraction quality (F1) is stable across context lengths within each model but differs sharply between models (Qwen3-8B: 0.615 vs Gemma4-E4B: 0.319); (4) Gemma4-E4B maintains near-constant throughput across all context lengths (35–41 tok/s) thanks to its MoE architecture, while Qwen3-8B degrades to 8.0 tok/s at 32K due to partial CPU offloading. These results provide concrete, hardware-specific guidance for quantization format selection in production deployments on 8 GB VRAM.

1. Introduction

Demand for local LLM inference is growing rapidly. Privacy requirements, latency constraints, and cloud API cost pressures are driving enterprises to run open-weight models on their own infrastructure. Quantization is the enabling technology: a 7–8B model at FP16 requires ~14 GB of VRAM, but 4-bit quantization compresses it to 3–6 GB, making consumer GPUs viable.

Two quantization formats currently dominate local deployment: GGUF (the llama.cpp ecosystem, CPU/GPU/Apple Silicon portable) and AWQ INT4 (NVIDIA GPU-optimized, used by vLLM, TGI). AWQ is widely reported to offer higher quality retention (~95%) than GGUF Q4_K_M (~92%), based on English general-purpose benchmarks such as MMLU and HumanEval. However, these benchmarks do not answer the question practitioners actually face:

On an 8 GB VRAM consumer GPU, what actually runs — and what is the quality on Japanese business document tasks?

1.1 Contributions

- First controlled comparison of GGUF Q4_K_M vs AWQ INT4 on Japanese business document tasks (meeting summarization and structured JSON extraction)
- Quantitative characterization of GPU offload behavior across 8K/16K/32K context

lengths on 8 GB VRAM

- Empirical demonstration that AWQ INT4 (vLLM) cannot launch at any practical context length on 8 GB VRAM, with a complete KV cache budget analysis
- Practical decision guidelines for quantization format selection under memory constraints

2. Background

2.1 Quantization Basics

LLM weights stored at FP16 consume 2 bytes per parameter. A 7–8B model requires ~14 GB at full precision. Quantization reduces per-weight storage to 4 bits (INT4), compressing the model to ~3.5 GB. Quality degradation depends on the quantization algorithm and model size; larger models are generally more robust to low-bit quantization due to weight redundancy.

2.2 GGUF Q4_K_M

GGUF is the native format of the llama.cpp project, supported by Ollama, LM Studio, and most local inference tools. The K-Quant family uses mixed-precision block quantization, allocating slightly more bits to attention and embedding layers deemed critical. The Q4_K_M variant ("Medium") is the de facto standard for consumer deployment, offering a balance of compression and quality. When VRAM is insufficient, llama.cpp supports hybrid CPU+GPU execution by offloading layers to system RAM; however, PCIe bandwidth then becomes the throughput bottleneck.

2.3 AWQ INT4

Activation-aware Weight Quantization (AWQ) identifies weights with high activation magnitudes — i.e., weights whose values most affect model output — and preserves them at higher precision while aggressively quantizing the remainder. This produces better quality retention than naive uniform quantization at the same bit width. AWQ requires NVIDIA CUDA and is optimized for GPU tensor core throughput. The vLLM serving framework, which is the standard deployment runtime for AWQ, pre-allocates a KV cache of configurable size at startup, consuming a fixed VRAM budget before any inference begins.

2.4 Context Length and KV Cache

The KV cache stores attention keys and values for all tokens in the current context. Its size scales linearly with context length. For an 8B-class model, the KV cache requires approximately 1.12 GiB at 8K tokens, 2.25 GiB at 16K, and 4.50 GiB at 32K. On 8 GB VRAM, the competition between model weights and KV cache is the central design constraint of this study.

3. Experimental Setup

3.1 Hardware

Component	Specification
GPU	NVIDIA RTX 4060 Laptop GPU
VRAM	8 GB (8,188 MiB)
CUDA	12.8
Driver	580 series
OS	Ubuntu Linux
Runtimes	llama-cpp-python 0.3.26 (CUDA cu124) / vLLM 0.22.1

3.2 Models

Model	Parameters	Architecture	Weight size (Q4_K_M)
Qwen3-8B	8B (Dense)	36 layers	~5.0 GB
Gemma4-E4B	4B effective (MoE)	42 layers	~5.4 GB

Qwen3-8B is Alibaba's dense 8B model with strong multilingual performance, particularly in Japanese (Shisa evaluation: 8.08/10). Gemma4-E4B is Google's Mixture-of-Experts model with ~4B active parameters per token, which significantly reduces KV cache requirements. Both models are licensed under Apache 2.0.

3.3 Tasks

Task A: Meeting Summarization

Given a synthetic Japanese meeting transcript, generate a 200–300 character summary. Evaluated via ROUGE-L against a reference summary.

Task B: Action Item Extraction (JSON)

Extract structured action items from the same transcript in the following JSON schema:

```
{
  "action_items": [
    {
      "task": "string",
      "assignee": "string or null",
      "due_date": "YYYY-MM-DD or null",
      "priority": "high | medium | low"
    }
  ]
}
```

Evaluated on JSON validity rate and field-level F1 score against ground truth. Chat templates were applied for all inference runs. Qwen3 thinking mode was disabled via the chat template parameter.

3.4 Test Data

ID	Tokens

	Meeting type		Action items	Context conditions
T1	Weekly standup	2,591	7	8K, 16K, 32K
T2	Monthly review	6,515	12	16K, 32K
T3	Quarterly planning	12,681	20	32K only

All test data are entirely synthetic Japanese meeting transcripts. No real meeting recordings or confidential information were used. Each transcript includes a mix of action items with explicit assignees and due dates alongside items with missing information (`null` fields), to test extraction under realistic ambiguity.

3.5 Experiment Matrix

Model	Format	8K	16K	32K	Trials
Qwen3-8B	GGUF Q4_K_M	✓	✓	✓	12
Gemma4-E4B	GGUF Q4_K_M	✓	✓	✓	12
Qwen3-8B	AWQ INT4	**FAILED**	**FAILED**	**FAILED**	3 (all OOM)

3.6 Metrics

Metric	Task	Definition
Throughput (tok/s)	Both	Average tokens/second during generation phase
TTFT (s)	Both	Time from prompt submission to first output token
Peak VRAM (GB)	Both	Maximum VRAM usage measured via pynvml
GPU offload layers	Both	Layers loaded on GPU / total layers
JSON validity rate	Task B	Fraction of outputs parseable by json.loads()
Action item F1	Task B	Token-level partial match F1 on task field vs ground truth

4. Results

4.1 GGUF Q4_K_M: Throughput and Memory

Qwen3-8B (GGUF Q4_K_M)

Context	tok/s	TTFT (s)	Peak VRAM	GPU layers
8K	**28.6**	2.5	6.62 GB	36/36 (full)
16K	21.7	6.3	7.69 GB	33/36 (partial)
32K	8.0	18.4	7.46 GB	18/36 (partial)

At 8K, all 36 layers fit in VRAM and throughput reaches 28.6 tok/s. At 16K, the growing KV cache forces 3 layers to CPU, reducing throughput to 21.7 tok/s. At 32K, 18 of 36 layers are offloaded; PCIe transfer becomes the bottleneck and throughput collapses to 8.0 tok/s — a 3.6× slowdown versus 8K. TTFT increases from 2.5s to 18.4s (7.4×).

Gemma4-E4B (GGUF Q4_K_M)

Context	tok/s	TTFT (s)	Peak VRAM	GPU layers
8K	**41.0**	1.3	4.50 GB	42/42 (full)
16K	38.5	2.4	5.11 GB	42/42 (full)
32K	35.8	4.2	6.49 GB	42/42 (full)

Gemma4-E4B's MoE architecture results in a smaller KV cache footprint; all 42 layers remain on GPU at every context length. Throughput degrades only 13% from 8K (41.0 tok/s) to 32K (35.8 tok/s), and TTFT stays under 4.2s even at 32K — a qualitatively different scaling profile from Qwen3-8B.

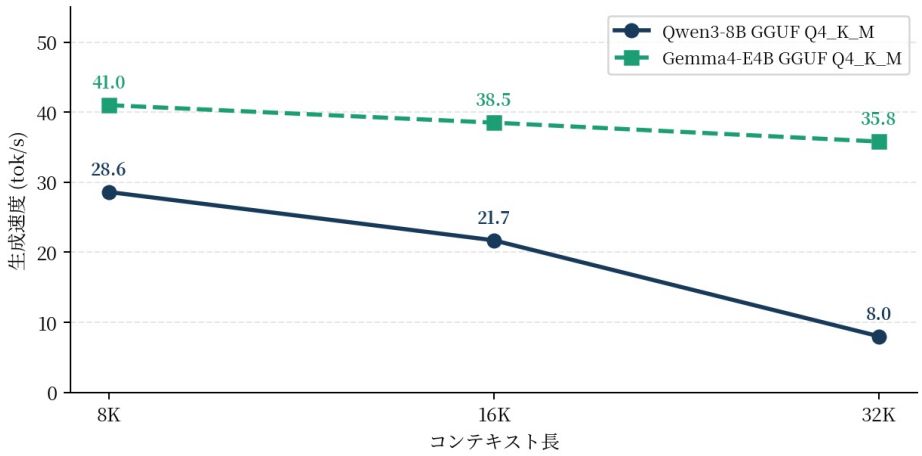

Figure 1: Generation throughput (tok/s) by context length

4.2 AWQ INT4: Launch Failure Analysis

> **Result: AWQ INT4 (Qwen3-8B-AWQ, vLLM 0.22.1) cannot launch at any experimental context length on 8 GB VRAM.** Model weights occupy 5.71 GiB on GPU. With `gpu_memory_utilization=0.97`, only 0.73 GiB remains for KV cache — insufficient for 8K context (requires 1.12 GiB).

gpu_memory_utilization	KV available	Max context	8K launch
0.90	0.19 GiB	~1,392 tokens	**Failed**
0.93	0.42 GiB	~3,056 tokens	**Failed**
0.95	0.57 GiB	~4,160 tokens	**Failed**
0.97	0.73 GiB	~5,280 tokens	**Failed**

To confirm the failure is hardware-capacity limited rather than a software misconfiguration, we verified AWQ functionality at a reduced context: `ctx=4096, util=0.97` launched successfully and completed inference at 22.67 tok/s with valid JSON output. The 4,096-token context fits within the 5,280-token KV budget; the 8,192-token context does not.

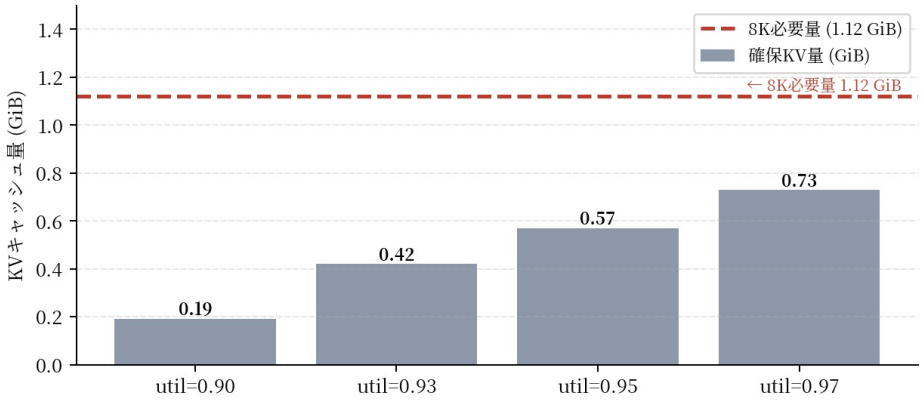

Figure 5: AWQ INT4: available KV cache vs 8K requirement across gpu_memory_utilization values

4.3 Quality: JSON Extraction

JSON validity rate was 100% across all 24 GGUF trials. All outputs were parseable without post-processing.

Action Item Extraction F1

Model	8K	16K	32K	Mean
Qwen3-8B	0.615	0.612	0.619	**0.615**
Gemma4-E4B	0.250	0.365	0.343	0.319

Qwen3-8B achieves stable F1 of 0.61–0.62 regardless of context length — a notable finding, as the 3.6× throughput degradation at 32K has no corresponding quality degradation. Gemma4-E4B's lower F1 is attributable to low recall: field-level accuracy for assignee, due_date, and priority is 1.0, but the model omits action items present in ground truth. This reflects a trade-off between the model's tendency toward concise, high-precision output and comprehensive coverage.

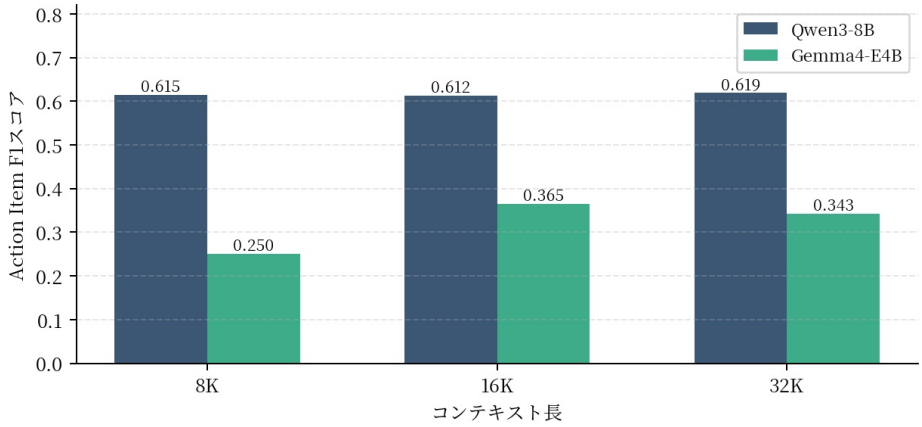

Figure 3: Action item extraction F1 score by context length

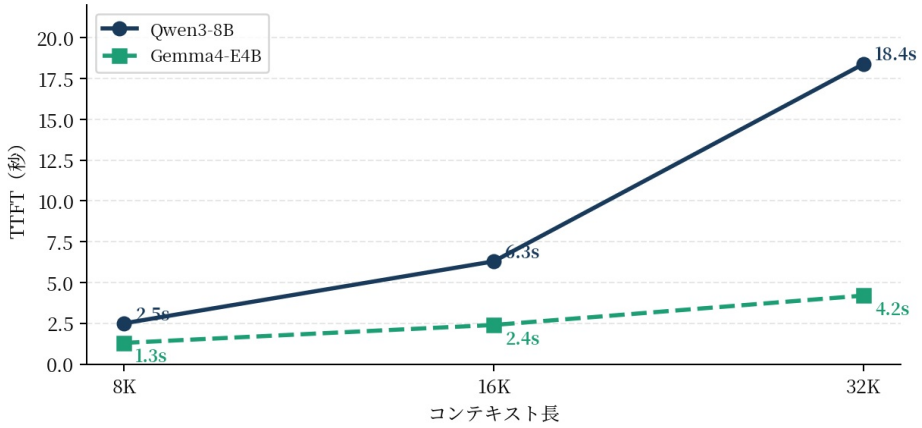

Figure 4: Time to first token (TTFT, seconds) by context length

4.4 Summary: Speed–Quality Trade-off

Model	Format	8K tok/s	32K tok/s	Mean F1	VRAM @ 32K
Qwen3-8B	GGUF Q4_K_M	28.6	8.0	**0.615**	7.46 GB
Gemma4-E4B	GGUF Q4_K_M	41.0	35.8	0.319	6.49 GB
Qwen3-8B	AWQ INT4	—	—	—	**launch failed**

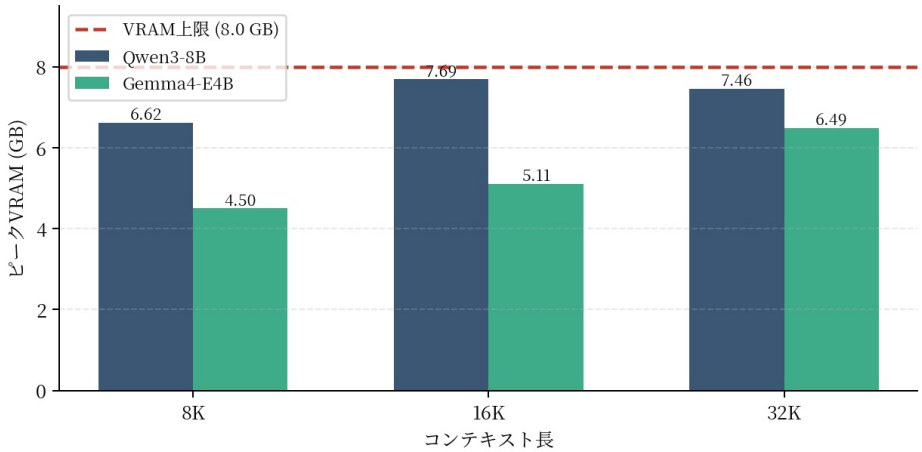

Figure 2: Peak VRAM usage (GB) by context length — dashed line = 8 GB limit

5. Discussion

5.1 The AWQ Quality Advantage Does Not Apply at 8 GB VRAM

AWQ is consistently reported to offer better quality retention than GGUF Q4_K_M (~95% vs ~92% on English benchmarks). Our results do not refute this on hardware where AWQ can run. However, on 8 GB VRAM with an 8B model, AWQ INT4 via vLLM cannot launch at the minimum practical context length. The quality advantage is irrelevant when the runtime cannot start.

> Quantization format selection must be evaluated at the **system level** — runtime × hardware × context length — not solely on model quality metrics.

5.2 Context Length Affects Throughput, Not Quality

For Qwen3-8B, throughput drops 3.6× from 8K to 32K while F1 remains within 0.007 (0.612–0.619). This decoupling is practically important: long-context meeting transcripts can be processed at acceptable quality, but operators must budget for the latency cost. At 32K, the 18.4s TTFT makes Qwen3-8B unsuitable for real-time applications but acceptable for batch offline processing.

5.3 Architecture Determines VRAM Scaling

Gemma4-E4B's MoE design activates only ~4B parameters per token, producing a smaller KV cache that fits on GPU even at 32K. Qwen3-8B's dense architecture scales KV cache with context length, triggering layer offloading at 16K and 32K. This demonstrates that the appropriate unit of analysis for VRAM planning is active parameter count and attention architecture, not total parameter count.

5.4 Practical Decision Guide

```
Quantization selection for 8 GB VRAM:

Speed-critical, context < 16K?
  → Gemma4-E4B GGUF Q4_K_M  (41 tok/s, 4.5 GB VRAM)

Quality-critical, context ≤ 8K?
  → Qwen3-8B GGUF Q4_K_M    (28.6 tok/s, 6.6 GB VRAM)

Quality-critical, context 16K–32K?
  → Qwen3-8B GGUF Q4_K_M    (accept throughput penalty)

AWQ INT4 (vLLM)?
  → Requires ≥16 GB VRAM for 8B models at 8K context
```

6. Limitations and Future Work

- **Synthetic test data.** All transcripts are artificial. Performance on real enterprise meeting recordings may differ, particularly on domain-specific vocabulary and informal speech patterns.
- **Model coverage.** Results are specific to Qwen3-8B and Gemma4-E4B. Other model families (Llama-3, Phi-4, Mistral) may exhibit different VRAM and quality profiles.
- **Runtime specificity.** AWQ was evaluated under vLLM 0.22.1. Alternative AWQ runtimes (ExLlamaV2, llama.cpp with AWQ support) may achieve different memory efficiency.

- **Future work.** We plan a Phase 2 evaluation on Apple Silicon (Mac mini, 64 GB unified memory) comparing GGUF Q4_K_M vs MLX 4-bit on larger models (Qwen3-32B, Qwen3-30B-A3B MoE), and will add ROUGE-L and LLM-as-judge evaluation for the summarization task.

7. Conclusion

We conducted a controlled evaluation of GGUF Q4_K_M and AWQ INT4 on Japanese business document tasks under 8 GB VRAM constraints. The main findings are:

1. **AWQ INT4 is not viable on 8 GB VRAM** for 8B models at practical context lengths. The combination of 5.71 GiB model weights and vLLM's pre-allocated KV cache leaves no room for the minimum required context. This is a pure hardware capacity limitation confirmed by successful inference at reduced context (4K).
2. **GGUF Q4_K_M achieves 100% JSON validity** across all 24 trials, demonstrating reliable structured output without constrained decoding.
3. **Quality is stable across context lengths; throughput is not.** Qwen3-8B F1 varies only 0.007 across 8K–32K while throughput drops 3.6×. The quality cost of long-context inference is latency, not accuracy.
4. **MoE architecture provides throughput stability under context pressure.** Gemma4-E4B sustains 35–41 tok/s across all context lengths at the cost of lower extraction recall (F1: 0.319 vs 0.615).

Appendix A: Environment

Component	Version / Value
GPU	NVIDIA RTX 4060 Laptop (8,188 MiB)
CUDA Toolkit	12.8
Python	3.11.15
llama-cpp-python	0.3.26 (CUDA cu124)
vLLM	0.22.1
rouge-score	latest
pynvml	latest

AWQ-specific environment flags: `VLLM_WORKER_MULTIPROC_METHOD=spawn` (avoids CUDA re-initialization in forked subprocess); `VLLM_USE_FLASHINFER_SAMPLER=0` (native sampler, no JIT compilation required).

Appendix B: Test Data Statistics

ID	Type	Tokens	Action items	Null due_date
T1	Weekly standup	2,591	7	2
T2	Monthly review	6,515	12	3
T3	Quarterly planning	12,681	20	2

Citation

```
@techreport{ohashi2026quantization,
  title     = {Comparative Evaluation of LLM Quantization Formats
               for Japanese Business Document Tasks},
  author    = {Ohashi, Kou},
  institution = {Shabbe Lab, Inc.},
  year      = {2026},
  url       = {https://github.com/kouohhashi/llm-quantization-benchmark}
}
```