

COMET Sensitivity-Guided Mixed Precision Quantization for the WMT26 Model Compression Task

Angad Ripudaman Singh Bajwa

Independent Researcher

angadisthere@gmail.com

Abstract

The WMT26 Model Compression shared task asks participants to reduce the footprint of a general-purpose large language model for machine translation (MT) deployment while preserving translation quality, under a fixed evaluation budget of model size, VRAM usage, and inference speed. We report a systematic comparison of post-training compression strategies applied to the constrained-track base model, google/gemma-3-12b-it, across the three required language directions (Czech→German, English→Chinese (Simplified), English→Arabic (Egyptian)). We evaluate an uncompressed BF16 baseline and a vLLM-served engine-level control against six compression strategies: two bitsandbytes baselines (8-bit and 4-bit), two calibrated weight-only quantization methods (AWQ and GPTQ, both W4A16), vLLM’s native dynamic FP8 weight quantization, and a novel COMET-sensitivity-guided mixed-precision scheme that assigns each of the 48 decoder layers an independent BF16/INT8/INT4 tier via a size-budget knapsack. Quality is measured with chrF and three COMET variants. Speed is full-process wall-clock time (including model load) on a held-out blind test set, and size is the on-disk safetensors footprint. Our mixed-precision submission achieves the best quality-size-speed trade-off in our pool: it is the fastest system measured (1.76 sentences/sec, batch 16), within 0.4 chrF and a few COMET points of the uncompressed baseline across all three language pairs, at roughly 40% of the baseline’s on-disk size (9.15 GB vs. 23 GB).

1 Introduction

Large language models (LLMs) are competitive general-purpose machine translation (MT) systems, but their size makes them expensive to deploy: memory footprint, VRAM, and decoding latency all scale with parameter count. The WMT26 Model Compression shared task (WMT26 Model

Compression Shared Task Organizers, 2026) targets this: participants take a fixed general-purpose LLM and compress it as aggressively as possible while retaining translation quality, with the trade-off evaluated along quality (COMET, MetricX, and an LLM-as-judge), model size (on-disk and VRAM), and inference speed (wall-clock time). The *constrained* track we enter fixes the base model (google/gemma-3-12b-it) and the calibration/fine-tuning data pool, so that only the compression technique differs, across three directions: Czech→German (ces-deu), English→Chinese Simplified (eng-zho_Hans), and English→Arabic Egyptian (eng-ara_EG). Submissions are scored by the organizers on a single NVIDIA H100 (up to 80 GB), with per-pair results and quality-size/quality-speed Pareto plots.

We evaluate eight systems along the task’s three axes:¹ an uncompressed BF16 baseline (HuggingFace Transformers), the same weights served by vLLM as an engine-level control, and six compression variants, namely bitsandbytes 8-/4-bit, calibrated AWQ and GPTQ (both W4A16), vLLM dynamic FP8, and a novel per-layer, quality-sensitivity-guided mixed-precision scheme. Our contributions are: (i) a controlled, apples-to-apples comparison of six compression strategies against an uncompressed baseline, using the organizers’ own harness (modelzip) on all three language pairs; (ii) a COMET-sensitivity-guided mixed-precision method that measures per-layer translation-quality sensitivity directly (rather than via a weight-magnitude heuristic) and solves an exact size-budget knapsack, giving the best quality-size-speed trade-off in our pool; and (iii) a characterization of how the task’s fresh-subprocess timing protocol makes load-time vs. checkpoint-time quantization the dominant factor in the speed rank-

¹Code and all submissions: <https://github.com/angadbajwa23/wmt26-model-compression>

ing, separating otherwise comparable 4-bit methods by more than $2\times$.

2 Related Work

Post-training quantization. Weight-only quantization stores weights at low bit-width while keeping activations in half precision. Dettmers et al. (2022) introduced 8-bit inference (LLM.int8()) with outlier-preserving mixed-precision decomposition, underlying the bitsandbytes library behind our bnb-q8/bnb-q4 baselines (the latter using the NF4 data type from QLoRA, Dettmers et al., 2023). GPTQ (Frantar et al., 2023) uses approximate second-order information to minimize layer-wise rounding error, and AWQ (Lin et al., 2024) protects activation-salient channels via per-channel scaling. Both yield accurate 4-bit W4A16 checkpoints and back our awq/gptq systems. 8-bit floating-point (FP8) formats retain more dynamic range than INT8 and can be applied online with per-tensor scales, as in vLLM’s *dynamic* mode (our vllm-fp8); all our vLLM systems use its PagedAttention engine (Kwon et al., 2023).

Sensitivity-guided mixed precision. Uniform-precision quantization is suboptimal because layers contribute unequally to task quality, so spending more bits on the most sensitive layers recovers most of the lost quality at a similar average bit-width. Our comet-mixed-precision submission follows this idea but, unlike proxy-based approaches (weight/activation statistics or perplexity), measures the actual downstream metric (COMET) after independently quantizing each of the 48 decoder layers, then solves an exact dynamic-programming knapsack over per-layer byte cost under a size budget.

The WMT model compression shared task. WMT26 (Gowda et al., 2026) is the second edition of this task; the first ran at WMT25 (Gaido et al., 2025) under the same quality-size-speed criteria we target here. Both participating teams that year approached compression through *depth* reduction rather than precision reduction: Moslem et al. (2025) apply iterative layer pruning guided by a layer-importance analysis, and Ponce et al. (2025) search the space of layer-collapse operations with an evolutionary method (GeLaCo) using a module-wise similarity fitness function, followed by fine-tuning and knowledge distillation to recover quality, with quantization applied as an

additional size reduction on top. Our submissions are complementary: we hold depth fixed and vary precision instead, but share the premise that layers are unequally important; our per-layer sensitivity sweep (Section 3.3) plays the role their importance/similarity criteria play in deciding which layers to remove.

3 Methodology

3.1 Task Definition and Constraints

The constrained track requires each system to be derived from google/gemma-3-12b-it (quantization, pruning, and distillation are all permitted), using only publicly released WMT26 data and past WMT test sets for calibration/fine-tuning. We use the WMT25 General MT test set as a reference-scored quality proxy and its blind source set for speed benchmarking, both provided by the organizers’ repository.

3.2 Calibration and Development Data

All methods requiring offline calibration (AWQ, GPTQ, and the final mixed-precision pass) draw from a single shared pool built from constrained-track data only. WMT26 training data is fetched per language pair with mtdata and normalized into the organizers’ paragraph-level JSONL schema. From this pool we build a per-pair calibration set mixing a seeded random sample of WMT26 training examples with the WMT25 General MT sources (always included), rendered through the model’s own chat template so that calibration activations match inference-time inputs. The sensitivity development set used by comet-mixed-precision (Section 3.3) is held out and disjoint from this pool, so per-layer sensitivity is not measured on data the quantizer fits to.

3.3 Compression Methods Evaluated

Table 1 summarizes the eight systems in our pool; all share the same base weights and differ only in compression method and serving engine.

bnb-q8/bnb-q4 apply bitsandbytes quantization at model-load time, with no offline calibration. awq/gptq are calibrated offline (via llm-compressor) on the shared calibration set of Section 3.2, producing static INT4, group-size-128 checkpoints served natively by vLLM’s compressed-tensors backend. vllm-fp8 uses vLLM’s *dynamic* FP8 weight quantization, applied at load time rather than baked into the checkpoint,

Table 1: Compression methods evaluated. All systems are derived from the same constrained-track base model.

System	Precision	Method	Serving engine
baseline	BF16	none (reference)	HF Transformers
bnb-q8	INT8	LLM.int8() (Dettmers et al., 2022)	bitsandbytes
bnb-q4	FP4 (NF4)	QLoRA-style NF4 (Dettmers et al., 2023)	bitsandbytes
awq	INT4, group size 128	AWQ (Lin et al., 2024)	vLLM
gptq	INT4, group size 128	GPTQ (Frantar et al., 2023)	vLLM
vllm-bf16	BF16	none (engine baseline)	vLLM
vllm-fp8	FP8 (dynamic, weights)	vLLM online FP8	vLLM
comet-mixed-precision	per-layer BF16/INT8/INT4	COMET-sensitivity knapsack + GPTQ	vLLM

so its compression appears only in VRAM and compute at serving time, not on disk.

comet-mixed-precision is our contribution, and the idea is simple: find out which layers actually matter for translation quality, then spend the bits there. It runs in three stages.

Measuring sensitivity. We quantize one decoder layer at a time, fake-quantizing its linear weights round-to-nearest to INT4 while the other 47 stay at BF16, translating the held-out development set, and recording how far wmt22-cometkiwi-da falls relative to the unquantized model before restoring the layer and moving on. Forty-eight passes give a per-layer price for dropping to 4 bits. We use the reference-free wmt22-cometkiwi-da as the guide signal so that every candidate can be scored directly on the calibration sources without references; it is a different metric from the reference-based wmt22-comet-da and the surface-level chrF we report at evaluation, so the allocation is not tuned on its own scorecard.

Assigning tiers. Each layer’s exact size at each precision comes from the safetensors header, so every choice has a known byte cost. A dynamic-programming knapsack then places each layer in BF16, INT8, or INT4, picking the assignment with the smallest total predicted quality drop that still fits a 9 GB budget; INT8 sensitivity is not measured separately but assumed to cost half the measured INT4 drop.

Quantizing. A single GPTQ pass with per-layer config_groups carries out the plan, touching only the INT8- and INT4-tier layers. Because the knapsack decides only decoder weights, the finished model lands at 9.15 GB rather than 9 (Section 4.3): the embedding and LM head, tokenizer, and config files sit outside its decision space.

3.4 Evaluation Protocol

All systems are evaluated with the organizers’ modelzip harness (evals/evaluate.sh / modelzip.evaluate), which invokes each submission’s run.sh as an independent subprocess with a fixed lang-pair, batch-size, input, and output, validating that output and input line counts match before accepting a run.

Quality. We report chrF (Popović, 2015), computed with sacrebleu, and three COMET (Rei et al., 2020) checkpoints computed with pymarian-eval: wmt22-comet-da (reference-based), and the reference-free wmt22-cometkiwi-da and wmt23-cometkiwi-da-xl. Quality is measured on the WMT25 reference-scored test set at batch size 32, one run per system per pair.

Speed. Per the organizer protocol, speed is measured separately from quality on ces-deu, using the WMT25 blind source set (456 sentences, no reference). After a warm-up, each system is timed at batch size 16 for 3 repetitions. Critically, each timed repetition is a fresh subprocess, so wall time spans the entire subprocess lifetime and includes model loading (safetensors deserialization, quantization setup, and vLLM engine init), not just translation of the 456 sentences. This deliberately rewards low cold-start-to-finish latency and penalizes expensive per-load overhead (e.g., on-the-fly dequantization) even when steady-state throughput is competitive. We report mean/min/max wall time over the 3 runs and a derived sentences-per-second figure (456 divided by the mean).

Size. Size is the on-disk footprint of each submission’s serialized weights directory (du -sh over safetensors shards plus tokenizer/config), the task’s stated size criterion.

3.5 Hardware

All results were collected on a single NVIDIA L40 (48 GB), our available development hardware, not the organizers’ official H100 (80 GB). Since all systems are compared under identical conditions, we expect relative rankings to be largely preserved, but *absolute* wall-clock numbers should not be taken as predictive of official timings; the FP8 submissions in particular may behave differently on the H100’s native FP8 tensor cores.

4 Results

4.1 Translation Quality

Table 2 reports chrF and the three COMET variants for all eight systems on all three language pairs.

No method causes a large quality collapse: every system is within 0.8 chrF of baseline, and in several cases slightly *above* it (bnb-q8 on ces-deu and eng-zho_Hans; both vLLM systems on eng-ara_EG). comet-mixed-precision is notably consistent, staying within 0.4 chrF of baseline on all three pairs and never falling more than 0.032 wmt22-comet-da below it, at ~40% of baseline size. The allocation is guided only by wmt22-cometkiwi-da, yet the quality it retains is visible in chrF, a surface metric with no COMET component, which is mild evidence that it is not simply overfitting its guide signal.

4.2 Inference Speed

Table 3 reports wall-clock speed on the wmt25-blind set (L40 GPU; see the hardware caveat in Section 3.5).

The ranking is dominated by *load-time* cost rather than steady-state throughput, a direct consequence of the fresh-subprocess-per-run protocol. The bitsandbytes systems are slowest by a wide margin as bnb-q8 takes $2.8\times$ the baseline’s wall time and $5.9\times$ that of comet-mixed-precision. This is because bitsandbytes re-quantizes weights on every load rather than reading a pre-quantized checkpoint. The fact that bnb-q8 is $2.0\times$ slower than bnb-q4 despite a $1.7\times$ larger checkpoint confirms the dominant term is per-load quantization work, not bytes moved. The pre-quantized, vLLM-native systems (awq, gptq, comet-mixed-precision) avoid this tax and beat the baseline outright, with comet-mixed-precision fastest overall.

4.3 Model Size

vllm-fp8’s on-disk size equals the BF16 baseline (Table 4): its FP8 is applied dynamically at load rather than serialized, so the size criterion gives it no credit for its VRAM-time compression. The lesson is general: compression that materializes only after load is invisible to the size metric and must be baked into the checkpoint to count.

4.4 Quality-Size-Speed Trade-offs

Table 5 combines macro-averaged quality with size and speed for all eight systems.

comet-mixed-precision is the one system in our pool that no other system beats on all three axes at once. It is the fastest we measured, it takes 40% of the baseline’s disk, and it gives up very little quality doing so: 0.30 macro-averaged chrF and 0.017 wmt22-comet-da against the baseline (the best-quality system we have), a smaller gap than bnb-q4 (0.53 chrF) or awq (0.76 chrF) pay at the same size or smaller.

The comparison that matters most is against uniform INT4. comet-mixed-precision spends 14% more disk than awq/gptq and buys back 0.20–0.46 chrF, while still running faster than either. Uniform quantization taxes every layer equally, whether or not that layer can afford it; the knapsack instead pushes the tolerant layers down to INT4 and keeps the handful of quality-critical ones at BF16 or INT8.

The bitsandbytes systems lose outright: bnb-q4 saves only 0.2 GB over awq/gptq at under half their speed, and bnb-q8 is $2.8\times$ slower than baseline. This is the cost of per-load re-quantization, not a fault of LLM.int8() or NF4, which target memory-constrained fine-tuning rather than repeated cold-start batch inference.

5 Conclusion

We compared six compression strategies for a 12B constrained-track MT model on the WMT26 task’s three axes, using the organizers’ harness. Our COMET-sensitivity-guided mixed-precision method gives the best overall trade-off in our pool: fastest measured inference (1.76 sent/s, 22% ahead of the next system), 40% of baseline size (9.15 GB), and quality within 0.4 chrF of baseline on all three pairs. This favours sensitivity-guided bit allocation over uniform low-bit quantization and over on-the-fly schemes not designed for cold-start inference.

One direction is especially promising. Since

Table 2: Translation quality on the WMT25 General MT test set (batch size 32, single run per system). Best value per column per language pair in **bold**.

Lang. pair	System	chrF	comet22-da	kiwi22-da	kiwi23-xl
ces-deu	baseline	63.5	0.6871	0.6682	0.5467
	bnb-q8	63.6	0.7118	0.6283	0.5230
	bnb-q4	63.2	0.6573	0.6357	0.5294
	awq	62.8	0.7106	0.6539	0.5400
	gptq	62.9	0.6141	0.6489	0.5339
	vllm-bf16	63.2	0.6468	0.6465	0.5340
	vllm-fp8	63.2	0.6362	0.6253	0.5209
	comet-mixed-precision	63.1	0.6606	0.6536	0.5364
eng-ara_EG	baseline	38.8	0.6272	0.5696	0.5036
	bnb-q8	38.6	0.5428	0.5741	0.5031
	bnb-q4	38.2	0.5731	0.5689	0.4957
	awq	38.0	0.5292	0.5507	0.4783
	gptq	38.3	0.6083	0.5762	0.5075
	vllm-bf16	39.2	0.5915	0.5786	0.5055
	vllm-fp8	39.1	0.5855	0.5777	0.5117
	comet-mixed-precision	38.7	0.5956	0.5723	0.4972
eng-zho_Hans	baseline	28.3	0.6772	0.6965	0.6076
	bnb-q8	28.4	0.6694	0.7091	0.6164
	bnb-q4	27.6	0.6550	0.6795	0.5823
	awq	27.5	0.6273	0.7000	0.5992
	gptq	27.9	0.6221	0.7214	0.6215
	vllm-bf16	28.0	0.6806	0.6946	0.5997
	vllm-fp8	28.2	0.6436	0.6954	0.5955
	comet-mixed-precision	27.9	0.6845	0.6850	0.5939

Table 3: Inference speed at batch size 16 (ces-deu, wmt25-blind, 456 sentences, mean of 3 runs). wall_time_sec is the full subprocess lifetime, including model loading. Sorted fastest to slowest.

System	Mean (s)	Min (s)	Max (s)	Sentences/s
comet-mixed-precision	259.4	244.4	284.0	1.76
gptq	315.8	289.5	333.4	1.44
awq	369.4	307.1	419.0	1.23
vllm-bf16	401.6	393.2	410.1	1.14
vllm-fp8	450.2	332.1	575.3	1.01
baseline	550.5	542.1	563.4	0.83
bnb-q4	767.6	744.6	795.5	0.59
bnb-q8	1534.1	1520.9	1554.2	0.30

all our methods are post-training and can only *lose* quality, SFT with knowledge distillation from the BF16 teacher could recover some of it and let the knapsack target a tighter budget, using the constrained-track data of Section 3.2.

Limitations

A few caveats bound how these results should be read. (i) All measurements used an NVIDIA L40 (48 GB), not the official H100 (80 GB); as discussed in Section 3.5, relative rankings should hold but absolute timings, FP8 especially, may shift on the official hardware. (ii) The INT8-tier sensitivity is estimated as half the measured INT4 drop rather than measured directly, to keep the sweep tractable (~ 1.5 – 2 h vs. 4 – 5 h); measuring it directly would

mainly refine the tier boundaries. (iii) The sweep measures each layer in isolation and the knapsack treats the drops as additive; modelling interaction effects between simultaneously quantized layers is a natural refinement. (iv) Sensitivity uses a single reference-free signal (wmt22-cometkiwi-da); guiding the allocation with the official metrics or a non-COMET signal (e.g., activation or curvature statistics) could give a better assignment, and comparing these signals is a natural next step.

References

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. [LLM.int8\(\): 8-bit matrix multiplication for transformers at scale](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.

Table 4: On-disk model size.

System	Size	Precision	Notes
baseline	23 GB	BF16	uncompressed reference
vllm-bf16	23 GB	BF16	identical weights to baseline, vLLM-served
vllm-fp8	23 GB	BF16 on disk	FP8 applied dynamically at load time, not serialized
bnb-q8	13 GB	INT8	bitsandbytes, quantized at load time
comet-mixed-precision	9.15 GB	mixed BF16/INT8/INT4	COMET-sensitivity knapsack, GPTQ-quantized
awq	8.0 GB	INT4, group size 128	AWQ, serialized checkpoint
gptq	8.0 GB	INT4, group size 128	GPTQ, serialized checkpoint
bnb-q4	7.8 GB	FP4 (NF4)	bitsandbytes, quantized at load time

Table 5: Combined trade-off summary. Quality columns are macro-averages across the three language pairs (wmt25, batch 32). Speed and size as in Tables 3–4. Best value per column in **bold** (baseline and bnb-q8 tie on average chrF).

System	Size (GB)	Sent/s	chrF (avg)	comet22-da (avg)
comet-mixed-precision	9.15	1.76	43.23	0.6469
gptq	8.0	1.44	43.03	0.6148
awq	8.0	1.23	42.77	0.6224
vllm-bf16	23.0	1.14	43.47	0.6396
vllm-fp8	23.0	1.01	43.50	0.6218
baseline	23.0	0.83	43.53	0.6638
bnb-q4	7.8	0.59	43.00	0.6285
bnb-q8	13.0	0.30	43.53	0.6413

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. [QLoRA: Efficient finetuning of quantized LLMs](#). In *Advances in Neural Information Processing Systems (NeurIPS)*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2023. [GPTQ: Accurate post-training quantization for generative pre-trained transformers](#). In *International Conference on Learning Representations (ICLR)*.

Marco Gaido, Roman Grundkiewicz, Thamme Gowda, and Matteo Negri. 2025. [Findings of the WMT 2025 shared task on model compression: Early insights on compressing LLMs for machine translation](#). In *Proceedings of the Tenth Conference on Machine Translation (WMT 2025)*, pages 484–494, Suzhou, China. Association for Computational Linguistics.

Thamme Gowda, Marco Gaido, Roman Grundkiewicz, and Matteo Negri. 2026. Findings of the WMT 2026 shared task on model compression: No free lunch at extreme compression. In *Proceedings of the Eleventh Conference on Machine Translation (WMT 2026)*, Budapest, Hungary.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with PagedAttention](#). In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. [AWQ: Activation-aware weight quantization for](#)

[LLM compression and acceleration](#). In *Proceedings of Machine Learning and Systems (MLSys)*.

Yasmin Moslem, Muhammad Hazim Al Farouq, and John D. Kelleher. 2025. [Iterative layer pruning for efficient translation inference](#). In *Proceedings of the Tenth Conference on Machine Translation (WMT 2025)*, pages 1022–1027, Suzhou, China. Association for Computational Linguistics.

David Ponce, Harritxu Gete, and Thierry Etchegoyhen. 2025. [Vicomtech@WMT 2025: Evolutionary model compression for machine translation](#). In *Proceedings of the Tenth Conference on Machine Translation (WMT 2025)*, pages 1011–1021, Suzhou, China. Association for Computational Linguistics.

Maja Popović. 2015. [chrF: character n-gram F-score for automatic MT evaluation](#). In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. [COMET: A neural framework for MT evaluation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702. Association for Computational Linguistics.

WMT26 Model Compression Shared Task Organizers. 2026. [WMT26 model compression shared task](#). Organizer repository: <https://github.com/thammegowda/wmt26-model-compression>.