

FBK’s Submission to WMT26 Model Compression Task: Improving Inference Speed for Quantized Models

Dhairya Suman^{1,2} and Luisa Bentivogli¹

¹Fondazione Bruno Kessler, ²University of Trento
{dsuman,bentivo}@fbk.eu

Abstract

This paper presents FBK’s submission to the WMT26 Shared Task on Model Compression. The task requires compressing the Gemma 3 12B model for machine translation on the Czech-German language pair under the task’s constrained track. Post-training quantization (PTQ) methods such as GPTQ have become an industry standard for model compression, but in their standard, weight-only form they reduce only the on-disk size of the model; leaving the activations are in full-precision. As part of this submission, we quantize both weights and activations to 8-bits (*W8A8*), aiming to also improve inference speed in addition to memory savings, while preserving translation quality. We submit two systems: both quantizing activations using round-to-nearest quantization, one of which additionally applies SmoothQuant’s channel-wise smoothing before quantization. Our experiments demonstrate that both the *W8A8* systems increase throughput by more than 20% while only slightly reducing in quality relative to the full precision model.

1 Introduction

The success of Large Language Models (LLMs) in their ability to handle a wide range of tasks across many languages is highly attributed to their scale (Lyu et al., 2024; Pang et al., 2025; Zhu et al., 2024). Their huge parameter counts, in the tens of billions, give these models the capacity to generalise across domains, languages and modalities with a single set of weights (Zhu et al., 2024). This scale, however, comes at a cost. Serving a model with billions of parameters in full precision (typically *FP16* or *BF16*) requires large amounts of GPU memory for weight storage, and the matrix multiplications that dominate a forward pass are correspondingly expensive to execute at full precision, both of which increase inference latency and reduce deployability. This is especially noticeable in the case of Machine Translation (MT) which

is often deployed in resource-constrained scenarios where a large LLM cannot be simply served in full precision (Frantar et al., 2023). For instance, LLMs require an average of 30 seconds for inference, whereas Encoder-Decoder models require only 0.3 seconds on average, indicating a nearly 100-fold difference (Pang et al., 2025). Hence, it becomes imperative to reduce the memory footprint and inference latency of MT using LLMs while maintaining translation quality.

Towards this aim, this paper describes FBK’s submission to the WMT26 Shared Task on Model Compression¹ (Gowda et al., 2026). The goal of the task is to compress a general-purpose LLM for MT while preserving a strong balance between compactness, speed and MT performance. The compressed models are evaluated on three dimensions: translation quality, size of the model on disk and vRAM usage and the inference speed. We participated in the constrained track, where the goal was to compress the model Gemma 3 12B. Specifically, we targeted the language pair Czech-German (*ces* $\rightarrow$ *deu*).

We adopt post-training quantization (PTQ) as our compression technique, since it targets all the evaluation axes: (1) quantizing the weights reduces the model size while preserving the performance gains brought by the scale of the model; (2) quantization can be applied to activations as well as weights, which as we discuss below is what makes inference-speed gains possible (Xiao et al., 2024); (3) most importantly, PTQ methods have shown to preserve accuracy close to the full precision model even at 8-bit and 4-bit precision level (Marchisio et al., 2024; Marie and Fujita, 2025). Another major advantage of using PTQ over compression methods like pruning (Cheng et al., 2024) and distillation (Xu et al., 2024) is that PTQ can directly oper-

¹<https://www2.statmt.org/wmt26/model-compression.html>

ate on the released checkpoint using only a small calibration set and no gradient-based training.

Within PTQ, we take GPTQ (Frantar et al., 2023) as the starting point for our submission. GPTQ is a layer-wise quantization method. It is the baseline for our submission: it reduces the checkpoint substantially at almost no cost in translation quality. However, GPTQ is standardly deployed in weight-only form (*W8A16*), in which activations remain in *BF16* and the quantized weights are dequantized before every matrix multiplication. In contrast, quantizing both activations and weights can accelerate compute-intensive operations, giving nearly double the throughput of matrix multiplications compared to their *FP16* counterparts (Xiao et al., 2024). Closing this gap is the goal of our participation, and both of our submitted systems therefore extend GPTQ with activation quantization so that both operands of each linear layer are in *INT8*, thus improving inference speed.

We discuss two systems:

1. **W8A8-GPTQ-RTN**: In addition to quantizing the weights from *BF16* to *INT8* using GPTQ, the activations are also quantized to *INT8* using a dynamic, per-token round-to-nearest scheme (RTN). This system tests the hypothesis that simply placing both operands in *INT8* is enough to dispatch the linear layers to *INT8* GEMM kernels and obtain the associated throughput gains, and that at 8 bits the resulting quantization error is small enough to leave translation quality substantially intact.
2. **W8A8-SQ-GPTQ**: This system applies SmoothQuant rescaling (Xiao et al., 2024) to the weights and activations before quantizing the weights using GPTQ and the activations using the same dynamic, per-token RTN scheme. By migrating part of the activation dynamic range into the weights, which are distributed far more uniformly and are quantized accurately even at low precision, we expect to reduce the activation quantization error and recover quality relative to *W8A8 – GPTQ – RTN*.

2 Background

Quantization maps a real-valued tensor $\mathbf{X}$ to a low-bit integer representation $\mathbf{X}^{\text{INT}}$. Uniform symmetric quantization assumes that the distribution

of values taken by the tensor $\mathbf{X}$ is approximately centered at 0, it can be expressed as:

$$X^{\text{INT8}} = \left\lfloor \frac{X}{\Delta} \right\rfloor \quad \text{and} \quad \Delta = \frac{\max(|X|)}{2^{N-1} - 1} \quad (1)$$

where N is the number of bits, in our case 8, $\lfloor \cdot \rfloor$ is the rounding function and Δ is the step size. Similarly, we can **dequantize** from *INT8* to real-valued tensor:

$$X^{\text{dequant}} = \Delta \cdot X^{\text{INT}} \quad (2)$$

Post-training quantization (PTQ) refers to converting an already-trained model to lower precision without further gradient-based training, typically using a small, unlabeled calibration set to estimate the statistics needed to choose accurate quantization parameters. When these parameters are obtained simply by applying the equations above directly with no further correction, it is called **round-to-nearest (RTN) quantization**. GPTQ (Frantar et al., 2023) and SmoothQuant (Xiao et al., 2024), described below, are also PTQ methods, but use more sophisticated calibration procedures to reduce the error introduced due to quantization. PTQ methods can be categorized along these two axes:

1. **What is being quantized?** *Weight-only quantization* (Lin et al., 2026; Dettmers et al., 2022; Frantar et al., 2023) reduces the model size by quantizing the weights from *FP16* to a lower bit-width (for example *INT8*, *INT4*). However, the weights are again dequantized back to the activation dtype (*FP16*) during matrix multiplication. *Weight and Activation quantization* quantizes both activations and weights into *INT8* to leverage the integer kernels like *INT8* GEMM present in modern GPU architectures for matmul computation. In this way, not only is the model size reduced, but also the inference speed increases. GPTQ is a weight only quantization method, whereas SmoothQuant quantizes both activations and weights.
2. **How Δ is being computed?** In case of *static quantization*, Δ is calculated offline using the activations on a calibration dataset. While, for *dynamic quantization*, we use the values of each activation tensor at inference time to calculate Δ . GPTQ is a static quantization

method. In our experiments on both the quantizing weights and activations, the activations are dynamically quantized using RTN.

We discuss the PTQ methods used in our experiments below.

2.1 GPTQ

This method aims to improve over naive RTN quantization by formulating weight quantization as *layer-wise reconstruction problem*. Concretely, it aims to find a quantized weight matrix $\hat{W}^*$ for a given weight matrix W over a calibration set of activations A collected at that layer’s input:

$$\hat{W}^* = \arg \min_{\hat{W}} \|AW - A\hat{W}\|_F^2 \quad (3)$$

Since this objective 3 is quadratic in $\hat{W}$, its curvature is fully described by the Hessian $H = 2A^T A$ which can be estimated once per layer from the calibration activations.

Unlike RTN which doesn’t take into account the activations A , GPTQ quantizes the weights within a layer sequentially, one input channel at a time. After quantizing a specific weight w_q , the resulting error $e_q = w_q - \hat{w}_q$ is compensated: the remaining not-yet-quantized weights in the same row are updated in closed form by:

$$\delta = -\frac{e_q}{[H^{-1}]_{qq}} \cdot [H^{-1}]_{q,:} \quad (4)$$

This second-order error compensation is what allows GPTQ to reach substantially lower degradation than RTN at the same bit width, particularly at low bit widths such as 4-bit. GPTQ was originally proposed, and is most commonly deployed, as a weight-only method (e.g. *W8A16* or *W4A16*): activations are left in *FP16* and only $\hat{W}$ is quantized.

2.2 SmoothQuant

Simultaneously quantizing weights and activations is an effective strategy to reduce GPU memory usage and increase the throughput of matrix multiplications. However, activations can be difficult to quantize in LLMs due to systematic outliers with large magnitudes, weights, by contrast, are distributed far more uniformly and are quantized accurately even at low precision.

SmoothQuant exploits this asymmetry by migrating part of the activation dynamic range into the weights, offline and before any quantization takes

place, so that the two tensors are comparably difficult to quantize rather than one being much harder than the other. It introduces a per-input-channel factor s_j computed once from calibration statistics:

$$s_j = \frac{\max(|X_j|)^\alpha}{\max(|W_j|)^{1-\alpha}} \quad (5)$$

where $\alpha \in [0, 1]$ controls the migration burden on the weights. This smoothing factor is then used to transform the weights and activations, while keeping the mathematical equivalence of the linear layer:

$$Y = XW \quad (6)$$

$$Y = (X \text{diag}(s)^{-1}) \cdot (\text{diag}(s)W) = \hat{X}\hat{W} \quad (7)$$

After transformation of the activations and weights, RTN is used to finally quantize the weights and activations.

3 Methodology

In this section we describe each of our submissions in detail.

3.1 W8A8-GPTQ-RTN

Weight-only GPTQ quantizes only one of the two operands of each linear layer, leaving the activations in *BF16*; the quantized weights must therefore be dequantized before every matrix multiplication, and the multiplication itself is still executed in floating point. In this system we quantize the activations, in addition to the weights: the weights are quantized with GPTQ as described above, and at inference time the activations are quantized to *INT8* with symmetric dynamic per-token RTN. Both operands of each linear layer are then in *INT8*, the dequantization step is removed, and the multiplication is dispatched to an *INT8* GEMM kernel.

3.2 W8A8-SQ-GPTQ

This configuration has the same pipeline as 3.1 with an additional initial smoothing/rescaling step for the weights borrowed from SmoothQuant (Xiao et al., 2024) to reduce quantization difficulty for the activations. We chose $\alpha = 0.8$ (the *llmcompressor* library default). After this, the weights are again quantized using GPTQ and the activations using symmetric dynamic per-token RTN quantization.

Table 1: Description of the four evaluated configurations.

Configuration	Weights	Activations	Weight quantizer	Smoothing
Base (<i>BF16</i>)	16-bit	16-bit	—	—
<i>W8A16-GPTQ</i>	8-bit, per-channel	16-bit	GPTQ	—
<i>W8A8-SQ-GPTQ</i>	8-bit, per-channel	8-bit, dynamic per-token	GPTQ	$\alpha = 0.8$
<i>W8A8-GPTQ-RTN</i>	8-bit, per-channel	8-bit, dynamic per-token	GPTQ	none

4 Experimental Settings

4.1 Task and data

We evaluate on the *ces* $\rightarrow$ *deu* language pair from the WMT25 test suite. We used the segmentation technique used in (Gaido et al., 2025) to segment the original paragraphs resulting in a test set with 2868 segments.

4.2 Model and quantization pipeline

1. **Model:** We use the Gemma 3 12B model² (Team et al., 2025) as mandated by the constrained track of the shared task³.
2. **Calibration Set:** We use a subset of 512 segments from the WMT19 (Barrault et al., 2019) test set for the language pair *ces* $\rightarrow$ *deu* as the calibration dataset. This data was used to collect the GPTQ Hessian estimates and perform the smoothing in SmoothQuant.
3. **Quantization Library:** We used the *llm-compressor*⁴ (0.6.0.1) (AI and vLLM Project, 2024) library to quantize our models.
4. **Hardware Used:** All experiments were run on a single NVIDIA A100-SXM-64GB GPU with NVIDIA driver 535.274.02 (CUDA 12.2).
5. **Inference Engine:** We use vLLM⁵ (0.8.5.post1) (Kwon et al., 2023) as our serving engine to use its state-of-the-art *W8A8* kernels on the A100 GPU for faster inference.
6. **Serving configuration:** Each source segment is wrapped in the following zero-shot instruction, with {src} and {tgt} set to *Czech* and *German*:

You are a professional {src} {tgt} translator. Your goal is to accurately convey the meaning and nuances of the original {src} text while adhering to {tgt} grammar, vocabulary, and cultural sensitivities. Produce only the {tgt} translation, without any additional explanations or commentary. Translate the following {src} text into {tgt}:{text}

The full test set of 2,868 segments is passed to `LLM.generate()` in a single call and scheduled by vLLM’s continuous batching, so no batch size is fixed explicitly. All configurations were served with `gpu_memory_utilisation=0.9` i.e 90% of the total GPU memory was used at each stage. vLLM reserves this fraction of the device at startup and allocates the remainder after weights and activations to the KV cache.

4.3 Metrics

For each system, we provide the following evaluation scores on the WMT25 test suite for the language pair *ces* $\rightarrow$ *deu*:

1. **Translation Quality:** This is reported through xCOMET-XL⁶ (Guerreiro et al., 2024) and MetricX⁷ (Juraska et al., 2024).
2. **Speed:** end-to-end throughput reported as both *sentences/sec* and *tokens/sec* generated. Throughput was measured as an end-to-end wall-clock interval `time.perf_counter()` around a single `generate()` call over all 2,869 test segments, submitted together and scheduled by vLLM’s continuous batching. Reported figures⁸ are output tokens per second (total generated tokens divided by elapsed time) and sentences per second.

²<https://huggingface.co/google/gemma-3-12b-it>

³<https://www2.statmt.org/wmt26/model-compression.html>

⁴<https://github.com/vllm-project/llm-compressor>

⁵<https://github.com/vllm-project/vllm>

⁶<https://huggingface.co/Unbabel/XCOMET-XL>

⁷<https://huggingface.co/google/metricx-24-hybrid-xl-v2p6-bfloat16>

⁸We report the mean of three independent runs per configuration; individual runs varied by less than 1.5% in throughput.

Table 2: Results across all four configurations

Configuration	Translation Quality		Speed		Model Size in GB
	xCOMET-XL	MetricX ($\downarrow$)	toks/sec	sentences/sec	
Base (<i>BF16</i>)	57.37	5.04	2015.8	68.31	24.4
W8A16-GPTQ	57.46	5.07	1779.8	60.29	13.7
W8A8-SQ-GPTQ	56.45	5.18	2486.4	84.92	13.6
W8A8-GPTQ-RTN	56.38	5.26	2621.0	88.78	13.6

3. **Size:** The on-disk checkpoint size of the compressed model in GB. Since, this work primarily targets inference efficiency, we only report the on-disk checkpoint size. Moreover, all configurations were served with `gpu_memory_utilisation=0.9`, meaning a fixed fraction of the GPU memory is allocated to all of the models regardless of their actual size.

All the model configurations are summarised in Table 1.

5 Results

Table 2 reports the results for the two W8A8 configurations. In addition to that, we also report the scores for the given systems: (1) W8A16-GPTQ - baseline GPTQ quantized model, where the weights are quantized into *INT8*, but the activations are in native *BF16*; (2) the **base** uncompressed model. We include them to establish the quality and inference speed tradeoff. We note the following observations from our results:

- **Inference Speed vs. Translation Quality.** We see that the GPTQ baseline (W8A16) preserves translation quality almost exactly with a minimal increase in the MetricX scores. However, it faces a reduction in throughput when compared to the *BF16* baseline as seen in 2. This is because in case of weight-only quantization, the *INT8* weights must be de-quantized back to *BF16* before each matrix multiplication, which is then executed in *BF16* exactly as in the baseline, which adds computational overhead. In comparison, both the W8A8 quantized models remove this step entirely: both operands (the weights and the activations) are already in *INT8* and the multiplication is dispatched to *INT8* tensor-core GEMM kernels, yielding higher throughput than that of the full-precision and W8A16 model, at a cost of roughly one xCOMET-XL point.

- **Effect of smoothing.** Between both the W8A8 systems, we see that SQ-GPTQ shows slightly better performance, however, the un-smoothed system is faster.

6 Conclusions

This work presented FBK’s submission to the WMT 2026 Model Compression shared task, which focused on both compressing Gemma 3 12B model for machine translation across Czech-German language pairs within the constrained settings of the task. We focused on quantizing both the weights and activations in the model to reduce the size of the model and improving the inference efficiency while maintaining translation performance. Our experimental results demonstrate that we can successfully increase the inference speed while maintaining performance by quantizing both the activations and weights.

Acknowledgments

This paper has received funding from the European Union’s Horizon Europe programme grant agreement No. 101213369 (project DVPS) and European Union’s Digital Europe Programme (DIGITAL) under grant agreement No. 101198470 (project LLMs4EU). We also acknowledge the CINECA award under the ISCRA initiative (project MAP-MT), for the availability of high-performance computing resources and support.

References

- Red Hat AI and vLLM Project. 2024. [LLM Compressor](#).
- Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. [Findings of the 2019 conference on machine translation \(WMT19\)](#). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared*

- Task Papers, Day 1*), pages 1–61, Florence, Italy. Association for Computational Linguistics.
- Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. [A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations](#). *Preprint*, arXiv:2308.06767.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. [Llm.int8\(\): 8-bit matrix multiplication for transformers at scale](#). *Preprint*, arXiv:2208.07339.
- Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2023. [Gptq: Accurate post-training quantization for generative pre-trained transformers](#). *Preprint*, arXiv:2210.17323.
- 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*, 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.
- Nuno M. Guerreiro, Ricardo Rei, Daan van Stigt, Luisa Coheur, Pierre Colombo, and André F. T. Martins. 2024. [xCOMET: Transparent machine translation evaluation through fine-grained error detection](#). *Transactions of the Association for Computational Linguistics*, 12:979–995.
- Juraj Juraska, Daniel Deutsch, Mara Finkelstein, and Markus Freitag. 2024. [MetricX-24: The Google submission to the WMT 2024 metrics shared task](#). In *Proceedings of the Ninth Conference on Machine Translation*, pages 492–504, Miami, Florida, USA. Association for Computational Linguistics.
- 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 ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2026. [Awq: Activation-aware weight quantization for llm compression and acceleration](#). *Preprint*, arXiv:2306.00978.
- Chenyang Lyu, Zefeng Du, Jitao Xu, Yitao Duan, Minghao Wu, Teresa Lynn, Alham Fikri Aji, Derek F. Wong, and Longyue Wang. 2024. [A paradigm shift: The future of machine translation lies with large language models](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1339–1352, Torino, Italia. ELRA and ICCL.
- Kelly Marchisio, Saurabh Dash, Hongyu Chen, Dennis Aumiller, Ahmet Üstün, Sara Hooker, and Sebastian Ruder. 2024. [How does quantization affect multilingual llms?](#) *Preprint*, arXiv:2407.03211.
- Benjamin Marie and Atsushi Fujita. 2025. [The uneven impact of post-training quantization in machine translation](#). *Preprint*, arXiv:2508.20893.
- Jianhui Pang, Fanghua Ye, Derek Fai Wong, Dian Yu, Shuming Shi, Zhaopeng Tu, and Longyue Wang. 2025. [Salute the classic: Revisiting challenges of machine translation in the age of large language models](#). *Transactions of the Association for Computational Linguistics*, 13:73–95.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. [Gemma 3 technical report](#). *Preprint*, arXiv:2503.19786.
- Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2024. [Smoothquant: Accurate and efficient post-training quantization for large language models](#). *Preprint*, arXiv:2211.10438.
- Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. [A survey on knowledge distillation of large language models](#). *ArXiv*, abs/2402.13116.
- Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. [Multilingual machine translation with large language models: Empirical results and analysis](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2765–2781, Mexico City, Mexico. Association for Computational Linguistics.