Model Compression for Machine Translation in Large Language Models

Jort Vincenti, Matteo Nulli, Antonios Tragoudaras, Zoe Tzifa-Kratira, and Angel Bulalance Gomez
University of Amsterdam

1 Introduction

Small traditional machine translation models can produce high-quality translations in a one-to-one language setting (NLLBTeam et al., 2022). However, when scaling these models to handle multiple languages, they often struggle to perform well due to the limited amount of parallel data and the complex architectures required to fully understand the task. In contrast, Large Language Models (LLMs) (Brown, 2020) can handle complex settings due to their large architectures and overcome smaller datasets by leveraging their large English training data. As a result, ALMA is a fine-tuned LLM designed for machine translation across multiple language directions from and to English and is the first LLM to become competitive with traditional machine translation models (Xu et al., 2023).

Despite performing remarkably well in translation, ALMA incurs significant computational and memory costs, making even inference prohibitively expensive to scale to consumer hardware (Zhu et al., 2024). To address these challenges, several compression techniques have been proposed for LLMs. Among many others, these include quantization (Gray and Neuhoff, 1998; Xiao et al., 2023) and pruning (LeCun et al., 1989; Han et al., 2015, 2016) methods. However, none of these techniques have been applied to ALMA and evaluated in terms of translation quality.

In this work, we explore applying compression techniques to ALMA to preserve its translation quality. We experiment with five distinct compression methods: GPTQ (Frantar et al., 2023), Q-LoRA (Dettmers et al., 2024), SmoothQuant (Xiao et al., 2023), Wanda (Sun et al., 2024), and DSnot (Zhang et al., 2024). To evaluate the effectiveness of these techniques, we assess their translation quality, memory usage, and inference time, providing a comprehensive analysis of the trade-offs involved in compressing LLMs fine-tuned for translation.

2 Background and Methodology

2.1 ALMA model

ALMA is a many-to-many multilingual translation model trained on 10 language pairs. ALMA achieves increased translation performance through a two-stage fine-tuning process (Xu et al., 2023). In the first stage, ALMA addresses the English-centric bias of LLaMA by fine-tuning the model on non-English monolingual data, enhancing its ability to handle these languages effectively. In the second stage, the model is further refined using high-quality parallel data, translating between English and the target languages in both directions. This two-step approach strengthens ALMA's overall language comprehension and translation capabilities.

2.2 Quantization methods

Quantization methods reduce the precision of a model's parameters while minimizing performance degradation (Gray and Neuhoff, 1998). Due to the extensive training time required for ALMA, this study will focus on Post-Training Quantization (Zhu et al., 2024), which allows the model to be quantized without the need for re-training. In this study, we explore three such methods:

GPTQ GPTQ (Frantar et al., 2023) quantizes each row of the weight matrix independently, aiming to minimize the quantization error. The weights are reduced to int4 through a calibration dataset and dynamically dequantized to FP16 during inference. This approach reduces memory usage by up to 4x, as the int4 weights are dequantized within a fused kernel, avoiding reliance on the GPU's global memory.

Q-LoRA quantization In (Dettmers et al., 2024), two novel quantization mechanisms are introduced. The first involves a new 4-bit NormalFloat type (NF4) that maps FP32 weights to normalized values

from -1 to 1, allowing for a more precise representation of low-precision weights. The second mechanism utilizes block-wise quantization to store quantization constants in memory, significantly reducing overall memory requirements (Dettmers et al., 2021).

SmoothQuant In SmoothQuant (Xiao et al., 2023), the authors observe that quantizing weights is much easier than quantizing activations. To address this, they propose a method that smooths the activation outliers by relocating them to the weight domain via a transformation. This simplifies the quantization process and improves model performance.

2.3 Pruning methods

The goal of pruning methods is to remove the weights that have minimal impact on the model's output (LeCun et al., 1989). Similar to section 2.2, we will focus only on methods that do not require re-training. The following two methods are presented:

Wanda (Sun et al., 2024) is a pruning method that focuses on the weights and the input activations to determine which weights to prune. By multiplying the magnitude of the weights with the norm of their input activations, Wanda effectively removes the weights with minimal impact on the model's output.

DSnot DSnot (Zhang et al., 2023), inspired by Dynamic Sparse Training (DST), iteratively adjusts sparsity by pruning and growing weights based on their contribution to the reconstruction error. This method has demonstrated improved performance scores and lower perplexity at high sparsity levels.

3 Experimental Setting

Data We evaluate the compressed versions of ALMA on its original test set (Xu et al., 2023). This dataset contains 100,000 examples of human-written parallel data across 10 translation directions[1] collected from WMT'21, WMT'22, and Flores-200 (Rei et al., 2022; Kreutzer et al., 2022; Ortiz Suarez et al., 2019).

Metrics We evaluate the translation quality of the compressed models using BLEU (Papineni et al., 2002) and COMET (Rei et al., 2020), as these were the metrics reported in the original ALMA paper.

[1] cs ↔ en, de ↔ en, is ↔ en, zh ↔ en, ru ↔ en

In addition, we assess the performance improvements by analyzing memory reduction and measuring the latency per sample to evaluate the gains achieved through compression.

Computational requirements All experiments were carried out with a NVIDIA A100-SXM4-40GB GPUs. We report the memory footprint that corresponds solely to loading the model on the GPU. An extensive technical study regarding the memory requirement for running the baseline ALMA model and the quantized versions can be found in Appendix C.

4 Results & Discussion

Table 1 presents the results for the five different compression techniques. A noticeable performance gap can be observed between the en→xx and xx→en translation directions. As noted in the ALMA paper (Xu et al., 2023), this disparity is likely due to the large amount of English data used during the training of LLAMA 2 (Touvron et al., 2023), resulting in a bias towards better performance when translating into English.

4.1 Quantization

All quantization methods retained significant levels of performance in terms of BLEU and COMET compared to the baseline, while reducing memory requirements by $\approx 70\%$. A possible reason for this behavior could be attributed to ALMA's inherently low BLEU score, which indicates that its weights are not finely tuned for best performance in each language. Instead, the weights exhibit high variability to accommodate for the multilingualism of the task. As a result, we also infer that when the model is quantized the weights generalize better rather than causing performance degradation in specific languages.

The overall inference time for the 4-bit quantization methods is notably higher than the baseline, which may initially seem counter-intuitive. However, this can be attributed to GPU nodes performing multiplications in FP16 rather than using the quantized integers, as FP16 operations are more efficient on the hardware. Although integer representations require less memory than FP16 values, this does not necessarily result in improved runtime performance.

Method (Precision/Sparsity)	Avg en -> x		Avg x -> en		Avg Time	Memory
	BLEU%	*COMET%*	*BLEU%*	*COMET%*	*(seconds)*	*(MB)*
ALMA-7B (16bit/NA)	25.96	**85.45**	**34.54**	**82.86**	3.77	12860
GPTQ (4bit/NA)	25.74	85.15	33.86	82.64	15.60	3769
Q-LoRA (NF-4bit/NA)	25.60	85.13	34.05	82.66	7.01	3740
SQ + Q-LoRA (NF-4bit/NA)	**26.11**	85.21	33.90	82.70	6.83	**3726**
Wanda (NA/40%)	23.96	82.94	31.84	81.28	**1.72**	12852
Wanda + GPTQ (4bit/40%)	24.20	83.78	32.42	81.72	15.40	3769

Table 1: Summary of calculated metrics across all analyzed model compression techniques. All results are calculated on ALMA-7B and the Method column highlights the method used along with a precision/sparsity level (NA = Not Applicable, NF4 =4-bit NormalFloat. We report BLEU (↑) and COMET (↑) scores for en ↔ x. Additionally, we cover the average time per token in seconds (↓) and the memory allocation in megabytes (MB) (↓) after each compression technique is applied.

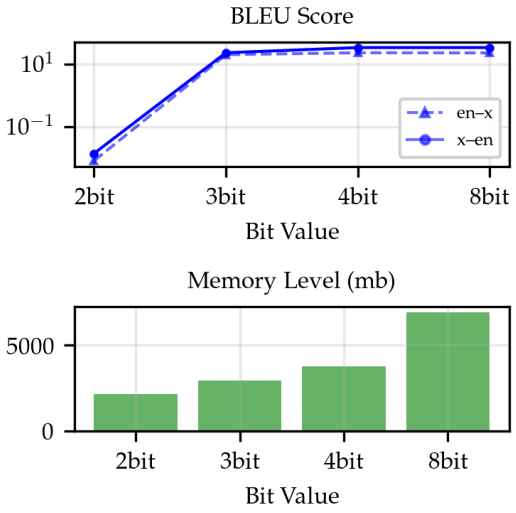

Figure 1: Top graph shows average BLEU Score for both directions of en ↔ x across different GPTQ bit-quantization levels. The bottom bar chart represents the memory allocation across different precision values. Both results are based on ALMA-7B.

4.1.1 GPTQ

A key aspect of the GPTQ approach is selecting the appropriate bit level for quantization. To determine the most suitable level, we conducted a small-scale ablation study. Figure 1 illustrates that the 4-bit quantized version displays the best performance-to-memory trade-off. The significant drop at the 2-bit level may be due to the limited number of representative values, which induces the model to not store enough relevant information to produce meaningful generation outputs. Interestingly, we observed a significant increase in inference time, which was not mentioned in the original paper. A possible explanation is that, in this study, we perform quantization and evaluation in two different steps. As a result, the fused kernel, which 'avoids reliance on the GPU's global memory' was not implemented, likely contributing to the additional inference time (Appendix B). While we use ALMA's data for calibration, we defer the analysis to different methods.

4.1.2 Q-LoRA like quantization

The authors of (Dettmers et al., 2024) highlight that the Q-LoRA quantization scheme produces floating-point quantization levels optimized for normally distributed weight data. This insight could explain the minimal impact on translation performance in our experiments. As with other quantization methods, the weights are de-quantized back to FP16. However, since the quantization factors are stored using block-wise quantization, the mapping process is more efficient, reducing overall computation time and resulting in faster inference compared to GPTQ (Dettmers et al., 2021).

4.1.3 SmoothQuant with LLM.Q-LoRA like Quantization

SmoothQuant has not yet been implemented for Llama-based models in terms of compression. However, we recognize the value of using activation smoothing as part of the quantization process. Therefore, we applied SmoothQuant (i.e., transferring activations to the weight domain) and used Q-LoRA scheme as a proxy for weight quantization. Surprisingly, this method outperformed the plain ALMA-7B, suggesting that quantization can act as a noise suppressor, enhancing the model's translation performance (Bondarenko et al., 2024).

4.2 Pruning

Regarding pruning methods, Wanda exhibited relatively low BLEU scores but to the benefit of low latency. While its memory usage was substantially

lower than the full ALMA-7B model, it did not achieve the memory footprint seen with quantization techniques. This is because Wanda selectively zeros out a portion of the weights rather than applying aggressive quantization. As a result, during inference, the number of matrix multiplications remains the same as in the pre-pruned model, but the introduced sparsity accelerates the computations.

4.2.1 Wanda + DSnoT

The findings of Zhang et al. (2023) demonstrate that DSnoT significantly enhances the perplexity scores of Wanda on reasoning tasks, particularly at very high sparsity levels. However, when evaluated using BLEU and COMET scores, DSnoT showed negligible improvements, even in cases where perplexity improved (see Appendix 5).

This discrepancy can be attributed to the fact that perplexity is highly sensitive to how well the model predicts the probability distribution over the vocabulary. DSnoT might improve the likelihood of individual word predictions without necessarily enhancing the overall quality of the translation.

As a result, DSnoT might be better suited for general LLM tasks, and for redundancy purposes, we do not include it in the main results table 1. The Wanda experiments were conducted at a sparsity level of 40%, as this was the threshold right before a significant drop in performance (as shown in 5).

4.2.2 Wanda + GPTQ

Shao et al. (2024) applied quantization and pruning to the LLaMA-7B model, using SparseGPT at 50% sparsity (Frantar and Alistarh, 2023) combined with GPTQ at 4-bit precision, observing better perplexity compared to plane quantization. This approach inspired us to apply the same approach to the ALMA model.

As shown in Table 1, this method results in a slight improvement over Wanda, with memory consumption now comparable to other quantization methods. However, the average inference time has also increased. Therefore, the choice between these methods depends on the specific hardware constraints and requirements of the translation task at hand.

4.3 Effects of Calibration

Figure 2 displays the performance metrics using different sample sets (details in Appendix 3 and 4). We observe minimal changes in performance between the different sample sizes. This can be explained by the fact that translation tasks may rely more on pattern recognition and direct word mappings, meaning that smaller calibration datasets can still capture enough of the key activation patterns needed for the model to perform well.

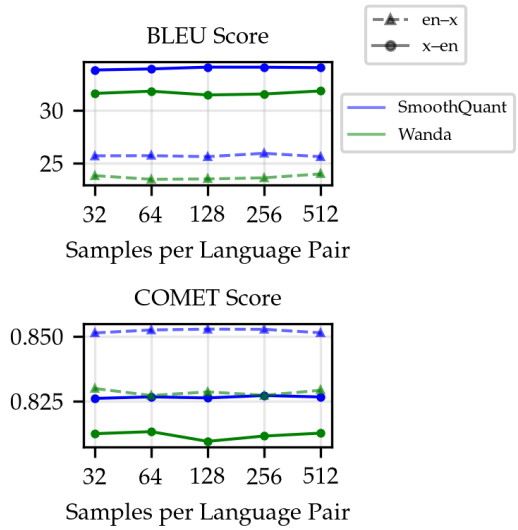

Figure 2: The top and bottom graphs display the BLEU and COMET scores, respectively, for both translation directions (en ↔ x) across varying calibration sizes and different compression methods. All results are based on the ALMA-7B model.

However, since all methods are post-training, the calibration process only needs to be performed once before inference. Therefore, using the best-performing model, in our case 512 samples per language, is acceptable despite its latency. Nevertheless, as model complexity grows, calibration can become costly in terms of memory and runtime, making techniques like Q-LoRA advantageous by eliminating the need for calibration.

5 Conclusion

In this work, we explored the application of various compression techniques to ALMA. Our study focused on five methods: GPTQ, Q-LoRA, SmoothQuant, Wanda, and DSnot. The results show that quantization techniques offer reductions in memory usage with minimal impact on translation quality. Pruning methods like Wanda provided faster inference times but at the cost of translation performance. The findings highlight the trade-offs between model efficiency and performance, demonstrating that compression methods can make large models like ALMA more accessible for deployment without sacrificing translation quality.

References

Yelysei Bondarenko, Riccardo Del Chiaro, and Markus Nagel. 2024. Low-rank quantization-aware training for llms.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Robert M. Gray and David L. Neuhoff. 1998. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1135–1143, Cambridge, MA, USA. MIT Press.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, and et al. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

NLLBTeam, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, and et al. 2022. No language left behind: Scaling human-centered machine translation.

Pedro Javier Ortiz Suarez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019*, pages 9 – 16.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Ricardo Rei, José GC De Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André FT Martins. 2022. Comet-22: Unbabel-ist 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation.

Hang Shao, Bei Liu, and Yanmin Qian. 2024. One-shot sensitivity-aware mixed sparsity pruning for large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11296–11300. IEEE.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and et al. 2023. Llama 2: Open foundation and fine-tuned chat models.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. 2023. A paradigm shift in machine translation: Boosting translation performance of large language models.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. 2023. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Sun Yunyun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. 2024. Dynamic sparse no training: Training-free fine-tuning for sparse LLMs. In

The Twelfth International Conference on Learning Representations.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models.

A ALMA data Appendix

Here we insert additional results and plots regarding ALMA (Xu et al., 2023).

	Parallel Data				Monolingual Data	
	Train	Development	Test (from English)	Test (to English)	# Words	Sampling Ratio
German (de)	14211	1002	2037	1984	73.8B	20%
Czech (cs)	12076	1002	2037	1448	9.7B	14%
Icelandic (is)	2009	-	1000	1000	0.3B	8%
Chinese (zh)	15406	1002	2037	1875	44.4B	19%
Russian (ru)	15000	1002	2037	2016	78.0B	22%
English (en)	-	-	-	-	523.9B	17%

Figure 3: Table from ALMA (Xu et al., 2023) detailing the exact data used for training/testing ALMA models.

B GPTQ Appendix

Here we insert additional details, results and plots regarding GPTQ (Frantar et al., 2023).

B.1 Implementation details

Our implementation of GPTQ quantization started from the repository GPTQ Repo. We initially tried quantizing the model through this repository code, but after unsuccessful attempts, we switched to GPTQConfig. While this library has a paramter `use_cuda_fp16` to actually employ the optimized cuda kernel for fp16 model, this did not bring any additional time efficiency gains, thus the poor time performance at inference.

C Memory Consideration - Technical Study

In Section 3, all experiments report the memory footprint of the baseline ALMA and there is a comparison to the memory saved using the compression techniques we studied. However, the memory requirements reported there only capture the number of bytes (MBytes) needed for representing each model parameters. Specifically, as our implementation is based on PyTorch framework, this is the model parameters and torch buffers.

As we wanted to compute the realistic hardware requirements needed for one to use our compression techniques, we employ nvprof. The profiler can capture precisely the memory allocated for the computation

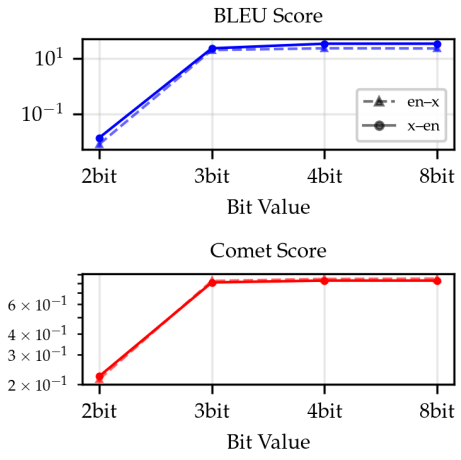

Figure 4: Complementary plots to Figure 1 showing also COMET Score as a function of bit level.

Model	*en-cs*		*cs-en*		*en-de*		*de-en*		*en-is*		*is-en*	
	BLEU	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*
ALMA-7B (32bit/NA)	30.68	88.42	43.83	84.73	30.31	83.75	30.28	82.58	25.30	85.00	36.07	83.87
Q-LoRA like Quant.	29.16	88.10	43.20	84.48	30.16	83.72	29.84	82.52	25.12	84.87	34.94	83.69
SQ + Q-LoRA like Quant.	29.78	88.10	43.48	84.47	29.65	83.52	29.92	82.46	24.84	84.80	35.52	83.70
Wanda	26.73	0.8605	40.09	0.8337	27.85	0.8180	29.06	0.8168	21.52	0.8209	33.06	0.8231

Model	*en-ru*		*ru-en*		*en-zh*		*zh-en*	
	Bleu	*Comet*	*Bleu*	*Comet*	*Bleu*	*Comet*	*Bleu*	*Comet*
ALMA-7B (32bit/NA)	27.71	85.96	38.61	84.14	15.79	84.13	23.90	78.98
Q-LoRA like Quant.	26.79	85.67	38.47	84.16	19.31	83.67	23.05	78.67
SQ + Q-LoRA like Quant.	27.09	85.49	38.37	84.16	16.63	83.72	22.98	78.53
Wanda	25.21	0.8357	37.62	0.8318	18.50	0.8120	19.38	0.7585

Table 2: Full Results for the Quantized methods on ALMA's dataset with BLEU and COMET scores

Calibration size	*en-cs*		*cs-en*		*en-de*		*de-en*		*en-is*		*is-en*	
	BLEU	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*
32	29.47	88.08	42.86	84.51	29.74	83.65	29.60	82.40	25.15	84.82	34.85	83.62
64	29.65	88.14	43.40	84.50	30.02	83.73	29.83	82.53	25.17	84.83	35.15	83.70
128	29.79	88.33	43.89	84.51	30.02	83.75	29.74	82.50	24.90	84.97	35.39	83.67
256	29.30	88.23	43.46	84.48	29.30	83.72	29.91	82.49	25.19	84.89	35.19	83.88
512	29.78	88.10	43.48	84.47	29.65	83.52	29.92	82.46	24.84	84.80	35.52	83.70

Calibration size	*en-ru*		*ru-en*		*en-zh*		*zh-en*	
	BLEU	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*
32	26.88	85.54	38.38	83.97	17.15	83.45	23.45	78.55
64	27.20	85.76	38.30	84.01	16.43	83.71	22.75	78.43
128	27.15	85.67	34.42	83.97	16.16	83.62	23.08	78.49
256	29.39	85.74	38.53	84.04	17.02	83.70	23.38	78.71
512	27.09	85.49	38.37	84.16	16.63	83.72	22.98	78.53

Table 3: Calibration of the smoothquant + Q-LoRA like Quant. models with BLEU and COMET scores

graph model[2] of PyTorch during inference. This results in accurate VRAM estimates, allowing us to determine whether the quantized models can be employed on commercial hardware, rather than being limited to powerful GPUs specifically tailored for Deep Learning applications.

Figure 5 illustrates the total memory footprint of the baseline ALMA-7B during inference runs across all ten language-pair test sets. In contrast, Figure6 shows the memory requirements for the same inference runs using the Q-LoRA-like quantized ALMA-7B model. The memory footprint of Q-LoRA-like quantized models ranges from 6 GB to 7.5 GB, whereas the baseline ALMA-7B requires up to 20 GB in the best-case scenario.

In this technical study, we compare only the Q-LoRA like quantization method to the baseline ALMA, due to the scarcity of computational resources. We anticipate that the other two compression methods, namely GPTQ and SmoothQuant combined with Q-LoRA, to exhibit a significantly smaller memory footprint than the baseline ALMA.

[2]The computation graph dynamically records the operations required for the forward pass and inference, which directly impacts memory usage.

Calibration size	*en-cs*		*cs-en*		*en-de*		*de-en*		*en-is*		*is-en*	
	BLEU	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*
32	26.85	0.8609	39.82	0.8328	29.10	0.8197	29.33	0.8164	21.19	0.8194	32.34	0.8218
64	26.72	0.858	39.51	0.8323	27.93	0.8181	29.19	0.8173	21.17	0.8153	32.99	0.8213
128	27.11	0.8593	39.47	0.8306	27.87	0.8179	29.44	0.8172	21.24	0.8204	32.82	0.8206
256	26.88	0.8601	39.70	0.8326	27.72	0.8160	29.46	0.8181	20.70	0.8156	32.62	0.8222
512	26.73	0.8605	40.09	0.8337	27.85	0.8180	29.06	0.8167	21.52	0.8209	33.06	0.8230

Calibration size	*en-ru*		*ru-en*		*en-zh*		*zh-en*	
	BLEU	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*	*BLEU*	*COMET*
32	25.57	0.8374	37.45	0.8322	17.30	0.8121	19.09	0.7596
64	25.28	0.8354	36.99	0.8307	16.17	0.810	19.40	0.7579
128	25.06	0.8359	37.36	0.8310	16.17	0.8096	19.58	0.7596
256	25.45	0.8358	37.52	0.8304	17.21	0.8087	18.45	0.7549
512	25.21	0.8357	37.62	0.8318	18.50	0.8109	19.37	0.7585

Table 4: Calibration of the Wanda model with BLEU and COMET scores

Method / Sparsity	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Wanda									
BLEU	43.1	42.86	42.4	40.17	23.1	0.584	0.006	0.003	0
COMET	0.844	0.844	0.843	0.834	0.754	0.35	0.127	0.203	0.367
Wanda + DSnoT									
BLEU	-	-	-	-	22.6	0.316	0	0.0003	0.0004
COMET	-	-	-	-	0.749	0.341	0.364	0.153	0.2

Table 5: BLEU and COMET scores across different sparsity levels for Wanda and Wanda + DSnoT.

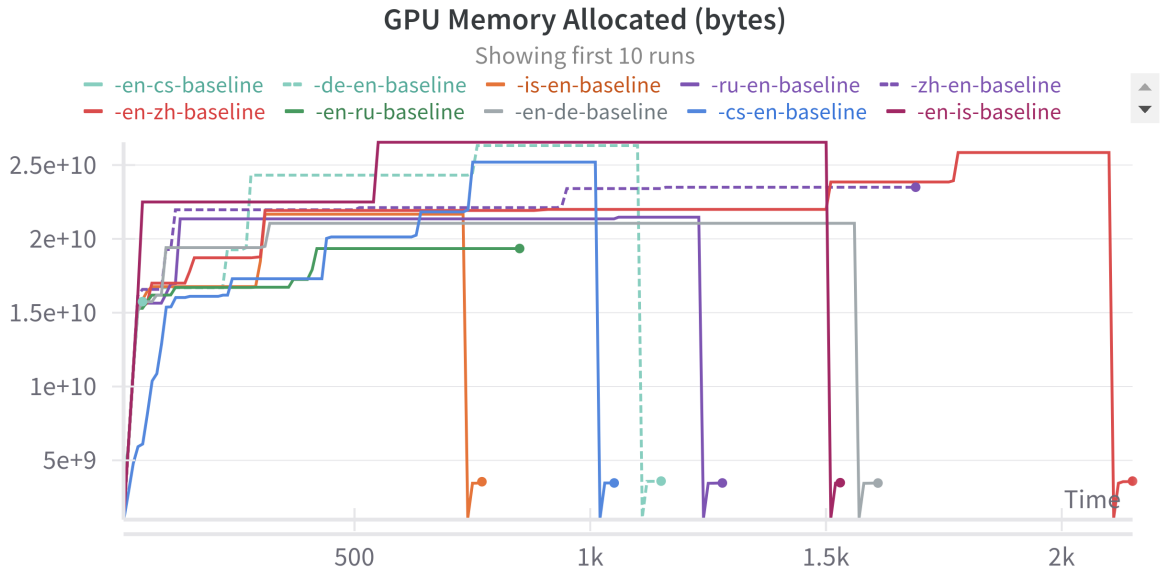

Figure 5: GPU Memory Allocation for an inference run with baseline ALMA. Hyperparameters selected: batch size of 2, max token length of 256.

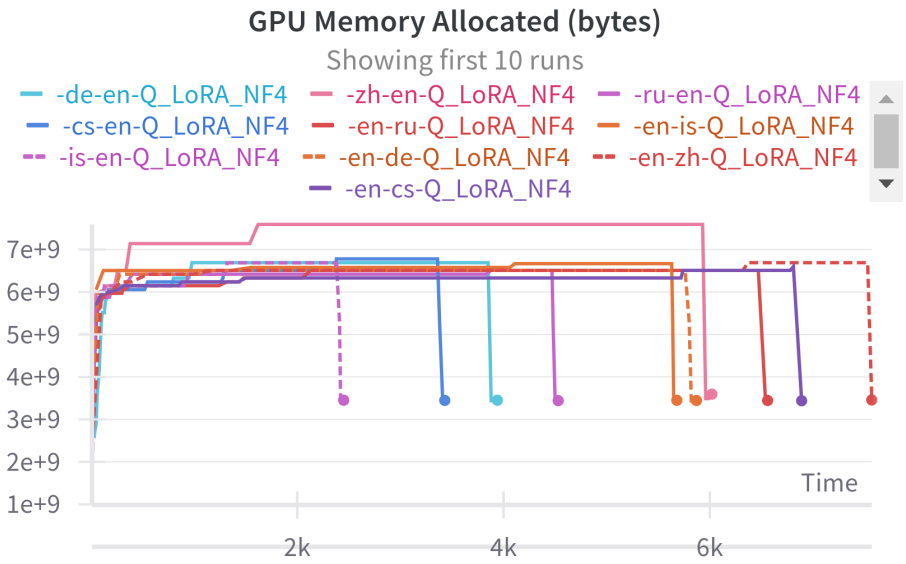

Figure 6: GPU Memory Allocation for an inference run with LoRA-like quantization. Hyperparameters: batch size of 2, max token length of 256.