

TMU-onono at WMT26: DiBA-Based Model Compression for Czech-to-German Machine Translation

Nobutaka Ono

Department of Computer Science, Tokyo Metropolitan University
6-6 Asahigaoka, Hino-shi, Tokyo 191-0065, Japan
onono@tmu.ac.jp

Abstract

We describe TMU-onono’s Czech-to-German systems submitted to the constrained track of the WMT26 Model Compression shared task. We apply DiBA (Diagonal and Binary Matrix Approximation) (Ono, 2026), which approximates a dense matrix A as $D_1 B_1 D_2 B_2 D_3$, where D_1 , D_2 , and D_3 are real diagonal matrices and B_1 and B_2 are 0/1 binary matrices. We replaced 337 Gemma 3 12B weight matrices, including the tied embedding/lm-head, and retuned only diagonal entries on translations generated by the original model. Each 2.51 GB system package excludes base-model files used during setup and provides $9.72\times$ compression relative to the original 24.37 GB BF16 weights. The primary `diba-triton_direct` directly applies bitpacked factors for low-memory inference; the contrastive `diba-cached_unpacked` uses unpacked caches for higher throughput. Both were slower than the original. To avoid out-of-memory errors, we used relatively short sequences for retuning. The submitted systems also limited inputs to 384 tokens including the prompt and outputs to 128 new tokens. Primary achieved 17.73 BLEU and 48.44 chrF on 128 held-out short segments, versus 8.25 and 32.16 on 256 WMT25 development paragraphs. Relaxing inference limits alone did not consistently improve quality, suggesting that short-sequence retuning may have limited performance on longer translation units.

1 Introduction

When large language models (LLMs) are used for machine translation, practical cost depends not only on translation quality but also on distribution size, GPU memory, and inference speed. The WMT Model Compression shared task provides a common setting for adapting general-purpose LLMs to a translation direction while jointly assessing compactness and translation performance. WMT25 introduced the first shared task combining model size, translation quality, and inference speed (Gaido

et al., 2025). WMT26 requires executable systems and evaluates Gemma 3 12B (Gemma Team et al., 2025) in the constrained track; the official shared-task results and cross-team comparison are reported in the Findings paper (Gowda et al., 2026).

Neural-network compression includes pruning (Han et al., 2016; Frantar and Alistarh, 2023), quantization (Frantar et al., 2023; Lin et al., 2024), low-rank factorization (Jaderberg et al., 2014), and knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016). In previous work, we developed DiBA (Diagonal and Binary Matrix Approximation), a matrix-factorization approach to neural-network weight compression (Ono, 2026). DiBA approximates a dense weight matrix A as $D_1 B_1 D_2 B_2 D_3$, where D_1 , D_2 , and D_3 are real diagonal matrices and B_1 and B_2 are 0/1 binary matrices. The binary factors define shared selection-and-summation patterns, while the diagonal factors provide real-valued scaling.

Closely related prior methods include Double Binary Factorization (DBF) (Boža and Macko, 2025), LittleBit (Lee et al., 2025), and Binary Quadratic Quantization (BQQ) (Kuroki et al., 2025). DBF uses two $\{-1, +1\}$ sign factors with multidirectional scales and is algebraically close to DiBA. LittleBit binarizes latent low-rank factors and incorporates row-, column-, and latent-direction scaling. It uses quantization-aware training (QAT), with Dual Sign-Value-Independent Decomposition (Dual-SVID) for initialization and residual compensation to reduce approximation error. BQQ uses linear combinations of multiple binary matrix products and is a more general binary-quadratic representation.

Although these related works appeared earlier, DiBA was conceived and developed independently rather than derived from them. A systematic comparison is left to future work. This paper instead focuses on compressing nearly all dense weights of Gemma 3 12B with DiBA factors and on con-

structing two executable systems for our WMT26 submission.

Our work has three main elements:

1. We replace the 336 linear-layer weight matrices and the tied embedding/lm-head weight of Gemma 3 12B with 337 DiBA representations.
2. Our submission includes two systems with different inference backends: the memory-oriented primary system `diba-triton_direct` and the throughput-oriented contrastive system `diba-cached_unpacked`.
3. We retune only the diagonal factors using teacher-generated translations, document checkpoint selection, and evaluate the submitted systems in terms of translation quality, storage, and runtime.

2 DiBA: Our Approach

2.1 Diagonal and binary matrix approximation

For a dense matrix $A \in \mathbb{R}^{m \times n}$, DiBA uses

$$\hat{A} = D_1 B_1 D_2 B_2 D_3, \quad (1)$$

where $D_1 \in \mathbb{R}^{m \times m}$, $D_2 \in \mathbb{R}^{k \times k}$, and $D_3 \in \mathbb{R}^{n \times n}$ are real diagonal matrices, and $B_1 \in \{0, 1\}^{m \times k}$ and $B_2 \in \{0, 1\}^{k \times n}$ are binary matrices (Ono, 2026). Letting d_1 , d_2 , and d_3 denote their diagonal entries, we have

$$\hat{A}_{ij} = d_{1,i} d_{3,j} \sum_{\ell=1}^k d_{2,\ell} B_{1,i\ell} B_{2,\ell j}. \quad (2)$$

This expression shows that DiBA represents a matrix as a weighted superposition of k rank-one binary matrices with additional row- and column-wise scaling. In our previous work, we showed that DiBA can represent any real matrix exactly for a sufficiently large intermediate dimension k (Ono, 2026).

If each real scalar is stored with Q bits and each binary entry with one bit, the dense representation requires $S_{\text{dense}}(Q) = Qmn$ bits, whereas the theoretical DiBA storage is $S_{\text{DiBA}}(k; Q) = k(m + n) + Q(m + k + n)$ bits. The theoretical storage ratio is therefore

$$\begin{aligned} \rho_{\text{DiBA}}(k; Q) &= \frac{S_{\text{DiBA}}(k; Q)}{S_{\text{dense}}(Q)} \\ &= \frac{k(m + n) + Q(m + k + n)}{Qmn}. \end{aligned} \quad (3)$$

Thus, the intermediate dimension k controls the storage–accuracy trade-off.

2.2 Optimization algorithm

The previously reported DiBA-Greedy algorithm (Ono, 2026) alternates between least-squares refits of the diagonal factors with B_1, B_2 fixed and exact one-bit improvement tests with the diagonal factors and the other binary factor fixed. Every accepted bit flip strictly decreases the squared Frobenius error of the corresponding local binary subproblem.

For the large WMT26 factorizations, we used **DiBA-ILS**, an iterated local search (ILS) extension of DiBA-Greedy. Starting from a DiBA-Greedy solution, DiBA-ILS combines simulated-annealing acceptance of selected increases in squared Frobenius error with repeated diagonal refits and optional tabu memory to discourage revisiting recent states. We describe here only its role in constructing the submitted factors; a complete formulation, analysis of convergence behavior, and systematic comparison with DiBA-Greedy and other binary factorizations are deferred to separate work.

2.3 Retuning only diagonal factors (DiBARD)

Matrix factorization minimizes the Frobenius reconstruction error with respect to the original dense weight, which is not the downstream task loss of the replaced network. DiBARD (DiBA with Retuning only Diagonal factors) therefore freezes B_1, B_2 after replacement and updates only d_1, d_2, d_3 on downstream data (Ono, 2026).

Replacing an $m \times n$ matrix with intermediate dimension k introduces only $m + k + n$ trainable continuous scalars. Because the binary structure is not searched during adaptation, standard gradient optimization can be used. Retuning only the diagonal entries also leaves the DiBA storage size unchanged. Under the strict diagonal-only protocol used here, every model parameter other than d_1, d_2, d_3 of the replaced factors is frozen.

3 Submitted Systems

3.1 Overview

We used the instruction-tuned `google/gemma-3-12b-it` checkpoint as the base model. We constructed the submitted systems in four stages: (i) DiBA factorization of the original dense weights, (ii) generation of German teacher sequences by the original Gemma 3 12B, (iii) retuning of the diagonal factors, and (iv) export of the retuned diagonal

factors together with bitpacked binary factors.

3.2 Compressed components and factorization

The language-model component of Gemma 3 12B consists of 48 Transformer layers (Gemma Team et al., 2025). In every layer, we replaced all four attention projections—query, key, value, and attention output—and all three multilayer perceptron (MLP) projections—gate, up, and down. We also compressed the weight shared by the input embedding and lm-head using one shared DiBA representation, for a total of $48 \times 7 + 1 = 337$ representations. We chose k so that the theoretical compression ratios were broadly comparable across matrix types and constrained it to multiples of eight for byte-aligned bitpacking. Table 1 summarizes the final factors.

For each dense target weight, we first obtained a DiBA-Greedy solution (Ono, 2026). We then applied DiBA-ILS to representative matrices of each type, inspected the signal-to-noise ratio (SNR) trajectory and computation time, and empirically selected a type-specific iteration budget. Finally, we applied DiBA-ILS to all target weights with those settings.

Gemma 3 ties the raw input-embedding and lm-head weights, while applying a scale of $\sqrt{3840} \simeq 61.97$ only to the input lookup. The runtime therefore shares one DiBA representation but applies this scale only on the input path, not on the lm-head projection.

3.3 Training data

We constructed a 32,768-pair training set from 47 corpora in the WMT26 Czech–German parallel train-parts. Let $C = 47$ be the number of corpora and N_c the number of sentence pairs in corpus c . We used the sampling weight

$$w_c = \frac{1}{2} \cdot \frac{1}{C} + \frac{1}{2} \cdot \frac{\sqrt{N_c}}{\sum_{j=1}^C \sqrt{N_j}}, \quad (4)$$

which preserves small corpora more strongly than sampling in direct proportion to corpus size while still reflecting corpus scale. Within each corpus, row indices for candidate sentence pairs were sampled without replacement using a fixed seed. Source and target strings were normalized with Unicode NFKC (compatibility decomposition followed by canonical composition); NUL characters were replaced by spaces, and consecutive whitespace was collapsed. We retained candidates for

which both the source and target had 2–2000 characters and the longer-to-shorter character-length ratio was at most four, and then shuffled the retained candidates with another fixed seed.

To interleave corpora, we used a deterministic largest-deficit scheduler that keeps every prefix of the generated sequence close to the mixture in Eq. (4). Let t denote the next accepted output position, $\mathcal{A}_t$ the set of corpora with remaining candidates, and $u_c(t-1)$ the number accepted from corpus c up to position $t-1$. The next candidate is drawn from the corpus maximizing

$$\Delta_c(t) = t \cdot \frac{w_c}{\sum_{j \in \mathcal{A}_t} w_j} - u_c(t-1). \quad (5)$$

If a candidate duplicates a normalized source or source–target pair that has already been accepted, it is skipped without advancing t , and the deficit is recomputed for the next candidate. Hence, not only the final counts but also arbitrary prefixes remain close to the target mixture.

Using the same scheduler and one shared global deduplication state continuously, we constructed, in order, a 32,768-pair training set and a 512-sentence checkpoint-selection set. The two sets have no overlap in normalized sources or normalized source–target pairs. The checkpoint-selection set spanned 42 of the 47 retained corpora, and no single corpus contributed more than 47 of its 512 examples.¹ Because it was drawn from the same WMT26 training-pool corpora as the retuning set, we regard it as an in-domain development set rather than evidence of out-of-domain generalization. The WMT test data provided for the shared task were used for neither training nor checkpoint selection.

In a preliminary experiment, we retuned the same initial checkpoint for one 16,384-example pass using either the reference translations supplied with the parallel data or translations generated by the original Gemma 3 12B model. On a separate 256-example development set, distinct from the WMT25 paragraphs evaluated below, training on model-generated translations improved chrF by 4.41 points, BLEU by 0.77 points, and COMET by 0.0649 relative to training on the supplied references. We therefore adopted model-generated translations as the targets for diagonal retuning.

For each Czech source x in the training set, the original uncompressed Gemma 3 12B generated

¹The five largest components were OPUS OpenSubtitles (47 examples), CCMatrix (45), NLLB (45), MultiParaCrawl (32), and MultiCCAligned (20).

Target	Shape ($m \times n$)	Count	k	Mean SNR [dB]
Attention query	4096×3840	48	3840	8.64
Attention key	2048×3840	48	2048	7.04
Attention value	2048×3840	48	2048	6.70
Attention output	3840×4096	48	3840	8.36
MLP gate	15360×3840	48	5120	6.59
MLP up	15360×3840	48	5120	6.54
MLP down	3840×15360	48	5120	6.37
Tied embedding/lm-head	262208×3840	1	6144	6.55
Total	–	337	–	–

Table 1: Final DiBA representations for the 337 compressed weight matrices. SNR is the mean Frobenius reconstruction SNR within each target type after DiBA-ILS factorization and before downstream diagonal retuning; the tied embedding/lm-head is counted as one shared weight matrix.

one German teacher sequence $\tilde{y}$ by deterministic greedy decoding. The prompt began with “Translate the following Czech text into German. Output only the translation, with no explanation or alternatives.” The Czech source was then placed between the labels Czech: and German:, and the resulting user message was wrapped in Gemma’s chat template. To reduce memory use and avoid out-of-memory errors during teacher generation, we truncated prompt-inclusive inputs to at most 384 tokens and limited generation to 128 new tokens, stopping earlier if an end-of-sequence (EOS) token was produced. The resulting teacher targets averaged 14.4 whitespace-delimited words.

3.4 Diagonal retuning

After DiBA replacement, we froze the binary matrices in all 337 DiBA representations and trained only their d_1, d_2, d_3 . The trainable set comprised the diagonal entries of $337 \times 3 = 1,011$ diagonal matrices, totaling 5,666,624 scalar parameters. Biases, RMSNorm parameters, and all other model parameters remained frozen. The three diagonal matrices of the shared embedding/lm-head representation were included; gradients from input lookup and output projection therefore accumulated into the same diagonal entries.

Two representative structural-compression systems in the WMT25 system papers centered on decoder-layer pruning (Moslem et al., 2025) or layer collapse (Ponce et al., 2025). Neither reported the same setting of structurally approximating the tied embedding/lm-head weight and retuning only its compressed factors, nor an ablation of that choice. Relative to these two systems, compressing and retuning the shared weight is a distinguishing component of our system, although we did not compare against freezing the tied factor’s

diagonals.

Using one sequence generated by the original model as a hard target corresponds to a hard-target form of sequence-level knowledge distillation (Kim and Rush, 2016). Let θ_D be all DiBA diagonal parameters and T the target-token positions of the teacher sequence. We minimize the target-token cross-entropy (CE) loss:

$$\mathcal{L}_{\text{TCE}}(\theta_D) = -\frac{1}{|T|} \sum_{t \in T} \log p_{\theta_D}(\tilde{y}_t \mid x, \tilde{y}_{<t}). \quad (6)$$

Source-prompt and padding positions are excluded, and the loss is averaged only over target tokens. We did not combine reference-translation CE or Kullback–Leibler (KL) divergence between teacher and student output distributions with this objective.

For the first pass, we initialized the diagonal parameters with their factorization values and used a fresh AdamW optimizer. We used a learning rate of 3×10^{-7} , zero weight decay, batch size 1, BF16 computation, gradient checkpointing, and gradient-norm clipping at 0.3. To control training memory, the concatenated prompt–target sequence in this first pass was limited to 384 tokens; the encoder retained the rightmost 384 tokens when this limit was exceeded. This training-sequence limit is distinct from the separate input and generation limits used for teacher generation and inference. Training used fixed order without shuffling. We saved a checkpoint every 8,192 examples through the first 32,768-example epoch, then continued the second epoch in the same order while preserving the AdamW trajectory.

3.5 Submitted systems

The two systems included in our submission replace the same 336 Linear weights and the tied embedding/lm-head weight, and use the same 337

DiBA representations and retuned diagonal values. The B_1, B_2 factors are stored with little-endian bitpacking; only the inference execution method differs.

The primary `diba-triton_direct` is memory-oriented. During model loading, it drops the vision tower and multimodal projector that are unnecessary for text-only translation and stream-loads only the small set of non-DiBA text-side tensors from the base checkpoint. Multiplication by the bitpacked B_1, B_2 factors is executed directly by a Triton selected-sum kernel.

The contrastive `diba-cached_unpacked` is throughput-oriented. It evaluates the DiBA representation using standard dense matrix multiplications interleaved with element-wise diagonal scaling. The binary factors are unpacked into floating-point 0/1 matrices and cached in GPU memory. Different floating-point evaluation paths can yield different decoded outputs even with identical factors.

Both wrappers accept batch sizes 1, 4, 8, 16, and 32, with 32 recommended in both runtime manifests. Each source line produces one German output line. The submitted package contains two system directories, each with the factors, a minimal runtime, and setup/inference scripts; it does not bundle dense Gemma weights, training checkpoints, optimizer states, or WMT data. The dense base checkpoint is prepared in the execution environment during setup.

Both backends passed the organizer’s sanity checks at batch sizes 1 and 32.

4 Experiments and Results

4.1 Local checkpoint selection

We measured translation quality with chrF (Popović, 2015), sacreBLEU (Post, 2018), and COMET (Rei et al., 2020). The local evaluation used sacreBLEU 2.6.0² and unbabel-comet 2.2.7 with Unbabel/wmt22-comet-da. Table 2 reports the complete development-time checkpoint history on the aforementioned 512-sentence checkpoint-selection set. It is not an independent final evaluation. Each checkpoint is identified by its epoch and the number of examples processed within that 32,768-example epoch.

Immediately after DiBA replacement, the model was nearly unable to translate, but diagonal-only

State	CE	chrF	sBLEU	COMET	Ovl.
Original	3.8778	53.89	25.93	0.8181	2/512
Initial DiBA	15.3398	11.76	0.01	0.1952	490/512
E1 8,192	2.8965	42.28	14.27	0.6880	11/512
E1 16,384	2.8463	43.33	16.08	0.7102	5/512
E1 24,576	2.7926	45.98	18.65	0.7287	3/512
E1 32,768	2.5920	46.74	20.42	0.7332	3/512
E2 8,192	2.6926	46.83	19.31	0.7464	3/512
E2 16,384	2.6959	47.15	19.86	0.7344	2/512
E2 24,576*	2.7938	47.68	20.33	0.7465	1/512
E2 32,768	2.6411	48.19	20.90	0.7450	4/512

Table 2: Diagonal-retuning trajectory on the 512-sentence checkpoint-selection set. E1/E2 denote epochs and the following number is the processed-example count out of 32,768. CE denotes target-token cross-entropy under teacher forcing; sBLEU denotes sacreBLEU. Ovl. is the character-length heuristic $\text{len}(\text{pred}) > 2 \text{len}(\text{src})$, not a semantic hallucination metric. *Submitted. Metric values are rounded for presentation; checkpoint selection used the unrounded scores.

training recovered much of the performance. Compared with the original model, the submitted state was lower by 6.21 chrF points, 5.60 sacreBLEU points, and 0.0716 in COMET. The single overlong flag for the submitted checkpoint was a false positive: the source had six characters, and a finite, natural-length German output simply exceeded the ratio threshold. Every Epoch 2 checkpoint produced zero empty outputs. Among the retuned checkpoints, the final Epoch 2 checkpoint maximized chrF (48.19) and sacreBLEU (20.90), but COMET decreased slightly to 0.7450 and the overlong count rose to 4/512. Manual inspection of the final Epoch 2 checkpoint’s outputs also found two substantive cases of repetition or redundancy. We therefore submitted the 24,576-example checkpoint, which had the highest observed COMET among the retuned checkpoints (0.7465) and no visually identified runaway generation on this selection set under the development decoding configuration. The nonmonotonic trends show that teacher-forced CE alone is insufficient for free-generation checkpoint selection.

4.2 Storage reduction

Table 3 reports physical file sizes. The original BF16 dense weight files occupy 24.37 GB, whereas a one-system compressed-model package excluding the dense base checkpoint occupies 2.51 GB. Almost all of the package size is accounted for by the 337 bitpacked factor files. Based on unrounded

²Reconstructed BLEU signature: BLEU|nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.6.0.

Representation	Size [GB]
Original BF16 dense weight files	24.37
Compressed-model package	2.51

Table 3: Physical size comparison for one submitted system. GB denotes decimal 10^9 bytes.

file sizes, this corresponds to a $9.72 \times$ physical compression ratio; the factor payload is equivalent to approximately 1.65 bits per original BF16 scalar. The 2.51 GB value is the submitted package size for one system, not the complete installed disk footprint after setup; an installation may additionally retain its Python environment and the separately obtained Gemma checkpoint.

4.3 Evaluation setup

Compared methods. We denote the uncompressed Gemma 3 12B model by Original, diba-triton_direct by Primary, and diba-cached_unpacked by Contrastive. Setting S retains the submitted limits of 384 input tokens, including the translation prompt, and 128 new tokens, adopted to avoid out-of-memory errors. Setting R removes input truncation and permits up to 2,048 new tokens; it changes inference only, without repeating teacher generation or retuning. We compare Original S, Original R, Primary S, and Primary R on both evaluation sets below, with S as the main evaluation and R as a supplementary diagnostic. For runtime, we compare Original S, Primary S, and Contrastive S on the same 128 short segments.

Evaluation data. The *held-out parallel-text set* contains 128 Czech–German segments sampled from the same 47-corpus pool as the retuning data. We excluded normalized-source matches with the retuning set and development exclusion lists, using the NFKC/whitespace normalization, character-length filters, and corpus weights in Section 3.3. With seed 20260908, candidates were hash-ordered within corpora and sampled without normalized-source duplication, yielding segments from 37 corpora. The supplied German translations serve as references. This is an in-pool held-out evaluation, not an out-of-domain test. No token-length filter was applied; all inputs fit within 384 tokens, while two references exceed 128 tokens.

The *WMT25 development set* contains 256 paragraphs selected from the organizer-distributed 457-paragraph Czech–German file (Gaido et al., 2025).³

³data/wmt25/wmt25.cs-de_DE.paragraphs.jsonl in

System	Short segments $N = 128$		WMT25 paragraphs $N = 256$	
	BLEU	chrF	BLEU	chrF
Original S	25.73	55.32	20.39	44.60
Original R	25.67	55.55	38.07	63.94
Primary S	17.73	48.44	8.25	32.16
Primary R	8.62	44.14	6.46	36.31

Table 4: Translation quality on the two evaluation sets. S uses the submitted length limits; R relaxes inference limits only, as defined in Section 4.3.

Before generation, we fixed a seed-20260908 ordering stratified by input/reference lengths, approximately preserving the file’s length-group proportions, and used its first 256 records. Paragraphs were neither split nor concatenated; only new-lines and surrounding whitespace were normalized. Median prompt-inclusive input/reference lengths are 58/17.5 tokens for the short segments and 224/166.5 for the paragraphs. WMT25 provides a longer-unit evaluation but is not length-matched to the WMT26 test inputs.

Common implementation and metrics. All evaluations in Sections 4.3–4.5 used one NVIDIA RTX 6000 Ada Generation GPU (48 GB class), BF16 computation, scaled dot-product attention (SDPA), and batch size four.⁴ The runtime comparison used this common batch size because Contrastive exhausted GPU memory during the batch-32 warm-up. We used the same tokenizer, chat prompt, left padding, greedy decoding, and post-processing; Primary retained row block size 1024. Quality is measured by corpus BLEU and chrF with sacreBLEU 2.5.1 (Post, 2018), using full references and all outputs, including capped or repetitive outputs.⁵ No output was empty.

4.4 Translation quality

Table 4 summarizes the four quality conditions. Primary S achieved 17.73 BLEU and 48.44 chrF on the short segments, but 8.25 BLEU and 32.16 chrF on the WMT25 paragraphs. The within-WMT25 breakdown in Table 5 shows a similar pattern: Primary S achieved 17.51 BLEU and 47.48 chrF on

<https://github.com/thammegowda/wmt26-model-compression>, revision 652d2bf7da880bccb30a7753b7649cdcc72ddf17.

⁴AMD EPYC 9374F host; Linux; NVIDIA driver 580.173.02; Python 3.12.3; PyTorch 2.8.0+cu128 (CUDA build 12.8); Transformers 4.56.2; Triton 3.4.0.

⁵BLEU: nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.5.1. chrF ($\beta = 2$): nrefs:1|case:mixed|eff:yes|nc:6|nw:0|space:no|version:2.5.1.

WMT25 length group	N	Original S	Original R	Primary S	Primary R	Primary R cap
Short	69	30.61/59.99	30.48/60.23	17.51/47.48	17.26/47.39	0/69
Intermediate	167	21.02/45.92	34.60/62.31	8.99/34.76	4.92/35.26	11/167
Long input	20	4.25/25.58	55.39/73.58	0.25/6.92	15.90/33.49	2/20

Table 5: Within-WMT25 breakdown, with BLEU/chrF in each score cell. The three groups partition the same 256 paragraphs: short has input ≤ 384 and reference ≤ 128 tokens; intermediate has input ≤ 384 and reference > 128 ; long input has input > 384 . Input lengths include the prompt. The final column counts Primary R outputs reaching 2,048 tokens.

System	Time [s]	Ex./s	Gen. peak [GiB]
Original S	79.4	1.61	23.23
Primary S	306.3	0.42	5.30
Contrastive S	124.5	1.03	40.74

Table 6: Matched runtime on the 128 short segments under the common settings in Section 4.3. Time is the median of three passes after warm-up; Ex./s is 128/Time; Gen. peak is peak CUDA-allocated memory (1 GiB = 2^{30} bytes).

the 69 short paragraphs. Cross-dataset differences also involve corpus and content, so they cannot be attributed solely to length.

Relaxing inference limits substantially improved Original on the paragraphs but did not consistently recover Primary’s quality: chrF increased while BLEU decreased. For the short segments, 124 of Primary’s 128 outputs were identical across S/R, while two outputs already repetitive under S continued substantially longer under R. Primary R reached the 2,048-token cap on one short segment and 13 paragraphs. Thus, inference-only relaxation is insufficient to address the submitted system’s difficulties on longer outputs.

4.5 Runtime comparison

Each system ran in a fresh process, with one full-dataset warm-up followed by three measured passes over the fixed-order 128 short segments. CUDA synchronization bracketed each pass. Resident caches were retained, but key–value (KV) caches were not reused across requests. Timing includes tokenization, transfers, generation, decoding, and post-processing, but excludes model loading, warm-up, prompt construction, scoring, and file I/O. Table 6 reports median time, throughput, and the maximum CUDA-allocated memory over the measured passes, resetting peak counters before each pass. This memory measure excludes loading and warm-up and is not total process GPU memory.

Primary S reduced allocated peak memory by 77.2% relative to Original S. Contrastive S was

2.46 times faster than Primary S but used more memory than Original S; both compressed systems were slower than Original S. No competing compute process was observed. These are translation throughputs, including differences in generated length, rather than isolated kernel speeds.

5 Conclusions

We described TMU-onono’s WMT26 systems, combining DiBA compression of Gemma 3 12B with diagonal-only retuning. The submitted `diba-triton_direct` and `diba-cached_unpacked` provide memory-oriented and throughput-oriented execution of the same factors. The resulting systems substantially reduce package size while offering a memory–throughput trade-off.

The evaluation highlights the importance of retuning data for recovering translation quality after compression. Our use of relatively short teacher translations and length-constrained retuning may have substantially limited performance on longer translation units. The inference-only comparison does not isolate the contribution of retuning data, and these results do not establish an inherent limitation of the DiBA approach.

Future work includes a complete formulation and systematic evaluation of DiBA-ILS, comparisons with related matrix-factorization and model-compression methods, and evaluation on additional translation directions and LLMs. Retuning with data covering a broader range of sequence lengths is an important next step.

Acknowledgments

We thank the WMT26 Model Compression shared-task organizers for providing the submission execution environment and evaluation framework. This work was supported by JST SICORP Grant Number JPMJSC2306, Japan.

References

- Vladimír Boža and Vladimír Macko. 2025. [Addition is almost all you need: Compressing neural networks with double binary factorization](#). *arXiv preprint arXiv:2505.11076*.
- Elias Frantar and Dan Alistarh. 2023. [SparseGPT: Massive language models can be accurately pruned in one-shot](#). In *Proceedings of the 40th International Conference on Machine Learning*.
- 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*.
- 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.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, et al. 2025. [Gemma 3 technical report](#). *arXiv preprint arXiv:2503.19786*.
- 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.
- Song Han, Huizi Mao, and William J. Dally. 2016. [Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding](#). In *International Conference on Learning Representations*.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. [Distilling the knowledge in a neural network](#). *arXiv preprint arXiv:1503.02531*.
- Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. [Speeding up convolutional neural networks with low rank expansions](#). In *British Machine Vision Conference*.
- Yoon Kim and Alexander M. Rush. 2016. [Sequence-level knowledge distillation](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.
- Kyo Kuroki, Yasuyuki Okoshi, Thiem Van Chu, Kazushi Kawamura, and Masato Motomura. 2025. [Binary quadratic quantization: Beyond first-order quantization for real-valued matrix compression](#). In *Advances in Neural Information Processing Systems* 38.
- Banseok Lee, Dongkyu Kim, Youngcheon You, and Youngmin Kim. 2025. [LittleBit: Ultra low-bit quantization via latent factorization](#). In *Advances in Neural Information Processing Systems* 38.
- 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*.
- Yasmin Moslem, Muhammad Hazim Al Farouq, and John Kelleher. 2025. [Iterative layer pruning for efficient translation inference](#). In *Proceedings of the Tenth Conference on Machine Translation*, pages 1022–1027, Suzhou, China. Association for Computational Linguistics.
- Nobutaka Ono. 2026. [DiBA: Diagonal and binary matrix approximation for neural network weight compression](#). *arXiv preprint arXiv:2605.05994*.
- 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*, 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.
- Matt Post. 2018. [A call for clarity in reporting BLEU scores](#). In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. 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*, pages 2685–2702, Online. Association for Computational Linguistics.