SAME RECIPE, DIFFERENT WEIGHTS: QUANTIZATION AND DECISION STABILITY ACROSS BACKENDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Can one weight-only recipe be reapplied across tensor backends without changing its checkpoint, decisions, or cost? We evaluate symmetric INT8 and grouped INT4 on ten pretrained checkpoints spanning text, vision, audio, dual-encoder, and generative vision–language architectures in one multi-backend implementation on an A100. All 30 cross-backend FP32 state pairs match exactly; none of the 60 independently quantized pairs does. Exact PyTorch packed-state transplantation succeeds for eight ViT/CLIP target interventions and preserves all 800 held-out image decisions per intervention, yet a prespecified 99% calibration-agreement selector misses its holdout target on ViT and CLIP. Across three language-model families, all FP32 and INT8 16-token continuations agree across backends on eight fixed prompts, whereas minimum pairwise INT4 agreement is 5/8 on Qwen3 and Gemma3. Image-conditioned Qwen3-VL continuations agree across all backends for FP32 and INT8, but the worst INT4 pair agrees on only 15/20 images. Median stored-variable bytes fall to 0.254 and 0.184 of FP32 under INT8 and INT4, but portable direct-eager INT4 execution is slower across all three backends. These findings separate recipe reproducibility, exact state transfer, observed decision stability, and deployment cost without treating a changed tensor hash as a changed prediction.

1 INTRODUCTION

Shared multi-backend APIs such as Keras 3 make portability possible (Chollet et al., 2024), but floating-point evaluation and quantization still depend on numerical implementation details (Goldberg, 1991; Higham, 2002). Reproducible quantization therefore needs more than a matching high-level recipe (Li et al., 2021).

A model implementation that runs on several tensor backends is not necessarily a single numerical program. Even when pretrained FP32 weights match bit-for-bit, weight quantization executes reductions, divisions, rounding, casting, and packing in backend-specific kernels. A recipe such as "per-channel INT8" or "grouped INT4" specifies an algorithm but not necessarily an identical integer checkpoint. This distinction matters when an engineer quantizes a model on one stack and serves it on another: a successful model load proves neither the identity of the packed state nor the stability of task decisions.

There is a second, orthogonal promise to test. Quantization reduces stored weight bytes, but a portable implementation may unpack and dequantize on every forward pass instead of using native low-bit matrix kernels. In that case smaller state need not imply lower latency. Reporting only compression or only speed hides this tradeoff; reporting tensor error alone does not reveal whether a class, retrieval choice, or next token changed.

We ask three falsifiable questions: (1) Does reapplying one weight-only recipe to the same FP32 checkpoint produce identical packed state on PyTorch, JAX, and TensorFlow? (2) When state or arithmetic differs, which measured decisions change, and which are provably stable under the observed logit perturbation? (3) What storage–latency–agreement choices are available on one A100 under strict FP32 and a separately controlled TF32 execution mode? A packed-state transplant tests whether quantization-time differences or inference-time arithmetic account for observed output dif-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ferences. A calibration/holdout split then tests a prespecified decision-constrained recipe selector rather than merely reporting a post hoc Pareto curve.

Our contribution is an experimental decomposition, not a new low-bit kernel or a claim that the margin inequality is new. We provide (i) tensor-complete state and output evidence across three backends and ten checkpoints spanning encoders, image classifiers, dual encoders, text decoders, an encoder–decoder, and a generative vision–language model, (ii) an exact packed-state transfer intervention on two checkpoints, (iii) measured decision and certificate outcomes on natural images and greedy generation, and (iv) synchronized timing interpreted alongside stored-variable bytes. Identifying implementation and converted-checkpoint repository revisions are omitted from this review copy; neutral source variants and evaluated-state fingerprints appear in Appendix A.5.

2 Related work

MQBench established that quantization should be evaluated for reproducibility and deployability, not only academic accuracy (Li et al., 2021). GPTQ and AWQ provide stronger quality–compression tradeoffs for language models using second-order or activation-aware information (Frantar et al., 2023; Lin et al., 2024). Our INT8/INT4 recipes are intentionally simpler and calibration-free: they isolate backend transfer rather than compete with these algorithms. The fact that weight-only state is smaller does not imply speed without suitable execution kernels; our comparisons measure the particular portable implementation, not a hardware limit.

Cross-framework validation has already shown that nominally corresponding operators or models can diverge (Pham et al., 2019; Wang et al., 2020; Deng et al., 2023). Keras 3 exposes a common model interface over PyTorch, JAX, and TensorFlow (Chollet et al., 2024), making a shared implementation possible but not automatically equivalent. Deep500 and MLPerf motivate controlled fairness and synchronized measurement (Ben-Nun et al., 2019; Reddi et al., 2020). We extend this line of work to a specific causal question: whether differences arise while constructing quantized state or while executing an exactly shared packed state. Standard floating-point analysis explains why bitwise output equality is too strong (Goldberg, 1991; Higham, 2002); margin-based stability is established (Tsuzuku et al., 2018). We apply it to measured backend and recipe perturbations, not to adversarial input balls.

3 Formal comparison

Let θ be one verified FP32 checkpoint and $b \in \{P, J, T\}$ a backend. The fixed recipe q is either identity (FP32), symmetric per-output-channel INT8, or symmetric block-wise INT4 with nominal group size 32 and two four-bit values per stored byte. Independently applying it gives $\widehat{\theta}_{b,q} = Q_{b,q}(\theta)$. We compare every stored model variable by canonical path, shape, dtype, and SHA-256. A matching FP32 manifest establishes common source weights; a changed quantized manifest identifies a state-construction difference without, by itself, proving a decision failure.

For input x, let $z_{b,q}(x)$ be a task head: class logits for ViT/SwinV2, image–text similarity scores for CLIP/SigLIP2, or next-token logits for Qwen, Gemma, Qwen3-VL, and Whisper. BERT has an encoder representation but no native task decision head, so only numerical comparisons apply. We measure $\delta(x) = \|z_{a,q}(x) - z_{b,r}(x)\|_\infty$ and check the actual argmax. The first system quantity is $S_{b,q}$, the sum of bytes in materialized model variables; it is *not* total process or GPU memory. The second is synchronized direct-eager median latency $T_{b,q}$ on a fixed workload. We keep these axes separate.

Proposition 1 (Observed-perturbation top-1 stability)**.** *Let* $i = \arg\max z_a$ *have margin* $m = z_{a,i} - \max_{k \neq i} z_{a,k} > 0$. *If* $m > 2\|z_a - z_b\|_\infty$, *then* $\arg\max z_b = i$.

Proof. For every $k \neq i$, $z_{b,i} - z_{b,k} \geq z_{a,i} - z_{a,k} - 2\delta \geq m - 2\delta > 0$. □

The margin-stability idea has prior precedent (Tsuzuku et al., 2018); our contribution is its measured cross-backend application, not this inequality.

This certificate is sufficient, not necessary: a failed inequality is inconclusive, and observed decisions are still checked directly. For greedy generation we compare logits only while the two trajectories

Pretrained model	Role	State MiB			Diagnostic outcome
		FP32	INT8	INT4	
BERT-base (Devlin et al., 2019)	Encoder	417.6	105.2	74.3	Representation probes; no task head
ViT-B/16 (Dosovitskiy et al., 2021)	Vision	330.2	85.4	54.4	INT8 vs. FP32, same backend: 781/800
CLIP ViT-B/32 (Radford et al., 2021)	Multi-modal	577.1	152.4	107.8	INT8 vs. FP32, same backend: 790/800
Qwen2.5-0.5B-Instruct (Qwen Team et al., 2025)	Decoder	1884.6	473.1	344.0	INT8 vs. FP32, same backend: 4/8
Whisper-base (Radford et al., 2023)	Enc.–dec.	276.9	75.4	59.4	Decoder-logit probes; no task accuracy test
SwinV2-Tiny (Liu et al., 2022)	Vision	108.3	27.6	17.3	INT8 vs. FP32, same backend: 783/800
SigLIP2-B/16 (Tschannen et al., 2025)	Dual enc.	1431.2	362.4	298.3	INT8 vs. FP32, same backend: 798/800
Qwen3-0.6B (Yang et al., 2025)	Decoder	2273.8	570.5	411.7	INT4, cross-backend: 5/8
Gemma3-1B-it (Gemma Team, 2025)	Decoder	3814.3	956.7	705.4	INT4, cross-backend: 5/8
Qwen3-VL-2B (Qwen Team, 2025)	Gen. VLM	8115.9	3227.8	2576.2	INT4, cross-backend: 15/20

Table 1: All ten pretrained checkpoints, with citations to the originating model work and measured A100 state sizes. State is the sum of materialized model-variable bytes (MiB), equal across backends for each recipe; it is not peak GPU memory. Diagnostic counts are minima over three backends (same-backend INT8 versus FP32) or three backend pairs (same-recipe INT4), not pooled results. Image outcomes use an 800-image holdout; continuation outcomes use eight text prompts or 20 images. BERT and Whisper have numerical probes, not task-accuracy claims.

share the same prefix; if the inequality holds at each step, induction certifies the continuation for that tested prompt and horizon. If the trajectories diverge, later logits have different inputs and are not interpreted as pure backend drift.

4 Experimental design

Checkpoint and workload matrix. We use ten public pretrained variants loaded through one multi-backend model implementation: BERT-base, ViT-B/16, CLIP ViT-B/32, Qwen2.5-0.5B-Instruct, Whisper-base, SwinV2-Tiny, SigLIP2-B/16, Qwen3-0.6B, Gemma3-1B-it, and Qwen3-VL-2B-Instruct (Devlin et al., 2019; Dosovitskiy et al., 2021; Radford et al., 2021; Qwen Team et al., 2025; Radford et al., 2023; Liu et al., 2022; Tschannen et al., 2025; Yang et al., 2025; Gemma Team, 2025; Qwen Team, 2025). In each fixed model–recipe comparison, the same Keras 3-based model implementation runs with PyTorch, JAX, or TensorFlow as the selected execution backend (Chollet et al., 2024). The $10 \times 3 \times 3 = 90$ model–backend–recipe cells each run in a fresh process. The same named checkpoint and deterministic inputs are used within each model. We archive output heads, per-tensor state manifests, raw timing samples, failures, and package/GPU provenance. All FP32 source state comparisons are checked before interpreting quantized comparisons.

Table 1 lists all ten evaluated checkpoints, their stored-variable sizes under each recipe, and the task-level diagnostic available for each architecture.

Recipes and execution. For each eligible FP32 kernel block, the stored FP32 scale is $s = \max(\max|w|/Q, 10^{-7})$ and $q = \mathrm{clip}(\mathrm{round}_{\mathrm{even}}(w/s), -Q, Q)$, with $Q = 127$ for per-output-channel INT8 and $Q = 7$ for grouped INT4; there is no zero point. INT4 groups one contracting axis using the largest divisor of its length not exceeding 32, then packs adjacent signed codes low-nibble first. Built Dense/EinsumDense projections are eligible, but embeddings use per-row INT8 even under the INT4 model recipe; paths containing `lm_head`, biases, and other noneligible layers remain FP32. Appendix A.4 gives the exact rounding, zero-block, exclusion, and packing rules, including the CLIP token embedding. Both recipes are weight-only: portable layers dequantize weights during floating-point execution. No training, task-data calibration, AWQ, or GPTQ optimization is performed. Strict mode disables TF32 at the driver/framework level. A separate throughput mode enables TF32 for FP32 weights with the same timing protocol; a sanity probe checks that arithmetic changed.

Natural-image decisions and prespecified selection. From the CIFAR-10 test split (Krizhevsky, 2009), we sample 100 images per class without replacement using a fixed RNG seed. Twenty per class form a calibration subset (200 total); the remaining 80 per class are held out (800 total). ViT, CLIP, and SigLIP2 images are bicubically resized to 224×224; SwinV2 uses 256×256. Each uses its fixed preprocessing constants, specified with the seed, exact selection algorithm, and prompts in Appendix A.6. The 1,000-way ViT/SwinV2 heads are assessed for decision agreement, not CIFAR accuracy. CLIP and SigLIP2 compare each image with ten fixed class-name prompts, allowing zero-shot accuracy and agreement to be reported. The prespecified selector chooses the smallest stored-state recipe among FP32, INT8, and INT4 that reaches at least 99% calibration agreement in each backend against that backend's FP32 decisions *and* for every cross-backend pair under the same recipe. It is then evaluated without retuning on the 800 held-out images. We report actual agreement and separately report certificate coverage.

Packed-state intervention. For ViT and CLIP under INT8 and INT4, we export PyTorch's complete quantized variable state, including integer kernels, scales, and unaffected variables. On each target backend we build and independently quantize the same FP32 checkpoint, then overwrite every variable by canonical path with the source packed value. We require exact post-assignment tensor hashes before running the same image set. Comparing independent and transplanted executions *within the target backend* isolates the effect of state construction; comparing source and transplanted target executions isolates residual backend execution differences. The intervention does not benchmark checkpoint transfer time.

Greedy continuations and performance. Qwen2.5, Qwen3, and Gemma3 each generate 16 greedy tokens for eight fixed prompts. The two Qwen models use a fixed-width, masked full-prefix forward; Gemma3 uses its eager growing-prefix prefill. Neither reuses a KV cache. We retain token IDs, logits, margins, common-prefix lengths, and per-step timings; the latter include diagnostic capture and shape-compilation overhead and are not presented as serving throughput. Qwen3-VL additionally uses image-conditioned greedy full-prefix prefill for eight tokens on 20 fixed images (two per CIFAR-10 class, bicubically resized from 32×32 to 128×128) with the prompt "Describe this image in one sentence." This is a decision-agreement diagnostic, not visual question-answering accuracy. The primary latency benchmark times the model call plus an explicit device synchronization after warm-up (five samples per cell) on an A100-SXM4-40GB, excluding download and construction. Repeats within one process are correlated; we make no significance-test or universal framework-ranking claim. Appendix A.1–A.6 discloses the recorded environment, source variants, state fingerprints, and input construction.

5 Results

5.1 Reapplying one recipe does not preserve packed state

All 90 strict-mode model–backend–recipe cells completed with finite heads and synchronized timing samples; logged whole-model clone failures for Gemma3 and Qwen3-VL were resolved by applying the same quantization recipe in place. Every FP32 state manifest matches exactly across the three backend pairs for all ten checkpoints (30/30 pair checks). None of the 60 independently quantized INT8 or INT4 state-pair checks is exact. The mismatches are not merely different file names or tensor shapes: for CLIP INT4, 147 of 545 stored variables differ between PyTorch and JAX and 295 differ between PyTorch and TensorFlow; for Qwen2.5 INT4 the corresponding counts are 169 and 338 of 459. In the CLIP PyTorch–JAX comparison, 146 changed tensors are packed kernels and one is an intentionally INT8-quantized token embedding table, while the recorded scales agree. A changed hash alone says nothing about a task decision, so we evaluate outputs separately.

5.2 Smaller stored state is not faster in this implementation

Across all ten checkpoints, median stored-variable bytes are 0.254 of FP32 under INT8 and 0.184 under INT4. Yet within-backend primary-workload latency ratios are not proportional to storage. Median INT8/FP32 ratios over ten checkpoints are 1.05× on PyTorch, 1.13× on JAX, and 1.08× on TensorFlow; median INT4/FP32 ratios are 1.82×, 2.60×, and 4.61×, respectively. Individual expansion-cohort TensorFlow INT4 ratios reach 10.70× for Qwen3-VL. Medians are calculated

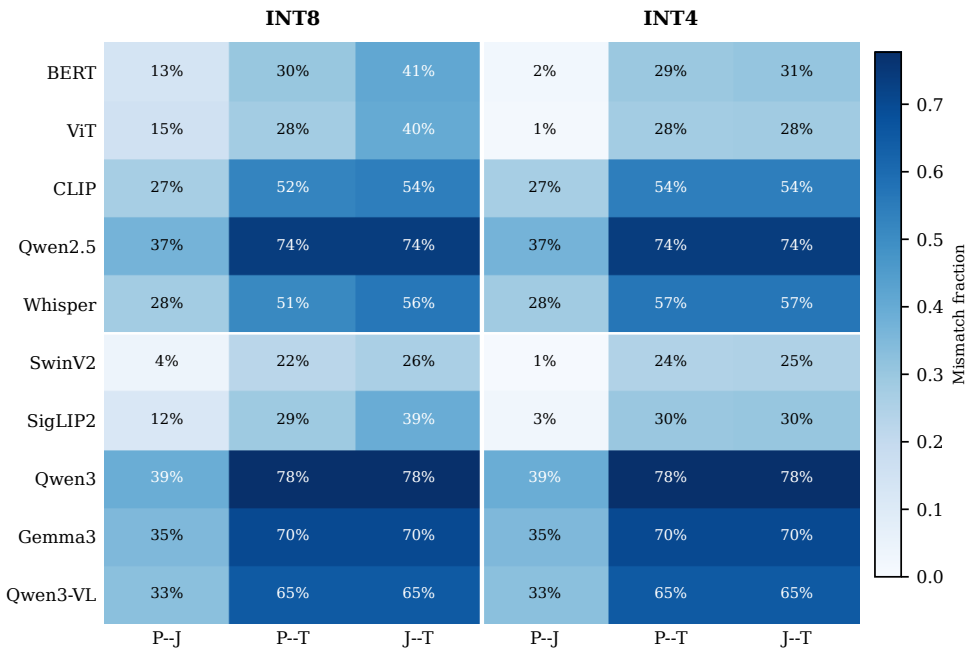

Figure 1: All ten checkpoints: fraction of complete model-state tensors with different hashes after independently applying the same recipe. P, J, and T denote PyTorch, JAX, and TensorFlow; all 30 FP32 pairs match exactly and are omitted. The horizontal divider separates the original and expansion cohorts. A changed hash is not equivalent to a changed decision.

	INT8 / FP32 latency			INT4 / FP32 latency		
Model	P	J	T	P	J	T
BERT-base	1.05	1.28	1.09	1.72	2.81	4.43
ViT-B/16	1.04	0.97	1.03	1.63	1.78	4.50
CLIP ViT-B/32	1.04	1.19	1.02	1.79	2.68	3.90
Qwen2.5-0.5B	1.06	1.21	1.12	1.91	2.79	5.81
Whisper-base	1.09	1.13	1.05	1.88	2.59	3.32
SwinV2-Tiny	1.04	0.99	1.05	1.53	1.85	2.46
SigLIP2-B/16	1.06	1.29	1.13	1.85	2.87	4.72
Qwen3-0.6B	1.05	1.12	1.08	1.86	2.61	5.80
Gemma3-1B-it	1.08	0.98	1.12	1.77	1.84	7.31
Qwen3-VL-2B	1.08	0.99	1.15	1.91	2.01	10.70

Table 2: Per-model A100 direct-eager latency ratios for all 90 strict-mode cells; P/J/T mean PyTorch/JAX/TensorFlow. Each ratio uses the median of five synchronized calls divided by the same model and backend's FP32 median. A ratio above one is slower, not faster. Workloads differ by model; absolute medians and shapes appear in Appendix A.2.

from unrounded model-level ratios before display rounding; they are descriptive, not uncertainty-adjusted framework rankings. They concern a portable weight-only path that dequantizes during forward execution; native INT4 kernels, compilation, and tuned inference stacks are outside the comparison.

5.3 A CALIBRATION-PASS RECIPE CAN MISS ITS HOLDOUT TARGET

All 36 natural-image cells for ViT, CLIP, SwinV2, and SigLIP2 completed on the same 1,000-image subset; their archives were verified against runtime SHA-256 manifests before analysis. The prespecified 99% calibration rule selects INT8 for ViT and CLIP: the worst required calibration agreement is 99.5% and 99.0%, respectively. On the untouched 800-image holdout, worst FP32 decision reten-

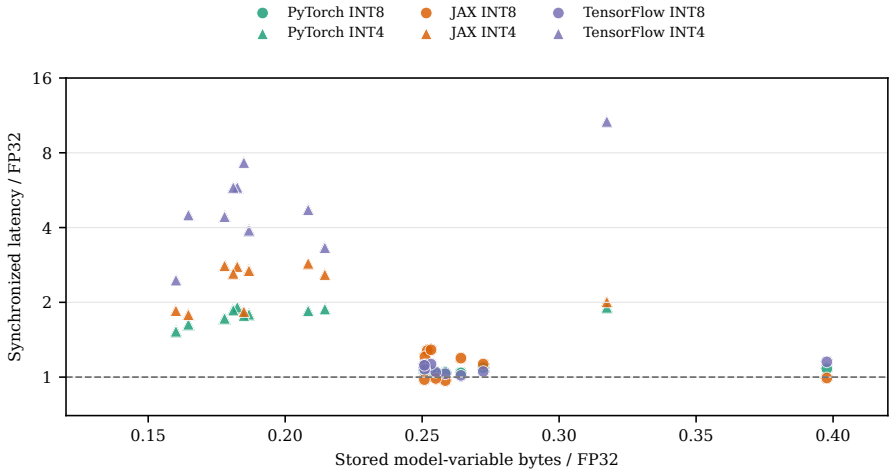

Figure 2: All ten checkpoints: within-backend cost of the two portable quantization recipes on A100, normalized to each model's own FP32 run. The vertical axis is logarithmic; the dashed line is equal latency. Reduced variable bytes do not imply an inference speedup without native low-bit kernels.

Model	Selected	Calib. min.	Holdout min.	Cross-backend	Zero-shot acc.
ViT-B/16	INT8	99.5%	97.625%	800/800	n/a
CLIP ViT-B/32	INT8	99.0%	98.75%	800/800	88.125%
SwinV2-Tiny	FP32	100%	100%	800/800	n/a
SigLIP2-B/16	INT8	99.0%	99.75%	800/800	95.625%

Table 3: Prespecified recipe selection on 200 calibration images and evaluation on 800 held-out images. Minima include within-backend FP32 retention and same-recipe cross-backend constraints. Cross-backend counts are minimum pairwise agreement for the selected recipe on holdout. Zero-shot accuracy uses ten fixed class prompts for the dual encoders; it is not an ImageNet evaluation.

tion falls to 97.625% (781/800) and 98.75% (790/800). The rule thus does *not* meet its 99% holdout target for these two models. For SwinV2, INT8 reaches only 96% minimum calibration agreement, so the selector retains FP32. For SigLIP2, it selects INT8 at exactly 99% minimum calibration agreement and achieves 99.75% (798/800) minimum holdout retention. INT4 fails the calibration constraint for all four models. This is a diagnostic split, not an estimate of performance on all natural images.

Despite the differing packed hashes, all three backend pairs make the same top-1 decision on all 800 held-out images for every tested image model and recipe. This observed equality should not be mistaken for a general certificate: for example, the sufficient margin bound certifies 797–799 of 800 pairwise INT8 decisions on CLIP/ViT, and only 772–785 of 800 CLIP INT4 decisions, while direct comparison finds all 800 equal. On the 1,000-image set, every SwinV2/SigLIP2 backend pair also agrees on all decisions; the INT4 certificate can still leave one agreed SigLIP2 decision uncertified. CLIP's holdout zero-shot accuracy is 87.625% in FP32, 88.125% in INT8, and 84.625% in INT4 on each backend; the corresponding SigLIP2 values are 95.75%, 95.625%, and 91.875%. These subset-level changes are descriptive, not evidence that quantization generally improves or degrades accuracy by these amounts.

5.4 Exact packed-state transfer separates quantization and execution

All eight target interventions (two models, two quantizers, two target backends) passed complete post-assignment hash verification: 201/201 variables for each ViT target and 545/545 for each CLIP target. The exported PyTorch states also match the corresponding independently archived PyTorch quantization runs exactly. On the 800 held-out images, each transplanted target agrees with the PyTorch source on all 800 decisions, and the observed-perturbation inequality certifies all 800 in

Model	Cross-backend, same recipe FP32	INT8	INT4	Within-backend vs. FP32 INT8	INT4
Qwen2.5-0.5B	8/8	8/8	8/8	4/8	0/8
Qwen3-0.6B	8/8	8/8	5/8	4/8	1/8
Gemma3-1B-it	8/8	8/8	5/8	6/8	0/8

Table 4: Minimum complete-continuation agreement over the three backend pairs (left) or three backends (right), out of eight fixed 16-token greedy prompts. Different recipes can change a continuation even when same-recipe backend decisions agree.

each intervention. Yet the logits are not generally bitwise identical, which exposes residual execution arithmetic after state is held fixed. Relative to each target's independently constructed state, all 1,000 ViT decisions and all CLIP INT8 decisions remain unchanged; CLIP INT4 changes one of 1,000 decisions on each target. This intervention shows that state construction can matter for a near-boundary decision, while the much larger set of differing state hashes need not change most decisions.

The isolated quantizer probe holds a single CLIP attention weight byte-identical across backends before quantization. In INT8, the PyTorch–JAX pair differs on only 21 of 589,824 integer values, each by one level, with identical scales. The corresponding INT4 pair differs on 37 of 589,824 unpacked values, again by one level with identical scales. TensorFlow produces small scale differences as well: at most 1.2×10^{-10} for INT8 and 1.9×10^{-9} for INT4 on this tensor. Thus a full-tensor hash failure can represent sparse boundary-level discrepancies, not wholesale corruption. The probe localizes the phenomenon to quantizer arithmetic on identical input bytes; it does not identify a unique low-level operation as the cause.

5.5 Backend agreement weakens at four-bit generation

The three language models' per-step logits and logit-derived greedy tokens were checked across all 27 generation cells; fixed-prompt tokenization is separately audited across backends. Table 4 reports complete 16-token continuation agreement on eight prompts, not token-level accuracy. For all three models, FP32 and INT8 continuations agree across all backends on every prompt, despite the INT8 state hashes differing. At INT4, the original Qwen2.5 still has 8/8 cross-backend agreement, but Qwen3 and Gemma3 have only 5/8 in the worst backend pair. Cross-backend INT4 execution therefore produces different greedy continuations for these two decoder families; unlike the ViT/CLIP transplant, this language-model comparison does not separately identify the contributions of state construction and execution arithmetic.

The sufficient margin certificate is distinct from observed agreement. Qwen2.5 has 128/128 certified cross-backend FP32 steps; its three INT8 pairs certify 126–127/128 and its INT4 pairs 52–72/128 even though all decisions agree. Qwen3 and Gemma3 likewise certify all 128 FP32 steps and at least 126 INT8 steps per backend pair. For INT4, the Qwen3 PyTorch–JAX, PyTorch–TensorFlow, and JAX–TensorFlow pairs certify 69/103, 61/98, and 69/113 common-prefix steps, respectively; Gemma3 certifies 82/111, 82/119, and 66/102 in the same pair order. These eight prompts are diagnostics, not a population estimate of generation reliability or quality.

5.6 Image-conditioned generation tests a different decision regime

All nine Qwen3-VL cells passed full-logit and token validation. Across 20 images and eight greedy tokens, all backend pairs agree completely in FP32 and INT8. At INT4, agreement is 16/20 (PyTorch–JAX), 19/20 (PyTorch–TensorFlow), and 15/20 (JAX–TensorFlow). Within each backend, INT8 retains 14/20 FP32 continuations and INT4 only 3–4/20: cross-backend agreement is not FP32 retention. The bound certifies 160/160 FP32, 153–154/160 INT8, and 84–99 of 132–154 comparable INT4 common-prefix decisions. These low-resolution diagnostics do not measure caption quality.

5.7 Execution precision is a separate cost–agreement axis

An isolated matrix-multiplication probe confirms that the strict and TF32-enabled settings produce different arithmetic on all three backends. In the nine FP32 model–backend primary workloads, TF32/strict median-latency ratios range from 0.87 to 1.07, with median 1.00; five timed calls in one process per mode do not establish a repeatable speedup. The change does perturb logits. Nevertheless, all six ViT/CLIP model–backend cells retain 800/800 held-out image decisions, and all three Qwen2.5 backend runs retain 8/8 sixteen-token continuations relative to strict execution. The image margin test certifies 783–800 of 800 decisions, depending on model and backend, and the matched-prefix Qwen test certifies 126/128 steps in each backend. This is an observed speed–decision point, not a claim that TF32 is universally decision-safe or faster. It also illustrates why equal argmax counts and a mathematical certificate should be reported separately.

6 Discussion and limitations

The experiment deliberately separates three meanings of "portable." A recipe can be portable as source code while producing different packed integer state; packed state can be transferred exactly while backend arithmetic still changes floating-point outputs; and changed outputs can leave a task decision unchanged. Neither a matching model API nor a single tensor tolerance settles all three. The intervention is particularly informative because it changes the quantized variables while holding the target backend, checkpoint source, and input protocol fixed. It cannot, however, by itself identify the specific primitive operation responsible for a quantization-time discrepancy.

The study is limited to ten checkpoints, one A100 accelerator type, one implementation revision, and weight-only symmetric INT8/INT4 recipes. It neither trains models nor compares against quality-optimized algorithms such as GPTQ and AWQ. The natural-image subset is deliberately class-balanced but modest and consists of low-resolution CIFAR images resized for ImageNet-trained models; image-classifier decision retention is not a CIFAR accuracy score. The text-generation sample uses eight fixed short prompts under greedy, no-cache decoding, and the VLM uses a small image-conditioned subset; neither predicts behavior under sampling, other domains, or long contexts. Timing is direct eager with five within-process samples per cell. Gradient tracking was not uniformly disabled across timed workers (Appendix A.1); recipe-level latency ratios may therefore partly reflect autograd bookkeeping or recipe-dependent trainable state and graph construction. We treat these ratios as implementation-specific observations, not controlled no-gradient performance comparisons. They also do not represent compiled execution, native low-bit kernels, production serving, or independent-runtime uncertainty. Stored-variable bytes are not total GPU memory. These boundaries are part of the claim, not exceptions to it.

7 Conclusion

Cross-backend quantization should be evaluated as a chain: verify the common FP32 source, inspect the packed state, measure numerical and task-level effects, and finally evaluate cost in the execution mode that will actually be used. This study provides a controlled protocol and an exact packed-state intervention for doing so, together with a prespecified holdout test of decision-constrained recipe selection. It does not equate a smaller checkpoint with a faster deployment or a changed hash with an incorrect prediction.

Reproducibility statement

Appendix A.1–A.6 specifies execution controls, all primary workloads, prompt and record schemas, quantization arithmetic, neutral upstream model identifiers, evaluated FP32 state fingerprints, and deterministic input construction. These details support protocol reconstruction and state verification. Code, converted checkpoint files, and raw run archives are not supplied with this review submission; consequently the paper does not claim turnkey independent rerun of its custom implementation. Identifying implementation locations and author-hosted checkpoint identifiers remain outside the review copy.

Ethics statement

The experiments use public pretrained checkpoints and a public image dataset; no new personal data are collected. A decision-stability certificate for selected outputs is not a claim of model safety, fairness, factuality, or deployment fitness. Speed results from one accelerator and implementation could mislead system selection if generalized beyond their stated mode.

AI Use Statement

Generative AI tools assisted with literature search, experiment design, the decision-stability proposition and proof, manuscript drafting and editing, figure preparation, benchmark and analysis code, and experiment orchestration. The author defined the research questions, selected the models and evaluation scope, directed and executed the experiments, and verified the methods, mathematical arguments, calculations, measurements, citations, and conclusions. All reported measurements were obtained from executed A100 runs and archived outputs; no experimental results were synthetically generated.

References

Tal Ben-Nun, Maciej Besta, Simon Huber, Alexandros Nikolaos Ziogas, Daniel Peter, and Torsten Hoefler. A modular benchmarking infrastructure for high-performance and reproducible deep learning. In 2019 IEEE International Parallel and Distributed Processing Symposium, pp. 66–77, 2019.

François Chollet et al. Introducing Keras 3.0. Keras project documentation, 2024. `https://keras.io/keras_3/` (accessed September 26, 2026).

Zizhuang Deng, Guozhu Meng, Kai Chen, Tong Liu, Lu Xiang, and Chunyang Chen. Differential testing of cross deep learning framework APIs: Revealing inconsistencies and vulnerabilities. In 32nd USENIX Security Symposium (USENIX Security 23), pp. 7393–7410, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT, pp. 4171–4186, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021.

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

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

David Goldberg. What every computer scientist should know about floating-point arithmetic. ACM Computing Surveys, 23(1):5–48, 1991.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2 edition, 2002.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yuhang Li, Mingzhu Shen, Jian Ma, Yan Ren, Mingxin Zhao, Qi Zhang, Ruihao Gong, Fengwei Yu, and Junjie Yan. MQBench: Towards reproducible and deployable model quantization benchmark. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, 2024.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12009–12019, 2022.

Hung Viet Pham, Thibaud Lutellier, Weizhen Qi, and Lin Tan. CRADLE: Cross-backend validation to detect and localize bugs in deep learning libraries. In Proceedings of the 41st International Conference on Software Engineering, pp. 1027–1038, 2019. doi: 10.1109/ICSE.2019.00107.

Qwen Team. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In International Conference on Machine Learning, pp. 8748–8763, 2021.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In Proceedings of the 40th International Conference on Machine Learning, volume 202, pp. 28492–28518, 2023.

Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, et al. MLPerf inference benchmark. In 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), pp. 446–459, 2020.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, et al. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. arXiv preprint arXiv:2502.14786, 2025.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In Advances in Neural Information Processing Systems, volume 31, 2018.

Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, and Dongdi Zhang. Deep learning library testing via effective model generation. In Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020. doi: 10.1145/3368089.3409761.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

A Protocol, inclusion rules, and raw records

Each model–backend–recipe worker starts in a fresh process so the Keras backend is fixed before import. The same model class and named checkpoint are loaded at FP32, then quantization is applied in memory. The archived tensor state includes non-trainable variables, integer kernels, floating scales, and embeddings. A completed core-probe cell requires state, timing, decision, and metadata files with finite expected-shape output arrays; image and generation cells have their own required-array checks. Failed and incomplete cells are excluded from summaries. One canonical input and additional fixed synthetic probes are recorded for every model. BERT, ViT, CLIP, Qwen2.5, SwinV2, SigLIP2, Qwen3, and Gemma3 have 24 probes each; Whisper and Qwen3-VL have eight. These are numerical diagnostics, not task-accuracy evaluations. The Qwen3-VL processor's patch size is explicitly set to match the checkpoint vision tower. The public whole-model functional-clone quantizer fails for Qwen3-VL and Gemma3; the same public recipe is applied in place to their built towers or nested layers, and all comparisons use those eager paths. Failed attempts remain in the private ledger.

The original-cohort primary synchronized workloads are BERT batch 1 and sequence 128; ViT batch 1 at 224×224; CLIP batch 1 with one 224×224 image and one 77-token text; Qwen2.5

batch 1 and sequence 128; and Whisper one fixed 30-second-equivalent log-mel feature tensor with eight decoder tokens. The expansion uses SwinV2 batch 1 at 256×256; SigLIP2 four paired 224×224 images and token sequences for a score head; Qwen3 and Gemma3 a short fixed tokenized prompt; and Qwen3-VL an image-conditioned 128×128 prefill producing next-token logits. Thus absolute times are not comparable across model rows. Every timing excludes process startup, weight download, model construction, quantization, and input preparation, but includes the model forward call and a device barrier. We record two warm-ups and five measured calls per workload. No timing outliers are removed. A second batch or sequence setting is recorded where available in the original cohort but is not used for the primary latency ratio.

Natural-image inputs and CLIP text prompts are shared byte-for-byte across cells. The subset's original CIFAR test indices and pixel SHA-256 are retained. The first 20 seeded sampled images per class are calibration; the remaining 80 are held out. The selection threshold and tie-break rule were fixed before the holdout array was analyzed. The packed-state experiment exports every quantized PyTorch variable and requires a path/dtype/shape match and exact post-assignment hash on each target backend before any target output is counted.

Qwen2.5's original run records retain tokenizer output IDs and masks. A separate tokenizer audit archives Qwen3/Gemma3 prompt IDs and masks for each backend; their generation cells retain the fixed prompt text, padded prompt lengths, per-step full-vocabulary logits, greedy tokens, margins, and step times. After the first divergent step in a prompt, subsequent backend logits are not interpreted as matched-prefix perturbations. TF32 arithmetic is tested in new processes with an explicit global override and backend setting; the settings and a matrix-multiplication sanity probe are archived. All public-facing statistical statements are descriptive counts or sample medians, with no population-level confidence claim from these diagnostic samples.

An initial image-conditioned PyTorch INT8 diagnostic exhausted A100 memory after three images while autograd was enabled. We changed the inference-only harness to disable gradient tracking and reran all PyTorch VLM recipes. The corrected FP32 $20 \times 8 \times V$ logit array is bitwise identical to the pre-correction FP32 trace, including all 160 greedy tokens. The failed run and both FP32 traces remain in the audit ledger; only complete post-correction VLM cells enter the comparisons.

A.1 Execution environment and arithmetic-mode check

The A100-SXM4-40GB reports driver 580.82.07. The runtime used Python 3.13.15, Keras 3.15.1, PyTorch 2.11.0+cu128, JAX/JAXlib 0.11.1, TensorFlow 2.20.0, NumPy 2.1.3, Transformers 5.17.0, Pillow 11.3.0, and tokenizers 0.23.2. Each cell is a fresh process with `KERAS_BACKEND` set before importing Keras; the float32 dtype policy, Python/NumPy/Keras worker seed 1729, `PYTHONHASHSEED=0`, `TF_DETERMINISTIC_OPS=1`, and `CUBLAS_WORKSPACE_CONFIG=:4096:8` are set before the model call. PyTorch also sets `torch.use_deterministic_algorithms(True, warn_only=True)` and disables cuDNN benchmarking. These requests do not establish bitwise-deterministic arithmetic across backends.

Strict mode sets `NVIDIA_TF32_OVERRIDE=0`, disables PyTorch CUDA matmul/cuDNN TF32, and selects JAX `jax_default_matmul_precision=highest`; the original-cohort strict comparison workers additionally call TensorFlow's explicit TF32 disable API. The separate FP32 throughput-mode workers set the override to 1, enable PyTorch matmul/cuDNN TF32 with `set_float32_matmul_precision("high")`, select JAX `tensorfloat32`, or enable TensorFlow TF32, respectively. The same FP32 matrix sanity probe returned 1024.1046142578 in strict mode and 1024.0 with TF32 enabled on each backend, versus an FP64 reference of 1024.1024169922. Thus the arithmetic setting had a measurable effect, but no general speedup followed.

Calls are direct eager Keras forwards with `training=False` where the model-call interface accepts it; no explicit `torch.compile`, `jax.jit`, or `tf.function` wrapper was applied. Gemma3 and Qwen3-VL use their eager prefill interfaces. The corrected PyTorch image-conditioned Qwen3-VL diagnostic additionally runs under `torch.inference_mode()`; the other worker calls were not uniformly wrapped in a no-gradient context. The timed call includes a device barrier: `torch.cuda.synchronize()`, `jax.block_until_ready` on all output leaves, or `tf.experimental.sync_devices()` (falling back to copying an eager tensor to

host if unavailable). Each primary workload has two warm-ups and five unfiltered synchronized samples; initial load, conversion, quantization, and input preparation are excluded. No multi-process confidence interval is claimed.

A.2 ALL PRIMARY-WORKLOAD MEDIAN LATENCIES

Model	Backend	FP32 ms	INT8 ms	INT4 ms
BERT-base	PyTorch	73.3	77.0	126.4
	JAX	146.5	187.4	411.4
	TensorFlow	269.4	292.4	1192.2
ViT-B/16	PyTorch	63.1	65.8	103.0
	JAX	163.1	157.9	290.9
	TensorFlow	221.8	228.8	997.0
CLIP ViT-B/32	PyTorch	139.2	144.8	249.3
	JAX	271.1	323.2	727.4
	TensorFlow	526.2	534.7	2052.1
Qwen2.5-0.5B	PyTorch	136.4	144.7	260.9
	JAX	302.1	364.6	841.9
	TensorFlow	617.5	690.1	3584.8
Whisper-base	PyTorch	90.4	98.8	170.0
	JAX	196.5	221.8	508.2
	TensorFlow	344.3	362.6	1142.8

Table 5: Original five-checkpoint cohort: measured model-call latency (ms), median of five synchronized samples, rounded to 0.1 ms, with no outlier removal.

Model	Backend	FP32 ms	INT8 ms	INT4 ms
SwinV2-Tiny	PyTorch	110.9	114.8	169.6
	JAX	279.1	275.6	516.7
	TensorFlow	398.1	416.7	978.2
SigLIP2-B/16	PyTorch	138.5	146.5	256.2
	JAX	308.3	397.2	885.5
	TensorFlow	537.4	606.2	2536.1
Qwen3-0.6B	PyTorch	175.9	184.9	327.5
	JAX	452.3	507.5	1182.8
	TensorFlow	777.4	835.9	4510.1
Gemma3-1B-it	PyTorch	189.1	203.7	334.6
	JAX	676.6	661.1	1242.2
	TensorFlow	827.8	923.1	6049.1
Qwen3-VL-2B	PyTorch	285.0	308.1	544.1
	JAX	1023.8	1015.2	2056.7
	TensorFlow	1270.5	1463.6	13597.2

Table 6: Expansion cohort: measured direct-eager model-call latency (ms), median of five synchronized samples, rounded to 0.1 ms. Workloads differ by model and are specified above; only within-row recipe ratios are used in the main analysis.

A.3 FIXED PROMPT AND RECORD SCHEMA

The eight language-model prompts were: “The capital of France is”; “Water freezes at a temperature of”; “Complete the sequence: 2, 4, 8, 16,”; “Write one sentence explaining gravity.”; “Translate to French: good morning”; “In Python, a list is created using”; “Explain photosynthesis in one short sentence.”; and “Write a valid JSON object with one key named answer.” The separately archived Qwen3/Gemma3 tokenizer audit checks prompt IDs and masks across backends; comparisons require the same tokenization within each model, not across different tokenizers.

Each completed probe cell records a variable manifest with canonical path, shape, dtype, byte count, and SHA-256; per-workload timing JSON with the five unfiltered samples and available device-memory counters; probe-decision JSON; and full numerical head arrays. Each natural-image cell

adds a $1000 \times C$ head array and metadata listing calibration/holdout membership; the shared dataset file records original image indices. Each packed-state source saves the complete variable arrays and path/hash maps; each target saves post-assignment verification, the $1000 \times C$ head array, and timing. Each text-generation cell saves an 8×16 token array, a matching margin array, a $16 \times 8 \times V$ FP32 logit array, prompt text and lengths, and step timings; input IDs and masks are archived in the tokenizer audit. Each image-conditioned VLM cell saves corresponding 20×8 token/margin arrays, a $20 \times 8 \times V$ logit array, source image positions, and prompt/grid metadata. Archive manifests contain byte lengths and SHA-256 for every bundle; incomplete cells are excluded, never imputed.

A.4 Exact quantization specification

The workers pass the bare `int8` or `int4` mode with nominal group size 32 to one weight-only quantizer, without calibration or per-layer overrides. The default configuration quantizes eligible built Dense and EinsumDense kernels and Embedding tables, but skips any layer path containing `lm_head`. Biases, convolution and normalization weights, and other noneligible layer classes stay FP32. Embeddings are deliberately treated differently from Dense kernels: each token row uses the INT8 rule even when the model recipe is INT4. Thus an "INT4 model" contains packed INT4 projection kernels, INT8 embedding tables, float32 scales, and unaffected FP32 variables. The CLIP text token embedding follows this rule, not an exception that leaves it unquantized.

For a reduction block B of FP32 weights, each backend evaluates $M_B = \max_{j \in B} |w_j|$, $s_B = \max(M_B/Q, \epsilon)$, and $q_j = \mathrm{clip}(\mathrm{round}(w_j/s_B), -Q, Q)$, with $\epsilon =$ `keras.config.epsilon()` $= 10^{-7}$ and no zero point. The operations use backend implementations of `keras.ops` in FP32; `round` is nearest-integer with ties to even for exact halfway inputs. The clip follows rounding and uses the closed integer interval shown. An all-zero block stores $s_B = 10^{-7}$ and all-zero codes. Scales are stored as float32 and are cast to the layer's compute dtype when the integer kernel is dequantized for floating-point execution. This specifies the mathematical recipe, not a claim that backend floating-point reductions and divisions are bitwise identical.

INT8 sets $Q = 127$, stores signed int8 codes in $[-127, 127]$, and reduces over the Dense kernel's input axis or the EinsumDense kernel's contracting axis or axes, leaving one scale per output coordinate. Embedding scales reduce over the feature axis, leaving one scale per token row. INT4 sets $Q = 7$ and requires one contracting axis of even length K. After moving this axis to the end, it chooses $g = \max\{d \in \mathbb{N} : d \le 32,\ d \mid K\}$; consecutive blocks of g values each receive one scale. This is the exact divisor fallback rather than padding a nondivisible dimension; for accepted positive even K, $g \ge 2$. Odd K raises an error. The generated signed codes are in $[-7, 7]$; -8 is unused. For adjacent codes along that axis, the stored uint8 byte is $b_i = (q_{2i} \bmod 16) + 16(q_{2i+1} \bmod 16)$: the earlier value occupies the low nibble. On unpacking, nibble values above 7 are converted to signed values by subtracting 16. No native low-bit matmul is used in the timed portable path.

For each model variable, the worker converts its value to a NumPy array without changing dtype, makes it C-contiguous, views its bytes as unsigned 8-bit values, and computes SHA-256 over those bytes. Path, shape, dtype, and byte count are recorded separately; file serialization does not enter the per-tensor digest. The comparisons ran on x86-64 A100 hosts and do not test cross-endian portability. The CLIP embedding's FP32 hash agrees across backends. Within each backend, its INT8 and INT4 table hashes agree; across PyTorch and JAX, the integer table hashes differ while the scale hashes match. Thus this changed table is quantized state, not an unexpectedly modified FP32 variable.

A.5 Checkpoint origins and evaluated configurations

Table 7 names the neutral third-party upstream variants underlying the ten evaluated FP32 states. Except for Qwen2.5, the runs loaded public, preconverted Keras weight snapshots, not the upstream files directly. Those hosted snapshot identifiers and repository revisions would expose the implementation's provenance, so they are omitted from this anonymous review copy. The exact evaluated variables are instead bound by the FP32 state fingerprints in Table 8. An upstream variant alone is *not* an exact substitute for a converted snapshot, and the absence of a downloadable anonymous snapshot or converter code limits independent re-execution.

Evaluated model	Neutral upstream variant	Relevant evaluated setting
BERT-base	`google-bert/bert-base-uncased`	Uncased encoder; 128-token timing input
ViT-B/16	timm `vit_base_patch16_224.augreg_in21k_ft_in1k`	Patch 16; 224×224
CLIP ViT-B/32	`openai/clip-vit-base-patch32`	Patch 32; 224×224; text width 77
Qwen2.5-0.5B-Instruct	`Qwen/Qwen2.5-0.5B-Instruct`	Direct source conversion; 128-token timing input
Whisper-base	`openai/whisper-base`	80 mel bins; 3000 frames; eight decoder tokens
SwinV2-Tiny	timm `swinv2_tiny_window8_256.ms_in1k`	Window 8; 256×256
SigLIP2-B/16	`google/siglip2-base-patch16-224`	Patch 16; 224×224; text width 64
Qwen3-0.6B	`Qwen/Qwen3-0.6B`	Plain-text tokenizer; five-token timing prompt
Gemma3-1B-it	`google/gemma-3-1b-it`	Plain-text tokenizer with BOS; six-token timing prompt
Qwen3-VL-2B-Instruct	`Qwen/Qwen3-VL-2B-Instruct`	Vision patch 16, spatial merge 2, temporal patch 2; 128×128 image

Table 7: Source model variants, not a claim that the independently downloaded upstream files are byte-identical to the evaluated converted states. The Qwen3-VL processor was explicitly initialized with patch size 16 to match the checkpoint vision tower rather than the hosted processor configuration's 14-pixel value; the merge and temporal patch sizes were 2.

The Qwen2.5 direct upstream revision recorded at run time is `7ae557604adf67be50417f59c2c2f167def9a775`. Immutable upstream revisions for the nine converted variants are not present in the retained benchmark provenance. The converted snapshots were architecture-mapped into float32 Keras variables, including tensor-layout changes required by the target layers; no task-data fine-tuning was performed. We cannot claim that naming the upstream variant and its architecture reconstructs the exact converted weights without the unpublished mapping and snapshot. For an auditable, identity-neutral identifier, Table 8 reports SHA-256 over the newline-joined sequence, in model-variable order, of `path|comma-separated shape|dtype|per-tensor SHA-256` for every FP32 variable. The per-tensor digest itself is over contiguous raw bytes as specified in Appendix A.4. These fingerprints matched across all three backend loads.

Model	Evaluated FP32 state fingerprint (SHA-256)
BERT-base	`cb1e28932ad4f413e7f945d4c9ed8d3f6418ec83ac69577d30e1e06f3aa8089a`
ViT-B/16	`e2ec6f9c0890d7f6c76f3706f276e4eff8d2e6b4f4dbcf05869d68473f86d109`
CLIP ViT-B/32	`e589a42ef152d330756a984e309c8549352a00b455ffc5f178d9063f6ab0c0da`
Qwen2.5-0.5B	`a7e6ed7c72dfb7ec0b54f1fca10c2d3c23083e8905ca86a0782244e822dfb4b9`
Whisper-base	`b04934077364b0ed3f7d0f511a211413afedb2332e55d5e9f959c2888112e4a6`
SwinV2-Tiny	`5f0d8e9050e4c840975944f282a537b687f53e405af6deb4cb62e1b7e03a8a38`
SigLIP2-B/16	`e0dc4a4e233505a523d624cd2efc78b89fbdd467130b09e101cc324a018ae5d2`
Qwen3-0.6B	`83149c27dfaf15852ed049468f2644eb3febac72a5ea76736240dad08834f960`
Gemma3-1B-it	`8c15c1f2201b1a5ee170cfc9d18e739bb4d677a91657d1d3419368ca2e89238d`
Qwen3-VL-2B	`fb346eed09afff786fe290ae1f7f2366a714311be550b3d25b160fac063a1d66`

Table 8: Identity-neutral fingerprints of the exact FP32 model-variable states used by all three backends. These are not upstream repository commit hashes.

A.6 Input construction and preprocessing

The natural-image diagnostic uses `uoft-cs/cifar10`, file `plain_text/test-00000-of-00001.parquet`, whose downloaded bytes have SHA-256 `841389e6f2d64f28bf17310e430aebac20ec3ba611a3c5e231dc93c645ce84de`. Decode each image as RGB uint8 and retain the Parquet row order. With NumPy 2.1.3, construct `rng=np.random.default_rng(2027)` (PCG64), then for class IDs 0 through 9 in order call `rng.choice(np.flatnonzero(labels==class_id), size=100, replace=False)`. Preserve each returned order and concatenate by class; the first 20 sampled positions per class form calibration and the next 80

holdout. The resulting $1000 \times 32 \times 32 \times 3$ uint8 array has raw-byte SHA-256 `eff83f04c0776b6388b45ab110ac3c7d83fdd83ab9f42adca226e44da2ff1276`. This algorithm, the source-file hash, and the array hash specify all 1,000 original test indices without a separate data file. The image-conditioned diagnostic uses sampled positions $0, 1, 100, 101, \ldots, 900, 901$; their original test indices in that order are 9695, 3421, 4431, 1260, 1567, 6352, 8077, 8594, 3498, 6265, 2026, 8211, 6662, 300, 8206, 9063, 2043, 2429, 7024, and 2810.

For ViT, CLIP, and SigLIP2, each RGB image is resized directly to 224×224 with Pillow 11.3.0 `Image.Resampling.BICUBIC`; SwinV2 uses 256×256. There is no center crop or explicit padding and no separate antialiasing flag. Convert the resulting HWC uint8 array to float32 in $[0, 1]$ by division by 255, then apply $(x - \mu)/\sigma$ channelwise. ViT and SigLIP2 use $\mu = \sigma = (0.5, 0.5, 0.5)$; CLIP uses $\mu = (0.48145466, 0.4578275, 0.40821073)$ and $\sigma = (0.26862954, 0.26130258, 0.27577711)$; SwinV2 uses $\mu = (0.485, 0.456, 0.406)$ and $\sigma = (0.229, 0.224, 0.225)$. The natural-image workers use batches of 10 for ViT/CLIP and 40 for SwinV2/SigLIP2. No classifier fine-tuning or CIFAR-specific normalization is applied.

CIFAR class	CLIP prompt	SigLIP2 prompt
airplane	a photo of an airplane.	a photo of a airplane.
automobile	a photo of an automobile.	a photo of a automobile.
bird	a photo of a bird.	a photo of a bird.
cat	a photo of a cat.	a photo of a cat.
deer	a photo of a deer.	a photo of a deer.
dog	a photo of a dog.	a photo of a dog.
frog	a photo of a frog.	a photo of a frog.
horse	a photo of a horse.	a photo of a horse.
ship	a photo of a ship.	a photo of a ship.
truck	a photo of a truck.	a photo of a truck.

Table 9: Literal class prompts used by the two dual-encoder diagnostics; the unusual SigLIP2 articles are retained because they are what the worker tokenized.

CLIP uses its fast tokenizer with a 77-token limit, truncation, begin/end tokens, and end-token padding; its attention mask accompanies text IDs. SigLIP2 uses its fast tokenizer with a 64-token limit, an appended `<eos>` token, and right `<pad>` padding, without a separate attention-mask input to the score head. The eight plain language prompts are listed in Appendix A.3. They are passed as strings, without a chat template or added generation prompt, to the checkpoint tokenizer. Qwen2.5/Qwen3 do not automatically add special tokens; Gemma3 prepends `<bos>`. Tokenizer batches initially right-pad with zero IDs and a binary mask; generation workers extract the nonpad IDs and left-align them within a common prompt width before greedy decoding. The Qwen2.5 and Qwen3 real prompt lengths are $(5, 6, 17, 6, 6, 8, 9, 11)$, with padded width 17; Gemma3 lengths are $(6, 7, 18, 7, 7, 9, 8, 12)$, with padded width 18. No truncation or maximum-length override is applied to these eight prompts. Generation runs for exactly 16 argmax steps without EOS-based early stopping or KV-cache reuse.

For Qwen3-VL, each selected CIFAR image is first bicubically enlarged from 32×32 to 128×128 RGB. The processor uses patch size 16, spatial merge 2, temporal patch 2, an 8×8 spatial patch grid, CLIP mean/std as above, float32 scaling by $1/255$, CHW ordering before patch packing, and the inherited smart-resize limits of 56^2 and $28^2 \cdot 1280$ pixels; at 128×128, smart resize leaves the dimensions unchanged. Its ChatML input has the default system text "You are a helpful assistant.", one user image, the literal user text "Describe this image in one sentence.", and an assistant generation prefix; the 16 merged image tokens are expanded into the input sequence. The recorded diagnostic has 44 input IDs per image and generates exactly eight greedy tokens without EOS stopping. The separate primary timing input uses a seeded synthetic 128×128 image and the text "What is shown in this image? Answer briefly."; the same processor constructs 47 input IDs. The hosted processor configuration's 14-pixel value was overridden with the model's 16-pixel patch size.

The primary timing inputs are synthetic, not the natural-image or language-decision sets. BERT uses one 128-token row, with IDs $((17i + 103) \bmod 30000)$ except first/last IDs 101/102, an all-one mask, and zero token types. ViT uses one 224×224 image; CLIP uses one such image with a

77-token row; Qwen2.5 uses one 128-token row with IDs $((37i + 1000) \bmod 150000)$ except first ID 151644. Whisper uses one 80×3000 float32 log-mel tensor sampled from a seeded standard normal and multiplied by 0.1, with decoder IDs $(50258, 50259, 50359, 50363, 220, 318, 257, 1332)$. SwinV2 uses one 256×256 synthetic image; SigLIP2 uses four 224×224 images and four 64-token synthetic rows with IDs $((31i + 101) \bmod 256000)$ except final ID 1; Qwen3 and Gemma3 time the tokenized plain text "The capital of France is" at lengths 5 and 6, respectively; Qwen3-VL uses the 47-token image-conditioned prefill above. Synthetic image arrays are float32 normal draws scaled by 0.2, using PCG64 seeds 1729 for original-cohort images and $1729 +$ size for expansion images; the VLM timing image uses PCG64 seed 9917 to draw RGB uint8 pixels. These shapes and sequences are the denominators for the within-model latency ratios, not a common workload across different models.