COMPACT: A UNIFIED FRAMEWORK FOR MULTIMODAL SPLIT LEARNING WITH COMPRESSED ACTIVATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Split learning enables collaborative training of vision–language models without sharing raw data, but transmitting intermediate activations across distributed clients can incur substantial communication overhead. We investigate a lightweight multi-client setting in which client-side modules are frozen, eliminating gradient transmission and parameter synchronization, and focus on activation compression at the client-server boundary. We introduce `COMPACT`, a unified framework that supports heterogeneous vision–language backbones with interchangeable compression modules, and evaluate seven compression strategies across four vision–language models on four medical VQA benchmarks. We find that compression effectiveness is strongly model-dependent, and blockwise NormalFloat activation quantization (Ql) provides the most consistent performance across backbones. Amongst compression configurations, larger cosine distance from uncompressed representations is consistently related to downstream accuracy, with Spearman correlations ranging from -0.72 to -0.85. Layer-wise analysis further shows that internal representation drift caused by reconstruction errors accumulates through frozen layers. Moreover, we identify two recurring phenomena associated with performance degradation: unstable training of reconstruction modules under short training budgets and poor utilization of quantization levels. Finally, in a simulated cross-silo setting, `COMPACT` reduces communication by 72.5% for 4-bit Ql, while preserving 98.5% baseline performance. Together, these results show that compression effectiveness depends on both compression rate and its interaction with pretrained representations and downstream adaptation.

1 INTRODUCTION

As general-purpose foundation models are increasingly adapted to specialized domains like healthcare and finance, access to domain-specific data has become essential for effective fine-tuning (Bommasani et al., 2022). Such data, however, is distributed across institutions and cannot be directly pooled due to privacy or regulatory constraints (Li et al., 2022). For example, financial institutions aim to adapt risk models using private customer data (McKelvey, 2000), hospitals seek to jointly fine-tune medical foundation models without exchanging patient records (de Jong et al., 2025).

Federated learning (FL) provides a standard approach to collaborative model training without centralizing raw data at the central server, but horizontal FL methods typically require each participating client to locally execute the whole model and periodically synchronize trainable updates (McMahan et al., 2017; Kairouz et al., 2021). To reduce the communication cost of foundation models, recent FL methods exchange only prompts, low-rank adapters, or reasoning traces (Qiu et al., 2024; Sun et al., 2024; Yao et al., 2026), yet each client must still host and run the full backbone. This cost is especially high for vision-language models (VLMs), whose number of visual tokens grows with input resolution (Dong et al., 2024) and whose training memory grows with sequence length (Dao et al., 2022). In contrast, split learning (SL) is better suited to settings where data remain decentralized but large-scale model computation can be centralized: clients execute only a lightweight front end, while most model storage, computation, and optimization remain on a shared server (Vepakomma et al., 2018). This asymmetric deployment reduces client-side resource requirements,

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.

but places high-dimensional intermediate representations and their gradients on the communication path, creating a substantial bottleneck during model adaptation (Oh et al., 2025).

Prior work reduces this communication cost through sparsification, quantization, and learned compression. Randomized Top-K sparsification (Zheng et al., 2023) selectively transmits intermediate activations, while SplitFC (Oh et al., 2025) combines adaptive feature selection and quantization. More recent work explores structured and learned compression strategies (Lin et al., 2026; Fang et al., 2026; Meuwissen et al., 2026). However, these methods are studied under different model architectures and experimental settings, and all of them are evaluated on unimodal tasks rather than on VLMs. It remains unclear why the same compression strategy succeeds on one VLM but fails on another under the same communication budget.

We address this gap with `COMPACT`, a unified framework for communication-efficient VLM fine-tuning in SL via COMPressed ACTivations (Figure 1). `COMPACT` keeps the vision encoder in client frozen, as in visual instruction tuning (Liu et al., 2023), so no gradients are returned to clients and all cut-layer traffic consists of forward activations, which COMPACT compresses with interchangeable modules. In contrast to the unimodal benchmarks of prior work, we systematically evaluate seven compression strategies across four VLMs and two compression rates, nominally 4 and 2 bits. Our results reveal substantial cross-model variation in compression effectiveness. For dynamic per-tensor quantization (DQ) on the MedXpert benchmark, COMPACT retains 99.7% of the uncompressed baseline on Qwen3-VL-4B (22.41/22.47), but only 18.3% on PaliGemma2-10B (3.34/18.22). However, Ql, adapted from QLoRA (Dettmers et al., 2023), retains at least 98% of baseline performance on all four VLMs at 4 bits and at least 92% at 2 bits.

To understand when compression fails, we measure the cosine distance between how far compressed activations deviate from their uncompressed counterparts. Across configurations, larger cosine distance is consistently associated with lower retained downstream accuracy, with Spearman correlations ranging from -0.72 to -0.85 across backbones. We further identify two failure modes associated with severe representation drift and performance collapse: instability introduced by difficult reconstruction optimization and underutilized quantization levels. Finally, we simulate a three-client multi-hospital SL setting to evaluate `COMPACT` in a realistic scenario. It reduces cut-layer traffic by 72.5% relative to 16-bit activations while retaining 98.5% of the performance of uncompressed SL.

Our contributions are summarized as follows:

- We propose `COMPACT`, **a SL framework and benchmark for VLMs** that freezes the client-side encoder to eliminate *backward traffic* and *compresses* the remaining *forward activations* with interchangeable modules across heterogeneous backbones.
- We show that **the performance of activation-compression in SL is strongly model-dependent**: methods that are effective for one backbone may degrade substantially on another, whereas Ql remains consistently robust across the evaluated models.
- We **connect compression failure to representation drift**, showing that performance degradation is linked with the cosine distance between compressed and uncompressed activations (Spearman ρ from -0.72 to -0.85) and further tracing severe failures due to *reconstruction instability* and *underutilized quantization levels*.
- We **validate our benchmark findings in a realistic three-client cross-silo deployment**, where 4-bit Ql reduces activation communication by 72% while retaining 99% of uncompressed performance and lowers the uplink required for full-speed training by 3.3×.

2 Background and Related Work

Split learning and multi-client training. SL enables collaborative training while keeping raw data with their owners, motivated by privacy-sensitive and data-governance-constrained settings such as healthcare (Gupta & Raskar, 2018; Vepakomma et al., 2018). Unlike conventional horizontal FL, which replicates model execution across clients and periodically synchronizes trainable updates, SL partitions computation between data-owning clients and a shared server. This avoids deploying the full model-training stack at every client and reduces the dependence of synchronous training on heterogeneous client settings (Bonawitz et al., 2019; Li et al., 2020). SL instead concen-

trates most model storage, optimization, and computation at a shared server while clients execute only a small portion of the model (Thapa et al., 2022; Pal et al., 2021; Lin et al., 2024c).

Split learning for foundation models. The increasing compute and memory requirements of foundation-model adaptation have motivated split-learning systems that distribute model computation between clients and the server. Recent approaches extend SL to LLM fine-tuning using parameter-efficient adaptation, while retaining trainable client-side parameters and periodically aggregating client-side adapters (Lin et al., 2024b; Zhang et al., 2024). Other approaches decouple client- and server-side optimization using local objectives, thereby reducing or eliminating gradient transmission across the split boundary (Mu & Shen, 2023; Nair et al., 2025; Zhang et al., 2026). SplitFrozen instead freezes the device-side model layers so that clients perform only forward computation while parameter-efficient fine-tuning is concentrated on the server (Ma et al., 2026). Prior SL work on VLMs assigns only the image modality to the client and processes the paired text on the server (Guo et al., 2025). However, private images and paired text are usually held by the same data owner, so it is unrealistic for one party to process only the images while another handles the text.

Activation compression in split learning. Offloading model computation to the server shifts a key systems cost to communication, since intermediate representations must be repeatedly transmitted across the client–server boundary. Prior work has explored adaptive feature dropping and quantization (Oh et al., 2025), mixed-bit activation quantization (Song et al., 2025), sparsification (Zheng et al., 2023), clustering- and vector-quantization-based compression (Wang et al., 2022a), learned bottleneck representations (Shao & Zhang, 2020), and batch-wise compression (Hsieh et al., 2022). However, these approaches are typically evaluated under different model architectures, client–server partition locations, datasets, and training protocols, making it difficult to determine whether a compression strategy that preserves downstream performance for one pretrained model remains effective for another. This question is particularly relevant to pretrained VLMs, whose intermediate representations can differ substantially across architectures. We refer to Appendix A, for more detailed related work and comparative analysis of SL for VLMs.

3 COMPACT FRAMEWORK

3.1 MULTIMODAL SPLIT-LEARNING FRAMEWORK

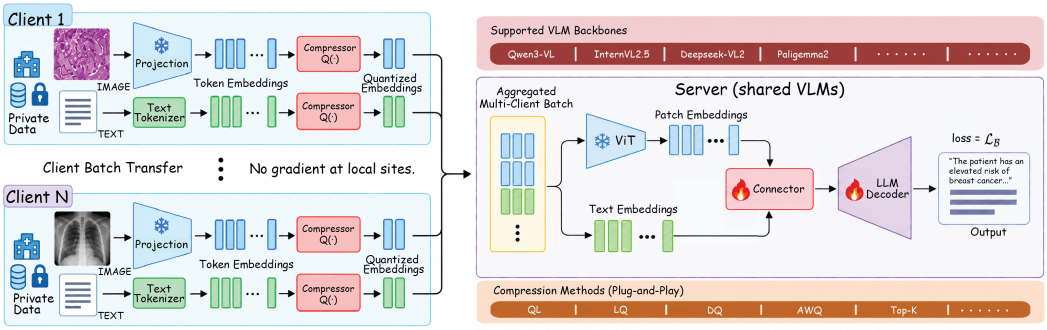

Figure 1: **Overview of the COMPACT framework.** Multiple clients keep raw multimodal inputs local and transmit compressed intermediate representations and tokenized supervision to the server. Clients execute only frozen lightweight front ends, with vision and text representations split after patch projection and token embedding, respectively. The server performs the remaining VLM computation, loss calculation, and model optimization. Since the client VLMs are frozen, no gradients or updated parameters need to be transmitted back to the clients. A common compression interface enables controlled comparison of activation-compression methods across heterogeneous VLM backbones. Snowflake & flame symbols denote frozen and trainable components, respectively.

We consider N clients $\{\mathcal{C}_i\}_{i=1}^{N}$ and a shared compute server. Client $\mathcal{C}_i$ holds a local dataset $\mathcal{D}_i = \{(x_{i,j}^v, x_{i,j}^t, y_{i,j})\}_{j=1}^{n_i}$ of vision inputs, text inputs, and task supervision. The raw vision and text

inputs are never transmitted, while task supervision is de-identified and transmitted to the compute server for loss computation [1]. Rather than running and optimizing the pretrained VLM locally, each client computes $h^m_{i,j} = f^m_c(x^m_{i,j})$ for each sample j and modality $m \in \{v, t\}$, where f^m_c denotes the frozen client-side layers up to the split point. Then it compresses $h^m_{i,j}$ with a modular compressor Q_ϕ, and transmits $z^m_{i,j} = Q_\phi(h^m_{i,j})$ together with the tokenized supervision. The server reconstructs $\hat{h}^m_{i,j} = Q^{-1}_\phi(z^m_{i,j})$ and runs the remaining vision processing, multimodal fusion, and language-model computation f_s [2]. For a mini-batch $\mathcal{B}$, the server computes the fine-tuning objective as

$$\mathcal{L}_\mathcal{B} = \frac{\sum_{(i,j)\in\mathcal{B}} \sum_{k\in\mathcal{S}_{i,j}} \ell_{\mathrm{CE}}\Big(f_s(\hat{h}^v_{i,j}, \hat{h}^t_{i,j})_k,\, y_{i,j,k}\Big)}{\sum_{(i,j)\in\mathcal{B}} |\mathcal{S}_{i,j}|},$$

where $\mathcal{S}_{i,j}$ denotes the set of supervised answer-token positions for sample (i, j). Thus, the batch loss is the token-level cross-entropy summed over all supervised answer tokens in the mini-batch and normalized by the total number of supervised tokens.

This formulation separates the client–server partition from the compression mechanism, allowing different compressors to be evaluated under a common split-learning architecture. Across heterogeneous VLM backbones, we standardize the partitioning at an early, modality-specific representation and use fixed lightweight model adapters to account for architectural differences among models. The exact model-specific vision & text cut points are provided in Appendix B. This design keeps the communication interface fixed across models and compressors, while raw inputs remain local to each client and the computationally expensive VLM fine-tuning is concentrated on the server.

3.2 Activation-compression strategies

Under this interface, every compressor maps a dense activation ($h \in \mathbb{R}^{L\times D}$), containing L tokens of width D, to a reconstruction $\hat{h}$ of the same shape. We express the quantization step as

$$q = \mathcal{Q}_d\left(\frac{T(h) - \gamma}{s}\right), \qquad \hat{h} = \mathcal{R}(q; s, \gamma),$$

where T is an element-wise transform, s and γ are the scale and offset, and $\mathcal{Q}_d$ maps each scalar to the nearest of d quantization levels. The evaluated methods differ along four main design choices (Table D.1). First, they may apply different transforms T before quantization. Second, the normalization parameters s, γ may be computed globally over the full activation tensor, locally over contiguous blocks of G activation scalars, or fixed in advance. Here, a block refers to a group of G consecutive scalar values that share the same quantization parameters. Third, the quantization levels may form either a uniform grid or the NormalFloat NF codebook consisting of d standard-normal quantiles. Finally, the reconstruction operator $(\mathcal{R})$ may be analytical or learned. For parameter-free methods, $\mathcal{R}(q; s, \gamma) = T^{-1}(sq + \gamma)$, with LQ (logarithmic quantization) additionally transmitting and restoring the sign. In contrast, the FSQ family uses a learned linear reconstruction map (W) (Mentzer et al., 2023; Guo et al., 2025). Because these methods incur different amounts of auxiliary information, we compare them using the *effective communication rate*, defined as the average number of transmitted bits per activation element, including method-specific metadata, normalized by the number of activation scalars (Table D.1). We evaluate methods at two target operating points, 2 and 4 nominal bits per activation scalar, to enable comparison under matched communication budgets. The 4-bit setting reflects a standard low-bit regime widely used in LLM quantization (Frantar et al., 2023; Lin et al., 2024a; Dettmers et al., 2023), whereas 2 bits represents a more aggressive compression regime explored for activation compression during training (Chen et al., 2021; Wang et al., 2022b). Full specifications of all compression methods are provided in Appendix C.

4 Experimental setup

We use two experimental protocols. Sections 5.1 and 5.2 evaluate compressors in a controlled single-client benchmark. We use a single client for experimental convenience because the key factors

[1] We de-identify the task context to prevent the disclosure of patient identity. We also exclude exceptionally rare cases where the answer may uniquely identify the sample. These cases could instead compute the loss locally using server-returned output logits, which is negligible compared with the whole communication time.

[2] For methods without a separate decoder, $Q^{-1}_\phi(Q_\phi(h))$ simply denotes the returned to the clients.

underlying the relative performance differences among compression methods are not affected by the number of clients in our setting, as shown in Appendix F. Section 5.3 then examines a realistic three-client, one-server cross-silo deployment using a larger backbone and training corpus.

Backbones and data. The performance benchmark fine-tunes four pretrained backbones for one epoch: Qwen3-VL-4B-Instruct (Bai et al., 2025), InternVL2.5-8B (Chen et al., 2024b), DeepSeek-VL2-tiny (Wu et al., 2024), and PaliGemma2-10B-mix-224 (Steiner et al., 2024). Its $143{,}430$ medical VQA samples comprise SynthVision (Panahi et al., 2026) ($115{,}811$), PathVQA (He et al., 2020) ($19{,}654$), SLAKE (Liu et al., 2021) ($4{,}999$), RadVQA (Lau et al., 2018) ($1{,}562$), and MedXpertQA (Zuo et al., 2025) ($1{,}404$). The cross-silo run uses Qwen3-VL-32B-Instruct (Bai et al., 2025) and adds $646{,}750$ PubMedVision VQA samples (Chen et al., 2024a) to this corpus, randomly partitioned into three disjoint hospital-client shards. In both protocols, client-side modules are frozen and send activations to the server.

Training. Both protocols train in `bf16`, fully fine-tune the modality connector, and apply LoRA (Hu et al., 2022) to the language model's decoder layers. Following common VLM instruction-tuning practice (Liu et al., 2024; Chen et al., 2024b), we fine-tune all configurations for one epoch. In the performance benchmark, all configurations share one recipe and vary only in compressor and nominal rate. Training uses DeepSpeed ZeRO-2 (Rasley et al., 2020), LoRA with r=16 and α=32 on the last third of the decoder, and a frozen vision tower. We use a learning rate of 2×10^{-5} with cosine decay and 3% warmup, an effective batch size of 16, and four NVIDIA A40 (48 GB) GPUs, the benchmark runs completed in one month. In the cross-silo run, clients send Ql codes at d=16, 4 or `bf16` activations for the uncompressed control. The server uses LoRA with r=64 and α=128 on the last half of the decoder, learning rates of 10^{-4} for LoRA and 2×10^{-5} for the connector, cosine decay with 3% warmup, a global batch size of 96 (32 per client), and two NVIDIA H200 (141 GB) GPUs. The cross-silo processes run with one-step pipelining, in which each client prepares and sends the compressed activations for step $t{+}1$ while the server trains on step t.

Seeds. For the performance benchmark, we run every configuration with three random seeds, $\{1, 7, 42\}$. The seed controls LoRA initialization, the initialization of trainable compressor parameters, data ordering, and dropout. We report the mean and half-range across the three runs. We report half-range rather than standard error to summarize the observed spread across the three runs, highlighting variability without suggesting that uncertainty can be reliably estimated from so few runs. The cross-silo deployment uses only one seed (42). This choice is motivated by the low seed-to-seed variability of Ql 4-bit observed in the performance benchmark.

Benchmarks. Both protocols evaluate on PathVQA (6,719 questions), RadVQA (682 questions, overall and separately for closed- and open-ended questions), SLAKE (1,365 questions), and the MedXpertQA-MM test set (589 questions). No evaluation image is used in training. Because response styles vary across models, we use relaxed matching for PathVQA, RadVQA, and SLAKE, which include open-ended questions. Only the first sentence is evaluated. For yes/no questions, the first yes/no token must match the reference. For open-ended questions with short-phrase reference answers, the first sentence must contain the reference phrase with matching word boundaries, except for MedXpertQA-MM, which needs an exact match after text normalization.

5 Results

5.1 Effect of Compression on Performance

We first evaluate the effect of each compressor on accuracy with one client and one server. We evaluate seven compression methods on four VLM backbones at 4 and 2 nominal bits, together with an uncompressed `bf16` baseline (4-bit in Table 1, 2-bit in Appendix Table D.2). The Rate column gives the effective communication rate of Section C, which differs between methods of the same nominal bit width because of method-specific metadata.

Across the four backbones, compression performance is strongly backbone-dependent, with these differences becoming more evident at lower bit widths. At 4 bits, Ql is the most consistently robust method, remaining within 3 percentage points of the uncompressed baseline across all back-

Table 1: All model backbones at nominal 4-bit compression ($d = 16$). Accuracy (%) over three seeds, shown as mean ± half-range, where the half-range is $(\max - \min)/2$ over the seeds. The best quantized result in each column is shown in **bold**. Light blue highlights Ql, the most consistent method across backbones. "Rate" denotes the *effective communication rate* from Table D.1.

Method	Rate	PathVQA	Rad-all	Rad-closed	Rad-open	SLAKE	MedXpert
DeepSeek-VL2-tiny							
None	16.00	36.40 ± 1.92	35.78 ± 0.59	54.14 ± 0.54	14.00 ± 0.64	57.26 ± 0.33	20.71 ± 1.19
Ql	4.39	33.47 ± 1.12	$\mathbf{37.44 \pm 0.59}$	$\mathbf{55.59 \pm 0.68}$	$\mathbf{15.92 \pm 0.48}$	$\mathbf{58.07 \pm 0.37}$	20.37 ± 0.59
LQ	5.00	34.54 ± 0.86	36.22 ± 0.95	54.50 ± 0.95	14.53 ± 0.96	57.19 ± 0.18	19.75 ± 0.85
AWQ	4.00	28.91 ± 2.87	33.24 ± 0.51	53.06 ± 0.68	9.72 ± 1.60	50.67 ± 0.73	$\mathbf{21.11 \pm 1.02}$
DQ	4.00	$\mathbf{39.34 \pm 0.22}$	31.33 ± 0.95	53.87 ± 1.49	4.59 ± 0.48	39.73 ± 0.62	18.79 ± 1.02
Top-K	5.00	29.49 ± 0.96	25.90 ± 1.61	47.48 ± 3.11	0.32 ± 0.32	21.44 ± 0.18	3.23 ± 0.00
FSQ	4.00	12.72 ± 9.38	14.86 ± 10.48	26.67 ± 19.86	0.85 ± 0.96	7.62 ± 8.83	1.98 ± 1.36
RD-FSQ	4.00	18.31 ± 6.83	17.40 ± 13.05	31.62 ± 24.05	0.53 ± 0.16	13.33 ± 5.13	9.34 ± 0.51
InternVL2.5-8B							
None	16.00	36.31 ± 0.77	46.14 ± 0.95	67.75 ± 1.08	20.51 ± 1.12	68.52 ± 0.37	25.24 ± 0.34
Ql	4.39	$\mathbf{36.84 \pm 0.80}$	$\mathbf{46.14 \pm 0.81}$	$\mathbf{68.02 \pm 0.81}$	$\mathbf{20.19 \pm 0.80}$	$\mathbf{68.67 \pm 0.88}$	$\mathbf{25.92 \pm 0.42}$
LQ	5.00	26.88 ± 0.20	36.46 ± 0.88	53.87 ± 0.41	15.81 ± 2.24	60.17 ± 0.22	23.43 ± 0.34
AWQ	4.00	35.55 ± 1.20	45.11 ± 1.76	67.57 ± 2.30	18.48 ± 1.44	67.55 ± 0.66	24.17 ± 1.10
DQ	4.00	30.90 ± 0.45	40.22 ± 0.44	59.82 ± 0.95	16.99 ± 0.32	60.49 ± 0.37	23.60 ± 0.17
Top-K	5.00	28.24 ± 0.36	42.82 ± 0.51	64.59 ± 1.35	16.99 ± 0.48	60.93 ± 0.62	24.39 ± 0.59
FSQ	4.00	1.72 ± 1.09	8.31 ± 7.26	10.90 ± 12.57	5.24 ± 1.28	16.97 ± 7.33	14.26 ± 0.59
RD-FSQ	4.00	14.15 ± 8.38	17.45 ± 5.43	30.00 ± 11.62	2.56 ± 2.08	21.29 ± 7.66	17.54 ± 0.68
PaliGemma2-10B							
None	16.00	34.91 ± 0.92	38.56 ± 1.03	58.29 ± 2.03	15.17 ± 0.96	58.05 ± 1.14	18.22 ± 0.08
Ql	4.39	32.50 ± 1.47	$\mathbf{38.47 \pm 0.66}$	$\mathbf{59.37 \pm 1.49}$	13.68 ± 0.48	$\mathbf{56.31 \pm 1.65}$	$\mathbf{19.07 \pm 0.68}$
LQ	5.00	$\mathbf{35.39 \pm 0.80}$	35.97 ± 0.37	54.95 ± 0.41	13.46 ± 0.80	46.94 ± 0.88	7.02 ± 0.68
AWQ	4.00	30.63 ± 2.14	34.85 ± 2.64	51.62 ± 4.86	$\mathbf{14.96 \pm 0.16}$	51.75 ± 1.21	17.03 ± 0.59
DQ	4.00	22.38 ± 2.44	29.03 ± 0.44	46.94 ± 1.89	7.80 ± 1.28	35.34 ± 4.80	3.34 ± 1.27
Top-K	5.00	30.89 ± 1.53	30.01 ± 0.37	49.37 ± 1.08	7.05 ± 0.48	36.07 ± 1.17	14.54 ± 0.93
FSQ	4.00	19.63 ± 4.93	21.99 ± 8.50	39.46 ± 17.03	1.28 ± 1.60	16.09 ± 1.87	4.92 ± 1.95
RD-FSQ	4.00	29.80 ± 1.85	26.59 ± 1.69	48.83 ± 3.24	0.21 ± 0.16	27.64 ± 2.49	3.17 ± 0.25
Qwen3-VL-4B							
None	16.00	37.01 ± 0.39	49.46 ± 0.29	72.61 ± 0.68	22.01 ± 0.48	69.47 ± 0.40	22.47 ± 0.08
Ql	4.39	$\mathbf{37.38 \pm 0.77}$	$\mathbf{48.97 \pm 0.59}$	71.35 ± 0.68	$\mathbf{22.44 \pm 0.48}$	$\mathbf{68.77 \pm 0.77}$	22.35 ± 0.34
LQ	5.00	37.12 ± 0.91	48.83 ± 0.22	$\mathbf{71.53 \pm 0.41}$	21.90 ± 0.64	67.89 ± 1.10	$\mathbf{22.58 \pm 0.25}$
AWQ	4.00	33.70 ± 0.60	48.29 ± 0.37	70.99 ± 0.41	21.37 ± 1.12	65.54 ± 1.32	21.45 ± 0.34
DQ	4.00	32.67 ± 0.86	47.41 ± 0.07	69.37 ± 0.41	21.37 ± 0.32	64.25 ± 0.15	22.41 ± 0.76
Top-K	5.00	28.97 ± 0.37	40.08 ± 1.54	59.82 ± 2.97	16.67 ± 0.80	49.04 ± 1.17	21.05 ± 0.59
FSQ	4.00	15.82 ± 5.23	17.55 ± 7.62	30.72 ± 15.14	1.92 ± 1.60	10.50 ± 2.71	8.77 ± 1.19
RD-FSQ	4.00	12.13 ± 12.17	14.42 ± 15.40	22.43 ± 25.41	4.91 ± 3.53	20.63 ± 15.82	13.24 ± 4.07

bones and benchmarks. In contrast, FSQ and RD-FSQ show substantial degradation and instability. For example, FSQ reduces PathVQA accuracy from 36.31 to 1.72 on InternVL2.5-8B, while RD-FSQ reduces Qwen3-VL-4B from 37.01 to 12.13. For DQ, LQ, and AWQ, they exhibit more backbone-specific failures. For instance, DQ nearly preserves MedXpert performance on Qwen3-VL-4B (22.41 vs. 22.47) but drops sharply on PaliGemma2-10B (3.34 vs. 18.22). The 2-bit setting (Table D.2) is considerably more challenging. Ql remains robust overall, but its accuracy declines much more than 4-bit quantization on several benchmarks, including SLAKE. Other methods exhibit more severe failures, particularly on Qwen3-VL and PaliGemma2. Because 4-bit Ql is already within 3% points of the uncompressed baseline, we omit higher bit widths due to their added cost.

5.2 Effect of Compression through the Lens of Feature Drift

Section 5.1 evaluates the performance of different compressors, but does not explain how compression affects the internal representations of the model. To characterize this effect, we measure feature drift as $FD = 1 - \cos(h^m_{i,j}, \hat{h}^m_{i,j})$. We evaluate this drift at the vision and text cut points, at the LLM decoder input, and across the decoder layers of the LM. Each configuration is defined by a backbone, compression method, and bit width, and is evaluated with three random seeds.

Larger drift goes with lower accuracy. Figure 2 plots drift against accuracy retained: the accuracy of a compressed checkpoint as a fraction of its matched uncompressed checkpoint, averaged over the four benchmark datasets. At the two cuts, the FSQ family approaches a cosine distance of 1 at both bit widths, whereas Ql has the smallest drift at every stage and bit width (about 0.01 at the cuts at 4 bits, Appendix Figure D.2). In every panel, configurations with larger drift tend to retain less accuracy. The Spearman correlation over the configurations is on average -0.72 to -0.85 at the cuts and the LLM decoder input across the four backbone models (Appendix Figure D.1). To verify that this correlation is not confounded by compression method, bit width, or backbone, we

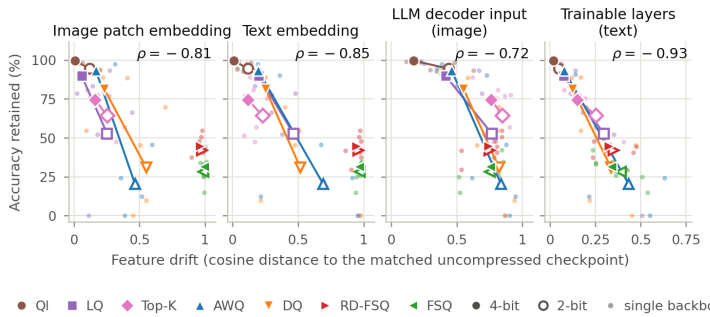

Figure 2: Feature drift versus accuracy retained at the four cuts. Large markers are method means over four backbones. Small dots are all the configurations. ρ is their Spearman correlation, with ranks taken within each backbone.

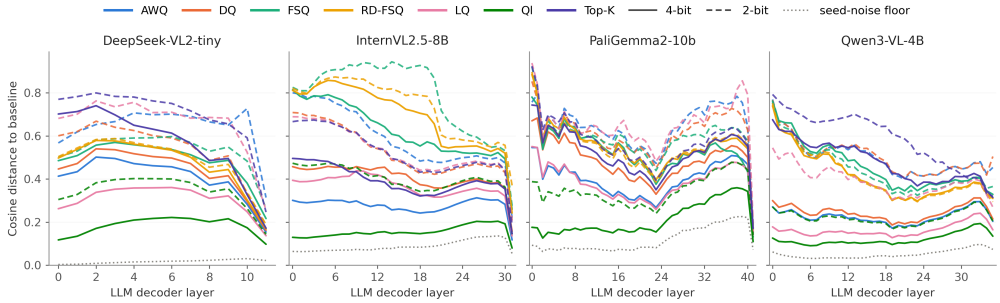

Figure 3: The change in feature drift on each layer. Each line represents a compression method-bit setting, averaged over 3 seeds.

regress these three factors out of both drift and retained accuracy respectively. The resulting partial correlation, averaged across the four benchmarks ($\bar{r}$; Appendix Table D.3), remains -0.49 for text-embedding drift and -0.47 for image drift at the LLM decoder input. The correlations remain stable when any single backbone or compression method is excluded, and variance decomposition analysis further shows that drift accounts for much of the variation otherwise attributed to method and bit width. (Appendix Tables D.4 and D.5).

Drift through the language model. Figure 3 shows that the feature drift generally accumulates through the LLM frozen decoder layers, while differences between compression methods persist across layers. The pattern for each trajectory is backbone-specific, but two features are shared by all four backbones. First, QI at 4 bits keeps the smallest drift at every layer, so the advantage seen at both input cuts carries through the whole model. Second, drift is lower at the final layer than at the first layer in 165 of the 168 checkpoints, so the model attenuates what it receives, which is consistent with all compression methods. Drift averaged over the finetuned last third of the LLM decoder layers correlates more strongly with accuracy ($\rho = -0.93$ at text tokens, Figure 2, right; $\bar{r} = -0.59$) than drift at either compression cut.

5.3 Evaluating the Effect of Quantization in Cross-Silo Split Learning

We study the effect of quantization in a three-hospital split-learning deployment, where each training step trainings one global batch of 96 samples (32 per client) to the server and a full epoch comprises 8,231 communication steps. Communication savings matter most over long runs, so we use Qwen3-VL-32B-Instruct (Bai et al., 2025). The co-located QI 4-bit epoch took 32.3 h, with server computation hiding most client compression and transfer (Appendix G). Compared with bf16, QI 4-bit cuts the data each client sends per step by 72.5% while retaining 98.5% of bf16 accuracy on average across the four benchmarks (Table 2).

In practice, hospital uplinks are limited, especially at small sites (NHS England Digital, n.d.; Federal Communications Commission, 2010). In addition, only part of a link can serve training. Health data

Table 2: 32B cross-silo accuracy (%) and data sent per client per communication step (MB), seed 42. Saved MB and Retained Performance (average over the four benchmarks) are relative to bf16.

	Data sent		Accuracy (%)						
Method	MB	Saved	Retained	PathVQA	Rad-all	Rad-closed	Rad-open	SLAKE	MedXpert
None (`bf16`)	123.7	–	–	42.63	58.21	81.08	31.09	75.09	28.52
Ql 4-bit	34.0	72.5%	98.5%	42.15	57.48	80.27	30.45	73.41	28.18
Ql 2-bit	18.6	85.0%	97.0%	40.11	56.45	80.27	28.21	73.11	28.35

must be protected in transit (U.S. Department of Health and Human Services, 2003), and other clinical traffic shares the link. In a replay of the measured trace (Appendix G), quantization cuts the uplink needed for full-speed training by $3.3\times$ at 4 bits and $6.5\times$ at 2 bits, and bf16 would consume nearly all of a small hospital's link. Each step waits for the slowest hospital, so this also saves time. If two large hospitals and one small hospital each allocate 10% of their link to training, the epoch takes 229.9 h with bf16, 68.0 h with Ql 4-bit, and 38.1 h with Ql 2-bit.

6 Two Recurring Failure Modes in Activation Compression

Building on the previous section, we analyze two recurring failure modes that make certain compression strategies particularly vulnerable in pretrained VLMs.

6.1 Failure mode I: learnable-compression instability

The FSQ family performs poorly across all four backbones, with collapse across all seeds. We interpret this behavior as an optimization failure caused by its trainable decoder. The dequantized activation passes through a randomly initialized linear map, while the largely frozen network, with only the last third of the language model fine-tuned, has only one epoch to adapt to this mismatch. As a result, performance can rely strongly on the decoder initialization.

Three observations support this interpretation. **1) Seed stability:** 4-bit PathVQA half-ranges average 1.11 points for parameter-free quantizers versus 6.23 for the FSQ family. **2) Bit width:** RD-FSQ on Qwen3-VL drops from 21.07 at 2 bits to 12.13 at 4 bits on PathVQA. **3) Format collapse:** Failed runs produce a small set of repetitive outputs rather than gradual degradation.

To characterize why higher code precision alone cannot resolve this instability, Proposition 1 analyzes the interaction between random decoder initialization and finite-step adaptation.

Proposition 1 (Decoder mismatch in finite steps)**.** *Let $\hat{h} = Wu_d$ reconstruct a frozen client activation $h \in \mathbb{R}^D$ from its dequantized code $u_d \in \mathbb{R}^m$ with decoder weight $W \in \mathbb{R}^{D\times m}$. If W has i.i.d. $\mathcal{N}(0, \sigma^2/D)$ entries, the cosine similarity $\cos(Wu_d, h)$ has mean 0 and variance $1/D$ at every bit width. Under gradient descent with step size η on $F(W) = \frac{1}{2}\mathbb{E}|A(Wu_d - h)|^2$ with a fixed linear downstream map A, each eigenmode of the decoder error contracts as $(1 - \eta\lambda_i\mu_j)^t$, where λ_i and μ_j are the eigenvalues of $A^\top A$ and $\mathbb{E}[u_d u_d^\top]$.*

We refer to Appendix E.1 for the proof. Proposition 1 shows that at different bit width,the randomly initialized reconstruction remains nearly orthogonal to the pretrained activation regardless of code precision. Moreover, under finite-step optimization, modes with small $\lambda_i\mu_j$ recover slowly and can remain strongly influenced by their initialization (Figure 2).

6.2 Failure mode II: per-tensor scale collapse

At 2-bit quantization, DQ and AWQ score at or near zero ($\leq 0.25\%$) on PathVQA and RadVQA-open for Qwen3-VL and PaliGemma2 (Table D.2), while LQ also collapses on PaliGemma2. This failure points to a shared vulnerability of these methods at very low precision. Each method determines its quantization range using statistics from the entire activation tensor. As a result, a small number of heavy-tailed values can substantially widen the range, increasing the spacing between quantization levels and leaving only a few effective levels for the majority of activation values.

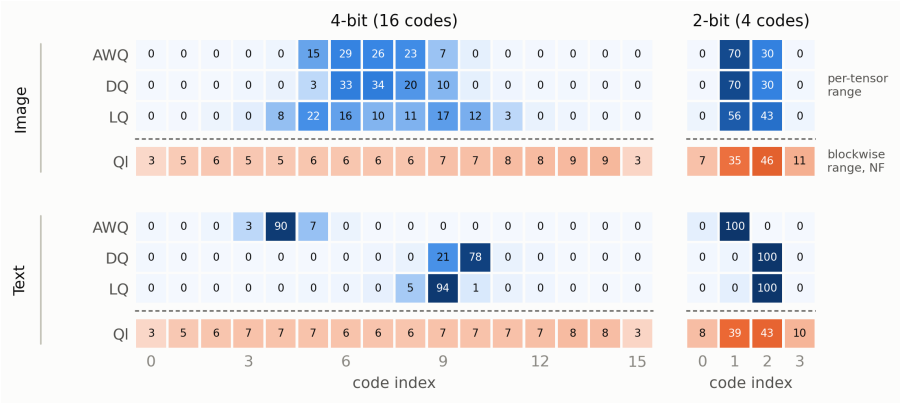

Figure 4: Code-usage frequency on PaliGemma2, measured as the percentage of activation values mapped to each quantization code, for image and text embeddings at 4-bit and 2-bit precision. AWQ, DQ, and LQ (blue) use a single range over the entire tensor, whereas Ql (orange) uses blockwise ranges over groups of 64 values together with NormalFloat quantization levels. Each cell reports the percentage of values assigned to that code; frequencies below 0.5% are rounded to zero.

Code utilization on PaliGemma2, where these methods fail most often, supports this explanation (Figure 4). At 2 bits, the image embeddings of AWQ, DQ, and LQ occupy only two of the four available codes, while their text embeddings collapse almost entirely to a single code. Proposition 2 explains this behavior: because these methods set a single quantization range over the whole tensor, heavy-tailed outliers enlarge the step size and force the bulk of the activation values onto only a few codes. Increasing precision to 4 bits reduces this concentration, but does not eliminate it. For text embeddings, at least 97% of the values still fall on only two codes. The same mechanism applies to LQ and AWQ after their respective transformations, since both retain a single global range.

Proposition 2 (Per-tensor concentration)**.** *Let $x \in \mathbb{R}^n$ be a flattened activation tensor with range $R = \max_i x_i - \min_i x_i > 0$. Quantize x by round-to-nearest, with any deterministic tie-breaking rule, onto $K \geq 2$ uniformly spaced levels that span $[\min_i x_i, \max_i x_i]$, so that the step size is $\Delta_K = R/(K-1)$. If $I \subset \mathbb{R}$ is an interval of width w, then the entries of x that lie in I are mapped to at most $\min\{K, \lfloor w/\Delta_K \rfloor + 2\}$ distinct quantization levels.*

We refer to Appendix E.2 for the proof. Ql avoids this failure mode through blockwise NormalFloat quantization. Its codebook provides finer resolution near zero, while scales are computed independently over blocks of 64 values, so an outlier affects only its local block rather than the entire tensor. Consistent with Proposition 3, this localizes the effect of extreme values and yields substantially broader code utilization across both 2-bit and 4-bit settings.

7 Conclusion and Limitation

We present COMPACT, a unified framework for communication-efficient multimodal SL, and evaluate seven activation-compression methods across four pretrained VLM families. By tracing feature drift through the network, we find that late-layer drift tracks accuracy loss and identify two failure modes, misaligned randomly initialized decoders and outlier-driven code concentration, that blockwise NormalFloat avoids. Across benchmarks and a 32B cross-silo deployment, the results show that compression for collaborative VLM adaptation should be judged jointly with the pretrained representation and the effective communication cost. Regarding limitations, COMPACT targets data locality and communication efficiency rather than formal privacy. It assumes a cooperating server trusted by the participating institutions, so privacy guarantees for the transmitted activations are beyond the scope of this work.

Reproducibility Statement

We provide the information needed to reproduce our experiments in the main paper and appendix. Section 3 describes the client–server interfaces and model-specific split points, Section 3.2 specifies the compression methods and their hyperparameters, and Section C describes the effective communication-rate calculation. The experimental setup provides the training configuration, dataset composition, random seeds, and evaluation protocol. Additional experimental results and implementation details are provided in the appendix. Anonymized source code, model adapters, run configurations, and seed-level results are available at `https://anonymous.4open.science/r/SL-53E6/`.

AI use statement

We used generative AI tools during the development and preparation of this work. These tools were used to provide feedback on experimental methodology, assist with the interpretation and presentation of experimental results, support literature search, and improve the clarity, organization, drafting, and editing of the manuscript. Generative AI tools were also used to assist with code implementation and debugging, as well as with portions of the mathematical derivations. All AI-assisted code, mathematical reasoning, and derived expressions were independently reviewed, verified, and, where necessary, revised by the authors before being incorporated into the experiments or manuscript.

Generative AI tools were not used to generate synthetic data or to independently formulate the research hypotheses or central scientific claims of this work.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022. URL `https://arxiv.org/abs/2108.07258`.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1: 374–388, 2019.

Timothy Castiglia, Anirban Das, Shiqiang Wang, and Stacy Patterson. Compressed-VFL: Communication-efficient learning with vertically partitioned data. In *International Conference on Machine Learning (ICML)*, 2022. arXiv:2206.08330.

Jianfei Chen, Lianmin Zheng, Zhewei Yao, Dequan Wang, Ion Stoica, Michael W. Mahoney, and Joseph E. Gonzalez. ActNN: Reducing training memory footprint via 2-bit activation compressed training. In *International Conference on Machine Learning (ICML)*, 2021. arXiv:2104.14129.

Junying Chen, Chi Gui, Ruyi Ouyang, Anningzhe Gao, Shunian Chen, Guiming Hardy Chen, Xidong Wang, Ruifei Zhang, Zhenyang Cai, Ke Ji, Guangjun Yu, Xiang Wan, and Benyou Wang. HuatuoGPT-Vision, towards injecting medical visual knowledge into multimodal LLMs at scale. *arXiv preprint arXiv:2406.19280*, 2024a.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024b.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Edwin D de Jong, Eric Marcus, and Jonas Teuwen. Current pathology foundation models are unrobust to medical center differences. *arXiv preprint arXiv:2501.18055*, 2025.

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

Jason A. Donenfeld. WireGuard: Next generation kernel network tunnel. In *Network and Distributed System Security Symposium (NDSS)*, 2017. doi: 10.14722/ndss.2017.23160.

Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Songyang Zhang, Haodong Duan, Wenwei Zhang, Yining Li, Hang Yan, Yang Gao, Zhe Chen, Xinyue Zhang, Wei Li, Jingwen Li, Wenhai Wang, Kai Chen, Conghui He, Xingcheng Zhang, Jifeng Dai, Yu Qiao, Dahua Lin, and Jiaqi Wang. InternLM-XComposer2-4KHD: A pioneering large vision-language model handling resolutions from 336 pixels to 4K HD. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Zhen Fang, Miao Yang, Zehang Lin, Zheng Lin, Zihan Fang, Zongyuan Zhang, Tianyang Duan, Dong Huang, and Shunzhi Zhu. NSC-SL: A bandwidth-aware neural subspace compression for communication-efficient split learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2156–2160, 2026. arXiv:2602.02696.

Federal Communications Commission. Health care broadband in america: Early analysis and a path forward. OBI Technical Paper 5, Federal Communications Commission, Omnibus Broadband Initiative, August 2010. Exhibit C.

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 (ICLR)*, 2023.

Timo Fudala, Vasileios Tsouvalas, and Nirvana Meratnia. Fine-tuning multimodal transformers on edge: A parallel split learning approach. *arXiv preprint arXiv:2502.06355*, 2025.

Jiajun Guo, Xin Luo, Jiayin Zheng, Yiqun Wang, Kai-Wei Chang, Wei Wang, and Jie Liu. Quantized-TinyLLaVA: a new multimodal foundation model enables efficient split learning. *arXiv preprint arXiv:2511.23402*, 2025.

Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. PathVQA: 30000+ questions for medical visual question answering. *arXiv preprint arXiv:2003.10286*, 2020.

Cheng-Yen Hsieh, Yu-Chuan Chuang, and An-Yeu Wu. C3-SL: Circular convolution-based batch-wise compression for communication-efficient split learning. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2022. arXiv:2207.12397.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1–2):1–210, June 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL https://doi.org/10.1561/2200000083.

Jason J. Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. A dataset of clinically generated visual questions and answers about radiology images. *Scientific Data*, 5:180251, 2018.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pp. 965–978. IEEE, 2022.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

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 (MLSys)*, 2024a.

Zehang Lin, Miao Yang, Haihan Zhu, Zheng Lin, Jianhao Huang, Jing Yang, Guangjin Pan, Dianxin Luan, Zihan Fang, Shunzhi Zhu, Wei Ni, and John Thompson. SL-FAC: A communication-efficient split learning framework with frequency-aware compression. *arXiv preprint arXiv:2604.07316*, 2026.

Zheng Lin, Xuanjie Hu, Yuxin Zhang, Zhe Chen, Zihan Fang, Xianhao Chen, Ang Li, Praneeth Vepakomma, and Yue Gao. SplitLoRA: A split parameter-efficient fine-tuning framework for large language models. *arXiv preprint arXiv:2407.00952*, 2024b.

Zheng Lin, Guangyu Zhu, Yiqin Deng, Xianhao Chen, Yue Gao, Kaibin Huang, and Yuguang Fang. Efficient parallel split learning over resource-constrained wireless edge networks. *IEEE Transactions on Mobile Computing*, 23(10):9224–9239, 2024c. arXiv:2303.15991.

Bo Liu, Li-Ming Zhan, Li Xu, Lin Ma, Yan Yang, and Xiao-Ming Wu. SLAKE: A semantically-labeled knowledge-enhanced dataset for medical visual question answering. In *IEEE International Symposium on Biomedical Imaging (ISBI)*, 2021.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llavanext: Improved reasoning, ocr, and world knowledge, 2024.

Jian Ma, Xinchen Lyu, Jun Jiang, Qimei Cui, Haipeng Yao, and Xiaofeng Tao. SplitFrozen: Split learning with device-side model frozen for fine-tuning LLM on heterogeneous resource-constrained devices. *IEEE Communications Magazine*, 64(3):172–179, 2026. arXiv:2503.18986.

Brandon McKelvey. Financial institutions' duty of confidentiality to keep customer's personal information secure from the threat of identity theft. *UC Davis L. Rev.*, 34:1077, 2000.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017. arXiv:1602.05629.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple, 2023. URL https://arxiv.org/abs/2309.15505.

Bas Meuwissen, Vasileios Tsouvalas, and Nirvana Meratnia. AutoEncoder-compressed parallel split learning for pre-trained model fine-tuning. *arXiv preprint arXiv:2607.17913*, 2026.

Microsoft. Azure network round-trip latency statistics. https://learn.microsoft.com/en-us/azure/networking/azure-network-latency, 2026. Accessed September 2026.

Yujia Mu and Cong Shen. Communication and storage efficient federated split learning. In *IEEE International Conference on Communications (ICC)*, pp. 2976–2981, 2023. arXiv:2302.05599.

Srijith Nair, Michael Lin, Peizhong Ju, Amirreza Talebi, Elizabeth Serena Bentley, and Jia Liu. FSL-SAGE: Accelerating federated split learning via smashed activation gradient estimation. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2505.23182.

NHS England Digital. HSCN connectivity options. https://digital.nhs.uk/services/health-and-social-care-network/hscn-technical-guidance/hscn-connectivity-options, n.d. Consumer scenarios 6–11. Accessed September 2026.

Yongjeong Oh, Jaeho Lee, Christopher G. Brinton, and Yo-Seb Jeon. Communication-efficient split learning via adaptive feature-wise compression. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6):10844–10858, 2025. arXiv:2307.10805.

Shraman Pal, Mansi Uniyal, Jihong Park, Praneeth Vepakomma, Ramesh Raskar, Mehdi Bennis, Moongu Jeon, and Jinho Choi. Server-side local gradient averaging and learning rate acceleration for scalable split learning. *arXiv preprint arXiv:2112.05929*, 2021.

Maziyar Panahi et al. SynthVision: Building a 110K Synthetic Medical VQA Dataset with Cross-Model Validation. https://huggingface.co/blog/OpenMed/synthvision, 2026. Accessed: 2026-09-25.

Chen Qiu, Xingyu Li, Chaithanya Kumar Mummadi, Madan Ravi Ganesh, Zhenzhen Li, Lu Peng, and Wan-Yi Lin. Federated text-driven prompt generation for vision-language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.

Jiawei Shao and Jun Zhang. BottleNet++: An end-to-end approach for feature compression in device-edge co-inference systems. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020. arXiv:1910.14315.

Yurun Song, Zhuoyi Yang, Ian G. Harris, and Sangeetha Abdu Jyothi. AMAQ: Adaptive mixed-bit activation quantization for collaborative parameter efficient fine-tuning. *arXiv preprint arXiv:2510.05468*, 2025.

Andreas Steiner, André Susano Pinto, Michael Tschannen, et al. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555*, 2024.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving LoRA in privacy-preserving federated learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. SplitFed: When federated learning meets split learning. In *AAAI Conference on Artificial Intelligence*, volume 36, pp. 8485–8493, 2022. arXiv:2004.12088.

U.S. Department of Health and Human Services. Health insurance reform: Security standards; final rule. Federal Register 68(34):8334–8381, codified at 45 CFR Part 164, Subpart C (transmission security: § 164.312(e)), 2003.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Jianyu Wang, Hang Qi, Ankit Singh Rawat, Sashank Reddi, Sagar Waghmare, Felix X. Yu, and Gauri Joshi. FedLite: A scalable approach for federated learning on resource-constrained clients. *arXiv preprint arXiv:2201.11865*, 2022a.

Jue Wang, Binhang Yuan, Luka Rimanic, Yongjun He, Tri Dao, Beidi Chen, Christopher Ré, and Ce Zhang. Fine-tuning language models over slow networks using activation compression with guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b. arXiv:2206.01299.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. DeepSeek-VL2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

Dixi Yao, Tahseen Rabbani, Manzil Zaheer, and Tian Li. Federation over text: Insight sharing for multi-agent reasoning. *arXiv preprint arXiv:2604.16778*, 2026.

Songge Zhang, Guoliang Cheng, Zuguang Li, and Wen Wu. SplitLLM: Hierarchical split learning for large language model over wireless network. In *IEEE Globecom Workshops (GC Wkshps)*, 2024. arXiv:2501.13318.

Zihan Zhang, Leon Wong, and Blesson Varghese. Ampere: Communication-efficient and high-accuracy split federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 2026. Early access. arXiv:2507.07130.

Fei Zheng, Chaochao Chen, Lingjuan Lyu, and Binhui Yao. Reducing communication for split learning by randomized Top-k sparsification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4665–4673, 2023. arXiv:2305.18469.

Yuxin Zuo, Shang Qu, Yifei Li, Zhangren Chen, Xuekai Zhu, Ermo Hua, Kaiyan Zhang, Ning Ding, and Bowen Zhou. MedXpertQA: Benchmarking expert-level medical reasoning and understanding. *arXiv preprint arXiv:2501.18362*, 2025.

A Extended Comparison with Related Work

Table A.1 complements the discussion in Section 2 by comparing prior work on SL for foundation models and activation compression in SL along the setting and compression dimensions relevant to COMPACT.

Table A.1: Comparison with prior work on split learning (SL) for foundation models and activation compression in SL. *Multimodal clients*: each client holds all modalities of its samples (✗: modalities are split across parties; –: unimodal data). *Frozen client*: the client-side model is not trained, so no gradients are returned to clients. *Multi-bb.*: evaluated on more than one pretrained backbone. *Drift analysis*: relates representation drift to downstream accuracy. ✓: yes; ✗: no; –: not applicable.

	Setting						Compression study			
Method	**Pretrained FM**	**VLM**	**Multimodal clients**	**Multi-client**	**Frozen client**	**Multi-bb.**	**Activation compression**	**Compares methods**	**Drift analysis**	**Theory**
SL for foundation models										
SplitLoRA (Lin et al., 2024b)	✓	✗	–	✓	✗	✓	✗	–	–	–
SplitFrozen (Ma et al., 2026)	✓	✗	–	✓	✓	✓	✗	–	–	–
MPSL (Fudala et al., 2025)	✓	✓	✓	✓	✗	✓	✗	–	–	–
Activation compression in SL										
Randomized Top-K (Zheng et al., 2023)	✗	✗	–	✗	✗	✗	✓	✓	✗	✓
C-VFL (Castiglia et al., 2022)	✗	✗	–	✓	✗	✗	✓	✓	✗	✓
SplitFC (Oh et al., 2025)	✗	✗	–	✓	✗	✓	✓	✓	✗	✓
Quantized-TinyLLaVA (Guo et al., 2025)	✓	✓	✗	✗	✗	✗	✓	✓	✗	✓
COMPACT (This work)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

B Model-swappable client–server interfaces

To provide a consistent interface across heterogeneous VLM architectures, we define one split point for each modality. The **vision cut** is placed immediately after the vision tower's patch embedding, while the **text cut** is placed immediately after the language model's input embedding lookup. These locations expose early dense modality-specific representations while keeping client-side computation minimal.

Because VLMs organize these components differently, we implement a lightweight model adapter that specifies the locations and hidden dimensions of the vision and text interfaces together with model-specific input processing. At each split point, a proxy module preserves the backbone's native forward interface while routing the intermediate representation through the selected compressor. The remaining backbone computation proceeds through its native architecture. Adding a new VLM therefore requires only a corresponding model adapter, without redesigning the client–server or compression interfaces.

C Compression methods in detail

Scalar quantization in brief. A scalar quantizer replaces every entry x of a tensor by one of d representable values and transmits only the index of that value, $\log_2 d$ bits instead of the 16 bits of a `bf16` number. The uniform affine quantizer, used by DQ, LQ, AWQ and the FSQ family, places the d values on an evenly spaced grid over a range $[z,\, z + (d-1)s]$ with step s:

$$q = \mathrm{clamp}\big(\mathrm{round}((x - z)/s),\, 0,\, d-1\big), \qquad \hat{x} = s\, q + z.$$

The range must be known to both sides. It is either recomputed from the data at every forward pass and sent as side information, as a scale and offset per tensor or per block of G entries, or fixed in advance, as in the FSQ family, which first maps its input into $[-1, 1]$. The step s bounds the rounding error of every entry by $s/2$, so one outlier that widens the range coarsens the quantization of all entries sharing that range; Section 6.2 analyses this mechanism. Ql instead uses the NormalFloat (NF) codebook of QLoRA (Dettmers et al., 2023): its $d = 2^b$ values are the quantiles of a standard normal distribution rescaled to $[-1, 1]$, so that the entries of a normalized, roughly Gaussian block fall on each code about equally often. Throughout, d is the number of levels and $b = \log_2 d$ the nominal number of bits per scalar; rates including side information are given in Table D.1. The seven compressors are specified below and, as client and server steps, in Algorithms 1–7.

Ql — blockwise NormalFloat with double quantization (Algorithm 1). QLoRA's NF4 scheme (Dettmers et al., 2023) generalized to $b = \log_2 d$ bits and applied to activations. Each row of the flattened activation is split into blocks of $G = 64$; each block is min–max normalized to $[-1, 1]$ and mapped to the nearest entry of an NFb codebook whose levels are the 2^b equiprobable quantiles of a standard normal. The per-block scale is itself quantized to 8 bits (double quantization); the per-block minimum is transmitted at the activation dtype.

DQ — dynamic per-tensor quantization (Algorithm 2). As a basic quantization method, DQ dynamically adapts the quantization range to the activation statistics of each forward pass, using a single scale and zero point for the entire tensor. Let $b = \log_2 d$, with $q_{\min} = 0$ and $q_{\max} = d - 1$. For each forward pass, the zero point $z = \min x$ and scale $s = \max((\max x - \min x)/(q_{\max} - q_{\min}), 10^{-8})$ are recomputed over the entire token matrix. The quantized and dequantized values are $\tilde{x} = \mathrm{clamp}(\mathrm{round}((x - z)/s), q_{\min}, q_{\max})$ and $\hat{x} = s\tilde{x} + z$, respectively.

LQ — logarithmic quantization (Algorithm 3). On the base of DQ, LQ's value mapped through $\log_2(|x| + 1)$, then uniformly quantized to d levels over the observed log range, and inverted; the sign is transmitted separately.

AWQ — AWQ-style activation-aware scaling (Algorithm 4). Inspired by AWQ (Lin et al., 2024a), we use calibration activations to construct a per-channel scaling vector, following the key idea of allocating different quantization ranges according to channel-wise activation magnitudes. Specifically, we estimate the mean absolute activation $\bar{a}$ for each channel over the first 16 forward passes and search over $s = \bar{a}^{\alpha}$ with $\alpha \in \{0, \frac{1}{20}, \ldots, 1\}$. The selected scale is fixed after calibration.

Top-K — randomized Top-K sparsification (Algorithm 5). Retain the $k = rD$ largest-magnitude entries per token with randomized relaxation, zero the rest, and rescale survivors by D/k Zheng et al. (2023). We set $r = \log_2(d)/16$ so the *nominal* payload $16r$ matches the quantizers' $\log_2 d$; §C shows the position mask breaks this match. During training a fraction of the retained slots is resampled from the non-top-K entries to reduce selection bias.

FSQ (Algorithm 6). Finite Scalar Quantization (FSQ) (Mentzer et al., 2023) maps continuous representations to a finite set of scalar levels without a learned codebook. We adapt FSQ to intermediate activation compression using scalar quantization followed by a trainable output projection that maps the reconstructed representation back to the backbone hidden space.

RD-FSQ (Algorithm 7). RD-FSQ Guo et al. (2025) replaces the bounded $\tanh$ normalization in FSQ with linear scaling before scalar quantization. In its original formulation, RD-FSQ additionally uses a communication-distortion regularizer L_{comm}. At the split points considered here, however, the client-side modules are frozen, making this term independent of the trainable parameters. We therefore omit L_{comm} in our implementation and evaluate the resulting linear-scaled FSQ variant under the same protocol as the other compressors.

Communication Cost Accounting Nominal bit width does not always reflect the actual communication cost, as some compression methods require additional side information for reconstruction .We therefore report the *effective communication rate*, defined as the total number of transmitted bits, including method-specific metadata, normalized by the number of activation scalars (Table D.1). For example, Ql introduces additional block and row-level quantization metadata, LQ transmits one sign bit per scalar, and Top-K additionally communicates the positions of retained entries. At the same nominal width, these overheads lead to different effective communication rates, which we use to compare communication performance trade-offs.

Algorithms. Each listing gives the client-side encoding, what is transmitted, and the server-side reconstruction of one activation tensor h with N tokens of width D; x denotes one entry of h and d the number of levels.

Algorithm 1 Ql: blockwise NormalFloat quantization with double quantization (block size $G = 64$, $b = \log_2 d$ bits)

Require: $h \in \mathbb{R}^{N \times D}$; NF codebook $\mathcal{N}_b$ of 2^b standard-normal quantiles rescaled to $[-1, 1]$

Client: split every row of h into blocks B_j of G entries
for each block B_j **do**
  $m_j \leftarrow \min B_j$; $r_j \leftarrow \max B_j - m_j$
  $u \leftarrow 2(x - m_j)/r_j - 1$ for every $x \in B_j$ (normalize the block to $[-1, 1]$)
  $q_x \leftarrow \arg\min_k |u - \mathcal{N}_b[k]|$ (b bits per entry)
end for
for every row: $r_{\min}, r_{\max} \leftarrow \min_j r_j,\ \max_j r_j$; $\tilde{r}_j \leftarrow \mathrm{round}\big(255\,(r_j - r_{\min})/(r_{\max} - r_{\min})\big)$ (8-bit block scales)
transmit q, $\tilde{r}$ (8 bits per block), m (16 bits per block) and $(r_{\min}, r_{\max})$ (16 bits each per row)
Server: $r_j \leftarrow r_{\min} + \tilde{r}_j\,(r_{\max} - r_{\min})/255$
$\hat{x} \leftarrow \frac{1}{2}\big(\mathcal{N}_b[q_x] + 1\big)\,r_j + m_j$ for every entry x of block B_j

Algorithm 2 DQ: dynamic per-tensor quantization

Require: h; levels d

Client: $z \leftarrow \min h$; $s \leftarrow \max\big((\max h - \min h)/(d-1),\ 10^{-8}\big)$ (recomputed at every forward pass)
$q \leftarrow \mathrm{clamp}\big(\mathrm{round}((h - z)/s),\ 0,\ d - 1\big)$
transmit q ($\log_2 d$ bits per entry) and (s, z)
Server: $\hat{h} \leftarrow s\,q + z$

Algorithm 3 LQ: logarithmic quantization

Require: h; levels d

Client: $y \leftarrow \log_2(1 + |h|)$; $\sigma \leftarrow \mathrm{sign}(h)$
$y_{\min}, y_{\max} \leftarrow \min y,\ \max y$; $s \leftarrow (y_{\max} - y_{\min})/(d-1)$ (range over the whole tensor, in the log domain)
$q \leftarrow \mathrm{round}\big((y - y_{\min})/s\big)$
transmit q ($\log_2 d$ bits per entry), σ (1 bit per entry) and $(s, y_{\min})$
Server: $\hat{y} \leftarrow y_{\min} + s\,q$; $\hat{h} \leftarrow \sigma\,(2^{\hat{y}} - 1)$

Algorithm 4 AWQ-style activation-aware scaling

Require: $h \in \mathbb{R}^{N \times D}$; levels d; calibration passes $C = 16$; grid $\alpha \in \{0, \frac{1}{20}, \ldots, 1\}$

Calibration (client, first C forward passes): $\bar{a}_c \leftarrow$ mean of $|h_{\cdot c}|$ over the calibration activations for every channel c, normalized by its mean over channels
for each α in the grid **do**
  $s_c \leftarrow \bar{a}_c^{\,\alpha}$; $e(\alpha) \leftarrow \big\|\mathrm{DQ}(h/s)\,s - h\big\|^2$ (per-tensor quantization of the rescaled tensor, Algorithm 2)
end for
fix s at the α with the smallest $e(\alpha)$ and share s with the server once
Client (after calibration): $x \leftarrow h/s$ channel-wise; $q, (s', z') \leftarrow$ per-tensor quantization of x as in Algorithm 2
transmit q ($\log_2 d$ bits per entry) and (s', z')
Server: $\hat{h} \leftarrow (s'\,q + z')\,s$ channel-wise

Algorithm 5 Top-K: randomized top-k sparsification ($r = \log_2(d)/16$)

Require: $h \in \mathbb{R}^{N\times D}$; keep ratio r; resampling fraction p (training only)

1: **Client:** $k \leftarrow \max(1, \lfloor rD \rfloor)$
2: **for** each token h_i **do**
3: $S_i \leftarrow$ indices of the k entries of h_i with the largest $|h_{ij}|$
4: during training, replace a fraction p of S_i by indices drawn at random from the other $D-k$ entries
5: $v_{ij} \leftarrow (D/k)\, h_{ij}$ for $j \in S_i$
6: **end for**
7: transmit the values v (16 bits each, $16r$ bits per entry) and the positions S (a D-bit mask per token or $\log_2 D$ bits per kept entry, whichever is smaller)
8: **Server:** $\hat{h}_{ij} \leftarrow v_{ij}$ for $j \in S_i$, and 0 otherwise

Algorithm 6 FSQ with a trainable decoder

Require: h; levels $\mathcal{G}_d = \{-1, -1 + \frac{2}{d-1}, \ldots, 1\}$; trainable linear decoder $W \in \mathbb{R}^{D\times D}$ on the server

1: **Client:** $u \leftarrow \tanh(h)$ entry-wise (maps every entry into $(-1, 1)$)
2: $q \leftarrow \arg\min_k |u - \mathcal{G}_d[k]|$ for every entry ($\log_2 d$ bits per entry)
3: transmit q
4: **Server:** $u_d \leftarrow \mathcal{G}_d[q]$; $\hat{h} \leftarrow W u_d$ (W is trained with the server-side loss)

Algorithm 7 RD-FSQ: linear scaling with a trainable decoder

Require: h; levels $\mathcal{G}_d$ as in Algorithm 6; trainable linear decoder W on the server

1: **Client:** $u \leftarrow$ linear rescaling of h into $[-1, 1]$ (range computed over the tensor; the distortion regularizer of the original method is omitted, see above)
2: $q \leftarrow \arg\min_k |u - \mathcal{G}_d[k]|$ for every entry; transmit q
3: **Server:** $u_d \leftarrow \mathcal{G}_d[q]$; $\hat{h} \leftarrow W u_d$

D TABLES & FIGURES

Table D.1: Effective communication cost in bits per activation scalar, including side information required for reconstruction. Here, D is the activation dimension, G the quantization block size, d the number of quantization levels, $b = \log_2 d$, and r the retained fraction for Top-K. For Ql, each block carries an 8-bit quantized scale and a 16-bit minimum, while two additional 16-bit constants per row are used to dequantize the block scales. LQ transmits one sign bit per scalar. TopK_r selects the $k = rD$ largest-magnitude entries per token; Top-K sends their values in 16 bits and their positions as indices or a binary mask, whichever is cheaper. Values use $D = 2560$ and $G = 64$.

Method	T / selection	Scale (s, z)	Levels	Decoder	Effective rate	d=4	d=16
None	–	–	–	–	16	16.00	16.00
Ql	–	block (G=64)	NF_b	–	$\log_2 d + 24/G + 32/D$	2.39	4.39
DQ	–	tensor	uniform	–	$\log_2 d$	2.00	4.00
LQ	$\log_2(1+\lvert x\rvert)$	tensor	uniform	–	$\log_2 d + 1$	3.00	5.00
AWQ	x/s_c	tensor	uniform	–	$\log_2 d$	2.00	4.00
FSQ	$\tanh$	fixed	uniform	trainable W	$\log_2 d$	2.00	4.00
RD-FSQ	–	tensor	uniform	trainable W	$\log_2 d$	2.00	4.00
Top-K	TopK_r	–	–	–	$16r + \min(r \log_2 D, 1)$	3.00	5.00

Table D.2: All model backbones at nominal 2-bit compression ($d = 4$). Accuracy (%) over three seeds, shown as mean ± half-range, where the half-range is $(\max - \min)/2$ over the seeds (Section 4). The best quantized result in each column is shown in **bold**. Light blue highlights Ql, the most consistent method across backbones. "Rate" denotes effective bits per activation scalar from Table D.1.

Method	Rate	PathVQA	Rad-all	Rad-closed	Rad-open	SLAKE	MedXpert
DeepSeek-VL2-tiny							
None	16.00	36.40 ± 1.92	35.78 ± 0.59	54.14 ± 0.54	14.00 ± 0.64	57.26 ± 0.33	20.71 ± 1.19
Ql	2.39	$\mathbf{34.04 \pm 0.52}$	$\mathbf{34.36 \pm 1.25}$	50.99 ± 2.30	$\mathbf{14.64 \pm 0.80}$	$\mathbf{49.82 \pm 0.59}$	$\mathbf{20.09 \pm 1.10}$
LQ	3.00	25.39 ± 2.20	29.57 ± 0.51	$\mathbf{54.50 \pm 0.95}$	0.00 ± 0.00	21.20 ± 2.09	4.13 ± 0.51
AWQ	2.00	12.93 ± 8.69	15.88 ± 9.24	29.28 ± 17.03	0.00 ± 0.00	9.82 ± 7.88	0.00 ± 0.00
DQ	2.00	24.36 ± 0.71	25.90 ± 1.10	47.57 ± 2.16	0.21 ± 0.16	17.31 ± 1.25	2.26 ± 0.76
Top-K	3.00	27.11 ± 0.02	27.17 ± 0.44	50.00 ± 0.68	0.11 ± 0.16	21.39 ± 0.11	0.40 ± 0.17
FSQ	2.00	7.89 ± 11.35	8.60 ± 12.90	15.77 ± 23.65	0.11 ± 0.16	5.15 ± 6.52	0.96 ± 0.51
RD-FSQ	2.00	15.09 ± 11.36	15.69 ± 13.34	28.38 ± 25.00	0.64 ± 0.48	13.28 ± 8.57	1.98 ± 0.42
InternVL2.5-8B							
None	16.00	36.31 ± 0.77	46.14 ± 0.95	67.75 ± 1.08	20.51 ± 1.12	68.52 ± 0.37	25.24 ± 0.34
Ql	2.39	$\mathbf{33.58 \pm 0.14}$	$\mathbf{42.38 \pm 0.37}$	$\mathbf{63.60 \pm 0.95}$	$\mathbf{17.20 \pm 0.48}$	$\mathbf{62.69 \pm 0.29}$	$\mathbf{24.67 \pm 0.34}$
LQ	3.00	25.96 ± 0.49	35.43 ± 0.29	56.31 ± 0.14	10.68 ± 0.48	41.20 ± 1.03	14.43 ± 1.19
AWQ	2.00	21.42 ± 4.70	22.92 ± 10.63	37.84 ± 18.92	5.24 ± 1.44	26.64 ± 3.08	8.83 ± 4.24
DQ	2.00	25.62 ± 0.68	34.90 ± 0.73	55.68 ± 1.62	10.26 ± 0.32	43.47 ± 0.84	17.43 ± 0.34
Top-K	3.00	25.33 ± 0.28	34.02 ± 0.44	52.61 ± 0.27	11.97 ± 0.64	48.28 ± 0.59	22.13 ± 0.68
FSQ	2.00	9.90 ± 8.21	14.37 ± 11.95	24.86 ± 21.08	1.92 ± 1.76	12.19 ± 9.16	10.58 ± 0.34
RD-FSQ	2.00	10.80 ± 4.97	15.10 ± 6.96	25.14 ± 12.70	3.21 ± 0.80	25.45 ± 5.82	15.96 ± 0.59
PaliGemma2-10B							
None	16.00	34.91 ± 0.92	38.56 ± 1.03	58.29 ± 2.03	15.17 ± 0.96	58.05 ± 1.14	18.22 ± 0.08
Ql	2.39	$\mathbf{34.87 \pm 0.42}$	$\mathbf{36.85 \pm 0.81}$	$\mathbf{57.21 \pm 1.22}$	$\mathbf{12.71 \pm 0.64}$	$\mathbf{52.99 \pm 0.40}$	$\mathbf{19.35 \pm 0.59}$
LQ	3.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
AWQ	2.00	0.25 ± 0.36	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
DQ	2.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.05 ± 0.07	0.00 ± 0.00
Top-K	3.00	29.78 ± 1.28	26.39 ± 1.47	48.65 ± 2.70	0.00 ± 0.00	19.22 ± 2.01	7.36 ± 0.25
FSQ	2.00	21.41 ± 1.80	13.54 ± 7.55	24.50 ± 14.19	0.53 ± 0.32	19.93 ± 2.31	1.58 ± 0.68
RD-FSQ	2.00	26.74 ± 3.16	27.66 ± 0.51	50.90 ± 1.08	0.11 ± 0.16	22.71 ± 5.53	2.83 ± 0.34
Qwen3-VL-4B							
None	16.00	37.01 ± 0.39	49.46 ± 0.29	72.61 ± 0.68	22.01 ± 0.48	69.47 ± 0.40	22.47 ± 0.08
Ql	2.39	$\mathbf{32.71 \pm 0.39}$	47.61 ± 0.88	68.20 ± 1.35	$\mathbf{23.18 \pm 0.80}$	63.35 ± 0.22	$\mathbf{21.51 \pm 0.25}$
LQ	3.00	32.60 ± 0.27	$\mathbf{48.39 \pm 0.81}$	$\mathbf{70.36 \pm 0.54}$	22.33 ± 1.12	$\mathbf{66.57 \pm 0.48}$	20.54 ± 0.17
AWQ	2.00	0.00 ± 0.00	17.25 ± 7.18	31.80 ± 13.24	0.00 ± 0.00	9.69 ± 6.08	0.23 ± 0.34
DQ	2.00	0.00 ± 0.00	13.00 ± 10.19	23.96 ± 18.78	0.00 ± 0.00	7.16 ± 6.81	0.57 ± 0.34
Top-K	3.00	30.41 ± 0.83	37.29 ± 0.81	57.84 ± 1.62	12.93 ± 1.44	44.27 ± 1.03	20.20 ± 0.42
FSQ	2.00	13.50 ± 8.68	18.38 ± 4.03	30.45 ± 7.70	4.06 ± 0.80	20.32 ± 12.09	7.53 ± 4.41
RD-FSQ	2.00	21.07 ± 2.46	20.82 ± 2.57	33.96 ± 4.59	5.24 ± 2.08	23.25 ± 8.13	13.19 ± 1.61

Partial-correlation calculation. For each feature stage s, let $\mathbf{d}_s$ contain the feature drift of the checkpoints in a table row, and let $\mathbf{a}_b$ contain their accuracy on benchmark b. Let X contain an intercept and dummy variables for compression method and bit width; for the pooled analysis, it also contains backbone dummies. We regress both quantities on the same controls and correlate their residuals:

$$\begin{aligned} \tilde{\mathbf{d}}_s &= \mathbf{d}_s - X(X^\top X)^{-1} X^\top \mathbf{d}_s, \\ \tilde{\mathbf{a}}_b &= \mathbf{a}_b - X(X^\top X)^{-1} X^\top \mathbf{a}_b, \\ r_{s,b} &= \frac{\tilde{\mathbf{d}}_s^\top \tilde{\mathbf{a}}_b}{\|\tilde{\mathbf{d}}_s\|_2 \, \|\tilde{\mathbf{a}}_b\|_2}, \\ \bar{r}_s &= \frac{1}{4} \sum_{b=1}^{4} r_{s,b}. \end{aligned} \tag{1}$$

The table also reports $[\min_b r_{s,b},\ \max_b r_{s,b}]$.

Reading Table D.3 as a regression. Each partial correlation in Table D.3 comes from the linear model accuracy ∼ drift + method + bit width + backbone, fitted per benchmark. Table D.5 reports what the same fits attribute to each factor. Most of the method effect is carried by drift: without a drift term the parameter-free methods trail Ql by 9 to 16 accuracy points and the FSQ family by 22 to 28, whereas with text-embedding drift held fixed no parameter-free method trails by more than 8 points and the 2-bit penalty shrinks from 10 to 5 points (the FSQ-family coefficients are unstable here because their drift is saturated near 1). The backbone effect does not shrink: its partial η^2 of 0.20 to 0.33 matches or exceeds that of drift. Part of it is the backbones' different baseline accuracy, but it stays at 0.12 to 0.21 when accuracy retained is the outcome, so backbones also differ in how

Table D.3: Partial correlation between feature drift (per-token cosine distance to the matched uncompressed checkpoint) and benchmark accuracy, computed using Eq. (1). Partial Pearson correlations are computed separately for PathVQA, RadVQA, SLAKE and MedXpertQA-MM; $\bar{r}$ is their mean and $[\min, \max]$ their range across the four benchmarks. The "All models" row pools all 168 checkpoints and controls for bit width, compression method and backbone identity; the per-model rows control for bit width and method (all dummy-coded). "Late layers (text)" is the drift at text-token positions averaged over the trainable LLM layers. Stars are conservative ($^{***}p < 10^{-3}$, $^{**}p < 10^{-2}$, $^{*}p < 0.05$) and require all four benchmarks to be significant (largest per-benchmark p).

Feature	Model (n)	Pearson $\bar{r}$	$[\min, \max]$
Image embedding	All models (168)	-0.25	$[-0.40, -0.11]$
	DeepSeek-VL2-tiny (42)	-0.48	$[-0.73, -0.19]$
	InternVL2.5-8B (42)	-0.46	$[-0.62, -0.27]$
	PaliGemma2-10b (42)	-0.61^{***}	$[-0.67, -0.53]$
	Qwen3-VL-4B (42)	-0.78^{***}	$[-0.82, -0.72]$
Text embedding	All models (168)	-0.49^{***}	$[-0.55, -0.35]$
	DeepSeek-VL2-tiny (42)	-0.58^{*}	$[-0.81, -0.35]$
	InternVL2.5-8B (42)	-0.74^{***}	$[-0.85, -0.63]$
	PaliGemma2-10b (42)	-0.85^{***}	$[-0.94, -0.67]$
	Qwen3-VL-4B (42)	-0.62^{***}	$[-0.65, -0.58]$
LLM-input image	All models (168)	-0.47^{***}	$[-0.59, -0.37]$
	DeepSeek-VL2-tiny (42)	-0.48	$[-0.77, -0.22]$
	InternVL2.5-8B (42)	-0.51^{**}	$[-0.63, -0.43]$
	PaliGemma2-10b (42)	-0.27	$[-0.33, -0.20]$
	Qwen3-VL-4B (42)	-0.74^{***}	$[-0.78, -0.66]$
Late layers (text)	All models (168)	-0.59^{***}	$[-0.68, -0.56]$
	DeepSeek-VL2-tiny (42)	-0.61	$[-0.91, -0.30]$
	InternVL2.5-8B (42)	-0.51	$[-0.71, -0.24]$
	PaliGemma2-10b (42)	-0.84^{***}	$[-0.88, -0.76]$
	Qwen3-VL-4B (42)	-0.79^{***}	$[-0.84, -0.73]$

Table D.4: Robustness of the correlation between feature drift and accuracy. Entries are partial correlations $\bar{r}$ computed as in Table D.3 (controlling for compression method, bit width and backbone; mean over the four benchmarks). Because the three seeds of a configuration have nearly identical drift (within-configuration variance below 0.4% of the total), we also report the analysis on the 56 seed-averaged configurations and a bootstrap that resamples whole configurations; leave-one-out rows give the range. "Late layers (text)" is the drift at text-token positions averaged over the trainable LLM layers. Stars as in Table D.3.

Analysis	n	Image emb.	Text emb.	LLM-input image	Late layers (text)
Main analysis	168	-0.25	-0.49^{***}	-0.47^{***}	-0.59^{***}
Rank-based	168	-0.17	-0.38^{***}	-0.57^{***}	-0.68^{***}
Seeds averaged	56	-0.27	-0.53^{*}	-0.52^{**}	-0.67^{***}
FSQ family excluded	120	-0.23	-0.41^{**}	-0.43^{***}	-0.60^{***}
4-bit only	84	$+0.10$	-0.39^{**}	-0.46^{**}	-0.39^{*}
2-bit only	84	$+0.18$	-0.21	-0.33	-0.39
Leave one backbone out	126	$[-0.44, -0.13]$	$[-0.69, -0.22]$	$[-0.55, -0.36]$	$[-0.69, -0.43]$
Leave one method out	144	$[-0.34, -0.13]$	$[-0.54, -0.42]$	$[-0.53, -0.42]$	$[-0.61, -0.57]$
95% CI (bootstrap)	168	$[-0.46, -0.02]$	$[-0.72, -0.19]$	$[-0.61, -0.27]$	$[-0.70, -0.42]$

much accuracy they lose at a given drift. Drift itself explains most once it is measured late in the language model.

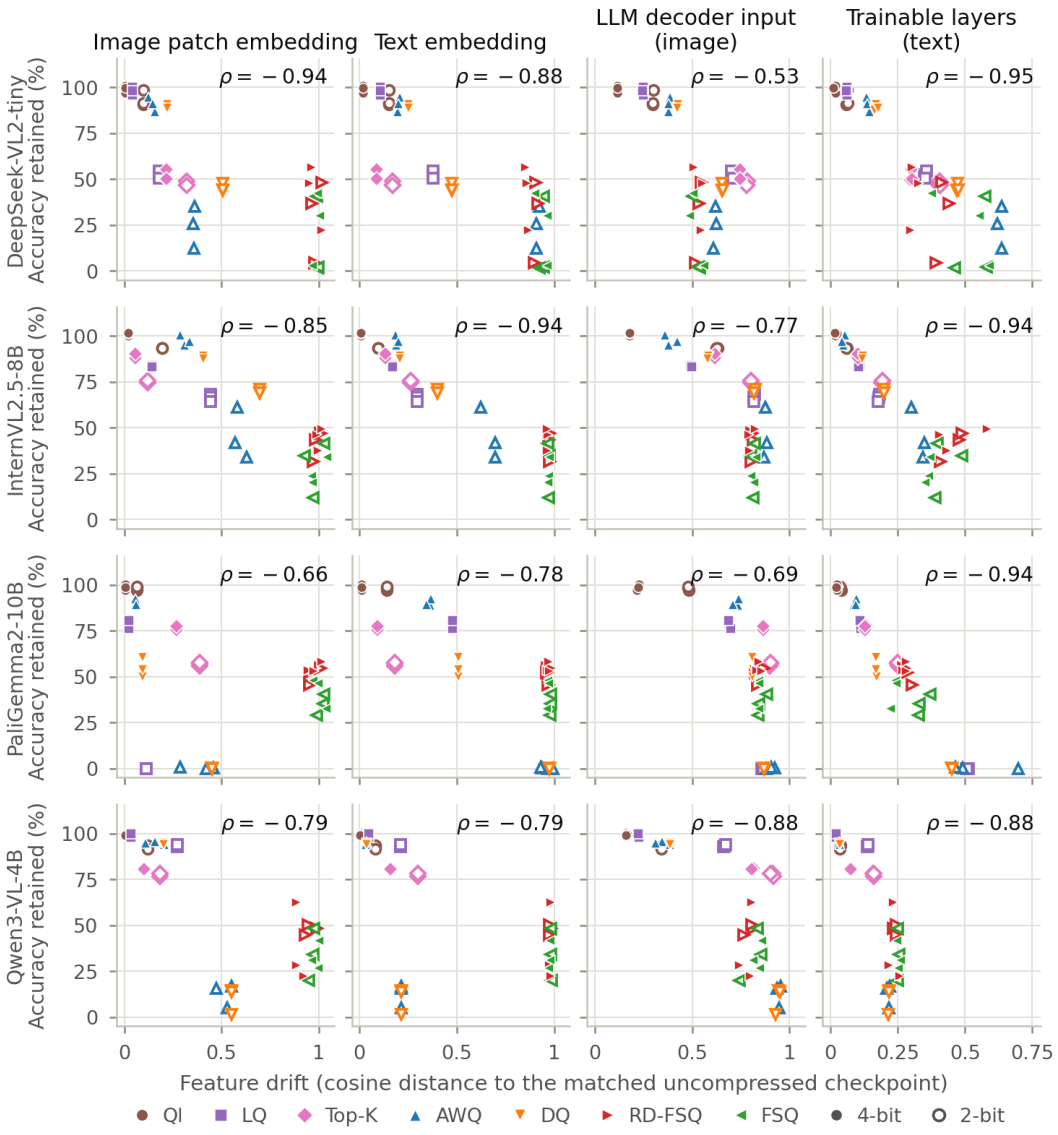

Figure D.1: Per-backbone version of Figure 2: every compressed checkpoint is one marker (three seeds per method and rate; filled: 4-bit, open: 2-bit). Axes and columns as in Figure 2. ρ is the Spearman correlation over the 14 seed-averaged method–rate configurations of each backbone. Drift at the LLM-input text positions is not shown because text embeddings enter the LLM unchanged.

Table D.5: What the regressions behind Table D.3 attribute to each factor (outcome: accuracy; mean over the four benchmarks). Top: partial η^2, the share of the variance left unexplained by the other three factors that a factor explains, with each drift measure in turn. Bottom: method, bit-width and backbone coefficients in accuracy points (reference: QI, 4 bits, Qwen3-VL-4B), without a drift term and with text-embedding or late-layer drift held fixed.

Partial η^2	Image emb.	Text emb.	LLM-input image	Late layers (text)
Drift	0.08	0.24	0.23	0.36
Method	0.17	0.18	0.31	0.11
Bit width	0.09	0.08	0.02	0.03
Backbone	0.28	0.20	0.33	0.25
Coefficient (points)	**No drift**	**Text emb. fixed**	**Late layers fixed**	
AWQ	−15.8	−4.3	−3.2	
DQ	−15.9	−6.2	−4.8	
LQ	−9.4	−1.2	−0.7	
Top-K	−11.1	−7.7	−1.5	
FSQ	−27.6	−0.2	−8.7	
RD-FSQ	−22.3	+4.6	−5.0	
2 bits vs. 4 bits	−10.2	−4.8	−2.1	
DeepSeek-VL2-tiny	−6.9	−4.3	+3.1	
InternVL2.5-8B	+0.9	+3.3	+5.6	
PaliGemma2-10B	−8.4	−1.2	−2.2	

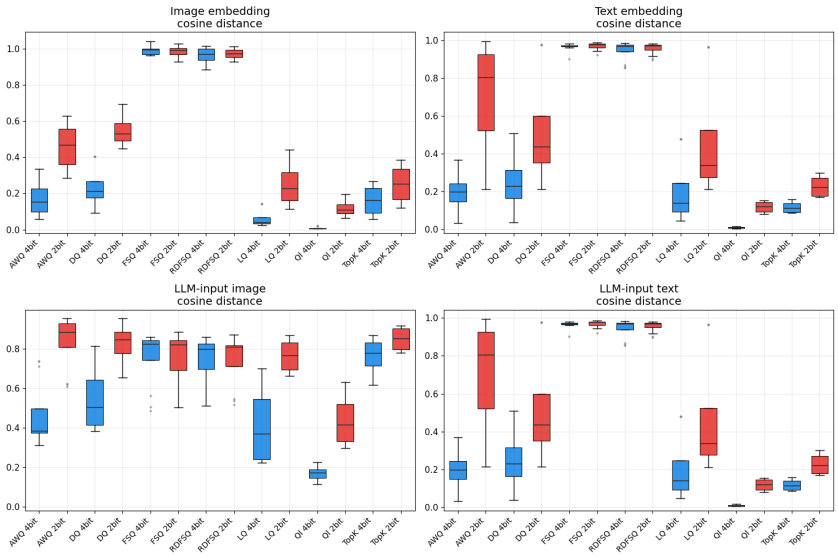

Figure D.2: Feature drift of every compressed checkpoint by method and bit width (blue: 4-bit, red: 2-bit), measured as the per-token cosine distance to the matched uncompressed checkpoint (same backbone and seed) at the image and text embeddings (the two cut points) and at the image- and text-token positions of the LLM input. Each box contains 12 checkpoints (four backbones × three seeds). The LLM-input text panel repeats the text-embedding panel because text embeddings enter the LLM unchanged.

E THEORETICAL ANALYSIS

E.1 DECODER MISMATCH IN FINITE STEPS

Proof of Proposition 1

Proof. Step 1: initialization mismatch. Conditioned on a realization of (u_d, h) with $u_d \neq 0$ and $h \neq 0$. Since W_0 is independent of (u_d, h) and has i.i.d. entries $\mathcal{N}(0, \sigma^2/D)$,

$$W_0 u_d \mid (u_d, h) \sim \mathcal{N}\left(0, \frac{\sigma^2 \|u_d\|^2}{D} I_D\right).$$

Let

$$\tau = \frac{\sigma \|u_d\|}{\sqrt{D}}, \qquad g = \frac{W_0 u_d}{\tau}.$$

Then $g \mid (u_d, h) \sim \mathcal{N}(0, I_D)$ and $W_0 u_d \neq 0$ almost surely. Writing $\bar{h} = h/\|h\|$,

$$\cos(W_0 u_d, h) = \frac{\bar{h}^\top g}{\|g\|}.$$

Choose an orthogonal matrix U such that $U^\top \bar{h} = e_1$. By orthogonal invariance of the standard Gaussian, $\tilde{g} = U^\top g \mid (u_d, h) \sim \mathcal{N}(0, I_D)$. Hence

$$\cos(W_0 u_d, h) \overset{d}{=} \frac{\tilde{g}_1}{\sqrt{\tilde{g}_1^2 + \cdots + \tilde{g}_D^2}}.$$

The numerator is symmetric about zero, so

$$\mathbb{E}_{W_0}\left[\cos(W_0 u_d, h) \mid (u_d, h)\right] = 0.$$

Moreover, $\tilde{g}_1^2, \ldots, \tilde{g}_D^2$ are identically distributed. Therefore,

$$\begin{aligned} D\,\mathbb{E}_{W_0}\left[\frac{\tilde{g}_1^2}{\sum_{j=1}^D \tilde{g}_j^2}\middle| (u_d, h)\right] &= \mathbb{E}_{W_0}\left[\frac{\sum_{j=1}^D \tilde{g}_j^2}{\sum_{j=1}^D \tilde{g}_j^2}\middle| (u_d, h)\right] \\ &= 1. \end{aligned}$$

Thus,

$$\mathrm{Var}_{W_0}\left[\cos(W_0 u_d, h) \mid (u_d, h)\right] = \frac{1}{D}.$$

Since this conditional distribution does not depend on the magnitude or direction of u_d, the cosine statistics induced by the random initialization are independent of the quantization bit width whenever $u_d \neq 0$.

Step 2: Reconstruction Objective. Define

$$\Gamma = A^\top A, \qquad \Sigma_d = \mathbb{E}[u_d u_d^\top], \qquad C = \mathbb{E}[h u_d^\top],$$

where the expectations are over the data distribution. The Reconstruction Objective is

$$F(W) = \frac{1}{2}\mathbb{E}\left[\|A(W u_d - h)\|^2\right] = \frac{1}{2}\mathbb{E}\left[(W u_d - h)^\top \Gamma (W u_d - h)\right].$$

Differentiating with respect to W gives

$$\nabla_W F = \Gamma(W \Sigma_d - C). \tag{2}$$

For any $z \in \mathrm{null}(\Sigma_d)$,

$$0 = z^\top \Sigma_d z = \mathbb{E}[(z^\top u_d)^2],$$

so $z^\top u_d = 0$ almost surely and hence $Cz = 0$. Therefore,

$$C = C \Sigma_d^+ \Sigma_d.$$

Thus

$$W_\star = C \Sigma_d^+$$

satisfies

$$W_\star \Sigma_d = C$$

and hence $\nabla_W F(W_\star) = 0$. Since F is a convex quadratic, $W_\star$ is a minimizer of F.

Step 3: mode-wise adaptation. Define the decoder error

$$B_t = W_t - W_\star.$$

Starting from W_0, population (full-batch) gradient descent gives

$$W_{t+1} = W_t - \eta \nabla_W F(W_t).$$

Using $\Gamma(W_\star \Sigma_d - C) = 0$, the error satisfies

$$B_{t+1} = B_t - \eta \Gamma B_t \Sigma_d.$$

Consider the orthonormal eigen decompositions

$$\Gamma = U \operatorname{diag}(\lambda_1, \ldots, \lambda_D) U^\top, \qquad \Sigma_d = V \operatorname{diag}(\mu_1, \ldots, \mu_m) V^\top.$$

Multiplying the recursion on the left by $U^\top$ and on the right by V gives

$$U^\top B_{t+1} V = U^\top B_t V - \eta \operatorname{diag}(\lambda_1, \ldots, \lambda_D)(U^\top B_t V) \operatorname{diag}(\mu_1, \ldots, \mu_m).$$

Define

$$\beta_{ij,t} = \left(U^\top B_t V\right)_{ij}.$$

Then

$$\beta_{ij,t+1} = (1 - \eta \lambda_i \mu_j) \beta_{ij,t},$$

and hence

$$\beta_{ij,t} = (1 - \eta \lambda_i \mu_j)^t \beta_{ij,0}.$$

For every mode with $\lambda_i \mu_j > 0$, if

$$0 < \eta < \frac{2}{\lambda_{\max}(\Gamma) \mu_{\max}(\Sigma_d)},$$

then

$$|1 - \eta \lambda_i \mu_j| < 1,$$

so the mode contracts geometrically. Modes with $\lambda_i \mu_j = 0$ remain unchanged. The contraction factors and the minimizer $W_\star$ may depend on the quantization bit width through the distribution of u_d. □

E.2 Per-tensor scale collapse

Proof for Proposition 2

Proof. Write $x_{\min} = \min_i x_i$ and $\Delta = \Delta_K$. The quantization levels are $\ell_j = x_{\min} + j\Delta$ for $j = 0, \ldots, K-1$, and the cell boundaries are the midpoints $b_j = x_{\min} + (j - \frac{1}{2})\Delta$ for $j = 1, \ldots, K-1$. Set $b_0 = -\infty$ and $b_K = +\infty$. Round-to-nearest maps $y \in \mathbb{R}$ to ℓ_j exactly when $y \in C_j$, where the cells $C_0, \ldots, C_{K-1}$ partition $\mathbb{R}$ and each finite boundary b_j is assigned to exactly one of C_{j-1} and C_j by the tie-breaking rule. The argument uses only two properties of this partition:

$$b_{j+1} - b_j = \Delta \quad \text{for } 1 \le j \le K-2, \tag{3}$$

$$C_j \subseteq [b_j, b_{j+1}] \cap \mathbb{R} \quad \text{for } 0 \le j \le K-1. \tag{4}$$

Step 1: boundaries inside I. Let k be the number of boundaries $b_1, \ldots, b_{K-1}$ that lie in I. The boundaries increase with j and I is an interval, so these have consecutive indices $b_p, \ldots, b_{p+k-1}$. If $k \ge 1$, both b_p and b_{p+k-1} lie in I, and the equal spacing (3) gives $(k-1)\Delta = b_{p+k-1} - b_p \le w$. Since $k - 1$ is an integer, $k - 1 \le \lfloor w/\Delta \rfloor$. This also holds trivially when $k = 0$, so

$$k \le \lfloor w/\Delta \rfloor + 1. \tag{5}$$

Step 2: cells met by I. Let $j_1 < \cdots < j_r$ be the indices of the cells that intersect I. For $1 \le i < r$, pick $y \in I \cap C_{j_i}$ and $z \in I \cap C_{j_{i+1}}$. The containment (4) and the monotonicity of the boundaries give $y \le b_{j_i+1} \le b_{j_{i+1}} \le z$, and since I is an interval, $b_{j_{i+1}} \in [y, z] \subseteq I$. Hence $b_{j_2}, \ldots, b_{j_r}$ are $r - 1$ distinct boundaries lying in I, and $r - 1 \le k$.

Step 3: conclusion. Every entry of x in I is mapped to a level ℓ_{j_i} with $1 \le i \le r$. By Step 2 and the bound on boundaries (5), the number of distinct levels is at most $r \le k + 1 \le \lfloor w/\Delta \rfloor + 2$. It is also at most K, the total number of levels, which gives the bound $\min\{K,\ \lfloor w/\Delta_K \rfloor + 2\}$. □

Remark (scope of the argument). The proof uses only the equal boundary spacing (3) and the interval structure of the cells (4). The bound therefore holds for any position of the uniform grid, including asymmetric quantization with a rounded integer zero point (clipping at the ends only enlarges the two end cells), and for truncation quantizers, whose boundaries are the levels themselves. It does not hold for stochastic rounding, where an entry may be mapped to either neighboring level. For example, with levels at the integer multiples of Δ, the entries of $I = [-0.2\Delta, 1.3\Delta]$ can reach the four levels $-\Delta, 0, \Delta, 2\Delta$, whereas $\lfloor w/\Delta \rfloor + 2 = 3$. The same counting argument gives $\min\{K, \lceil w/\Delta_K \rceil + 2\}$ in that case.

The following bound, used in Section 6.2, compares the worst-case MSE bounds of per-tensor and blockwise quantization.

Proposition 3 (Blockwise versus global quantization)**.** *Let $x \in \mathbb{R}^n$ have range $R = \max_i x_i - \min_i x_i$, and let I be an interval of width $R_I < R$. Call the n_0 entries of x that lie outside I outliers. Consider two quantizers, both using round-to-nearest onto $K \geq 2$ uniformly spaced levels without clipping.*

- Global: *all entries share the range R, with step $\Delta_K = R/(K-1)$.*
- Blockwise: *assume G divides n and partition x into n/G contiguous blocks $\mathcal{B}_1, \ldots, \mathcal{B}_{n/G}$ of size G. Block g is quantized over its own range $R_g = \max_{i \in \mathcal{B}_g} x_i - \min_{i \in \mathcal{B}_g} x_i$ with step $\Delta_g = R_g/(K-1)$, and its metadata (offset and step) is stored exactly.*

Write $\mathrm{MSE}(Q) = \frac{1}{n}\|Q(x) - x\|_2^2$ and define

$$B_{\mathrm{global}} = \frac{R^2}{4(K-1)^2}, \qquad B_{\mathrm{block}} = \frac{G}{4n(K-1)^2} \sum_{g=1}^{n/G} R_g^2. \tag{6}$$

Then $\mathrm{MSE}(Q_{\mathrm{global}}) \leq B_{\mathrm{global}}$ and $\mathrm{MSE}(Q_{\mathrm{block}}) \leq B_{\mathrm{block}}$. Let m be the number of entries in blocks that contain at least one outlier. Then $m \leq \min(n, Gn_0)$,

$$B_{\mathrm{block}} \leq \frac{mR^2 + (n-m)R_I^2}{4n(K-1)^2}, \tag{7}$$

and, with $\rho = R_I/R$ and $\mu = \min(1, Gn_0/n)$,

$$\frac{B_{\mathrm{block}}}{B_{\mathrm{global}}} \leq \frac{m}{n} + \Big(1 - \frac{m}{n}\Big)\rho^2 \leq \mu + (1-\mu)\rho^2 \leq 1, \tag{8}$$

where the last inequality is strict whenever $Gn_0 < n$.

Proof for Proposition 3

Proof. Step 0: error of one uniform grid. Let $\ell_j = a + j\Delta$, $j = 0, \ldots, K-1$, and let $y \in [a, a + (K-1)\Delta]$. Then $\ell_j \leq y \leq \ell_{j+1}$ for some j (or $y = \ell_{K-1}$), so the nearest level lies within $\min(y - \ell_j, \ell_{j+1} - y) \leq \Delta/2$ of y, whatever the tie-breaking rule.

Step 1: global quantization. Every entry lies in $[\min_i x_i, \max_i x_i]$, which the global grid spans. By Step 0, $|Q_{\mathrm{global}}(x_i) - x_i| \leq \Delta_K/2 = R/(2(K-1))$ for every i. Squaring and averaging gives $\mathrm{MSE}(Q_{\mathrm{global}}) \leq R^2/(4(K-1)^2) = B_{\mathrm{global}}$.

Step 2: blockwise quantization. The grid of block g spans the range of that block, so Step 0 gives $|Q_{\mathrm{block}}(x_i) - x_i| \leq R_g/(2(K-1))$ for $i \in \mathcal{B}_g$. (If $R_g = 0$, the block is constant and is reproduced exactly.) Hence

$$\sum_{i \in \mathcal{B}_g} \big(Q_{\mathrm{block}}(x_i) - x_i\big)^2 \leq \frac{G\, R_g^2}{4(K-1)^2}.$$

Summing over the n/G blocks and dividing by n gives $\mathrm{MSE}(Q_{\mathrm{block}}) \leq B_{\mathrm{block}}$ as defined in (6).

Step 3: separating contaminated blocks. Let c be the number of blocks that contain at least one outlier, so that $m = Gc$. Each outlier lies in exactly one block, so $c \leq n_0$. Also $c \leq n/G$, and

therefore $m \le \min(n, Gn_0)$. A block without outliers has all of its entries in I, so $R_g \le R_I$. A contaminated block is a subset of x, so $R_g \le R$. Consequently

$$\sum_{g=1}^{n/G} R_g^2 \le cR^2 + \Big(\frac{n}{G} - c\Big)R_I^2 = \frac{mR^2 + (n-m)R_I^2}{G},$$

and substituting into the definition (6) proves the blockwise bound (7).

Step 4: the ratio. Dividing the blockwise bound (7) by B_{global} gives $B_{\text{block}}/B_{\text{global}} \le f(m/n)$ with $f(t) = \rho^2 + t(1-\rho^2)$. Because $\rho < 1$, f is strictly increasing. Since $m/n \le \mu \le 1$, we obtain $f(m/n) \le f(\mu) = \mu + (1-\mu)\rho^2 \le f(1) = 1$, with $f(\mu) < 1$ whenever $\mu < 1$, that is, whenever $Gn_0 < n$. This proves the ratio bound (8). □

Remark (worst-case and typical error). Proposition 3 compares worst-case error bounds. It does not order the realized errors for every tensor: if all entries of x happen to lie on the global grid, the global error is zero, while the block grids need not align with the data. Under the standard model in which the rounding error is uniform on $[-\Delta/2, \Delta/2]$, the expected errors are $R^2/(12(K-1)^2)$ and $\frac{G}{12n(K-1)^2}\sum_g R_g^2$. Their ratio equals $B_{\text{block}}/B_{\text{global}}$, so the ratio bound (8) also describes typical error under this model. With stochastic rounding, both per-entry bounds double and the ratio is unchanged. For example, with $n = 4096$, $G = 32$, $n_0 = 4$ and $\rho = 0.1$, the ratio bound (8) is about 0.041.

Remark (metadata). The proposition assumes exact block metadata and does not charge for storing it. If the stored step $\hat{\Delta}_g$ satisfies $\Delta_g \le \hat{\Delta}_g \le c\,\Delta_g$ and the grid still covers the block (for instance $c = 2$ when the scale is rounded up to a power of two), the per-entry error is at most $\hat{\Delta}_g/2$, and B_{block} and the ratio bound grow by at most a factor c^2. In addition, an s-bit scale per block costs s/G extra bits per entry. At an equal bit budget, the global quantizer could use roughly $2^{s/G}$ times as many levels, which reduces B_{global} by roughly $2^{-2s/G}$. For $s = 16$ and $G = 32$ this is a factor of about 2, so the example above becomes roughly 0.08.

Applying the conditions to LQ and AWQ. For LQ, the occupancy result applies to $z_i = \log_2(1 + |x_i|)$, using the range and bulk width of z. Even when this map is approximately linear in $|x_i|$, folding the signs changes the distribution, so its resolution ratio cannot be identified with that of x. A single reconstructed log-magnitude q gives $\hat{x}_i = \text{sign}(x_i)(2^q - 1)$, which can retain a nonconstant sign pattern. For AWQ, apply the result to $z_{tj} = x_{tj}/s_j$, where the calibrated scales s_j are fixed and positive, and undo the scaling after quantization. Channel scaling changes the relevant range and bulk width; it does not remove their dependence on all coordinates sharing the tensor scale.

For DQ, complete single-code collapse of a $T \times D$ tensor yields $q\mathbf{1}_T\mathbf{1}_D^\top$ and removes token-content contrasts within that tensor. The analogous AWQ reconstruction is $q\mathbf{1}_T s^\top$. A single-code bulk is weaker than this complete collision: rare coordinates, signs, sample-specific metadata, and other modalities can still carry information. Accordingly, the theory gives sufficient conditions for code concentration and bounds on local distortion; assigning the observed downstream failures to these mechanisms requires the corresponding activation statistics and complete reconstructions.

F Robustness of Method Comparison to Client Number

In this section, we justify why using a single client for model benchmarking does not affect our conclusions regarding the relative performance of different compression methods or their failure modes.

For FSQ and RD-FSQ, the observed degradation is primarily caused by optimization instability associated with the randomly initialized trainable decoder and the limited optimization budget, as discussed in Section 6.1. This failure mode arises from the difficulty of adapting the randomly transformed activations to the largely frozen downstream network and is not fundamentally dependent on the number of clients.

For DQ, AWQ, and LQ, the observed failure is caused by per-tensor scale collapse, where heavy-tailed activation values expand the global quantization range and leave only a small number of effective quantization levels for the bulk of the activations, as discussed in Section 6.2. Although splitting a batch across multiple clients changes the tensor-level statistics used by these methods, it does not remove the underlying scale-collapse mechanism, as shown in Figure F.1, this concentration persists across batch sizes of 8, 16, and 32: the central 99% of text activations occupies only about 4.7% of the full min–max range, while image activations occupy approximately 16–18%. In contrast, Ql avoids this failure mode through blockwise NormalFloat quantization, which localizes the effect of outliers to individual blocks. Therefore, the mechanisms underlying the relative performance differences among these methods persist when moving from the single-client benchmark to the multi-client setting.

For Top-K, the sparsification operation is independent across the batch dimension: each token is sparsified independently based on its activation values. Thus, splitting a batch across multiple clients does not alter the sparsification operation itself. Its suboptimal performance primarily results from information loss caused by setting a large fraction of activation values to zero.

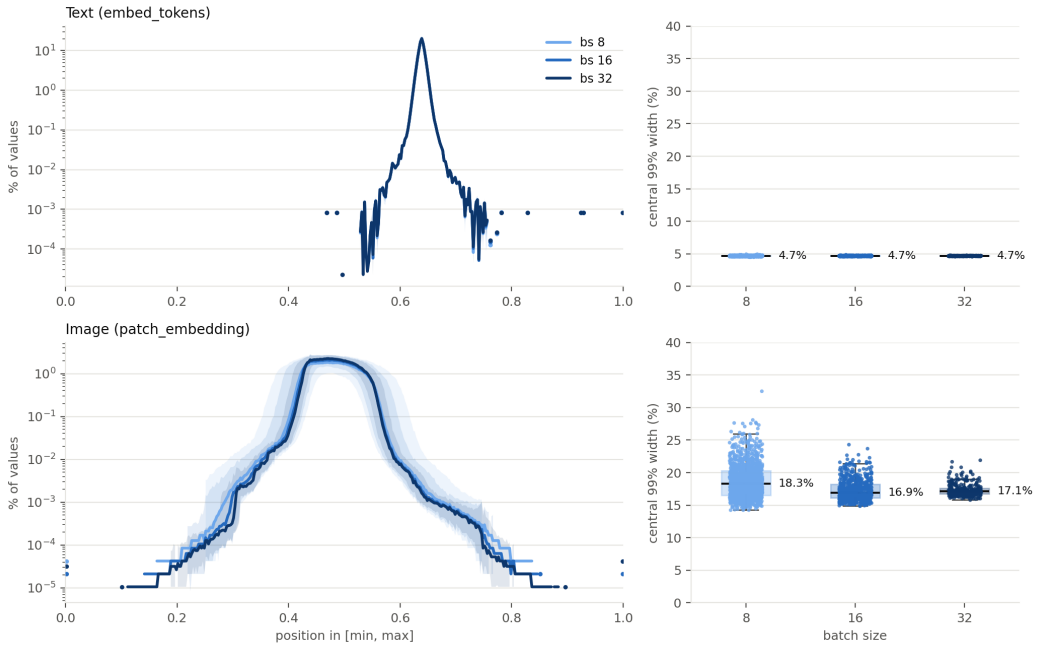

Figure F.1: **Activation distributions and quantization-range utilization at the vision and text cuts.** The batchs are partitioned from 10000 randomly selected samples from the traning set from the performance benchmark. **Top left:** Distribution of text token embeddings from PaliGemma2 at the text cut for batch sizes 8, 16, and 32. Values are normalized within each batch's min–max range. **Top right:** Fraction of the full min–max range occupied by the central 99% of values at the text cut. Across batch sizes, the central 99% occupies only about 4.7% of the available range. **Bottom left:** Distribution of image patch from PaliGemma2 embeddings at the vision cut. Image activations occupy a broader portion of the min–max range than text embeddings, although most values remain concentrated away from the extrema. **Bottom right:** Fraction of the full min–max range occupied by the central 99% of values at the vision cut, spanning approximately 16–18% across batch sizes. Together, the results show that a small number of extreme activations expand the per-batch dynamic range and leave much of the uniform quantization range underutilized, particularly at the text cut.

G 32B DEPLOYMENT AND TRACE-DRIVEN NETWORK SIMULATION

Setup. Table G.1 lists the configuration of the run in Section 5.3. It uses a larger corpus and a different recipe from the four-backbone study, so its accuracy (Table 2) is not comparable with Tables 1 and D.2. Clients and server ran on one node and exchanged frames over TCP on the loopback interface, with negligible latency and about 17 Gbps of bandwidth. We ran a full epoch (8,231 steps) with bf16, Ql 4-bit and Ql 2-bit activations; the measured epochs took 32.7 h, 32.3 h and 32.5 h. Server computation accounted for 14.1 s, 14.0 s and 14.1 s of the 14.3 s, 14.1 s and 14.2 s average steps, and this overlap hid most client preparation and transfer. Each client frame carries the Ql indices with the block and row metadata of Table D.1, the token skeleton needed to place the image features, and the labels; the server answers each step with a short acknowledgement.

Table G.1: Configuration of the 32B deployment.

Backbone	Qwen3-VL-32B-Instruct, 64 decoder layers
Clients	3 processes on CPU (8 threads each); frozen patch and token embeddings; bf16 or Ql (d=16, 4)
Server	2× NVIDIA H200 (141 GB), data parallel
Data	143,430 samples of Section 4 + 646,750 PubMedVision VQA samples = 790,180, split at random into three disjoint client shards
Batch	32 per client, 96 global; one epoch = 8,231 steps
Trainable	LoRA r=64, α=128, dropout 0.05 on the q, k, v, o, gate, up and down projections of layers 32–63; connector fully trained (313M parameters)
Optimization	AdamW, lr 10^{-4} (LoRA) and 2×10^{-5} (connector), cosine decay, 3% warmup, gradient clipping 1.0, `bf16` with fp32 master weights
Schedule	pipelined: clients prepare and send step t+1 while the server trains on step t

Trace-driven simulation. The simulation is a discrete-event model of the implementation's client and server loops, driven by the per-step measurements in Table G.2. A client prepares step t, is blocked while sending it for half the round-trip time plus the frame over its available rate, and starts step t+1 once at most one sent step is unanswered (pipelined) or once step t is answered (without pipelining). The server starts step t when all frames of step t have arrived, computes for the measured time, answers every client after half the round-trip time, and checkpoints at the measured steps. Transfers that share a server ingress divide it max-min fairly, each capped by its own uplink. The round-trip time is 29 ms, Azure's median latency between East US and Central US (Microsoft, 2026); bandwidth means goodput, and TCP dynamics, loss and jitter are not modeled. Simulating the measured run on its own co-located link reproduces its step time to within 0.1 ms.

Table G.2: Per-step inputs of the simulation. Frame sizes are per client; the critical uplink is the smallest per-institution rate at which the pipelined step stays within 1% of the co-located step, and the critical server ingress the smallest ingress, shared by the three institutions, with the same property. Frame sizes, uploads and server computation are measured over all 8,231 steps of each run.

	bf16	Ql 4-bit	Ql 2-bit
Frame per client and step (MB)	123.7	34.0	18.6
Upload per institution and epoch (TB)	1.02	0.28	0.15
Client preparation per step (s)	1.39	2.43	1.63
Server computation per step (s)	14.09	13.98	13.98
Critical uplink per institution (Mbps)	90.0	27.5	13.9
Critical server ingress, three institutions (Mbps)	240.9	72.3	37.0

Network scenarios. Health data must be protected in transit (U.S. Department of Health and Human Services, 2003), commonly with VPNs. In one benchmark on a 1,000 Mbps link, IPsec achieved 881 Mbps and OpenVPN 258 Mbps (Donenfeld, 2017), so an institution's available uplink is the lower of its link rate and its gateway throughput. Each link also serves clinical systems and the many departments of an institution, so training receives only a share of it, and multi-hospital collaborations usually mix large and small sites. Table G.3 gives the share of the available uplink that training needs to run as fast as on one node, and the epoch time when it receives 10%. Because the required rates of 90, 28 and 14 Mbps are set by the model, batch and server rather than by the institution, compression cuts every institution's requirement by the same 3.3× (Ql 4-bit) or 6.5× (Ql 2-bit). A large hospital behind OpenVPN uses 10.7% of its uplink instead of 34.9%, returning

62.5 Mbps to clinical use, and a small hospital uses 28% or 14% instead of 90%. With a 10% share, one small hospital slows a collaboration with two large ones to the pace of three small hospitals: an epoch takes 229.9 h under bf16, 7.1× the co-located time, against 68.0 h (2.1×) for Ql 4-bit and 38.1 h (1.18×) for Ql 2-bit.

Table G.3: Share of its available uplink that each institution must give training to keep the 32B deployment as fast as on one node, and the epoch time when training is limited to 10% of that uplink. Available uplink is the lower of the link rate and the VPN gateway throughput (IPsec 881 Mbps, OpenVPN 258 Mbps; Donenfeld, 2017); a collaboration's requirement is set by its smallest uplink. Co-located epochs take 32.6 h (bf16) and 32.3 h (Ql). Only the co-located Ql 4-bit time was measured; all other entries are trace-replay projections. All scenarios appear in Table G.4.

		Share for full speed (%)			Epoch at 10% share (h)		
Collaboration	Uplink (Mbps)	bf16	Ql 4-bit	Ql 2-bit	bf16	Ql 4-bit	Ql 2-bit
3 large hospitals, IPsec	881	10.2	3.1	1.6	33.0	32.3	32.3
3 large hospitals, OpenVPN	258	34.9	10.7	5.4	91.8	33.1	32.3
2 large (IPsec) + 1 small	881, 881, 100	90.0	27.5	13.9	229.9	68.0	38.1
3 small hospitals	100	90.0	27.5	13.9	231.1	68.6	39.1

Table G.4: One-epoch time (h) of the 32B deployment in every scenario (pipelined), with the speedup over bf16 in parentheses. Available uplink rates are scenario assumptions. IPsec and OpenVPN gateways are assigned 881 Mbps and 258 Mbps, based on one benchmark (Donenfeld, 2017); an institution's uplink is the lower of its link rate and gateway throughput, before applying the training share. Only the co-located Ql 4-bit time was measured; all other times are trace-replay projections.

Setting	Link	Uplink (Mbps)	bf16	Ql 4-bit	Ql 2-bit
Co-located baseline	–	–	32.6	32.3 (1.01×)	32.3 (1.01×)
Large hospitals, 25% share	1,000 Mbps, IPsec	220	32.6	32.3 (1.01×)	32.3 (1.01×)
Large hospitals, 10% share	1,000 Mbps, IPsec	88	33.0	32.3 (1.02×)	32.3 (1.02×)
Large hospitals, 25% share	1,000 Mbps, OpenVPN	65	39.7	32.3 (1.23×)	32.3 (1.23×)
Large hospitals, 10% share	1,000 Mbps, OpenVPN	26	91.8	33.1 (2.77×)	32.3 (2.84×)
Small hospitals, 25% share	100 Mbps, IPsec or OpenVPN	25	94.7	33.5 (2.82×)	32.3 (2.93×)
Small hospitals, 10% share	100 Mbps, IPsec or OpenVPN	10	231.1	68.6 (3.37×)	39.1 (5.92×)
Two large + one small, 10% share	Slowest site	10	229.9	68.0 (3.38×)	38.1 (6.03×)
Server reserves 100 Mbps total	Shared by 3 institutions	33	68.0	32.3 (2.10×)	32.3 (2.10×)