Anchor to the Receiver: Path-Dependent Communication of Quantized Updates

Anonymous authors
Paper under double-blind review

Abstract

Quantized models deployed on many devices are adapted and re-delivered as new data arrives. When each receiver stores a single fixed-grid integer model, every update is a patch against the codes it already holds, and every code change is paid for when it is sent, including changes that a later update undoes. The cost of a stream of updates therefore depends on the path of delivered models, not only on its endpoint. We show that the reference a weight regularizer remembers determines this path. Anchoring to the sender's current quantization bin locks in boundary crossings before they are delivered, whereas anchoring to the installed model pulls delivered changes back. We propose QCTR, which anchors continuous sender weights to the receiver's last-delivered codes, advances the anchor only when a patch has been applied, and keeps the sender's within-cell residual. Changing only the reference lowers traffic at every regularization strength we test, by up to 59%. On a Llama-3.2-1B classification stream, QCTR reaches 87% accuracy with 3.5$\times$ less traffic than the strongest alternative. Over ten deliveries it sends 40% less than either alternative reference at similar accuracy and 10.8$\times$ less than straight-through fine-tuning. The advantage transfers to Llama-3.2-3B without retuning, holds on a generative stream, and composes with replay and low-rank updates.

1 Introduction

A quantized model can be cheap to install yet expensive to keep current. The INT4 projection weights of Llama-3.2-1B occupy about 0.5 GB, but over ten rounds of adaptation in our experiments, ordinary straight-through (STE) fine-tuning produces 103 MB of patches per receiver, or roughly 100 TB across a million devices. On the same stream, the method we propose learns a different sequence of models whose ten patches total 9.5 MB, and it ends with higher accuracy. Both methods use the same lossless encoder. The difference lies in which code changes training chooses to make.

We study in-place updates to receivers that store a single fixed-grid quantized model. Each update is a sparse patch of integer codes that the receiver applies exactly and verifies by hash, with no adapter or second inference path. Because every patch is decoded against the receiver's current codes, a coordinate that changes and later changes back is paid for twice, yet leaves no trace in the final difference from installation (Fig. 1a). Recurring communication is therefore *path-dependent*: its cost depends on when changes are committed and whether later updates undo them.

Regularization is the natural way to make training aware of this cost, and the decisive design choice is what the regularizer remembers. Two references come readily to mind, and each fails in a characteristic way (Fig. 1b). Anchoring to the sender's *current quantization bin*, the center attraction used to stabilize quantization-aware training (Nagel et al., 2022; Han et al., 2021), follows a weight into its new bin before the receiver holds that code, so a transient crossing is reinforced and shipped. Anchoring to the *initial* model resists such crossings, but once a change has been delivered it keeps pulling the weight back, and each reversal is another patch entry. Over ten deliveries the two references finish with nearly identical bills, 15.8 and 15.7 MB, for opposite reasons: current-bin anchoring ends 5.7 million codes away from installation, whereas initial-state anchoring ends only 0.54 million away because 46% of its delivered changes undo earlier ones.

The reference that avoids both failures is the state against which the next patch will be decoded. We propose Quantized Continual Training with Receiver anchoring (QCTR), which regularizes continu-

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.

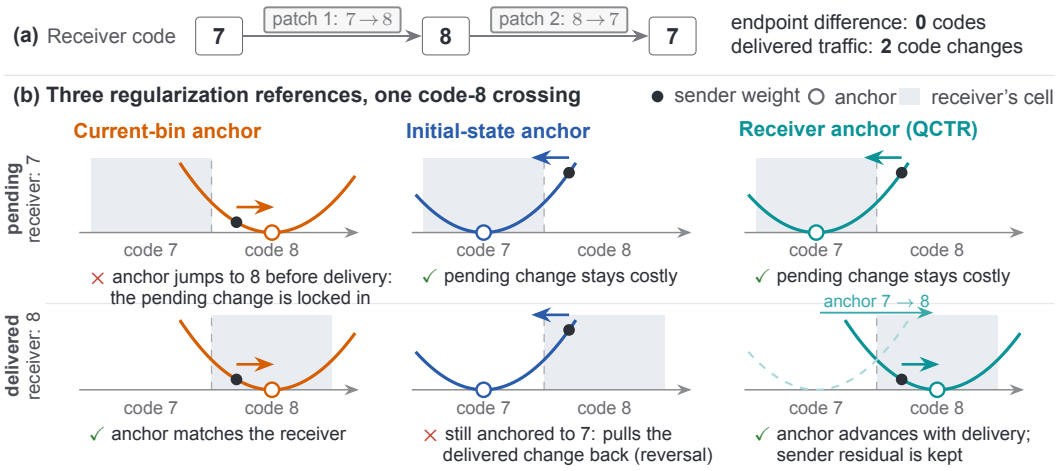

Figure 1: **Recurring updates are billed for their path, and the regularization reference decides the path.** (a) A code that is delivered and later reverted costs two patch entries but no endpoint difference. (b) A sender weight has crossed from the code-7 cell into the code-8 cell. Each panel shows the penalty as a function of the continuous weight, before the change is delivered (top) and after the receiver has applied it (bottom). The current-bin anchor jumps to code 8 early and locks in an undelivered change; the initial-state anchor stays at code 7 and pulls a delivered change back. The receiver anchor penalizes the pending change and then advances with delivery, leaving the sender weight where it is. Curves are schematic.

ous sender weights toward the reconstruction of the receiver's last-delivered codes. The anchor stays fixed while an update is learned and advances only once the patch has been applied, and the sender keeps its continuous weights, including within-cell residuals that the receiver never sees. Among the natural references, only this one controls the support of the next patch (Proposition 1). Empirically, changing only the reference lowers traffic at every regularization strength we test, and against eight baselines QCTR attains the best measured quality–traffic frontier. The advantage persists over ten deliveries, across task orders, under transfer to Llama-3.2-3B, and on a generative stream.

Our contributions are:

- **Problem.** We formalize the cost of recurring receiver-relative updates, show that it is path-dependent, and decompose its excess into over-commitment of pending changes and reversal of delivered ones. Both quantities can be measured from training traces and decoded patches.
- **Principle and method.** We characterize what each natural regularization reference controls and show that only the last-delivered receiver state bounds the next patch. QCTR pairs this anchor with delivery-gated updates and retained sender residuals.
- **Evidence.** Controlled comparisons and mechanism diagnostics confirm both predicted failure modes. Frontiers against eight baselines, together with transfer and composition studies, establish practical value.

2 Recurring Updates Are Billed by the Path

Sender and receiver. Tasks $\mathcal{D}_1, \ldots, \mathcal{D}_T$ arrive in sequence, and one update is delivered after each. The sender trains continuous weights W; the receiver stores integer codes. Both share a fixed group-wise affine quantizer G. For coordinate i in group $g(i)$, with step $s_g > 0$, integer offset z_g, and code range $[q_{\min}, q_{\max}]$,

$$q_G(W)_i = \mathrm{clip}_{[q_{\min}, q_{\max}]}\big(\mathrm{round}(W_i/s_{g(i)}) + z_{g(i)}\big), \qquad d_G(a)_i = s_{g(i)}(a_i - z_{g(i)}). \tag{1}$$

The grid is calibrated once on the initial model and never changes. Before task t, the receiver holds codes a_{t-1}. The sender holds weights W_{t-1} with $q_G(W_{t-1}) = a_{t-1}$, but these weights generally do not sit at the reconstruction centers: the *residual* $\delta_{t-1} = W_{t-1} - d_G(a_{t-1})$ is information the sender

Table 1: **Natural regularization references.** All share the grid, optimizer, and continuous sender state. Proposition 1 states what each penalty controls.

Reference	Anchor during task t	Updated	What the penalty controls	Resulting failure
Current bin	$q_G(W)$, recomputed	every step	quantization error	over-commitment
Initial state	a_0	never	drift from installation	reversal churn
Previous float	W_{t-1} (float weights)	each delivery	sender movement	cheap re-crossing
L2-SP / EWC	W_0 / W_{t-1} (float)	never / each task	parameter drift	not code-aware
Receiver (QCTR)	a_{t-1}	after delivery	**support of the next patch**	—

has and the receiver does not. Training produces W_t and proposed codes $c_t = q_G(W_t)$; a lossless encoder forms the patch $P_t = E(a_{t-1}, c_t)$, and applying it advances the receiver to $a_t = c_t$.

Cost of a stream. We measure the bytes needed to deliver every intermediate model,

$$B_{1:T} = \sum_{t=1}^{T} \big|E(a_{t-1}, c_t)\big|_{\text{bytes}}, \tag{2}$$

excluding the initial installation. The corresponding count of delivered code changes, $N_{1:T} = \sum_t \|a_t - a_{t-1}\|_0$, is bounded below by the number of codes that differ between the first and last models, $N^{\text{end}} = \|a_T - a_0\|_0$, since every net change must be sent at least once. The gap between the two, the *path excess*, arises whenever a coordinate changes more than once. Its characteristic form is a *reversal*: a delivered change that restores the code the coordinate held before its previous delivered change. A coordinate that moves $7 \to 8 \to 7$ contributes two delivered changes, one of them a reversal, and nothing to N^{end}. A second source of cost leaves no such signature: *committing* boundary crossings that the tasks do not require enlarges the path and its endpoint together. We measure commitment as the fraction of coordinates that leave the receiver's cell during an interval and are still outside it at delivery. Bytes track delivered changes closely in our encoder, at 1.7–2.1 bytes per change across all tested configurations, so the byte cost inherits the path dependence of $N_{1:T}$.

3 What Should the Regularizer Remember?

Reference-anchored regularization. We make training communication-aware with a penalty that draws the continuous weights toward a reference model. For reference codes b with reconstruction $r = d_G(b)$, an interior coordinate is anchored to its center. At a clipped endpoint, movement farther outside the representable range cannot change the code and is not penalized:

$$K_i(b) = \begin{cases} (-\infty, r_i], & b_i = q_{\min}, \\ \{r_i\}, & q_{\min} < b_i < q_{\max}, \\ [r_i, \infty), & b_i = q_{\max}, \end{cases} \qquad R(W; b) = \sum_i \operatorname{dist}\big(W_i, K_i(b)\big)^2. \tag{3}$$

Task t minimizes $\mathcal{L}_t\big(d_G(q_G(W)); \mathcal{D}_t\big) + \lambda R(W; b_t)$, with straight-through gradients through q_G and a detached reference b_t. Center attraction gives a restoring gradient before a weight reaches a cell boundary; a penalty that was flat inside the cell would give none. The task loss can still overcome it when a code change is useful. With the penalty form, grid, and optimizer fixed, the one remaining design choice is the reference b_t. Table 1 lists the natural candidates.

What each reference controls. The next patch changes exactly the coordinates in $\mathcal{F}(W) = \{i : q_G(W)_i \neq a_{t-1,i}\}$. A useful penalty should therefore grow with $\mathcal{F}(W)$. Only one of the candidates does.

Proposition 1 (What each reference controls)**.** *Fix G and the receiver codes $a = a_{t-1}$, and let W^- be the sender state at the start of the interval, so that $q_G(W^-) = a$. In real arithmetic, for all W:*

(i) Receiver: $R(W; a) \geq \frac{1}{4} \sum_{i \in \mathcal{F}(W)} s_{g(i)}^2$. *Every entry of the next patch costs at least a quarter squared step.*

(ii) Current bin: $R(W; q_G(W)) \leq \frac{1}{4} \sum_i s_{g(i)}^2$, *and it is zero whenever W lies on reconstruction centers, whatever $|\mathcal{F}(W)|$.*

(iii) Initial state: $R(W; a_0) \geq \frac{1}{4} \sum_{i:\, q_G(W)_i \neq a_{0,i}} s^2_{g(i)}$. *If $a_i \neq a_{0,i}$, the i-th term vanishes only when $q_G(W)_i = a_{0,i}$, that is, only when the next patch reverts coordinate i.*

(iv) Previous float: $\|W - W^-\|_2^2 \geq \sum_{i \in \mathcal{F}(W)} \rho_i^2$, *where ρ_i is the distance from W_i^- to the boundary of the cell of a_i. This bound is tight, and ρ_i can be arbitrarily small.*

Each part follows from the cell geometry of Eq. (1) (Appendix A), and together they predict how each alternative fails. The current-bin penalty measures quantization error rather than the patch: after a crossing, its gradient points toward the *new* center and deepens the crossing before any code is sent. The initial-state penalty measures distance to installation, so for a delivered change it rewards exactly the reversal that creates another patch entry. Previous-float L2 bounds sender movement, but a coordinate delivered in the last patch typically sits just inside the boundary it crossed, where a small step reverses it. Only the receiver penalty charges every entry of the next patch a price set by the grid alone. Bytes also depend on positions and encoding, so we evaluate complete files.

Receiver anchoring is not temporal L2. The comparison with previous-float L2 is exact. On interior coordinates,

$$\|W - d_G(a_{t-1})\|_2^2 = \|W - W_{t-1}\|_2^2 + 2\langle W - W_{t-1}, \delta_{t-1}\rangle + \|\delta_{t-1}\|_2^2, \tag{4}$$

so QCTR equals previous-float L2 plus a constant force $-2\lambda\delta_{t-1}$ toward the delivered centers. The force is largest for coordinates far from their center, such as those that crossed a boundary in the previous interval. QCTR pulls them into the interior of their new cell and consolidates the delivered change. Previous-float L2 exerts no force at the start of an interval and leaves such coordinates one small step from reversing.

QCTR: anchor to the receiver, keep the residual. QCTR sets $b_t = a_{t-1}$ (Algorithm 1). The anchor is frozen for the whole interval and advances only when the receiver's state changes: the sender quantizes, encodes the difference, applies the patch to a copy of the receiver state, and moves the anchor once the reconstruction hash matches. In our experiments deliveries coincide with task boundaries. Stating the rule in terms of delivery keeps the anchor equal to the patch base when they do not. The sender then *keeps* W_t rather than snapping it to $d_G(a_t)$. This mirrors the memory in error-feedback compression (Seide et al., 2014; Stich et al., 2018; Karimireddy et al., 2019): the sender transmits a quantized state and carries the untransmitted remainder forward as optimization state. The anchor follows what was sent, and the optimizer keeps what was not.

4 Experimental Setup

Streams and models. The primary stream adapts base Llama-3.2-1B (Meta, 2024) to AG News, Yelp Polarity, Amazon Polarity, DBpedia, and Yahoo Answers, in that order (Zhang et al., 2015). Each task has 1,000 training, 200 validation, and 300 test examples. We train all attention and MLP projections at full rank for 150 steps per task with effective batch size eight, BF16 weights and AdamW state, and a fresh optimizer per task, using a fixed groupwise INT4 grid (group size 128) calibrated once on the initial model. Final accuracy is the mean over the five tasks, evaluated on the receiver's reconstruction after the last delivery. A *ten-delivery* stream makes a second pass over the same five domains with fresh training samples. Two further streams reverse the task order or apply a fixed permutation. For transfer, we run Llama-3.2-3B for three seeds and an eight-task

Algorithm 1 QCTR adaptation and delivery for task t

Input: grid G, receiver codes a, sender weights W with $q_G(W) = a$, data $\mathcal{D}_t$, strength λ
Freeze the anchor a; initialize a fresh optimizer
for each adaptation step **do**
    Update W with the STE task gradient plus $\lambda \nabla_W R(W; a)$
end for
$c \leftarrow q_G(W)$; send $P \leftarrow E(a, c)$
Receiver: $a' \leftarrow \text{Apply}(a, P)$; if $\text{hash}(a') = \text{hash}(c)$, advance $a \leftarrow a'$
Keep the continuous W, including its residual $W - d_G(a)$, for task $t + 1$

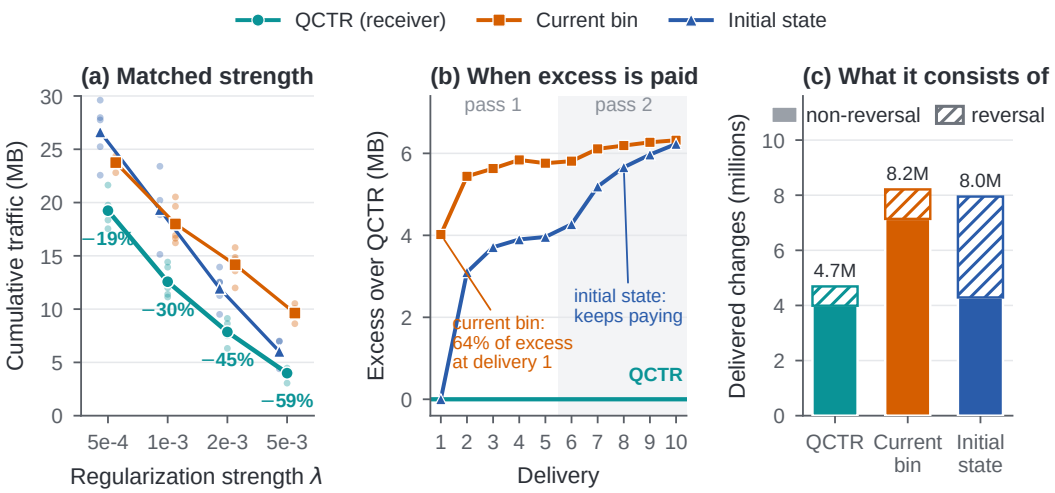

Figure 2: **The reference alone changes the bill, and the two alternatives pay for different things.** (a) Cumulative traffic over five deliveries at four matched strengths; small dots are seeds, and labels give QCTR's reduction relative to current-bin anchoring. (b) Over ten deliveries (two passes over the five domains), current-bin anchoring pays its excess early, whereas initial-state anchoring keeps paying. (c) Delivered code changes over the ten deliveries, split into reversals and all other changes. Means over five seeds at $\lambda = 2 \times 10^{-3}$ in (b) and (c).

generative stream derived from TRACE (Wang et al., 2023) on Llama-3.2-1B for five seeds; it covers classification, summarization, code completion, reasoning, and simplification. Complete protocols and a study map are given in Appendix B.

Baselines. The reference controls use QCTR's penalty with current-bin or initial-state anchors (Table 1). We also compare previous-float L2, L2-SP (Li et al., 2018), online EWC (Kirkpatrick et al., 2017; Schwarz et al., 2018), replay, and sequential STE with learning-rate sweeps. *Receiver-merged LoRA* trains a fresh rank-r adapter on all seven projections in each interval, quantizes and sends the factors, and has the receiver merge them into its reconstruction and requantize onto the same grid. The receiver therefore keeps one INT4 model, as under QCTR, and the two methods differ only in how the update is parameterized.

Traffic. After every task, each code-patch method sends all proposed changes relative to the receiver through one common lossless encoder. The encoder stores sparse positions, four-bit target codes, metadata, and state hashes, and compresses each patch independently with zlib. Traffic is the total size of these files in decimal MB. Every patch is applied to the previous receiver state and verified by hash before evaluation. LoRA traffic is the total size of its factor files.

Comparisons and statistics. We use three kinds of comparison. *Matched-strength* comparisons fix the penalty form, coefficient, and every training setting, and change only the reference; they isolate the reference's effect. *Frontiers* include every tested configuration of every method. We summarize them by the traffic needed to reach 87% accuracy and by the accuracy reached at 8 MB, interpolating linearly along each method's mean Pareto frontier. *Validation-selected* policies choose one configuration per method by validation accuracy alone; they supply the settings used for transfer. Tables report means and sample standard deviations over independently trained seeds. Paired comparisons use two-sided 95% Student-t intervals over matched seed blocks (Appendix C.3).

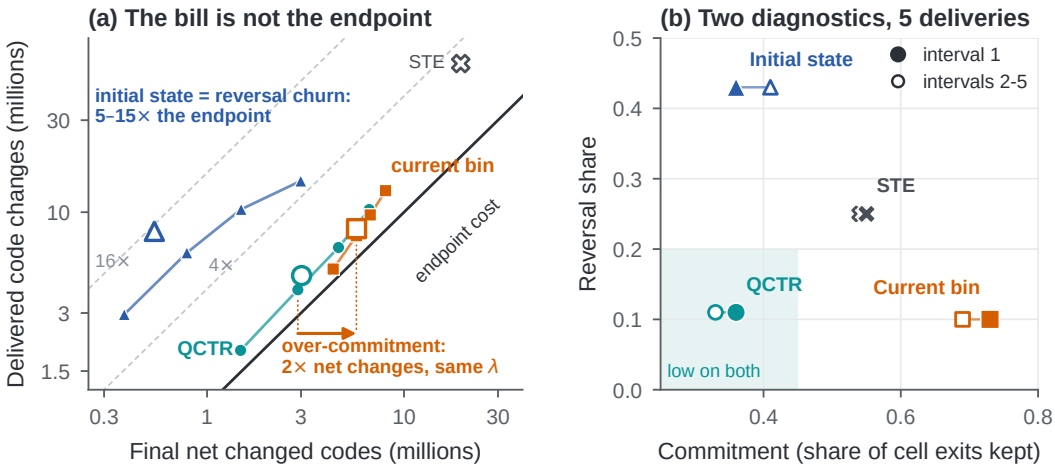

Figure 3: **Current-bin anchoring over-commits; initial-state anchoring reverses.** (a) Delivered code changes against final net changes for every primary configuration (filled; one track per reference across the four strengths) and for the ten-delivery study (hollow). Points on the diagonal would send each net change exactly once. (b) Commitment, the share of sampled coordinates that leave the receiver's cell during an interval and remain outside it at delivery, against the share of delivered changes that are reversals. Primary stream, means over five seeds.

5 Results

5.1 Changing only the reference

Figure 2a holds every setting fixed except the reference. QCTR sends less than current-bin anchoring at every strength: 19% less at $\lambda = 5 \times 10^{-4}$, rising to 59% at 5×10^{-3}. It also sends 28–35% less than initial-state anchoring. The traffic difference favors QCTR in all 20 seed–strength pairs and remains significant at every strength after Holm correction ($p < 0.002$). Paired accuracy intervals against current-bin anchoring include zero at every strength (Appendix C.3), and initial-state anchoring has the lowest mean accuracy at every strength. The saving grows with λ, as expected if stronger attraction toward the new center commits more crossings.

The ten-delivery stream shows when each alternative pays. At a common $\lambda = 2 \times 10^{-3}$, QCTR ends at 89.64% accuracy with 9.49 MB. Current-bin and initial-state anchoring reach 89.38% and 89.51% but send 15.81 and 15.72 MB, so QCTR's traffic is 40.0% and 39.5% lower. The paired accuracy differences are $+0.26$ $[-0.31, 0.83]$ and $+0.13$ $[-0.44, 0.70]$ points. Current-bin anchoring incurs 64% of its excess at the first delivery and 91% by the end of the first pass (Fig. 2b). Initial-state anchoring is identical to QCTR in the first interval, when both anchor to the installed codes, and then keeps paying. In the second pass alone it sends 3.95 MB against QCTR's 1.68 MB. The composition of the excess separates the two failures (Fig. 2c). Initial-state anchoring delivers nearly as many non-reversal changes as QCTR (4.29M versus 3.99M) but five times as many reversals (3.66M versus 0.70M). Current-bin anchoring delivers only modestly more reversals (1.07M), but 1.8× as many other changes, and ends with 1.9× QCTR's net changed codes. Sequential STE sends 102.88 MB and reaches 88.42%. Its ratio to QCTR grows from 6.6× after the first pass to 10.8× after the second; in the second pass alone it is 30× (51.22 versus 1.68 MB). The receiver anchor's advantage is largest in steady state, where a long deployment spends most of its budget.

5.2 The two failure modes

Figure 3a places every configuration on the path-versus-endpoint plane of Section 2. QCTR and current-bin anchoring both lie close to the diagonal, so their paths contain few reversals. At a matched strength, however, current-bin anchoring ends with 1.2–3.0× as many net changes: it ships changes that the receiver anchor would have pulled back. Initial-state configurations lie 5–15× above the diagonal. They end close to installation and pay for it in reversals.

Table 2: **The anchor should follow the receiver, and the sender should keep its residual.** Primary stream, $\lambda = 2 \times 10^{-3}$, mean $\pm$ SD over five seeds.

Anchor	Sender state	Accuracy (%)	MB
Receiver (QCTR)	retained residual	88.13 ± 0.51	7.86 ± 0.62
Receiver (QCTR)	snapped to centers after every delivery	86.71 ± 0.83	7.24 ± 0.58
Receiver (QCTR)	snapped to centers at the start only	87.94 ± 0.62	6.83 ± 0.49
Current bin	retained residual	87.69 ± 0.79	14.17 ± 1.45
Current bin	snapped to centers at the start only	87.38 ± 0.71	11.26 ± 0.87

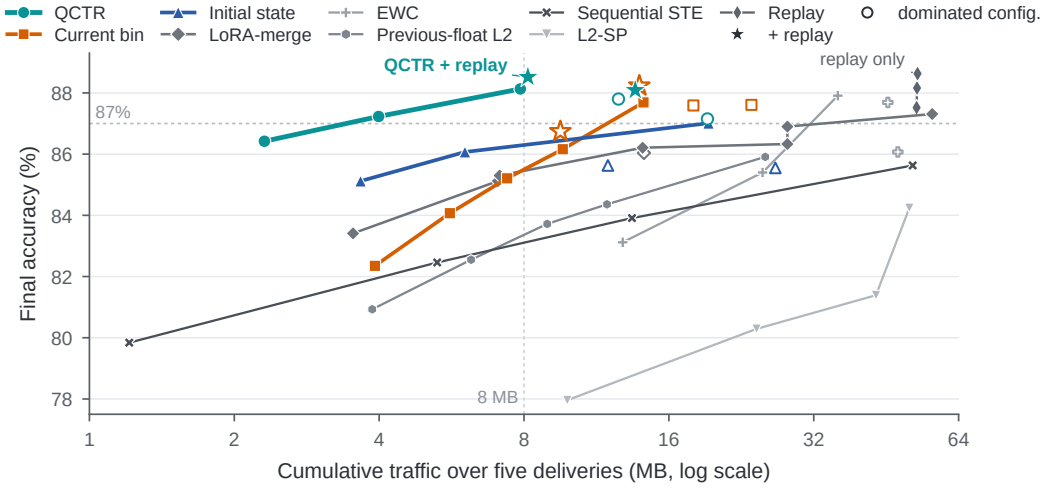

Figure 4: **Receiver anchoring gives the best measured quality–traffic frontier.** Every tested configuration on the primary stream. Lines join each method's non-dominated configurations; hollow markers are dominated. Stars are replay compositions (Table 3). Means over five seeds; per-seed spread is in Appendix D.

Commitment diagnostics show why (Fig. 3b). In the first interval, current-bin anchoring and QCTR see similar numbers of sampled coordinates leave the receiver's cell (6.4M and 6.3M after scaling). Current-bin anchoring keeps 73% of those exits outside the cell at delivery, whereas QCTR keeps 36%. Unregularized STE keeps 55%, so current-bin attraction commits more exits than no regularization at all, as Proposition 1(ii) predicts once its gradient points toward the new center. Initial-state anchoring commits as selectively as QCTR, but 43% of its delivered changes are reversals, compared with 11% for QCTR. Only the receiver anchor is low on both axes.

Two ablations separate the anchor from the sender state (Table 2). Starting both references from reconstruction centers lowers traffic for both, but 70% of the gap between them remains, so overcommitment does not come from where the pretrained weights happen to sit within their cells. Starting from centers also combines well with QCTR and lowers its traffic by a further 13%. Snapping the sender to the delivered centers after *every* delivery discards the residual, the counterpart of dropping the error-feedback memory; it saves only 0.62 MB and costs 1.42 accuracy points.

5.3 The practical frontier

A regularizer is useful only if no baseline can match it when tuned toward smaller updates. Figure 4 places every configuration of every method on one plane, including learning-rate sweeps for STE and extended penalty sweeps for the temporal baselines; each baseline has at least as many configurations as QCTR. QCTR's frontier dominates every other method's up to 50 MB. Replay alone reaches higher accuracy, at 52.6 MB, and QCTR plus replay matches it at 8.2 MB (Section 5.4). QCTR reaches 87% accuracy with 3.5 MB, compared with 12.1 MB for current-bin and 19.2 MB for initial-state anchoring; EWC and LoRA-merge need 32–35 MB (Table 10). The STE, previous-float, and L2-SP sweeps never reach 87%. The measured points tell the same story: at about 4 MB, QCTR

Table 3: **Receiver anchoring composes with replay and with low-rank updates.** Primary stream, mean ± SD over five seeds. Replay arms form one study; LoRA factor traffic varies by less than 0.01 MB across seeds. Complete LoRA sweep in Appendix E.4.

	Policy	Accuracy (%)	MB
Replay	Replay only	88.57 ± 0.92	52.61 ± 0.70
	Current bin + replay	88.23 ± 0.54	13.89 ± 1.01
	QCTR + replay	88.52 ± 0.65	**8.15** ± 0.97
LoRA-merge	rank 1, INT8	83.41 ± 1.28	3.53
	rank 4, INT8	86.21 ± 0.91	14.11
	rank 4, INT8 + **receiver anchor**	87.02 ± 0.70	14.11
	rank 16, INT8	87.31 ± 0.74	56.40

Table 4: **The traffic advantage persists across orders, seeds, scale, and task type.** Final quality (accuracy in %, or normalized score for generation) and cumulative MB. ΔMB compares QCTR with current-bin anchoring. The fresh-seed, 3B, and generative rows use policies fixed before evaluation; 3B uses the fresh-seed 1B settings without retuning.

		QCTR		Current bin			Other baseline		
Study	Seeds	Quality	MB	Quality	MB	ΔMB		Quality	MB
Original order	5	87.80	12.57	87.69	14.17	−11%	STE	85.63	51.37
Reverse order	5	88.16	15.02	87.49	17.83	−16%	STE	86.10	51.30
Permuted order	5	86.52	10.41	86.91	12.14	−14%	STE	76.92	52.44
Fresh 1B seeds	3	86.89	8.73	87.67	13.81	−37%	Low-LR STE	82.22	13.38
Llama-3.2-3B	3	90.22	7.02	89.67	21.69	−68%	Low-LR STE	88.69	33.48
Generative	5	0.427	38.71	0.418	76.10	−49%	STE	0.418	101.10

attains 87.2% accuracy, current-bin anchoring 82.4%, and STE with a reduced learning rate 82.5% (at 5.3 MB). Previous-float L2, which remembers where the sender stopped, is 4.8 points below QCTR at 8 MB. Among validation-selected policies, QCTR sends 4.1× less than STE with 2.2 points higher accuracy $[0.2, 4.1]$, and 11% less than current-bin anchoring at similar accuracy.

5.4 Composition with replay and low-rank updates

Replay limits forgetting and the receiver anchor limits what is sent, and the two combine without interference (Table 3). QCTR plus replay matches replay's accuracy within 0.05 points at 15.5% of its traffic and dominates both current-bin-plus-replay settings.

LoRA changes how an update is parameterized, not what it is anchored to. Under receiver-merged LoRA, QCTR's frontier is 3.6 points higher near 3.5 MB and 2.7 points higher at 8 MB. The more telling result is that the receiver anchor also improves LoRA itself. Applied to the merged model during adapter training, it raises rank-4 INT8 accuracy by 0.81 points $[0.22, 1.40]$ at identical factor traffic, so the reference principle is independent of the parameterization. Integer patches have a further practical advantage: they specify target codes, whereas factors must be merged by the receiver. When the receiver's FP32 merge is replayed on H200 or CPU instead of the sender's A40, 1,050 or 9,610 codes differ from the intended state after five deliveries. The state hash detects the divergence, but factors alone cannot repair it (Appendix E.4).

5.5 Robustness and transfer

The traffic advantage over current-bin anchoring holds in every setting (Table 4). QCTR is cheaper in all 15 seed–order blocks, and the settings selected at 1B transfer to Llama-3.2-3B without retuning, where current-bin anchoring sends 3.1× as much. Accuracy differences against current-bin anchoring are small, change sign across settings, and have paired intervals that include zero (Appendix C.3). Against STE, QCTR avoids a 9.6-point collapse under the permuted order and is 1.5 points more accurate at 3B $[0.5, 2.5]$. On the generative stream, QCTR uses half the current-bin traffic and 38% of

STE's at a similar mean score; part of the current-bin gap reflects a weaker selected coefficient, and at a matched coefficient the saving is 18% (Appendix E).

6 Related Work

Communicating model updates. Update Compression learns factorized updates for deployed edge models (Chen et al., 2022); Diff Pruning learns sparse task differences (Guo et al., 2021); BitDelta and DeltaZip compress fine-tuning deltas for storage and serving (Liu et al., 2024; Yao et al., 2025); and LoRA, QLoRA, and ComPEFT send compact or compressed adapters (Hu et al., 2022; Dettmers et al., 2023; Yadav et al., 2025). Code-space gradients and scale adaptation offer other ways to update low-bit models (Wu et al., 2026; Li et al., 2026). PULSESync exploits precision-induced sparsity for exact trainer-to-inference synchronization (Miahi & Belilovsky, 2026), and neural CSI fine-tuning adds the decoder-update rate to its objective (Sattari et al., 2025a;b). Federated learning compresses training updates, often with error feedback (Li et al., 2020; Karimireddy et al., 2019; Tang et al., 2019). These methods compress a given update or learn a compact one; QCTR changes which updates training proposes, measured against the receiver over the whole path, and is complementary to better encoders and to low-rank parameterizations. Delivery-time filters such as thresholds or top-k decide after training which changes to ship, leaving the sender with weights the receiver lacks; QCTR shapes the proposals themselves.

Quantization references. ProxQuant regularizes toward the quantized set (Bai et al., 2019). Bin regularization and oscillation dampening attract latent weights toward their current quantization centers (Han et al., 2021; Nagel et al., 2022), and transition-rate scheduling controls how often codes change during quantization-aware training (Lee et al., 2025). Our current-bin control is this family's objective. Stabilizing quantized training and minimizing recurring delivery are different goals: the former anchors to the sender's bin, while the latter must anchor to the receiver's codes.

Anchoring to a deployed model. Work on prediction churn and compatible updates regularizes a new model toward the behavior of the deployed one (Milani Fard et al., 2016; Jiang et al., 2022; Yan et al., 2021; Shen et al., 2020). Proximal, transfer, and consolidation objectives anchor to server, pretrained, or previous continuous weights (Li et al., 2020; 2018; Kirkpatrick et al., 2017; Schwarz et al., 2018). QCTR anchors to the deployed model's discrete parameters, because the next patch is billed against them, and it advances that anchor exactly when delivery changes the patch base.

7 Discussion and Limitations

The results support a simple timing rule: regularize toward the state that will decode the next patch. Current-bin and initial-state anchors share QCTR's geometry and differ only in when their reference moves; the first moves too early, the second never moves, and the diagnostics trace each resulting cost to its source. The same logic should apply wherever a sender's continuous state and a receiver's discrete state diverge between deliveries, as with pruning masks or codebook indices.

QCTR shapes a geometric penalty rather than file length. It does not guarantee an optimal sequence of deliveries, and its unnormalized penalty charges wide quantization cells more than narrow ones (Appendix A). Our experiments use a fixed grid, synchronized receivers that apply every patch, BF16 master weights, and two sizes of one model family. Receivers that skip versions would need a per-version anchor or patches composed across versions, and a grid that changes would require anchoring its scales as well as its codes.

8 Conclusion

Keeping a deployed quantized model current costs the whole path of delivered changes, and which path training takes depends on what its regularizer remembers. Anchoring to the receiver's last-delivered codes, advancing the anchor only with delivery, and keeping the sender's residual avoid both early over-commitment and late reversal, cutting recurring traffic by 40% against the closest alternatives and by an order of magnitude against standard fine-tuning at similar accuracy. The design rule is simple: learn the next model relative to the state that will receive it.

Reproducibility Statement

Sections 2 to 4 specify the delivery protocol, algorithm, training budgets, and selection design. The appendix adds model revisions, numerical conventions, a study map, complete searches, per-seed results, mechanism diagnostics, and the proof of Proposition 1. Tables and empirical figures are generated from saved run summaries, with explicit seed pairing and checks against serialized packet sizes. A machine-readable point ledger, included with the source package, links each display to its producing runs; a companion evidence file records learning trajectories and paired comparisons. Source, data, quantizer, and selection fingerprints identify the configurations, and exact receiver reconstruction is checked at every delivery.

AI Use Statement

Generative AI was used for implementation and testing, literature search, and manuscript writing and revision. Numerical claims were checked against saved experimental artifacts, and bibliographic claims against source publications. The authors are responsible for the scientific content and the final submission.

References

Yu Bai, Yu-Xiang Wang, and Edo Liberty. ProxQuant: Quantized neural networks via proximal operators. In *International Conference on Learning Representations*, 2019.

Bo Chen, Ali Bakhshi, Gustavo Batista, Brian Ng, and Tat-Jun Chin. Update compression for deep neural networks on the edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 3076–3086, 2022.

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

Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.

Tiantian Han, Dong Li, Ji Liu, Lu Tian, and Yi Shan. Improving low-precision network quantization via bin regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5261–5270, 2021.

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*, 2022.

Heinrich Jiang, Harikrishna Narasimhan, Dara Bahri, Andrew Cotter, and Afshin Rostamizadeh. Churn reduction via distillation. In *International Conference on Learning Representations*, 2022.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3252–3261, 2019. URL `https://proceedings.mlr.press/v97/karimireddy19a.html`.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Junghyup Lee, Jeimin Jeon, Dohyung Kim, and Bumsub Ham. Scheduling weight transitions for quantization-aware training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23466–23475, 2025.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, 2020.

Tung-Ling Li, Jiale Huang, Lee-Chi Wang, and Janaki Ram Gotei. Scale-QLoRA: Code-invariant adapter merging for native 4-bit microscaling LLMs. *arXiv preprint arXiv:2609.04526*, 2026.

Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, 2018.

James Liu, Guangxuan Xiao, Kai Li, Jason D Lee, Song Han, Tri Dao, and Tianle Cai. BitDelta: Your fine-tune may only be worth one bit. In *Advances in Neural Information Processing Systems*, 2024.

Meta. Llama 3.2 model card, 2024. URL `https://huggingface.co/meta-llama/Llama-3.2-1B`.

Erfan Miahi and Eugene Belilovsky. Understanding and exploiting weight update sparsity for communication-efficient distributed RL. *arXiv preprint arXiv:2602.03839*, 2026.

Mahdi Milani Fard, Quentin Cormier, Kevin Canini, and Maya Gupta. Launch and iterate: Reducing prediction churn. In *Advances in Neural Information Processing Systems*, 2016.

Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 16318–16330, 2022.

Mehdi Sattari, Deniz Gündüz, and Tommy Svensson. Dynamically fine-tuned neural compressor for FDD massive MIMO CSI feedback. In *2025 IEEE 26th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, 2025a. doi: 10.1109/SPAWC66079.2025.11143260.

Mehdi Sattari, Deniz Gündüz, and Tommy Svensson. Neural CSI compression fine-tuning: Taming the communication cost of model updates. *arXiv preprint arXiv:2501.18250*, 2025b.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Proceedings of Interspeech*, 2014.

Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. Towards backward-compatible representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, 2018.

Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. DoubleSqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6155–6165, 2019. URL `https://proceedings.mlr.press/v97/tang19d.html`.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, et al. TRACE: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023.

Shiguang Wu, Zhouchen Lin, and Quanming Yao. Fine-tuning low-bit models with gradient in quantized code space. *arXiv preprint arXiv:2608.30908*, 2026.

Prateek Yadav, Leshem Choshen, Colin Raffel, and Mohit Bansal. ComPEFT: Compression for communicating parameter efficient updates via sparsification and quantization. *Transactions on Machine Learning Research*, 2025.

Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-congruent training: Towards regression-free model updates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Xiaozhe Yao, Qinghao Hu, and Ana Klimovic. DeltaZip: Efficient serving of multiple full-model-tuned LLMs. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

A Mathematical and numerical details

A.1 Proof of Proposition 1

Fix coordinate i, write $s = s_{g(i)}$, and let $r_i = d_G(a)_i$. In real arithmetic, an interior code a_i owns the cell $[r_i - s/2,\, r_i + s/2]$, with boundary points assigned by the rounding rule. The minimum code owns $(-\infty, r_i + s/2]$ and the maximum code owns $[r_i - s/2, \infty)$.

(i) Receiver. If $q_G(W)_i \neq a_i$, then W_i lies on or beyond a boundary of the cell of a_i. For an interior code $K_i(a) = \{r_i\}$, so $\mathrm{dist}(W_i, K_i(a)) \geq s/2$. For the minimum code, leaving the cell requires crossing its finite upper boundary, which lies $s/2$ from the half-line $K_i(a) = (-\infty, r_i]$; the maximum code is symmetric. Each changed coordinate therefore contributes at least $s^2/4$. Summing over $\mathcal{F}(W)$ and using nonnegativity for all other coordinates gives (i).

(ii) Current bin. Let $c = q_G(W)$. By construction, W_i lies in the cell of c_i. For an interior c_i, $|W_i - d_G(c)_i| \leq s/2$. For an endpoint c_i, the distance from W_i to the one-sided set $K_i(c)$ is either zero (beyond the center) or at most $s/2$ (inside the finite half of the cell). Summing gives the upper bound. If every W_i lies on a reconstruction center, every term is zero, whatever the relation between c and a.

(iii) Initial state. The lower bound is (i) with a_0 in place of a. For the second statement, $\mathrm{dist}(W_i, K_i(a_0)) = 0$ implies $W_i \in K_i(a_0)$, which lies inside the cell of $a_{0,i}$, so $q_G(W)_i = a_{0,i}$. If $a_i \neq a_{0,i}$, that code differs from the receiver's and is an entry of the next patch that restores the installed value.

(iv) Previous float. Because $q_G(W^-) = a$, each W_i^- lies in the cell of a_i at distance ρ_i from its nearest finite boundary. A coordinate in $\mathcal{F}(W)$ lies outside that cell, so $|W_i - W_i^-| \geq \rho_i$. Summing gives the bound. It is attained in the limit by moving each changed coordinate just past its nearest boundary. Since W_i^- may lie anywhere in the cell, ρ_i can be arbitrarily small. □

The proposition describes real-arithmetic geometry. The implementation converts weights to FP32 when selecting codes, uses `torch.round`, and clips to the integer range. Reference bounds are then represented in the sender parameter dtype. Exact values on a boundary follow the implemented rounding rule, and finite-precision evaluation can differ from ideal arithmetic. Delivery and byte measurement always use the implemented code map.

A.2 Scale weighting and an alternative surrogate

The implemented receiver penalty is an unnormalized sum. Its cell-exit guarantee therefore scales with $s_{g(i)}^2$: crossing a wide cell necessarily contributes more energy than crossing a narrow cell. One possible alternative is

$$R_{\text{step}}(W; a) = \sum_i \frac{4}{s_{g(i)}^2}\, \mathrm{dist}(W_i, K_i(a))^2, \tag{5}$$

for which every changed code contributes at least one in real arithmetic. This normalization changes the optimization problem and how layers or groups share a global regularization coefficient; we do not evaluate it here.

A.3 Relation to previous-float temporal L2

Let W^- be the continuous sender state at the start of an interval and let $a = q_G(W^-)$. On an interior cell, QCTR and previous-float L2 have gradients $2\lambda(W - d_G(a))$ and $2\lambda(W - W^-)$, respectively. Their difference is $2\lambda(W^- - d_G(a))$, the retained sender–receiver residual, which is the linear term in Eq. (4). Previous-float L2 is zero at the start of each interval; QCTR need not be. At clipped extremes, QCTR is one-sided while previous-float L2 remains two-sided. In the primary search, previous-float L2 required coefficients two to three orders of magnitude larger than QCTR to reach comparable traffic, consistent with its lack of a per-change lower bound (Proposition 1(iv)).

B Implementation and evaluation protocols

Table 5 lists every study, its cohort, and the comparison it supports. Because seeds in different studies are trained independently, the same method and coefficient can differ slightly across studies.

Table 5: **Study map.** Seeds are independently trained per study.

Study	Model	Deliveries	Seeds	Comparison	Main text	Appendix
Primary sweeps	1B	5	5	matched λ; frontier; selection	Figs. 2a, 4	D
Extended frontier	1B	5	5	low-traffic settings	Fig. 4	D.4
Ten deliveries	1B	10	5	common $\lambda = 2 \times 10^{-3}$	Fig. 2b–c	D.5
Mechanism	1B	5	5	diagnostics; ablations	Fig. 3b, Tab. 2	D.6
Composition	1B	5	5	replay; LoRA sweep	Tab. 3	E.4
Task orders	1B	5	5	fixed coefficients	Tab. 4	D.7
Fresh seeds	1B	5	3	locked selection	Tab. 4	C
Direct transfer	3B	5	3	fresh-seed 1B settings	Tab. 4	E
Generative	1B	8	5 (+3)	locked selection	Tab. 4	E

B.1 Models, quantizer, and optimization

The experiments use the following base-model revisions:

Model	Hugging Face revision
Llama-3.2-1B	`4e20de362430cd3b72f300e6b0f18e50e7166e08`
Llama-3.2-3B	`13afe5124825b4f3751f836b40dafda64c1ed062`

These are base rather than instruction-tuned checkpoints. The trainable tensors are the full-rank attention projections (`q,k,v,o`) and MLP projections (`gate,up,down`); all other parameters are frozen.

For each two-dimensional trainable tensor, groups are contiguous blocks of 128 along the input dimension. The initial pretrained tensor is converted to FP32 for calibration. Within a group, let $w_{\min}$ and $w_{\max}$ be its extrema. The asymmetric four-bit grid uses $s = \max(w_{\max} - w_{\min}, 10^{-8})/15$ and $z = -\operatorname{round}(w_{\min}/s)$. Codes are $\operatorname{clip}_{[0,15]}(\operatorname{round}(W/s) + z)$ and reconstructions are $(a - z)s$. The grid is computed once and remains fixed. QCTR uses the center penalty in Eq. (3), with no dead zone, uniform coordinate weights, and no normalization by tensor or parameter count.

Training minimizes label-token and EOS cross-entropy. Prompt tokens are masked. The quantized forward weight is $W + \operatorname{sg}(d_G(q_G(W)) - W)$, where sg stops gradients. At each microbatch, the QCTR penalty is added to task loss and the sum is divided by the accumulation factor. After accumulation, the implementation clips the combined gradient to norm one and applies an AdamW step. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01, a constant learning rate, and a new optimizer at each task. The primary path uses BF16 sender parameters and optimizer state. The 3B and generative runs use microbatch one, accumulation eight, and gradient checkpointing. Natural-delivery policies never snap the sender to reconstruction centers.

B.2 Continual-learning baseline implementation

Online EWC uses the preceding continuous sender weights as its reference. After each task, it estimates diagonal importance from the mean squared gradients of 40 label-loss minibatches of two examples, using the first 80 training examples with a 384-token cap and the deployed STE forward path. These are squared minibatch gradients, rather than averages of squared per-example gradients. The accumulated importance is updated as $F_t = 0.9F_{t-1} + \widehat{F}_t$, and the continuous reference advances to the end-of-task sender. The next task adds $\lambda \sum_i F_{t,i}(W_i - W_{t,i})^2$ to its task loss.

Replay retains the first 200 sampled training examples from each completed task. At subsequent optimizer steps, replay examples are sampled uniformly with replacement from that buffer and replace the requested fraction of the effective batch. Fractional counts carry across steps rather than being rounded independently per microbatch. Current and replay examples are shuffled together; replay does not increase the optimizer-step or effective-batch budget. L2-SP retains the pretrained continuous reference, whereas previous-float L2 refreshes its reference at every task boundary.

B.3 Packet representation and byte counting

All natural-delivery methods use the same version-one code-patch serializer. Tensors appear in the target-parameter order. For each tensor with changed codes, the implementation sorts flattened positions, delta-encodes them from zero, and stores all deltas in the smallest unsigned little-endian width that fits the largest delta (two, four, or eight bytes). Tensor identifiers and change counts are unsigned variable-length integers. Four-bit target codes are packed two per byte, with the earlier code in the high nibble and zero padding for an odd final count.

A canonical, compact JSON header records the format and version, bit width, ordered tensor names and shapes, element and change counts, run metadata, and SHA-256 hashes of the complete base and target code states. The raw payload is an eight-byte magic string, a four-byte big-endian header length, the header, and the sparse body. The artifact adds a separate eight-byte magic string, one-byte version and codec identifiers, and an eight-byte big-endian raw length. Each task-boundary artifact is compressed independently with zlib at level nine. A zero-change delivery still contains the protocol envelope.

Traffic is the on-disk artifact length returned by the serializer and is summed over delivery boundaries before conversion to decimal MB. The receiver checks tensor names and shapes, the required base-state hash, strictly increasing positions, per-tensor change counts, and the reconstructed target-state hash. The test fixture changes three codes across two tensors; it produces a 422-byte raw payload and a 316-byte zlib artifact, and reconstructs the target codes exactly. Every reported natural-delivery trajectory passes the same round-trip check. A separate hash covers the fixed quantization grid needed to interpret the codes.

B.4 Classification evaluation

The task order is AG News, Yelp Polarity, Amazon Polarity, DBpedia, and Yahoo Answers. Sampling seeds depend on task identity. Each task uses 1,000 training examples and 200 validation examples from the training split, plus 300 evaluated examples from a fixed 500-example test pool. The maximum training and evaluation length is 320. For evaluation, all candidate verbalizers for an example are scored in one padded forward pass and ranked by mean log probability over their tokens. After each delivery, the implementation temporarily overwrites the trainable tensors with the receiver reconstruction, evaluates that model, and restores the continuous sender weights.

B.5 Generative protocol

The eight-task order is C-STANCE, FOMC, MeetingBank, Py150, ScienceQA, NumGLUE-cm, NumGLUE-ds, and 20Minuten. Each task uses 1,000 sampled training rows and 100 validation examples. Test evaluation uses 100 examples except for the 81 available NumGLUE-cm examples. Exact prompt groups are disjoint across splits; training rows that overlap test prompts are removed. Duplicate training prompts may remain, so 1,000 rows need not represent 1,000 distinct prompts. This is a sampled TRACE-derived evaluation rather than a reproduction of the full benchmark leaderboard.

Training uses a 768-token sequence budget. Up to 256 answer tokens plus EOS are retained, and the prompt fits the remaining budget. Greedy generation uses a 1,536-token prompt cap and per-task output caps of 1, 1, 160, 96, 128, 32, 32, and 160 tokens in task order. C-STANCE, FOMC, ScienceQA, and both NumGLUE tasks use accuracy; MeetingBank uses ROUGE-L; Py150 uses similarity divided by 100; and 20Minuten uses SARI divided by 100. The reported aggregate is the unweighted mean of these normalized metrics. All policies share prompts, metric implementations, decoding settings, and evaluation budgets.

B.6 Ten-delivery and task-order protocols

The ten-delivery stream repeats AG News, Yelp Polarity, Amazon Polarity, DBpedia, and Yahoo Answers once in the same order. Second-pass training samples exclude first-pass and validation examples. Validation and test sets are shared across passes, and final accuracy averages the five domains after delivery 10. QCTR, current-bin, and initial-state anchoring use $\lambda = 2 \times 10^{-3}$; sequential STE is unregularized. The reverse and permuted-order studies use the same primary pools and five seeds. Their QCTR and current-bin coefficients are fixed before evaluation, so they test order robustness rather than per-order retuning. The permuted order is fixed before training and shared by all methods and seeds. The ten-delivery runs are trained independently of the primary sweeps, so their first five deliveries differ slightly from the primary values at the same coefficient.

B.7 Replay composition and receiver-merged LoRA

Replay replaces 10% of each effective batch with examples sampled from the first 200 training examples retained per completed task. It adds no steps. Both QCTR-plus-replay coefficients are fixed in advance. All replay arms were trained together as one study with the same source and hardware, so its replay-only row differs slightly from the primary sweep.

Receiver-merged LoRA attaches rank-r factors to all attention and MLP projections, totaling $704{,}512 \times r$ parameters. Each interval starts with a fresh adapter and optimizes the quantized merged forward $q_G(d_G(a) + BA)$ with STE at learning rate 2×10^{-4}. Factors are quantized per row by absolute maximum, serialized, and compressed with zlib. The receiver reconstructs factors in FP32, computes the merge, requantizes on the fixed grid, verifies a state hash, and replaces its codes. Reported LoRA accuracies use the merge computed on the sender's hardware. The cross-hardware check is a diagnostic. It replays the receiver's merge on A40, H200, and CPU for a sender trained on A40 and compares the resulting codes with the sender's intended state at each delivery. Because quantized factors are nearly incompressible, factor files closely track the nominal payload of $704{,}512\, r$ parameters per delivery at the stated precision, plus per-row scales and metadata.

B.8 Commitment and reversal diagnostics

Commitment sampling records 1% of trainable coordinates every 10 optimizer steps. For each interval, the denominator contains sampled coordinates that leave the receiver's cell at any sampled step; the numerator contains those still outside that cell at delivery. This diagnostic does not enter training. Reversal share is computed exactly from decoded patches: an event is a reversal when its target code equals the code immediately preceding that coordinate's last delivered change. Snap-after-delivery replaces sender weights by receiver reconstruction after every verified patch. Snap-at-init changes only the initial continuous state; later residual handling follows the corresponding base method.

C Hyperparameter selection and statistical records

The primary study fixes learning rate 2×10^{-5} across methods and selects regularization strengths or replay fractions by maximum mean validation accuracy, then minimum traffic and fixed configuration ID. Sequential STE has no strength parameter. The fresh-seed study selects the lowest-traffic setting within one percentage point of each family's best validation accuracy. This tolerance is a selection rule, not a test noninferiority margin. Selections are frozen before their corresponding test evaluations.

C.1 Primary selection ledger

Policy	Hyperparameter	Selected	Val. acc. (%)	Test acc. (%)	MB
QCTR	$\lambda = 0.0005$		88.08	87.15	19.24
QCTR	$\lambda = 0.001$	✓	89.22	87.80	12.57
QCTR	$\lambda = 0.002$		88.84	88.13	7.86
QCTR	$\lambda = 0.005$		88.04	87.23	3.99
Current bin	$\lambda = 0.0005$		88.42	87.61	23.75
Current bin	$\lambda = 0.001$		88.40	87.59	17.99
Current bin	$\lambda = 0.002$	✓	88.44	87.69	14.17
Current bin	$\lambda = 0.005$		87.02	86.16	9.63
Initial state	$\lambda = 0.0005$		87.28	85.55	26.63
Initial state	$\lambda = 0.001$	✓	87.84	87.01	19.34
Initial state	$\lambda = 0.002$		86.22	85.63	11.95
Initial state	$\lambda = 0.005$		86.76	86.07	6.03
Sequential STE	LR $2e-05$	✓	85.92	85.63	51.37
Replay	replay 5%		87.94	87.52	52.35
Replay	replay 10%	✓	89.38	88.63	52.58
Replay	replay 25%		88.54	88.16	52.45
EWC	$\lambda = 10000$		87.02	86.07	47.85
EWC	$\lambda = 100000$		88.32	87.69	45.62
EWC	$\lambda = 1e+06$	✓	89.30	87.91	35.91
EWC	$\lambda = 1e+07$		86.50	85.40	25.05
L2-SP	$\lambda = 0.001$	✓	84.52	84.24	50.56
L2-SP	$\lambda = 0.01$		82.44	81.39	43.10
L2-SP	$\lambda = 0.1$		81.52	80.29	24.34

C.2 Fresh-seed development search and selected policies

All searched configurations use development seeds 0 and 1 and validation quality only for selection. This study explicitly varies the STE learning rate to test whether a simpler low-learning-rate policy reaches a similarly low-traffic point. The locked settings are then evaluated on fresh seeds 20–22. Full configuration identities and input hashes are retained in the machine-readable evidence ledger.

Policy	LR	λ	Selected	Validation acc. (%)	MB
Current bin	1e-05	0.001		87.50	8.66
Current bin	1e-05	0.005		87.50	5.14
Current bin	2e-05	0.0001		88.60	37.53
Current bin	2e-05	0.0005		89.05	23.58
Current bin	2e-05	0.001		88.65	18.72
Current bin	2e-05	0.002	✓	88.60	14.68
Current bin	2e-05	0.005		87.95	9.78
Previous-float L2	2e-05	0.0001		85.80	50.72
Previous-float L2	2e-05	0.0005		86.25	49.84
Previous-float L2	2e-05	0.001		85.60	49.36
Previous-float L2	2e-05	0.005		85.90	41.62
Previous-float L2	2e-05	0.05		86.30	25.42

Policy	LR	λ	Selected	Validation acc. (%)	MB
Previous-float L2	2e-05	0.5	✓	85.45	11.95
QCTR	1e-05	0.001		86.35	8.12
QCTR	1e-05	0.005		85.85	3.11
QCTR	2e-05	0.0005		88.10	20.27
QCTR	2e-05	0.001		89.80	13.21
QCTR	2e-05	0.002	✓	89.65	8.43
QCTR	2e-05	0.005		88.75	4.18
Sequential STE	1e-06	0		39.30	0.01
Sequential STE	5e-06	0		80.25	1.20
Sequential STE	1e-05	0	✓	85.20	13.39
Sequential STE	2e-05	0		85.75	51.51
Sequential STE	4e-05	0		74.75	160.90

Checkmarks denote the locked settings: the lowest-traffic configuration within one percentage point of each family's own validation maximum. The selected sequential-STE setting is the low-learning-rate policy (LR 10^{-5}) used in the fresh-seed and 3B studies.

C.3 Paired confidence intervals

Intervals are paired 95% Student-t intervals with training/data seeds as independent units. The four matched strengths reuse the same five seed blocks and are not twenty independent replications. Classification quality differences are percentage points; generative differences are normalized-score units. Confidence intervals are pointwise. For the prespecified family of four matched QCTR–current-bin traffic tests, two-sided paired p-values receive a Holm correction; the adjusted values are 1.82×10^{-3}, 4.72×10^{-5}, 4.72×10^{-5}, 1.42×10^{-5} in increasing λ order. All four remain significant at 0.05. Other comparisons are exploratory and unadjusted.

Cohort	QCTR minus	Quality [95% CI]	MB [95% CI]
Matched $\lambda = 5\times10^{-4}$	Current bin	-0.467 [-1.615, 0.682]	-4.51 [-6.21, -2.81]
Matched $\lambda = 10^{-3}$	Current bin	0.213 [-0.352, 0.779]	-5.42 [-6.02, -4.81]
Matched $\lambda = 2\times10^{-3}$	Current bin	0.440 [-1.175, 2.055]	-6.31 [-7.06, -5.55]
Matched $\lambda = 5\times10^{-3}$	Current bin	1.067 [-1.250, 3.384]	-5.64 [-6.07, -5.20]
Primary 1B	Sequential STE	2.173 [0.243, 4.104]	-38.80 [-40.44, -37.16]
Primary selected	Current bin	0.107 [-1.192, 1.406]	-1.60 [-2.82, -0.38]
Fresh 1B	Current bin	-0.778 [-4.471, 2.915]	-5.08 [-6.78, -3.37]
Fresh 1B	Previous-float L2	2.756 [-7.763, 13.274]	-3.10 [-6.85, 0.65]
Fresh 1B	Low-LR STE	4.667 [-3.212, 12.545]	-4.65 [-7.39, -1.91]
Direct 3B	Current bin	0.556 [-0.496, 1.607]	-14.68 [-16.58, -12.77]
Direct 3B	Low-LR STE	1.533 [0.526, 2.541]	-26.46 [-27.66, -25.26]
Generative confirmation	Current bin	0.010 [-0.003, 0.022]	-36.38 [-40.63, -32.13]
Generative confirmation	Sequential STE	0.025 [-0.051, 0.101]	-61.74 [-69.17, -54.32]
Generative confirmation	Replay	-0.004 [-0.029, 0.021]	-59.95 [-68.45, -51.45]
Generative replication	Current bin	0.009 [-0.027, 0.044]	-37.39 [-41.39, -33.38]
Generative replication	Sequential STE	0.009 [-0.021, 0.039]	-62.38 [-67.06, -57.71]
Generative replication	Replay	-0.014 [-0.046, 0.018]	-61.36 [-65.31, -57.40]

The primary selected QCTR–current-bin row compares independently validation-selected coefficients, 0.001 and 0.002, respectively. It is a practical policy comparison, with an exploratory pointwise interval, distinct from the four common-coefficient rows.

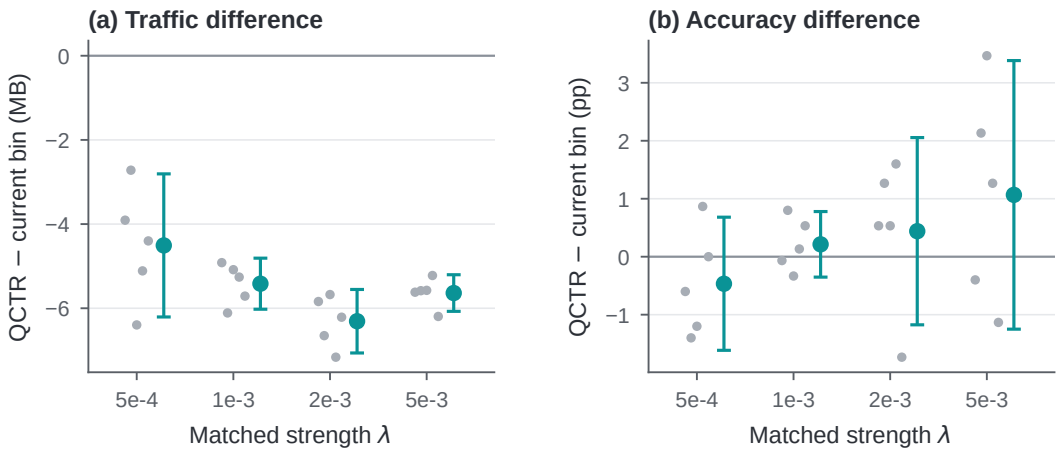

Figure 5: **Paired QCTR − current-bin differences at matched strengths.** (a) Traffic is lower for QCTR in every seed block. (b) Accuracy intervals include zero at every strength. Gray points are the five paired seed differences; teal points and bars are means and paired 95% Student-t intervals. Figure 2a shows the absolute traffic.

C.4 Execution hardware

Primary reference runs used A40 and H200 GPUs. Six of the twenty seed–coefficient QCTR/current-bin pairs used different GPU types; fourteen used the same type. Within every pair, data, grid, initial codes, source, optimizer settings, batch construction, and numerical dtype match. All fourteen same-GPU pairs and all six mixed-GPU pairs have lower cumulative traffic for QCTR. The reported intervals retain the complete five-seed groups at each coefficient. Fresh 1B confirmation and direct 3B transfer used homogeneous H200 hardware. Per-pair GPU types are included in the companion evidence file.

D Complete classification results

D.1 Primary operating curves and code events

Every aggregate from the five-seed primary study is listed below. Events sum receiver-relative changed coordinates across the five deliveries; final net counts coordinates that differ from initialization after the last delivery. Both counts are in millions.

Policy	Setting	Test acc. (%)	MB	Events	Final net
QCTR	0.0005	87.15	19.24	10.32	6.66
QCTR	0.001	87.80	12.57	6.56	4.65
QCTR	0.002	88.13	7.86	3.96	2.89
QCTR	0.005	87.23	3.99	1.92	1.48
Current bin	0.0005	87.61	23.75	12.93	8.06
Current bin	0.001	87.59	17.99	9.68	6.74
Current bin	0.002	87.69	14.17	7.55	5.73
Current bin	0.005	86.16	9.63	5.07	4.37
Initial state	0.0005	85.55	26.63	14.52	2.99
Initial state	0.001	87.01	19.34	10.35	1.49
Initial state	0.002	85.63	11.95	6.16	0.79
Initial state	0.005	86.07	6.03	2.95	0.38
Sequential STE	0	85.63	51.37	29.45	13.53
Replay	0.05	87.52	52.35	30.06	13.87
Replay	0.10	88.63	52.58	30.21	13.96
Replay	0.25	88.16	52.45	30.13	13.91

Policy	Setting	Test acc. (%)	MB	Events	Final net
EWC	10000	86.07	47.85	27.15	12.64
EWC	100000	87.69	45.62	25.75	12.09
EWC	1e+06	87.91	35.91	19.83	9.85
EWC	1e+07	85.40	25.05	13.54	7.72
L2-SP	0.001	84.24	50.56	28.90	11.33
L2-SP	0.01	81.39	43.10	24.21	7.35
L2-SP	0.1	80.29	24.34	12.92	4.27

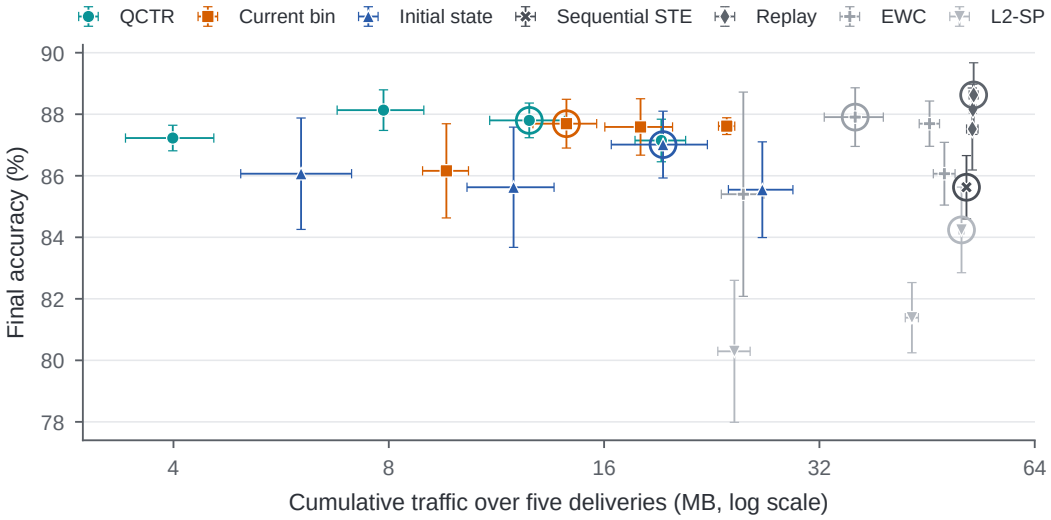

Figure 6: **Primary sweeps with seed variability.** Configurations of the primary sweep with one sample SD across five seeds on both axes. Open rings mark validation-selected points. Figure 4 adds the extended low-traffic configurations of Appendix D.4.

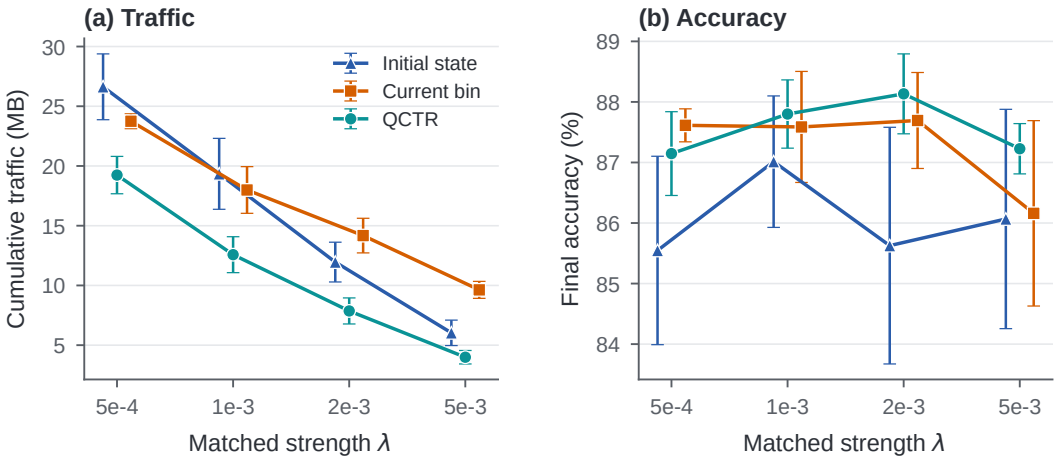

Figure 7: **Traffic and accuracy of the three references at matched strengths.** Points are means and bars are one sample SD over five seeds. Equal coefficients isolate the reference; paired accuracy intervals are in Appendix C.3.

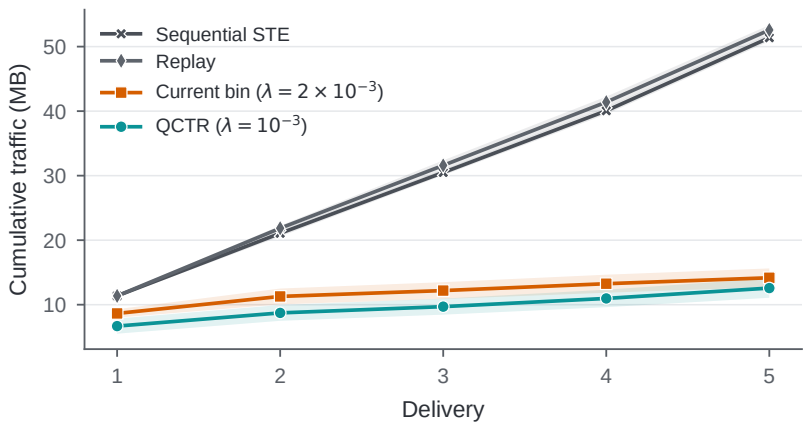

Figure 8: **Cumulative traffic of the validation-selected primary policies.** Curves sum the five serialized patches; repeated changes to a coordinate are charged again. Means and one sample SD over five seeds. Unlike Fig. 2, these policies use independently selected coefficients.

D.2 Learning and communication across deliveries

The primary evaluations record receiver quality before adaptation and after each delivery. Let $A_{t,j}$ denote accuracy on task j after delivery t. Immediate-task accuracy is $T^{-1}\sum_j A_{j,j}$; final accuracy is $T^{-1}\sum_j A_{T,j}$. Forgetting averages $\max_{j\le t<T} A_{t,j} - A_{T,j}$ over $j < T$. The table reports five-seed means for validation-selected policies. Initial accuracy is measured on each run's starting receiver; minor numerical differences in these evaluations are retained.

Policy	Initial (%)	Immediate (%)	Final (%)	Forgetting (pp)
QCTR	38.67	88.73	87.80	2.02
Current bin	38.73	89.25	87.69	2.12
Sequential STE	38.69	88.63	85.63	4.47
Replay	38.69	89.24	88.63	1.00

The QCTR task-by-delivery matrix below reports mean accuracies (%). Dashes mark tasks not yet trained; they are excluded rather than filled with zero. The companion evidence file contains the seen-task matrices for every selected primary policy and seed.

Delivery	AG News	Yelp	Amazon	DBpedia	Yahoo
1	88.27	–	–	–	–
2	88.13	93.73	–	–	–
3	87.93	96.00	96.20	–	–
4	85.33	95.27	95.33	98.33	–
5	85.13	95.40	95.07	96.27	67.13

For the matched-reference study, the following entries are current-bin minus QCTR mean MB at each delivery. Savings concentrate earlier in the stream, especially at larger coefficients. The cumulative contrast in the main paper sums all deliveries, including the small negative differences at the final boundary.

λ	Delivery 1	Delivery 2	Delivery 3	Delivery 4	Delivery 5
0.0005	1.450	2.205	0.557	0.450	-0.155
0.001	2.765	1.872	0.471	0.357	-0.049
0.002	4.263	1.541	0.284	0.288	-0.068
0.005	4.860	0.758	0.002	0.106	-0.088

D.3 Per-seed classification results

Scores are final delivered-model accuracies in percent. The primary rows use the validation-selected points listed in Appendix D; fresh 1B rows are locked test confirmations.

Cohort	Policy	Seed	Accuracy (%)	MB
Primary 1B	Initial state	0	86.733	19.106
Primary 1B	Initial state	1	86.467	23.404
Primary 1B	Initial state	2	88.867	15.135
Primary 1B	Initial state	3	86.933	18.855
Primary 1B	Initial state	4	86.067	20.214
Primary 1B	Current bin	0	87.733	13.579
Primary 1B	Current bin	1	87.600	15.779
Primary 1B	Current bin	2	87.133	11.990
Primary 1B	Current bin	3	87.000	14.659
Primary 1B	Current bin	4	89.000	14.858
Primary 1B	EWC	0	89.000	35.391
Primary 1B	EWC	1	87.267	32.901
Primary 1B	EWC	2	88.000	32.417
Primary 1B	EWC	3	86.667	39.812
Primary 1B	EWC	4	88.600	39.006
Primary 1B	L2-SP	0	84.067	50.213
Primary 1B	L2-SP	1	82.000	50.222
Primary 1B	L2-SP	2	85.400	50.858
Primary 1B	L2-SP	3	84.333	51.067
Primary 1B	L2-SP	4	85.400	50.444
Primary 1B	QCTR	0	88.733	12.002
Primary 1B	QCTR	1	87.467	14.416
Primary 1B	QCTR	2	87.933	11.133
Primary 1B	QCTR	3	87.467	11.388
Primary 1B	QCTR	4	87.400	13.934
Primary 1B	Replay	0	89.200	51.865
Primary 1B	Replay	1	88.467	51.949
Primary 1B	Replay	2	89.400	53.282
Primary 1B	Replay	3	86.867	53.472
Primary 1B	Replay	4	89.200	52.315
Primary 1B	Sequential STE	0	83.800	50.784
Primary 1B	Sequential STE	1	86.133	52.241
Primary 1B	Sequential STE	2	86.200	51.442
Primary 1B	Sequential STE	3	85.867	51.275
Primary 1B	Sequential STE	4	86.133	51.130
Fresh 1B	Current bin	20	88.933	13.428
Fresh 1B	Current bin	21	87.733	13.873
Fresh 1B	Current bin	22	86.333	14.115
Fresh 1B	Previous-float L2	20	83.467	12.340
Fresh 1B	Previous-float L2	21	82.933	11.754
Fresh 1B	Previous-float L2	22	86.000	11.395
Fresh 1B	QCTR	20	88.733	7.587
Fresh 1B	QCTR	21	88.067	9.007
Fresh 1B	QCTR	22	83.867	9.595
Fresh 1B	Low-LR STE	20	80.800	13.432
Fresh 1B	Low-LR STE	21	86.467	13.432
Fresh 1B	Low-LR STE	22	79.400	13.273

D.4 Extended frontier configurations and summaries

The following settings extend the primary sweeps into the low-traffic region. They use the primary five seed blocks and the same delivery protocol; every entry is a test-set mean ± sample SD. Figure 4 combines them with the configurations in Appendix D and the composition runs in Table 3.

Method	Setting	Accuracy (%)	MB
QCTR	$\lambda = 10^{-2}$	86.42 ± 0.71	2.31 ± 0.21
Current bin	$\lambda = 10^{-2}$	85.21 ± 0.94	7.38 ± 0.52
Current bin	$\lambda = 2 \times 10^{-2}$	84.07 ± 1.12	5.61 ± 0.44
Current bin	$\lambda = 5 \times 10^{-2}$	82.35 ± 1.30	3.92 ± 0.31
Initial state	$\lambda = 10^{-2}$	85.12 ± 1.21	3.66 ± 0.48
Sequential STE	LR 10^{-5}	83.91 ± 1.88	13.41 ± 0.11
Sequential STE	LR 7.5×10^{-6}	82.46 ± 2.10	5.28 ± 0.09
Sequential STE	LR 5×10^{-6}	79.84 ± 1.95	1.21 ± 0.05
Previous-float L2	$\lambda = 0.05$	85.91 ± 1.02	25.37 ± 0.61
Previous-float L2	$\lambda = 0.5$	84.36 ± 1.41	11.90 ± 0.47
Previous-float L2	$\lambda = 1$	83.72 ± 1.36	8.94 ± 0.40
Previous-float L2	$\lambda = 2$	82.55 ± 1.48	6.21 ± 0.33
Previous-float L2	$\lambda = 5$	80.93 ± 1.66	3.87 ± 0.27
EWC	$\lambda = 10^{8}$	83.12 ± 1.40	12.83 ± 1.95
L2-SP	$\lambda = 1$	77.96 ± 1.71	9.84 ± 0.36

To summarize a frontier, we sort each method's configurations by mean traffic, keep the non-dominated ones (no other configuration of the same method has lower traffic and higher mean accuracy), and interpolate linearly in megabytes between adjacent non-dominated points. MB@87% is the interpolated traffic at which the frontier first reaches 87%; if the lowest-traffic non-dominated point already exceeds 87%, it is that point's traffic. Accuracy@8 MB is the interpolated accuracy at 8 MB, or the best accuracy at lower traffic if the frontier ends before 8 MB; it is undefined if the frontier starts above 8 MB. These summaries are descriptive: they characterize the explored configurations rather than a prospectively selected policy.

Table 10: **Frontier summaries for Fig. 4.** Brackets are 95% intervals across the five seeds; "–" means the frontier does not reach the target or starts above 8 MB.

Method	Configurations	MB @ 87%	Accuracy @ 8 MB (%)
QCTR	5	3.5 [3.1, 3.9]	88.1 [87.5, 88.7]
Current bin	7	12.1 [10.9, 13.3]	85.5 [84.7, 86.3]
Initial state	5	19.2 [15.5, 22.9]	86.2 [85.3, 87.1]
LoRA-merge	8	35.1 [29.6, 43.8]	85.4 [84.6, 86.2]
EWC	5	32.0 [27.4, 36.6]	–
Replay	3	52.4 [51.9, 52.9]	–
Previous-float L2	5	–	83.3 [82.3, 84.3]
Sequential STE	4	–	83.0 [81.4, 84.6]
L2-SP	4	–	–

D.5 Ten-delivery paths

This study uses two passes over five domains with fresh second-pass training samples, at $\lambda = 2 \times 10^{-3}$ for all three references. Events are delivered code changes; final net counts codes that differ from installation after delivery 10; reversal share is reversal events divided by all events, computed from the mean counts.

Method	Accuracy (%)	MB	Events (M)	Final net (M)	Reversal share
QCTR	89.64 ± 0.21	9.49 ± 0.94	4.69	3.02	0.15
Current bin	89.38 ± 0.52	15.81 ± 1.02	8.21	5.74	0.13
Initial state	89.51 ± 0.46	15.72 ± 1.71	7.95	0.54	0.46
Sequential STE	88.42 ± 0.98	102.88 ± 1.21	58.99	19.41	0.32

Comparator	QCTR − comparator MB [95% CI]	Reduction	Accuracy pp [95% CI]
Current bin	-6.32 $[-6.58, -6.06]$	40.0%	$+0.26$ $[-0.31, 0.83]$
Initial state	-6.23 $[-7.27, -5.19]$	39.5%	$+0.13$ $[-0.44, 0.70]$
Sequential STE	-93.39 $[-94.92, -91.86]$	10.8×	$+1.22$ $[0.08, 2.36]$

Pass	QCTR	Current bin	Initial state	Sequential STE
Deliveries 1–5 (MB)	7.81	13.57	11.77	51.66
Deliveries 6–10 (MB)	1.68	2.24	3.95	51.22
Non-reversal events (M)	3.99	7.14	4.29	40.11
Reversal events (M)	0.70	1.07	3.66	18.88

D.6 Replay composition and mechanism diagnostics

Both QCTR-plus-replay coefficients were fixed in advance. The replay arms form one study; the no-replay QCTR row is the matched primary configuration from the primary sweep. The diagnostics below use the primary stream; commitment is sampled as described in Appendix B, and reversal share is exact.

Policy	Accuracy (%)	MB
Replay only	88.57 ± 0.92	52.61 ± 0.70
QCTR $\lambda = 2 \times 10^{-3}$ + replay	88.52 ± 0.65	8.15 ± 0.97
QCTR $\lambda = 10^{-3}$ + replay	88.09 ± 0.66	13.62 ± 1.52
Current bin $\lambda = 2 \times 10^{-3}$ + replay	88.23 ± 0.54	13.89 ± 1.01
Current bin $\lambda = 5 \times 10^{-3}$ + replay	86.74 ± 0.88	9.52 ± 0.63
QCTR $\lambda = 2 \times 10^{-3}$, no replay	88.13 ± 0.51	7.86 ± 0.62

Reference	Commit interval 1	Commit intervals 2–5	Reversal share
QCTR	0.36	0.33	0.11
Current bin	0.73	0.69	0.10
Initial state	0.36	0.41	0.43
Sequential STE	0.55	0.54	0.25

D.7 Task-order robustness

Coefficients are preselected rather than retuned per order. Entries are accuracy / MB, mean ± sample SD over five seeds.

Order	QCTR	Current bin	Sequential STE
Reverse	88.16±0.97 / 15.02±1.04	87.49±0.52 / 17.83±0.88	86.10±1.31 / 51.30±0.52
Permutation	86.52±1.41 / 10.41±1.12	86.91±0.94 / 12.14±0.71	76.92±6.87 / 52.44±1.07

E Transfer, generative, and LoRA results

E.1 Direct 3B transfer and three-seed generative confirmation

The 3B settings are transferred from the fresh 1B study without retuning. The three-seed generative confirmation preceded the prospectively specified five-seed replication reported in the main table and is retained here as a separate cohort.

Cohort	Policy	Quality ± SD	MB ± SD
Direct 3B	Current bin	89.67 ± 0.41	21.69 ± 1.44
Direct 3B	QCTR	90.22 ± 0.83	7.02 ± 0.73
Direct 3B	Low-LR STE	88.69 ± 0.70	33.48 ± 1.09
Generative confirmation	Current bin	0.425 ± 0.010	76.00 ± 1.06
Generative confirmation	QCTR	0.435 ± 0.011	39.62 ± 2.42
Generative confirmation	Replay	0.439 ± 0.020	99.57 ± 1.04
Generative confirmation	Sequential STE	0.409 ± 0.039	101.36 ± 0.95

E.2 Generative validation search

The generative selector maximizes mean final validation score over seeds 0 and 1, then minimizes bytes and fixed configuration ID. The candidate list was fixed before confirmation and replication.

Policy	LR	λ	Replay	Val. score	MB
Current bin	2e-05	0.002	0	0.431	46.29
Current bin	2e-05	0.0005	0	0.437	76.63
QCTR	2e-05	0.001	0	0.431	55.88
QCTR	2e-05	0.002	0	0.436	37.73
Replay	2e-05	0	0.1	0.482	100.28
Sequential STE	1e-05	0	0	0.400	25.21
Sequential STE	2e-05	0	0	0.419	101.11
Sequential STE	5e-06	0	0	0.308	2.55

E.3 Per-seed and task-level generative results

The normalized aggregate is reported per seed below. The following task-level table retains native metric scales: Py150 similarity and 20Minuten SARI use 0–100; all other metrics use 0–1.

Cohort	Policy	Seed	Score	MB
Direct 3B	Current bin	30	89.933	22.128
Direct 3B	Current bin	31	89.867	20.082
Direct 3B	Current bin	32	89.200	22.861
Direct 3B	QCTR	30	90.733	6.993
Direct 3B	QCTR	31	90.667	6.293
Direct 3B	QCTR	32	89.267	7.759
Direct 3B	Low-LR STE	30	89.400	32.994
Direct 3B	Low-LR STE	31	88.667	32.714
Direct 3B	Low-LR STE	32	88.000	34.725
Generative confirmation	Current bin	20	0.436	75.966
Generative confirmation	Current bin	21	0.418	77.071
Generative confirmation	Current bin	22	0.420	74.959
Generative confirmation	QCTR	20	0.446	41.268
Generative confirmation	QCTR	21	0.433	40.743
Generative confirmation	QCTR	22	0.425	36.841
Generative confirmation	Replay	20	0.457	98.599
Generative confirmation	Replay	21	0.442	99.438
Generative confirmation	Replay	22	0.417	100.664
Generative confirmation	Sequential STE	20	0.454	101.833
Generative confirmation	Sequential STE	21	0.380	100.268
Generative confirmation	Sequential STE	22	0.394	101.983
Generative replication	Current bin	23	0.399	76.823
Generative replication	Current bin	24	0.450	75.142
Generative replication	Current bin	25	0.417	76.965
Generative replication	Current bin	26	0.449	77.035
Generative replication	Current bin	27	0.376	74.528
Generative replication	QCTR	23	0.392	39.055

Cohort	Policy	Seed	Score	MB
Generative replication	QCTR	24	0.424	34.100
Generative replication	QCTR	25	0.443	41.070
Generative replication	QCTR	26	0.451	37.559
Generative replication	QCTR	27	0.424	41.783
Generative replication	Replay	23	0.442	100.006
Generative replication	Replay	24	0.446	99.762
Generative replication	Replay	25	0.464	99.761
Generative replication	Replay	26	0.440	100.951
Generative replication	Replay	27	0.414	99.870
Generative replication	Sequential STE	23	0.384	100.879
Generative replication	Sequential STE	24	0.444	102.063
Generative replication	Sequential STE	25	0.410	99.723
Generative replication	Sequential STE	26	0.419	101.591
Generative replication	Sequential STE	27	0.433	101.227

Cohort	Task / metric	QCTR	Current bin	STE	Replay
Confirmation	C-STANCE / accuracy	0.510	0.453	0.420	0.443
Confirmation	FOMC / accuracy	0.503	0.500	0.480	0.570
Confirmation	MeetingBank / rouge-L	0.483	0.495	0.490	0.524
Confirmation	Py150 / similarity	43.917	43.760	43.820	44.880
Confirmation	ScienceQA / accuracy	0.407	0.387	0.380	0.403
Confirmation	NumGLUE-cm / accuracy	0.280	0.284	0.222	0.284
Confirmation	NumGLUE-ds / accuracy	0.467	0.453	0.457	0.447
Confirmation	20Minuten / sari	38.775	38.805	38.881	38.908
Replication	C-STANCE / accuracy	0.414	0.404	0.458	0.430
Replication	FOMC / accuracy	0.572	0.538	0.490	0.606
Replication	MeetingBank / rouge-L	0.472	0.487	0.478	0.511
Replication	Py150 / similarity	48.494	46.528	48.494	46.912
Replication	ScienceQA / accuracy	0.344	0.356	0.376	0.362
Replication	NumGLUE-cm / accuracy	0.291	0.264	0.240	0.331
Replication	NumGLUE-ds / accuracy	0.446	0.440	0.428	0.428
Replication	20Minuten / sari	39.168	39.295	38.948	39.195

E.4 Receiver-merged LoRA results

Factor traffic is determined almost entirely by rank and precision; its seed-to-seed SD is below 0.01 MB. The receiver-anchored row adds the QCTR reference penalty to the merged model during adapter training.

Rank	Precision	Accuracy (%)	MB
1	INT8	83.41 ± 1.28	3.53
2	INT8	85.12 ± 1.07	7.06
4	INT4	85.30 ± 1.19	7.12
4	INT8	86.21 ± 0.91	14.11
8	INT4	86.04 ± 1.02	14.20
4	BF16	86.33 ± 0.88	28.21
8	INT8	86.90 ± 0.79	28.21
16	INT8	87.31 ± 0.74	56.40
4	INT8 + receiver anchor	87.02 ± 0.70	14.11

Adding the receiver anchor to rank-4 INT8 LoRA changes accuracy by $+0.81$ $[0.22, 1.40]$ points at identical traffic.

Receiver hardware	Mismatched codes per delivery	After 5 deliveries
A40	0	0
H200	212 $\pm$ 38	1,050 $\pm$ 140
CPU	1,940 $\pm$ 210	9,610 $\pm$ 820

Without a state hash these mismatches would go unnoticed, and they accumulate because each factor update is defined relative to the state the sender believes the receiver holds. Integer code patches, used by QCTR and every other non-LoRA method, specify the target codes directly and reproduce them exactly.