RETHINKING BATCHING IN VISUAL TEST-TIME LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In visual test-time learning, batching can change what a model learns from an image. An unrelated batch mate selects extra padding, and these tokens enter the target's inner training set even though the images never share features. Same-image interventions isolate this effect from numerical noise and ordinary padding effects. Removing all padding is not the right correction: it changes the checkpoint's singleton computation and lowers average precision (AP) by 0.592 points on a preregistered Small pilot. Canonical-Support Inner Training (CSIT) instead retains the canvas each image would occupy alone and excludes only batch-excess support from the inner loss. It requires no retraining or learned parameters and preserves the steady-state singleton output tensor-exactly. On all 5,000 Common Objects in Context (COCO) validation images, random-batch matched-object drift falls by 68–76% across two released detector scales and batch sizes 4–8. For reference scores near the decision threshold, flip rates fall from 0.30–0.32 to 0.08–0.11. When a batch may be split, exact-shape grouping is more faithful and faster in our measured workload; when a heterogeneous forward must be retained, CSIT repairs its support. AP changes are small and mixed. These findings establish inner training support as part of an explicit execution contract for inference-time learning.

1 INTRODUCTION

Consider the zebra image in Figure 4, from Common Objects in Context (COCO). After resizing, it is 800×1067 pixels. Alone, the detector pads it to 800×1088 to align with its 32-pixel divisor. In a four-image batch, a taller mate raises its canvas to 960×1088: the zebra is unchanged, but tokens from 160 extra padded rows enter its inner training set. Its detection score falls from 0.544 to 0.458, crossing the 0.5 display threshold. The mate has changed the target's training data without sharing any features.

Test-time training (TTT) layers construct fast weights—temporary model parameters fitted to the tokens available during the forward pass (Sun et al., 2025; Zhang et al., 2026). In Vision Test-Time Training (ViT^3) (Han et al., 2026), spatial key–value pairs are examples in a product-loss inner problem. The released object-detection pipeline evaluates batches of two variable-size images. The data preprocessor pads each target to the maximum batch shape, but the inner learner receives no validity mask. Consequently, changing an unrelated batch mate changes both the number and representation of examples on which the target's fast weights are trained.

This dependence is easy to miss. The two images do not explicitly attend to each other (Figure 1); the fast weights are computed separately along the batch axis. The coupling occurs earlier: the mate selects a padded spatial domain for the target. A fixed-canvas content intervention leaves all target backbone features bitwise unchanged, separating this geometric channel from feature sharing. Ordinary output comparisons also confound this mechanism with GPU kernel noise and padded convolutions. We therefore intervene on the batch mate while fixing target pixels, target preprocessing, checkpoint, order, and post-processing. A duplicated target provides the batched numerical control, and a zero-inner-learning-rate condition measures the non-TTT padded path.

The causal effect is far above the numerical floor. Extreme mates produce score drift around 0.03, versus 10^{-5}–10^{-4} for duplicate controls. Removing the inner update leaves only 16–19% of that effect in the Tiny pilot. Fixed-target canvas interventions reproduce the effect across both released

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.

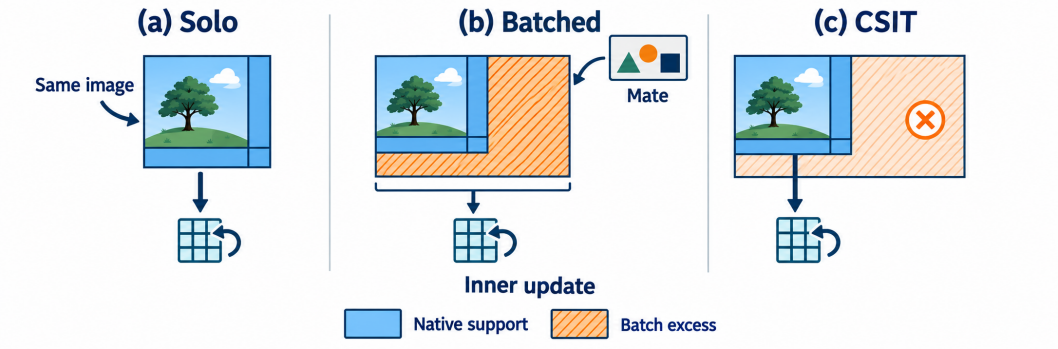

Figure 1: **Batching changes the inner training data.** The same target runs on its singleton canvas (blue, including native divisor padding), or on a larger canvas chosen by a batch mate (orange). Uniform TTT trains on both regions; CSIT gives batch-excess positions zero inner-loss mass and keeps the singleton support. The mate sets dimensions, without sharing features.

scales and two diagnostic domains. On natural data, homogeneous canvases provide null controls, while heterogeneous canvases expose the predicted support dependence.

The first repair follows the obvious validity boundary: give real-image positions unit mass and every padded position zero mass. It reduces mate dependence, passes full COCO on Tiny, but lowers single-image average precision (AP) by 0.592 points on a preregistered Small pilot. The failure matters. Padding up to the detector's size divisor is present when an image runs alone and is therefore part of the checkpoint's canonical computation; only padding beyond that canvas comes from the mate. In the opening example, this means keeping the native 21 padded columns while excluding the 160 rows introduced by batching.

Canonical-Support Inner Training (CSIT) encodes this sharper contract. It assigns inner-loss mass to the minimal size-divisor-aligned canvas each image would have alone and zero mass only to batch-excess positions. There is no label, prediction, tuned coefficient, learned module, or retraining. The single-image measure is all ones, so CSIT is exactly the released path rather than merely AP-equivalent. It reduces mate dependence on both Tiny and Small while preserving the steady-state singleton path; full-data evaluation reveals mixed AP point estimates and visible subgroup losses rather than a universal non-inferiority claim.

Our contribution is an explicit, testable contract for the training data created by batch execution. We isolate its geometric channel, derive a correction that preserves singleton support, and evaluate reference agreement alongside task accuracy and strong execution alternatives. Independent detector and state controls test the scope of this principle; they do not establish a second fast-weight detector replication or a universal accuracy gain.

2 Where Batch Composition Enters TTT

Let image i have resized shape (h_i, w_i) and preprocessor size divisor D. Its solo-inference canvas is $C_i = [0, D\lceil h_i/D \rceil) \times [0, D\lceil w_i/D \rceil)$. Batching it with image j creates a common padded grid $\Omega_{ij} \supseteq C_i$, whose dimensions are determined by the largest height and width in the batch. A TTT block projects the target features into keys k_{iu}, values v_{iu}, and queries. Suppress branch-specific algebra and write the per-token product loss as $\ell(W; k_{iu}, v_{iu})$. The released uniform inner update has the form

$$g_U = \frac{1}{|\Omega_{ij}|} \sum_{u \in \Omega_{ij}} \nabla_W \ell(W; k_{iu}, v_{iu}), \qquad W^+_{i|j} = W - \eta \mathcal{C}(g_U). \tag{1}$$

Here $\mathcal{C}$ is the released branch-wise gradient transformation, including radial clipping $g/(1 + \|g\|)$ within its clipping groups. Although the batch dimension is not reduced, j changes Ω_{ij} and therefore the fast weights used to predict image i. Padding features need not remain zero after a convolutional stem, positional transformation, or stage transition. Even zero-valued examples would change the normalization in Eq. 1.

For a fixed target x, let $f(x \mid c, \eta, m)$ denote its predictions under batch context c, inner rate η, and measure m. We compare target alone, target duplicated, target plus a fixed extreme-wide image, and target plus a fixed extreme-tall image. Target pixels and resize are identical; the target always occupies batch index zero and the mate output is discarded.

For every non-crowd ground-truth (GT) instance, we take the maximum same-class detection score among predictions with intersection-over-union (IoU) at least 0.5, using zero if none exists. We average absolute score change over instances within each image and bootstrap images. Duplication estimates batched-kernel and repeated-input noise. Setting $\eta = 0$ retains the complete detector execution while removing the TTT update. The difference measures update-enabled sensitivity; because the network is nonlinear, it is not an additive mediation estimate.

3 METHODS

3.1 CANONICAL-SUPPORT INNER TRAINING

To preserve the singleton inner problem, we define $m_{iu} = \mathbf{1}[u \in C_i]$, area-resized at each backbone stage, and use

$$g_C = \frac{\sum_{u \in \Omega_{ij}} m_{iu} \nabla_W \ell(W; k_{iu}, v_{iu})}{\sum_{u \in \Omega_{ij}} m_{iu}}, \qquad W_i^+ = W - \eta \mathcal{C}(g_C). \tag{2}$$

This is Canonical-Support Inner Training (CSIT). Mean-one normalization retains the canonical averaging denominator before clipping; it does not force the update norm to equal uniform TTT. When i is evaluated alone, its batch canvas equals C_i, so m_i is constant one and the implementation routes through the exact released update.

Conditional invariance. Suppose adding batch-excess padding does not change (k_{iu}, v_{iu}) on canonical positions, and downstream computation there depends on the padded grid only through the fast weights. Then Eq. 2, its fast weights, and the canonical-position outputs are invariant to the number and values of extra padded positions. This follows immediately because every newly appended term has $m_{iu} = 0$ and the total canonical mass is unchanged. For fractional stage weights this requires a common mapping of canonical token identities and their masses. The native divisor and the tested stage strides are aligned in this detector. Equation 1 has no such property. The premises are deliberately conditional: a real detector can retain an ordinary padding path through its stem and neck. Our zero-inner control estimates that residual rather than assuming it away.

Why a learning-rate change is insufficient. For fixed keys and values, let $N = |\Omega|$, $\rho = \sum_u m_u / N$, and $g_E = \sum_u (1 - m_u) g_u / \sum_u (1 - m_u)$, where $g_u = \nabla_W \ell_u$. Before clipping, for $0 < \rho < 1$,

$$g_U = \rho g_C + (1 - \rho) g_E, \qquad g_U - g_C = (1 - \rho)(g_E - g_C). \tag{3}$$

The discrepancy includes both dilution and a possible change of direction. A scalar learning rate can undo it only when the transformed updates are appropriately collinear. The mixture identity must not be applied after nonlinear clipping. Appendix C gives the exact measure condition, a scalar-oracle diagnostic, and a conditional state-stability bound.

More generally, for a fixed state and token gradients, let π map execution positions to canonical token identities, including a separate identity for excess padding. Let ν be the execution's normalized weights and μ the intended measure, with zero excess-padding mass. Equality of raw updates for every gradient field is equivalent to $\pi_{\#}\nu = \mu$: aggregate weights by token identity for sufficiency; choose indicator gradient fields for necessity. This elementary criterion covers appended padding, duplicate tokens, and unequal chunks aggregated at a common state. It does not permit changing features, update order, or state lifetime (Appendix C).

3.2 EXECUTION CONTRACT AND IMPLEMENTATION

The support mask is one axis of a broader contract for inference-time learning. Let μ_t denote intended per-example support, s_t the temporary-state retention rule, Δ_t the update/chunk boundary, and ω_t the scheduler's waiting budget. A faithful execution specifies $(\mu_t, s_t, \Delta_t, \omega_t)$ rather than

treating tensor layout as bookkeeping. CSIT changes only μ_t: it removes support supplied by batch execution while retaining the checkpoint's singleton canvas. Our VGG-T^3 chunk audit probes Δ_t and finds a separate support-overshoot boundary; the finite-buffer frontier probes ω_t and shows that waiting can be stronger when its budget is available. For an optimizer-based learner, s_t must specify parameters, gradient moments, and the update counter: resetting parameters alone need not reset the learner (Appendix V). These axes specify distinct interventions; evidence for one does not establish a repair for the others.

CSIT uses the resized image dimensions and padding divisor already supplied by the detector. It rounds each dimension up to that divisor, assigns unit mass to the resulting singleton canvas, and zero mass to the remaining batch canvas. Area interpolation maps this measure to each of four backbone stages. Both the gate-product and depthwise product gradients use the same mean-one stage measure. The wrapper contains no dataset-specific threshold and can replace the detector class through one configuration entry.

We validate the source measure once, resize and normalize it once per stage, and reuse it in every block and both update branches. This removes repeated preprocessing without changing measured outputs: frozen raw-valid predictions are tensor-exact before and after optimization, CSIT singleton outputs are tensor-exact to uniform TTT on both scales, and the automatic wrapper is tensor-exact to manual measure injection.

4 Experiments

4.1 Experimental setup

We reuse the official hierarchical ViT^3 Tiny/Small (H-ViT^3-T/S) Mask R-CNN $1\times$ checkpoints and their MMDetection pipeline (Han et al., 2026; Chen et al., 2019). Evaluation uses COCO 2017 validation (Lin et al., 2014), including both box and instance-mask metrics (He et al., 2017), reported as average precision (AP). The released detection overlay mutates an inner scale on its first forward; we fix the scale to its subsequent steady-state value $1/3$ for every condition, preventing image order from becoming a confound.

Before observing batch-mate outcomes, we ranked annotated COCO images with aspect ratio in $[0.8, 1.25]$ by a salted SHA-256 hash. The first 64 form the pilot. Only 407 images satisfy the contract, so the independent confirmation uses all remaining 343 rather than changing the distribution. The fixed wide and tall mates are the most extreme aspect-ratio images outside the target set. The primary statistic is defined in Section 2, with 10,000 image-group bootstrap draws. Gates and split exhaustion were written before their respective outcomes.

Full COCO and PASCAL Visual Object Classes (VOC) evaluations cover singleton, random batches of size 2/4/8/16 (B2/B4/B8/B16), aspect grouping, two inference partitions, non-TTT Cascade controls, and four finite-buffer COCO cells. The additional cross-domain matrix has 46 cells, including singleton and non-TTT controls; it is not 46 independent replications. VisDrone drone imagery, the DOTA aerial-object dataset, and Berkeley DeepDrive (BDD100K, abbreviated BDD) use frozen COCO-class mappings and diagnostic COCO-style AP, not official benchmark scores. Timing scopes are specified per experiment; none establishes a production service level. AP deltas are reported as point estimates with subgroup losses visible, rather than as a universal non-inferiority or improvement claim. Primary drift excludes images with no eligible ground-truth instances (4,952/5,000 scored on COCO); set agreement includes all images. Intervals are conditional on the recorded image set and partition: target-image resampling for controlled interventions, whole-batch resampling for paired natural B4 comparisons. Seeds change partitions of the same images, not training or independent test samples.

4.2 A batch mate changes the learned model for the target

Extreme-mate effects in Table 1 exceed duplicate controls by two to three orders of magnitude. Duplicate-control drifts range from 1.47×10^{-5} to 1.19×10^{-4}. All eight CSIT reduction intervals exclude zero positively. On confirmation, Tiny wide/tall intervals are $[0.01760, 0.02379]$/$[0.01942, 0.02674]$; Small intervals are $[0.01783, 0.02441]$/$[0.01885, 0.02702]$.

Table 1: Mean absolute change in matched-object score under a fixed unrelated mate. The independent confirmation contains every eligible image not used in the pilot. Every CSIT single-image output is tensor-exact to uniform.

scale	split	mate	uniform	CSIT	reduction
Tiny	pilot	wide	0.03111	0.00825	73.5%
		tall	0.02953	0.00824	72.1%
	confirm	wide	0.02708	0.00647	76.1%
		tall	0.03099	0.00802	74.1%
Small	pilot	wide	0.02044	0.00444	78.3%
		tall	0.02476	0.00408	83.5%
	confirm	wide	0.02594	0.00492	81.0%
		tall	0.02847	0.00558	80.4%

Controls: duplicate drift 1.47×10^{-5}–1.19×10^{-4}; Tiny pilot $\eta = 0$: 0.00507 (wide), 0.00563 (tall).

The effect, repair, and exact solo contract therefore replicate across both held-out images and model scale.

With $\eta = 0$ on the Tiny pilot, wide- and tall-mate changes fall from 0.03111/0.02953 to 0.00507/0.00563, leaving 16.3%/19.1% of the uniform-TTT effect. Bootstrap intervals for TTT-minus-no-inner are $[0.01736, 0.03514]$ and $[0.01576, 0.03309]$. The residual warns against an absolute invariance claim: ordinary padding still changes some features. The dominant additional path, however, is the inner update.

On 32 targets per scale, we fix the complete preprocessed canvas and replace only the mate tensor by a horizontal reflection or zeros. Across 512 comparisons (two scales, two geometries, two arms, two replacements), all four target backbone outputs remain bitwise identical. Box outputs need not be bitwise equal, but maximum matched-score drift is 2.09×10^{-7}. Changing the canvas on these targets instead gives uniform mean drift 0.0294–0.0347. This separates a geometry-mediated training-set change from cross-image content sharing in the backbone (Appendix P).

4.3 Task utility and the failed valid-pixel repair

On full COCO random B4 (partition 0), Tiny box/mask AP changes by $-0.054/+0.232$ points and Small by $+0.010/+0.190$. Under random B8, box AP changes by $-0.048/-0.060$ for Tiny/Small, while mask AP rises by $+0.290/+0.225$. VOC AP50 changes by $+0.254/-0.080$ on random B4. Thus better reference agreement does not imply better task accuracy. Appendix G reports the complete matrices and subgroup metrics; Table 6 retains the historical sequential-B2 evaluation.

The raw-valid predecessor reduced Tiny mate dependence by 72–82% and passed Tiny full COCO. On the preregistered Small pilot it also reduced dependence by 79–85%, but changed single-image AP by -0.592 points, failing the utility gate. That historical pilot continuation was superseded by the completed full-data COCO/VOC and cross-domain matrices, which preserve the same failure boundary rather than hiding it. CSIT was frozen only after this failure: preserving solo divisor padding makes single outputs exact and the final extension reports its utility and subgroup limits explicitly.

4.4 Canvas intervention and mechanism decomposition

Holding the resized image fixed, we enlarge only its bottom/right canvas (Figure 2). All 20 nonzero model/domain/dose combinations show positive paired reductions, spanning 63.3–97.6% (2,000 image-bootstrap draws per dose; intervals are unadjusted). On 256 COCO targets, factorial decomposition assigns more of the natural-B4 improvement to denominator correction than to support deletion. Yet the prescribed post-clipping $1/\rho$ correction increases drift, and a same-feature gradient probe retains a directional residual after an oracle scalar fit. Thus dilution alone does not explain the discrepancy; this does not rule out every scalar policy. Appendices D and C give the decomposition and direction controls. Natural workloads also show an attenuated association between excess support and drift after CSIT (Appendix O), separately from the causal canvas intervention.

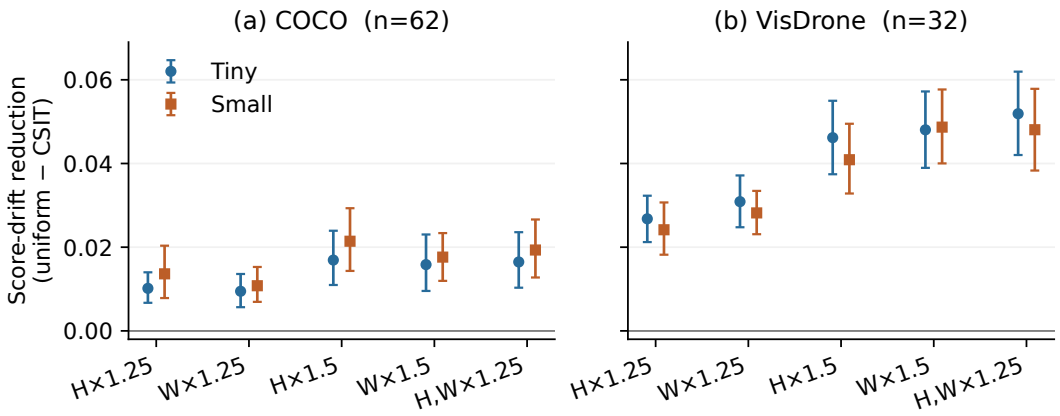

Figure 2: **Fixed pixels, changed execution canvas.** Paired score-drift reduction with 95% target-image bootstrap intervals for five bottom/right padding interventions, both scales, and two diagnostic domains. Every interval is positive; native-repeat drift is exactly zero. H/W denote height/width multipliers before divisor alignment. These are consistency endpoints, not AP.

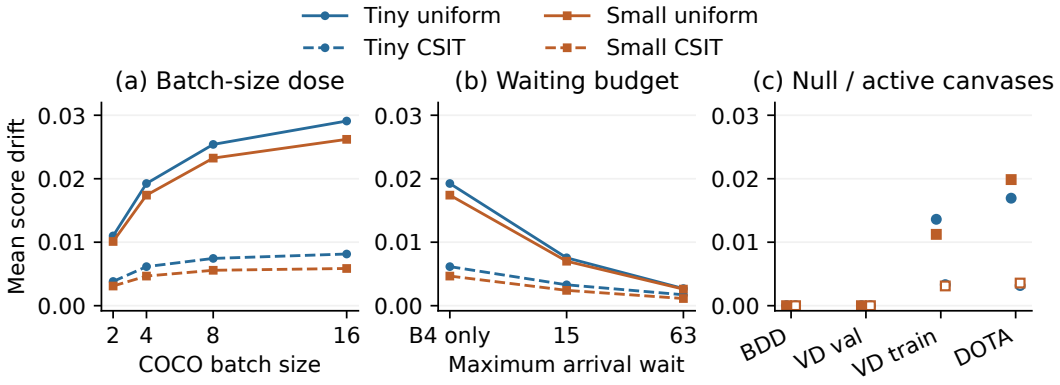

Figure 3: **Where the correction matters.** Full-data, partition-0 means. (a) Increasing random batch size increases drift. (b) Aspect sorting within 16/64-arrival windows sharply reduces it; B4 only has no extra lookahead. (c) BDD and VisDrone validation are canvas null controls; VisDrone train and DOTA are active axes. Filled/open markers denote uniform/CSIT. Absolute cross-domain AP is not compared. Lines connect measured operating points.

4.5 Reference preservation and execution alternatives

For singleton GT-matched scores within 0.05 of the 0.5 decision threshold, uniform flip rates are about 0.30–0.32 while CSIT rates are 0.08–0.11 on COCO random B4; away from the threshold both rates are below 0.02. Figure 4 makes these changes visible. After non-maximum suppression (NMS), class-aware prediction-set F1 (the harmonic mean of matching precision and recall) against singleton improves by 0.033–0.039 across Tiny/Small and the two original inference partitions. This F1 pools matching counts across images; the alternative arrival order in Table 10 reports mean per-image F1 instead. Comparing the same image under the two recorded B4 partitions gives 72.2% and 77.5% recomposition-drift reductions for Tiny and Small. Set and flip comparisons resample whole B4 batches; recomposition uses a two-way bootstrap over the two partition memberships. All intervals are conditional on the recorded partitions; they measure reference behavior, not ground-truth accuracy.

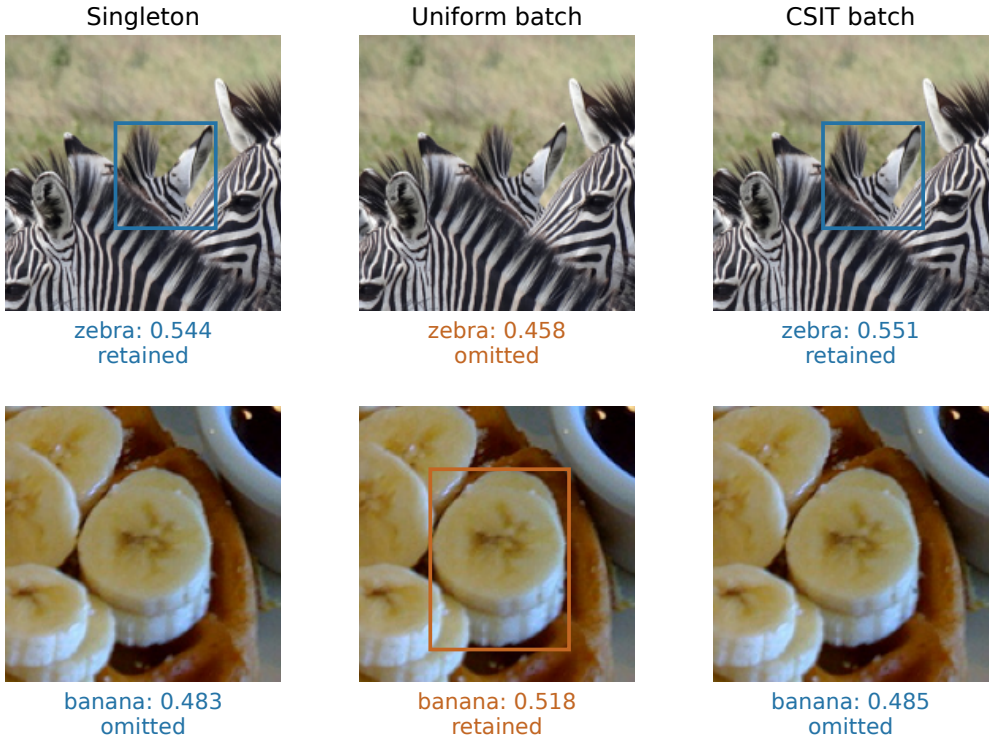

Figure 4: **A batch mate can change a threshold decision.** Each row shows identical COCO crops, Tiny partition 0. Only the inspected object's matched prediction is shown, at score ≥ 0.5; its score is printed even when the box is omitted. The top/bottom examples illustrate a detection removed/added by uniform batching relative to singleton. These cases were selected for near-threshold flips corrected by CSIT, not as representative accuracy gains.

The improvement is not confined to the mean. Across full COCO B4/B8, the 95th percentile of image drift falls by 65.8–72.9%, with positive paired whole-batch intervals. Nevertheless, 8.0–10.4% of images have higher drift, and Small retains a worst-case drift near 0.99. This matched-score endpoint can jump at the IoU boundary: selected adverse cases show matching changes rather than large confidence changes (Appendix R). CSIT is an aggregate repair, not an individual guarantee (Appendix Q). Re-extraction at IoU cutoffs 0.3/0.5/0.7 still gives 61.0–77.4% mean reductions.

Finite buffering is a strong alternative. With windows of 16 and 64 arrivals, uniform drift falls substantially before CSIT is applied. At window 64, CSIT still reduces the residual by 36.2% for Tiny and 56.7% for Small. This limits the practical value of CSIT when buffering is acceptable; the simple alternatives below further delimit its role when no extra waiting is allowed.

A geometry-only binary controller reduces drift on a partition holdout, but full CSIT is better and buffer64 uniform is stronger when waiting is allowed. Its two buffer windows support descriptive comparisons only; a retained Tiny outlier reverses the buffer64+CSIT mean-drift gain. Geometry exposes support risk without certifying AP or a best action (Appendix E).

On all 5,000 COCO validation images, exact-shape grouping within each incoming B4 preserves native singleton behavior to numerical precision and is faster than CSIT in this measured workload. CSIT keeps one heterogeneous forward and reduces drift from 0.01972/0.01742 to 0.00623/0.00453 (Tiny/Small). Fixed 1344×1344 execution is stable against its own singleton but has 0.03460/0.03128 drift from the native reference. Stability, reference preservation, and AP are therefore distinct (Table 2). Exact-shape grouping is a strong alternative whenever regrouping is acceptable; CSIT repairs support when retaining the heterogeneous batch is required.

Table 2: **Stability, reference preservation, and AP differ.** All 5,000 COCO validation images, with a separate salted arrival order (Appendix I), not the partition-0 grouping in Section 4.3 or Figure 3. Native drift uses released singleton; own-reference drift uses fixed-square singleton for that arm. Exact-shape grouping averages 3.075 calls per B4; the other rows use one. No extra lookahead.

Scale	Execution	Native drift	Own-ref. drift	Box AP
Tiny	Uniform B4	0.01972	0.01972	47.27
	CSIT B4	0.00623	0.00623	47.27
	Exact shape	4.04×10^{-8}	4.04×10^{-8}	47.26
	Fixed square B4	0.03460	9.55×10^{-8}	47.20
Small	Uniform B4	0.01742	0.01742	49.22
	CSIT B4	0.00453	0.00453	49.14
	Exact shape	3.45×10^{-8}	3.45×10^{-8}	49.21
	Fixed square B4	0.03128	8.27×10^{-8}	49.06

To quantify the waiting tradeoff, we measure all four policies on the same ordered 5,000 images per scale and replay identical synthetic arrivals against their measured execution times. Policy order is rotated and each measured forward is synchronized. The unchanged B4 policies reproduce the earlier per-image score vectors exactly; buffer64 changes membership only within each 64-arrival window. At Poisson arrival rates equal to 80% of measured CSIT capacity, median 95th-percentile (P95) latency across five arrival seeds is 535/871 ms for CSIT (Tiny/Small), 461/739 ms for exact-shape grouping, and 4594/7244 ms for buffer64. Buffer64 achieves lower drift, 0.00315/0.00286, and higher execution throughput, but its window incurs additional waiting (Appendix I.3). These latencies include assembly, queueing, and measured service time; they exclude decoding and networking and are simulated, not production measurements.

The waiting distinction has a simple explanation. For Poisson requests at rate λ, a complete cohort of k arrivals has expected per-request assembly wait $(k-1)/(2\lambda)$. Thus B4 incurs $1.5/\lambda$, whereas buffer64 incurs $31.5/\lambda$ before execution. Exact-shape grouping splits the current B4 without waiting for another cohort. Consequently, a restriction on lookahead alone does not favor CSIT over exact grouping: retaining a single heterogeneous forward is the relevant constraint. The scheduler comparison makes explicit which alternative spends waiting time, changes batch membership, or splits execution, rather than comparing drift at an unspecified systems budget.

Table 3: **Choose execution according to the permitted change.** Drift values are the full-COCO Tiny comparison in Table 2. Speed comparisons apply to the measured workload.

Constraint	Execution	Observed consequence
Retain one heterogeneous forward	CSIT	Drift 0.020 → 0.006; singleton unchanged; AP mixed.
May split the arriving group	Exact shape	Native reference to numerical precision; faster here.
May wait for similar shapes	Aspect buffer	Most drift reduced before support correction.
May change the canvas contract	Fixed square	Stable to its own reference, not the native one.

4.6 Evidence beyond the fast-weight detector family

Independent audits distinguish support, state, and task utility. Continual test-time adaptation for object detection (CTA-OD; Faster R-CNN R50) (Yoo et al., 2024) reduces confidence drift by 18.5–20.6% across three full-COCO orders through canonical-support pooling, with nearly unchanged AP. This tests a different adaptation mechanism: the released global-alignment loss pools features over the batch canvas, including excess padding. We change only that pooling support and retain the released foreground alignment, source statistics, and optimizer settings. Each order evaluates all 5,000 validation images, with labels used only for metrics and adaptation state retained through the stream. The reference computes each image's features on its singleton canvas but combines the same four images' statistics into the original joint loss and one optimizer update. Four independent

singleton learners would change the learning data as well as the canvas and would therefore be an inappropriate reference. In order 1, confidence drift falls from 0.01375 to 0.01108, while AP changes from 44.289 to 44.294; the other two complete orders confirm the drift reduction with mixed AP changes. Thus support correction transfers to optimizer-based detection without implying a task-accuracy gain (Appendix U). Because adaptation state persists, neighboring batches are not independent learning trajectories. We therefore report the three complete orders separately; the group-bootstrap interval within one order describes that realized stream and is not an independent-seed confidence interval.

VLOD-TTA adapts vision-language object detectors; our full-Watercolor audit separates optimizer moments from the step counter; complete state reset lowers AP (Appendix V). Support-only transfer to TTT for image restoration (TTTIR) fails (Appendix S), while full-split evaluation of the Segment Anything-based SAM-TTT (Yu et al., 2025) is stable under native fixed-canvas batching (Appendix T). These results broaden the mechanism evidence without establishing a second fast-weight detector replication.

5 Related Work

Classical TTT learns from an unlabeled test example through an auxiliary objective (Sun et al., 2020); Tent adapts normalization under distribution shift (Wang et al., 2021). TTT layers use inner learning as sequence computation (Sun et al., 2025; Zhang et al., 2026). In ViT^3, spatial tokens train a per-image fast-weight learner (Han et al., 2026). Vision-TTT, Transformer-to-TTT, and Modular TTT extend this view to visual sequences, pretrained weights, and composable graphs (Kong et al., 2026; Li et al., 2026a; Tang et al., 2026); VGG-T^3, Spatial-TTT, Mem3R, and TTT3R emphasize scene memory and streaming state (Elflein et al., 2026; Liu et al., 2026b;a; Chen et al., 2026). Our intervention concerns the execution-created training support of an existing learner.

SAST shows that unmasked padding leaks context in sparse attention for event-based detection (Peng et al., 2024). An attention mask, however, does not specify the empirical measure of an inner loss. Test-Time Padding learns pixels around adversarial CLIP inputs (Li et al., 2026b); CSIT adds no pixels or learned prompt and preserves native singleton padding. Padding efficiency in RF-DETR (Robinson et al., 2026) and numerical determinism in LLM serving (Zhao et al., 2026) concern different execution effects, which our duplicate and no-inner controls separate from learning-enabled sensitivity.

Detector test-time adaptation (TTA) adapts prompts or lightweight modules using pseudo-labels and entropy (Gao et al., 2025; Belal et al., 2025; Yoo et al., 2024). Tracing its loss support and state is necessary before transferring a repair; our CTA-OD audit holds shared learning data fixed. More broadly, unlabeled evidence may not identify the best adaptation action (Jhawar & Wang, 2026). Geometry can expose excess support without certifying AP or an optimal scheduler, motivating our separate evaluation of reference agreement and task utility.

6 Conclusion and Limitations

When input tokens train a model during inference, batching specifies part of its training data. CSIT preserves singleton support and excludes only batch-excess support. State lifetime, update boundaries, and scheduler waiting budgets likewise require explicit choices. Reference agreement and task accuracy remain separate objectives; execution policies must be judged on both.

Full COCO/VOC evaluation covers one fast-weight detector family. CTA-OD extends support correction to optimizer-based detection; VLOD-TTA and VGG-T^3 probe state and chunk semantics. They do not provide a second fast-weight detector replication. CSIT assumes canonical-feature preservation, leaves ordinary boundary effects, and yields mixed AP changes. Direct restoration transfer fails; MRI/LOL probes are small, and extreme canvases are interventions rather than production samples. Buffering and exact-shape grouping are strong alternatives. CSIT addresses the constraint of retaining a heterogeneous forward, without establishing universal repair or a production latency advantage.

AI USE STATEMENT

In this work, we used generative AI tools to help develop the conceptual framework, formulate mathematical claims, assist in the writing of proofs, refine hypotheses, design and provide feedback on the methodology and experiments, implement methods. We have not used generative AI tools to generate synthetic datasets, to provide critical ingredients for the proofs, or to assist with translation, and qualitative and thematic data analysis is not applicable to this work. Additionally, we used generative AI tools to aid and polish the writing, draft parts of the paper, retrieve and identify related work, edit software during implementation and debugging, prepare artifacts. We have reviewed all AI-assisted work. Reported measurements come from executed experiments and saved predictions; quantitative figures were plotted with Python. Checks include source verification, matched controls, checkpoint and result hashes, numerical formula checks, and replay of saved evaluations. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses existing research datasets and released checkpoints; it introduces no new human-subject recruitment or personal-data collection. Consistency with a reference model does not establish fairness, privacy, safety, or correctness. Detection systems can enable harmful surveillance, and support correction does not make such applications appropriate. Dataset and checkpoint use remains subject to the original terms.

REPRODUCIBILITY STATEMENT

The appendices provide the conditional assumptions and proofs (Appendix C), full-data protocols and results (Appendix G), execution alternatives (Appendix I), and independent detector/state audits (Appendices U and V). They specify evaluation settings, reference computations, model initialization, and numerical checks. Model and dataset sources are cited alongside the corresponding experiments.

REFERENCES

Anton Antonov, Andrey Moskalenko, Denis Shepelev, Alexander Krapukhin, Konstantin Soshin, Anton Konushin, and Vlad Shakhuro. RClicks: Realistic click simulation for benchmarking interactive segmentation. In *NeurIPS*, volume 37, pp. 127673–127710, 2024.

Atif Belal, Heitor R. Medeiros, Marco Pedersoli, and Eric Granger. VLOD-TTA: Test-time adaptation of vision-language object detectors. *arXiv:2510.00458*, 2025.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. MMDetection: Open MMLab detection toolbox and benchmark. *arXiv:1906.07155*, 2019.

Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. TTT3R: 3d reconstruction as test-time training. In *ICLR*, 2026.

Sven Elflein, Ruilong Li, Sérgio Agostinho, Zan Gojcic, Laura Leal-Taixé, Qunjie Zhou, and Aljosa Osep. VGG-T^3: Offline feed-forward 3d reconstruction at scale. In *CVPR*, pp. 36464–36474, 2026.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

Yingjie Gao, Yanan Zhang, Zhi Cai, and Di Huang. Test-time adaptive object detection with foundation model. In *NeurIPS*, volume 38, 2025.

Dongchen Han, Yining Li, Tianyu Li, Zixuan Cao, Ziming Wang, Jun Song, Yu Cheng, Bo Zheng, and Gao Huang. ViT^3: Unlocking test-time training in vision. In *CVPR*, 2026.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *CVPR*, pp. 5001–5009, 2018.

Kartik Jhawar and Lipo Wang. When is test-time adaptation identifiable from unlabeled evidence? *arXiv:2609.11235*, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. URL https://arxiv.org/abs/1412.6980.

Quan Kong, Yanru Xiao, Yuhao Shen, and Cong Wang. Vision-TTT: Efficient and expressive visual representation learning with test-time training. *arXiv:2603.00518*, 2026.

Yining Li, Dongchen Han, Zeyu Liu, Hanyi Wang, Yulin Wang, and Gao Huang. Linearizing vision transformer with test-time training. In *ICML*, 2026a.

Zhiwei Li, Yitian Pang, Weining Wang, Zhenan Sun, and Qi Li. TTP: Test-time padding for adversarial detection and robust adaptation on vision-language models. In *CVPR*, pp. 1513–1522, 2026b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

Changkun Liu, Jiezhi Yang, Zeman Li, Yuan Deng, Jiancong Guo, and Luca Ballan. Mem3R: Streaming 3d reconstruction with hybrid memory via test-time training. *arXiv:2604.07279*, 2026a.

Fangfu Liu, Diankun Wu, Jiawei Chi, Yimo Cai, Yi-Hsin Hung, Xumin Yu, Hao Li, Han Hu, Yongming Rao, and Yueqi Duan. Spatial-TTT: Streaming visual-based spatial intelligence with test-time training. In *ECCV*, 2026b.

Youssef Mansour, Xuyang Zhong, Serdar Caglar, and Reinhard Heckel. TTT-MIM: Test-time training with masked image modeling for denoising distribution shifts. In *ECCV*, pp. 341–357, 2024.

Yansong Peng, Hebei Li, Yueyi Zhang, Xiaoyan Sun, and Feng Wu. Scene adaptive sparse transformer for event-based object detection. In *CVPR*, 2024.

Isaac Robinson, Peter Robicheaux, Matvei Popov, Deva Ramanan, and Neehar Peri. RF-DETR: Neural architecture search for real-time detection transformers. In *ICLR*, 2026.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522. PMLR, 2025.

Bohao Tang, Zhen Qin, Yuqi Pan, Zheng Li, Pengfei Liu, and Ya Zhang. Modular TTT: Rethinking test-time training as composable modules. *arXiv:2608.07110*, 2026.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

Chen Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. In *BMVC*, 2018.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. URL https://arxiv.org/abs/1803.08494.

Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. DOTA: A large-scale dataset for object detection in aerial images. In *CVPR*, 2018.

Jayeon Yoo, Dongkwan Lee, Inseop Chung, Donghyun Kim, and Nojun Kwak. What, how, and when should object detectors update in continually changing test domains? In *CVPR*, pp. 23354–23363, 2024.

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020.

Zhenni Yu, Li Zhao, Guobao Xiao, and Xiaoqin Zhang. SAM-TTT: Segment anything model via reverse parameter configuration and test-time training for camouflaged object detection. In *ACM Multimedia*, pp. 4030–4038, 2025.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right. In *ICLR*, 2026.

Shiju Zhao, Jiacheng Yang, Qihang Chen, Junhao Hu, Jiaqi Zheng, Guihai Chen, and Xusheng Chen. CoRun: Padding is simple and efficient for deterministic LLM inference. *arXiv:2608.14376*, 2026.

Kaihang Zheng, Jun Li, Hang Guo, Hongyu Chi, Zimo Liu, Tao Dai, Jinpeng Wang, and Yaowei Wang. TTTIR: Unlocking instance-specific state evolution via test-time training for image restoration. *arXiv:2609.26151*, 2026.

Pengfei Zhu, Longyin Wen, Xiao Bian, Haibin Ling, and Qinghua Hu. Vision meets drones: A challenge. *arXiv:1804.07437*, 2018.

Appendices

Rethinking Batching in Visual Test-Time Learning

These appendices provide the conditional theory (Appendix C), full-data results (Appendix G), simple execution alternatives (Appendix I), serving contract (Appendix K), and additional content and tail audits (Appendices P and Q). Equations, figures, and tables retain the numbering resolved jointly with the main paper.

A Frozen decision gates

The pilot mechanism gate required an extreme-mate change of at least 0.005, at least five times the duplicate control, with a positive bootstrap separation. Method promotion additionally required at least 50% reduction, a positive reduction interval, and valid-single AP non-inferiority within -0.1 points. The confirmation required both axes to pass on a disjoint split. Raw-valid Tiny passed, but the separately preregistered Small pilot failed its single-image utility gate and stopped. The outcome-aware canonical-support hypothesis was then written before any CSIT output. It required tensor-exact solo inference, both axes on Tiny/Small pilot and confirmation, full box/mask non-inferiority within -0.1 points, and at most 5% overhead on both scales. No failed gate was rewritten after an outcome.

B Implementation-state control

The released detection overlay initializes a branch scale as $32^{-1/2}$ and assigns $1/3$ during its first forward. Consequently the first image after model construction is evaluated by a different function from later images. We use constant $1/3$, matching the steady-state forward and current classification implementation, for every condition. An all-one measure is routed through the literal uniform code path. Repeated single-image and all-one detector smoke tests are exactly equal. This control is not a method contribution, but it is necessary for image-locked causal comparisons.

C Measure preservation and conditional guarantees

The following elementary identities explain the intervention; they are not a new generalization theory or a certificate for the detector's full pipeline.

Exact inner-measure condition. Let U be a finite set of canonical token identities with probability mass μ. An execution layout contains copies of those tokens and padding. Let π map each copy to its identity and each padded position to $\perp$; extend μ by $\mu(\perp) = 0$. Let ν be normalized nonnegative layout weights. If copies retain their canonical gradient, then the raw inner updates agree for *every* gradient field if and only if $\pi_{\#}\nu = \mu$. To prove sufficiency, collect the weights of each identity in the finite gradient sum. To prove necessity, choose the scalar indicator of each identity as the gradient field. Identical raw updates imply identical deterministic clipping and fast weights. A particular model can have equal updates under unequal measures, so this universal statement is not a test of necessity from observed gradients alone. It requires a common initial state, features, update rule, and token correspondence.

This condition implies zero mass for appended padding, preservation of total mass under token replication, and mass-weighted aggregation of unequal chunks. It does not license changing positional encodings or convolutional neighborhoods. Our area-resized mask is fractional in the general formulation; canonical stage mass, rather than a pixel count substituted at every stage, is the normalizer in Eq. 2.

Scalar oracle. For fixed-feature transformed uniform and canonical updates a and b, the best nonnegative scalar is

$$\alpha^* = \max\{0, \langle a, b\rangle / \|a\|^2\}, \qquad r^* = \|b - \alpha^* a\| \quad (a \neq 0).$$

For $a = 0$, $r^* = \|b\|$. This follows by minimizing the convex quadratic $\|b - \alpha a\|^2$ over $\alpha \geq 0$. A nonzero residual rules out exact restoration by any nonnegative scalar for those fixed features. The

clipping groups must match the released operator; Eq. 3 itself holds before clipping. On the saved 16-image Tiny diagnostic, median relative post-clipping oracle residuals are 0.135 (gate) and 0.158 (depthwise), with 510 correlated gradient cases. These are operator diagnostics, not 510 independent images or a task-accuracy experiment.

Imperfect feature preservation. For native and masked updates of a fixed state space, write $W_{t+1} = W_t - \eta_t \mathcal{C}(G_t(W_t))$ and $W'_{t+1} = W'_t - \eta_t \mathcal{C}(G'_t(W'_t))$. Assume $\eta_t \geq 0$, G_t is L_t-Lipschitz, and $\|G'_t(W) - G_t(W)\| \leq \delta_t$ on the states in question. For each radial clipping group, the map $g/(1+\|g\|)$ has radial and tangential Jacobian eigenvalues $(1+\|g\|)^{-2}$ and $(1+\|g\|)^{-1}$; thus it is 1-Lipschitz in the joint Euclidean norm. Adding and subtracting $\mathcal{C}(G_t(W'_t))$ gives

$$e_{t+1} \leq (1+\eta_t L_t) e_t + \eta_t \delta_t, \qquad e_T \leq e_0 \prod_{s=0}^{T-1} (1+\eta_s L_s) + \sum_{t=0}^{T-1} \eta_t \delta_t \prod_{s=t+1}^{T-1} (1+\eta_s L_s),$$

where $e_t = \|W'_t - W_t\|$. The second expression follows by induction. Equal initial state and zero feature discrepancy imply equal fast weights; CSIT alone does not ensure $\delta_t = 0$. We have not certified these constants for a detector, and backbone stages with different dimensions are not repeated optimization of one W. NMS, proposal selection, and threshold decisions can amplify small remaining differences, so this bound is not an AP guarantee.

D Factorial mechanism evaluation

The five arms use identical targets and context templates. If the released helper divides by stage area N, uniform uses unit weights, support-only uses m, denominator-only uses $1/\rho$, and CSIT uses m/ρ. LR-only multiplies the *already clipped* uniform update by $1/\rho$; it therefore differs from denominator-only. With mean score drifts u, s, d, c, respectively, the order-averaged support contribution is $[(u-s)+(d-c)]/2$ and the denominator contribution is $[(u-d)+(s-c)]/2$. They sum to $u-c$ by construction. The joint interaction is $s+d-u-c$. Both repeats are averaged within each of 256 targets before 10,000 target-bootstrap draws. Intervals are exploratory, unadjusted, and conditional on the recorded context construction; they do not establish separate additive causal paths.

Table 4: Natural B4 factorial contributions; 95% target-bootstrap intervals.

Contrast	Tiny	Small
Support	0.00499 [0.00284, 0.00774]	0.00392 [0.00287, 0.00500]
Denominator	0.00993 [0.00728, 0.01306]	0.00975 [0.00724, 0.01273]
Interaction	0.00516 [-0.00039, 0.00947]	0.01341 [0.01054, 0.01640]
LR-only − uniform	0.00700 [0.00121, 0.01276]	0.03577 [0.02683, 0.04615]

E Adaptive execution and paired buffering

The evaluated controller is binary at batch level: $E(B) \geq 0.2$ selects CSIT, otherwise uniform. The implementation library contains a soft-weight proposal, but the evaluated runner maps every non-CSIT route to uniform; no continuous support-weight experiment is claimed. Threshold exploration used recorded partition-0 outputs; partition 1 changes membership while sharing the COCO image pool. Its first 128 images form the real-inference slice, with 126 eligible images for score drift. No new-image generalization claim is made. F1 is the mean per-image, same-class maximum-cardinality post-NMS matching F1 at score/IoU thresholds 0.5/0.5, with an empty–empty pair assigned one. This differs from the pooled-count F1 used for the full-data set endpoint.

Paired intervals resample all 32 B4 batches jointly across the three random execution arms. Adaptive-minus-uniform drift intervals are $[-0.01570, -0.00666]$ (Tiny) and $[-0.01459, -0.00602]$ (Small); F1 intervals are $[0.00592, 0.03391]$ and $[0.01059, 0.02636]$. The newly paired buffer uses the same seed-1 arrival set, then sorts each 64-image window by aspect ratio; outputs are joined by image ID. Because it has only two windows, its comparison is descriptive. "No extra lookahead" means no waiting beyond forming the base B4 batch, not zero request latency.

Table 5: Same 128-image arrival set. Drift averages 126 eligible images; macro F1 averages all 128. Buffer rows allow up to 63 arrival waits.

	Tiny		Small	
Execution	Drift	Macro F1	Drift	Macro F1
Random B4 / uniform	0.01795	0.96669	0.01632	0.97592
Random B4 / CSIT	0.00563	0.98842	0.00384	0.99551
Random B4 / adaptive	0.00714	0.98614	0.00624	0.99400
Buffer64 / uniform	0.00201	0.99327	0.00201	0.99808
Buffer64 / CSIT	0.00677	0.99959	0.00135	0.99842

Endpoint disagreement is retained. Tiny buffer64+CSIT has better macro F1 but worse GT-score drift than buffer64 uniform. The largest per-image drift is 0.74297 on image 127624; it is included in the mean, not trimmed. GT matching and prediction-set matching ask different questions and both include discrete IoU decisions. This counterexample precludes a per-image or every-scheduler improvement claim. In contrast, the separate full 5,000-image, partition-0 buffer64 study shows lower mean drift with CSIT on both scales. The 128-image follow-up must not replace that population or be pooled with it as independent evidence.

F Independent contract probes and negative boundaries

VGG-T^3: chunk boundaries. Using the released model (Elflein et al., 2026), the local audit fixes semantic support to $S-1$ views while applying the learned state to all S views. Sixteen sequences have $S=5$ and two have $S=3$; two updates and checkpoints are fixed. With three-view or large memory chunks, the public partial-window translation overshoots the intended support in all 18 sequences. Strict clipping removes that overshoot. For three-view chunks, public minus strict Chamfer distance is 0.00314 with a scene-cluster interval $[-0.00164, 0.00624]$; for large chunks it is -0.00059 $[-0.00581, 0.00423]$. Neither establishes better reconstruction quality. Strict-large and native repeat reproduce their unchunked references exactly in the online comparison; the raw unchunked tensors were not retained. Smaller support-preserving chunks can still differ numerically. The default full-support/large-chunk path is an exact control; this defect concerns partial semantic windows and is not a failure of the default benchmark protocol.

TTTIR: synthetic padding sensitivity. We load the released LOL-v1 checkpoint of TTTIR (Zheng et al., 2026) and verify complete state-dictionary coverage after key renaming, and use one seeded 256×256 random RGB input. Extra bottom/right reflect padding is cropped from the output. Disabling the inner update returns the unadapted transition weight and uses its own no-padding reference. Mean absolute output changes are:

Extra pixels	32	64	128
Update enabled	0.003413	0.003531	0.005072
Update disabled	0.001639	0.002005	0.002040

Native repeats are zero. This is one synthetic input, not a LOL evaluation, a CSIT repair, or a PSNR/SSIM result; the nonlinear difference of sensitivities is not an additive causal decomposition. TTT3R checkpoint loading succeeds but local attention execution fails, so it contributes no positive result.

G Full-data evaluation and reporting conventions

All checkpoints are frozen. The released resize keeps aspect ratio with $(1333, 800)$ bounds and the native size divisor. COCO uses all 5,000 validation images, including empty images for AP. VOC2012 (Everingham et al., 2010) uses 5,823 validation images, 20 mapped categories, difficult-instance handling, and continuous AP50. COCO-training/VOC-image overlap has not been ruled out; VOC is a diagnostic transfer evaluation, not a clean domain-generalization benchmark.

Table 6: Full COCO 2017 validation, two released checkpoints and sequential batch size two. Time is summed synchronized wall time across four paired shards. The negative timing delta is interpreted as parity, not speedup.

scale	measure	box AP	box AP_S	mask AP	mask AP_S	time
Tiny	uniform	47.292	31.121	42.737	26.170	392.17 s
	CSIT	47.273	30.885	42.859	26.264	388.94 s
	Δ	-0.019	-0.236	+0.122	+0.094	-0.82%
Small	uniform	49.209	34.247	44.106	28.939	684.24 s
	CSIT	49.180	34.243	44.155	29.039	677.05 s
	Δ	-0.029	-0.004	+0.049	+0.100	-1.05%

Cross-domain images are evaluated with the same COCO-trained detector: VisDrone validation (Zhu et al., 2018) (548 images) and BDD (Yu et al., 2020) (10,000) have homogeneous effective canvases; VisDrone train (6,471) and DOTA validation (Xia et al., 2018) (458) have heterogeneous canvases. No model is trained on these diagnostic splits. Train/validation contrasts are observational and may differ in content; only the fixed-image canvas intervention controls content directly. DOTA polygons become axis-aligned hulls; only airplane, boat, car, and truck are mapped. VisDrone merges pedestrian/people into person, van/truck into truck, and tricycle/motor into motorcycle, retains bicycle/car/bus, and drops other categories. BDD merges pedestrian/rider into person, retaining bicycle/bus/car/motorcycle/train/truck. These class mappings and the COCO 100-detection cap make the scores diagnostic, not official leaderboard AP. Crowd objects follow the metric-specific eligibility rules; input hashes were recorded during evaluation.

Natural partitions sort SHA-256 hex digests of the UTF-8 string `csit-expanded-20260919-v1:`s`:`i, with partition seed $s \in \{0, 1\}$ and image ID i written as unpadded decimal integers. Global aspect sorting is an offline boundary. Finite buffering sorts each disjoint arrival window by original aspect ratio, waits until its last arrival, and flushes it before the next window. Waits of 15/63 are arrival counts, not milliseconds; no service-level or isolated throughput claim is made. The 46-cell cross-domain matrix includes the two B16 COCO cells and eight non-TTT Cascade cells. The earlier COCO/VOC matrix contains 21 cells and three Cascade controls. A cell can contain both uniform and CSIT arms.

COMPLETE COCO/VOC EXTENSION (21 CELLS)

Table 7: Uniform (U) and CSIT (C); AP in points. VOC uses AP50; other datasets use COCO-style box AP. Drift is relative to singleton.

Model	Data / policy	B	Seed	AP U	AP C	Drift U	Drift C
Cascade	COCO/sort	4	0	40.339	–	0.00005	–
Cascade	COCO/rand	1	0	40.339	–	–	–
Cascade	COCO/rand	4	0	40.337	–	0.00251	–
Small	COCO/sort	4	0	49.225	49.213	0.00009	0.00006
Small	COCO/rand	1	0	49.213	–	–	–
Small	COCO/rand	2	0	49.232	49.158	0.01012	0.00310
Small	COCO/rand	4	0	49.126	49.136	0.01741	0.00467
Small	COCO/rand	4	1	49.145	49.116	0.01761	0.00453
Small	COCO/rand	8	0	49.187	49.127	0.02325	0.00558
Small	VOC/sort	4	0	89.711	89.708	0.00004	0.00003
Small	VOC/rand	1	0	89.708	–	–	–
Small	VOC/rand	4	0	89.734	89.654	0.00825	0.00275
Tiny	COCO/sort	4	0	47.262	47.253	0.00009	0.00006
Tiny	COCO/rand	1	0	47.256	–	–	–
Tiny	COCO/rand	2	0	47.340	47.251	0.01099	0.00382
Tiny	COCO/rand	4	0	47.272	47.218	0.01925	0.00615
Tiny	COCO/rand	4	1	47.354	47.262	0.01991	0.00639
Tiny	COCO/rand	8	0	47.303	47.255	0.02541	0.00745
Tiny	VOC/sort	4	0	88.938	88.941	0.00005	0.00004

Model	Data / policy	B	Seed	AP U	AP C	Drift U	Drift C
Tiny	VOC/rand	1	0	88.940	–	–	–
Tiny	VOC/rand	4	0	88.735	88.990	0.00998	0.00396

COMPLETE CROSS-DOMAIN EXTENSION (46 CELLS)

Table 8: Uniform (U) and CSIT (C); AP in points. VOC uses AP50; other datasets use COCO-style box AP. Drift is relative to singleton.

Model	Data / policy	B	Seed	AP U	AP C	Drift U	Drift C
Cascade	BDD/rand	1	0	21.208	–	–	–
Cascade	BDD/rand	4	0	21.208	–	0.00000	–
Cascade	DOTA/rand	1	0	0.714	–	–	–
Cascade	DOTA/rand	4	0	0.723	–	0.00043	–
Cascade	VD val/rand	1	0	12.419	–	–	–
Cascade	VD val/rand	4	0	12.419	–	0.00000	–
Cascade	VD train/rand	1	0	8.118	–	–	–
Cascade	VD train/rand	4	0	8.118	–	0.00041	–
Small	BDD/sort	4	0	25.967	25.967	0.00000	0.00000
Small	BDD/rand	1	0	25.967	–	–	–
Small	BDD/rand	4	0	25.967	25.967	0.00000	0.00000
Small	BDD/rand	8	0	25.967	25.967	0.00000	0.00000
Small	COCO/rand	16	0	49.181	49.121	0.02620	0.00586
Small	DOTA/sort	4	0	1.301	1.296	0.00081	0.00048
Small	DOTA/rand	1	0	1.287	–	–	–
Small	DOTA/rand	4	0	1.464	1.287	0.01986	0.00356
Small	DOTA/rand	4	1	1.489	1.286	0.01928	0.00296
Small	DOTA/rand	8	0	1.607	1.291	0.02644	0.00380
Small	VD val/sort	4	0	18.207	18.207	0.00000	0.00000
Small	VD val/rand	1	0	18.207	–	–	–
Small	VD val/rand	4	0	18.207	18.207	0.00000	0.00000
Small	VD val/rand	8	0	18.207	18.207	0.00000	0.00000
Small	VD train/sort	4	0	14.112	14.112	0.00001	0.00000
Small	VD train/rand	1	0	14.112	–	–	–
Small	VD train/rand	4	0	14.279	14.113	0.01124	0.00311
Small	VD train/rand	4	1	14.277	14.110	0.01113	0.00307
Small	VD train/rand	8	0	14.292	14.109	0.01264	0.00349
Tiny	BDD/sort	4	0	24.418	24.418	0.00000	0.00000
Tiny	BDD/rand	1	0	24.418	–	–	–
Tiny	BDD/rand	4	0	24.418	24.418	0.00000	0.00000
Tiny	BDD/rand	8	0	24.418	24.418	0.00000	0.00000
Tiny	COCO/rand	16	0	47.268	47.272	0.02909	0.00814
Tiny	DOTA/sort	4	0	1.039	1.016	0.00064	0.00032
Tiny	DOTA/rand	1	0	1.014	–	–	–
Tiny	DOTA/rand	4	0	1.235	1.012	0.01693	0.00318
Tiny	DOTA/rand	4	1	1.280	1.013	0.01673	0.00389
Tiny	DOTA/rand	8	0	1.398	1.018	0.02340	0.00378
Tiny	VD val/sort	4	0	16.588	16.588	0.00000	0.00000
Tiny	VD val/rand	1	0	16.588	–	–	–
Tiny	VD val/rand	4	0	16.588	16.588	0.00000	0.00000
Tiny	VD val/rand	8	0	16.588	16.588	0.00000	0.00000
Tiny	VD train/sort	4	0	13.095	13.095	0.00001	0.00001
Tiny	VD train/rand	1	0	13.095	–	–	–
Tiny	VD train/rand	4	0	13.256	13.095	0.01361	0.00332
Tiny	VD train/rand	4	1	13.243	13.093	0.01348	0.00327
Tiny	VD train/rand	8	0	13.270	13.093	0.01536	0.00369

COCO SIZE AND MASK UTILITY

Table 9: CSIT minus uniform AP points, full COCO partition 0. Subgroup losses are retained; these are point estimates, not non-inferiority tests.

Model	Policy/B	Box	Box S	Box M	Box L	Mask	Mask S
Small	aspect/4	-0.012	-0.000	-0.032	-0.013	-0.009	-0.007
Small	random/2	-0.075	+0.346	-0.123	-0.134	+0.058	+0.283
Small	random/4	+0.010	+0.338	-0.014	+0.240	+0.190	+0.338
Small	random/8	-0.060	-0.351	-0.032	+0.172	+0.225	+0.008
Tiny	aspect/4	-0.009	-0.006	-0.044	-0.003	-0.001	-0.001
Tiny	random/2	-0.089	+0.067	-0.182	+0.089	+0.096	+0.107
Tiny	random/4	-0.054	-0.148	-0.101	+0.521	+0.232	+0.306
Tiny	random/8	-0.048	-0.303	-0.139	+0.571	+0.290	+0.322
Small	random/16	-0.061	-0.255	-0.043	+0.194	+0.267	+0.151
Tiny	random/16	+0.003	-0.312	-0.110	+0.592	+0.364	+0.424
Small	buffer16/4	-0.054	+0.026	-0.152	-0.122	+0.051	+0.076
Small	buffer64/4	-0.062	-0.040	-0.127	-0.046	+0.006	-0.030
Tiny	buffer16/4	-0.075	-0.092	-0.199	-0.079	-0.000	+0.020
Tiny	buffer64/4	-0.061	-0.060	-0.150	+0.037	-0.070	+0.005

H REPRODUCIBILITY AND EVIDENCE MAP

These appendices report the assumptions, proofs, evaluation protocols, model settings, and numerical results. Complete COCO execution alternatives, CTA-OD stream comparisons, and Watercolor state interventions are distinguished from smaller mechanism probes. Analysis uses Python, NumPy, SciPy, and Matplotlib; detector inference uses PyTorch 2.7.1 with CUDA 12.8, MMCV 2.1, and the MMDetection pipeline. Timing scopes are reported with each study: earlier shared-GPU observations are separate from the later warmed, synchronized execution-alternative runs.

The adapter in Appendix K also supports independent GPU workers. Complete original batches are assigned to separate model processes; splitting a batch would change its canvas. On ten COCO images per scale, grouped as 4/4/2, one and two RTX 5090 GPUs produce identical boxes, scores, and labels. This checks the tested path and final partial batch, not throughput or arbitrary hardware equivalence. It applies to the frozen independent-image learner, not continual adaptation with shared optimizer state. Stage measures are reused within a forward; features, predictions, and fast weights are not cached across requests.

Claim	Evidence in these appendices
Canonical measure and scalar insufficiency	Appendix C
Full COCO/VOC reference and task results	Appendix G
Support/denominator intervention	Appendix D
Fixed canvas and exact-shape alternatives	Appendix I
Independent optimizer-based support repair	Appendix U
State history and optimizer counter	Appendix V

I SIMPLE ALTERNATIVES AND THE CHOICE OF REFERENCE

I.1 FULL-DATA QUESTION AND DESIGN

A support mask must be compared with avoiding heterogeneous padding. We extend an earlier 64-image diagnostic to all 5,000 COCO validation images on both released scales. Sorting SHA-256 hex digests of the UTF-8 string `csit-simple-baselines-20260924:`i, with i the unpadded decimal image ID, forms 1,250 consecutive B4 arrival groups. This differs from the original partitions in Appendix G. The protocol is recorded before this full run, but the earlier subset is known: this is an exploratory full-population follow-up, not an independent preregistered confirmation. No targets are selected by outcome. All images enter official COCO AP; 4952 images have eligible GT for score drift.

Six executions share image IDs, resize, checkpoint, groups and post-processing. *Singleton* runs each request on its native canvas. *Uniform B4* and *CSIT B4* retain the heterogeneous group in one forward. *Exact shape* groups only those four arrivals by divisor-rounded singleton shape, then restores output order. *Fixed square B4* pads normalized inputs to 1344×1344 and updates shape metadata; *fixed square solo* supplies its own reference. The last two arms intentionally change the checkpoint's native canvas.

Table 10: Full COCO execution alternatives, using the separate arrival order of Table 2. AP uses all 5,000 images; drift uses 4,952 images with eligible GT. F1 is mean per-image class-aware set agreement with native singleton at score/IoU thresholds 0.5/0.5 (empty–empty equals one), unlike the pooled-count F1 in Section 4.5. Timing is synchronized execution, not arrival-to-response latency.

Model	Execution	Drift	F1	Box AP	ms / B4	img/s
Tiny	Singleton	0.00000	1.0000	47.26	178.7	22.08
	Uniform B4	0.01972	0.9601	47.27	172.4	23.40
	CSIT B4	0.00623	0.9891	47.27	174.2	23.48
	Exact shape	4.04×10^{-8}	1.0000	47.26	156.1	24.90
	Fixed square B4	0.03460	0.9280	47.20	229.6	17.32
	Fixed square solo	0.03460	0.9280	47.20	257.2	15.53
Small	Singleton	0.00000	1.0000	49.21	298.7	13.33
	Uniform B4	0.01742	0.9663	49.22	276.6	14.86
	CSIT B4	0.00453	0.9920	49.14	280.0	14.73
	Exact shape	3.45×10^{-8}	1.0000	49.21	250.7	15.49
	Fixed square B4	0.03128	0.9401	49.06	378.3	10.55
	Fixed square solo	0.03128	0.9401	49.06	412.9	9.67

Two contiguous 2,500-image shards per scale retain complete B4 groups. Image IDs are verified disjoint and exhaustive; all predictions are merged before computing COCO metrics, rather than averaging shard AP. This gives 60,000 image–arm–scale evaluations on 5,000 unique images. Each arm is warmed on the first group of each shard, then evaluated once per group in rotated order, with CUDA synchronization. One process runs on each RTX 5090; the CPU host is shared. Timing includes preprocessing, support setup and prediction, but excludes image decoding, serialization and arrival waiting. It is a descriptive execution measurement, not a service-level or repeated-run timing confidence interval.

Exact-shape grouping averages 3.0752 calls per four arrivals versus one for heterogeneous B4. Its mean padded area is 907,771 pixels per image, versus 1,274,656 for heterogeneous B4 and 1,806,336 for fixed square. Grouping trades more launches for fewer pixels; launch count alone cannot establish a CSIT speed advantage.

I.2 WHAT THE ALTERNATIVES ESTABLISH

Exact-shape drift is 4.04×10^{-8}/3.45×10^{-8} (Tiny/Small), with native-reference set F1 of one. CSIT reduces drift from 0.01972/0.01742 to 0.00623/0.00453 while retaining the heterogeneous forward. The 10,000-draw paired B4-bootstrap intervals for uniform-minus-CSIT drift are [0.01260, 0.01440]

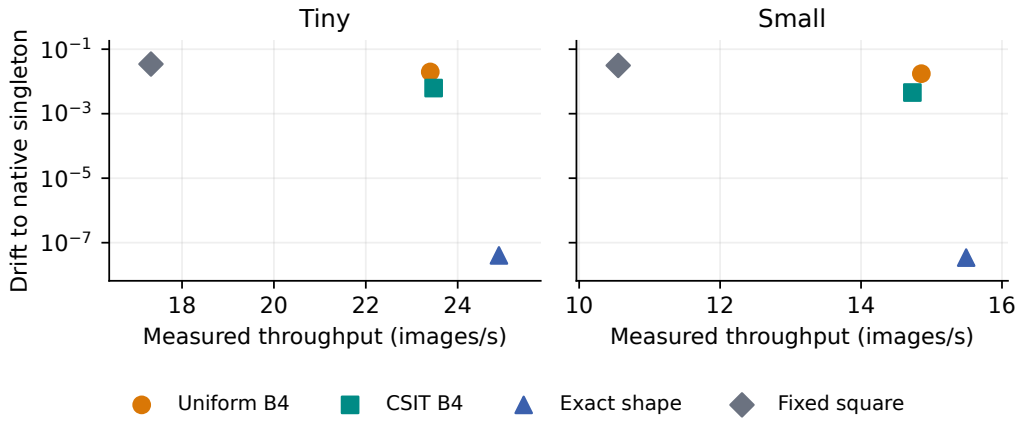

Figure 5: Full COCO comparison with strong execution alternatives. Exact-shape grouping is more consistent and faster in this measured workload. Fixed-square execution is stable relative to a different reference. The drift axis is logarithmic; throughput excludes arrival waiting and decoding.

and [0.01188, 0.01396]. They are exploratory, unadjusted, and conditional on this partition, not independent training runs.

Fixed-square B4 versus its own singleton gives drift $9.55 \times 10^{-8}/8.27 \times 10^{-8}$, but native-reference drift is 0.03460/0.03128. All AP outcomes are retained separately. The earlier subset suggested a fixed-square AP gain; that diagnostic did not justify a full-population gain claim. A system may adopt another fixed canvas, but must validate and declare that reference. Disagreement with native inference alone does not make an intentional contract change an error.

If native behavior is the priority and regrouping is acceptable, exact-shape execution is an attractive first baseline. If a system must retain a heterogeneous batch, CSIT repairs its inner-support axis without reordering it. If future arrivals may be buffered, aspect-ratio sorting is another strong option. No-extra-lookahead is not unique to CSIT, and these observations do not establish a universally optimal scheduler.

I.3 Arrival-time replay with paired full-data execution

To separate execution cost from waiting, we measure four policies on the same ordered 5,000 COCO images per scale: uniform B4, CSIT B4, exact-shape partitioning within each B4, and aspect sorting within disjoint 64-arrival windows. The final window contains eight images. The first three retain the previous B4 memberships; buffer64 changes membership within its window. We rotate policy order across windows, warm each policy, and synchronize each measured forward group. Each scale uses a dedicated GPU on a shared CPU host. These 40,000 image–policy–scale evaluations add no new unique images. AP includes every image; drift uses the same native singleton reference and 4,952 eligible images. Paired drift intervals resample the 79 complete arrival windows, with eligible-image weighting for the shorter final window. Uniform-minus-CSIT drift intervals are [0.01259, 0.01438] for Tiny and [0.01181, 0.01402] for Small (10,000 draws, exploratory and conditional on this order). All three unchanged policies reproduce the earlier per-image GT-matched score vectors exactly on both scales.

Arrival times are synthetic. We use independent exponential interarrival times, and a compound-Poisson alternative that releases eight requests at each event, at the same mean request rate. PCG64 seeds 20260926–20260930 generate 5,000 arrivals per process at 50%, 80%, and 95% of the measured CSIT execution capacity. All policies see identical arrivals within each comparison. This is a single-server replay with fixed group memberships: each group becomes eligible at its fourth arrival, or at the last arrival of its buffer64 window. Groups execute sequentially in policy order and release their four outputs together; exact-shape subgroups do not wait for another arrival cohort.

Table 11: Paired full-COCO scheduler comparison. AP uses all 5,000 images per scale; drift uses 4,952 eligible images. Execution rate is measured without arrival waiting. Latencies are simulated: each entry is the median of five trace quantiles for Poisson arrivals at 80% of the measured CSIT capacity for each scale. No additional per-call cost is imposed.

Model	Policy	AP	Drift	Exec. img/s	P95 (ms)	P99 (ms)
Tiny	Uniform B4	47.27	0.01972	23.87	520	648
	CSIT B4	47.27	0.00623	23.60	535	638
	Exact shape	47.26	$< 10^{-6}$	25.59	461	584
	Buffer64	47.28	0.00315	31.82	4594	5224
Small	Uniform B4	49.22	0.01742	14.99	832	1030
	CSIT B4	49.14	0.00453	14.78	871	1084
	Exact shape	49.21	$< 10^{-6}$	15.83	739	895
	Buffer64	49.25	0.00286	20.53	7244	8241

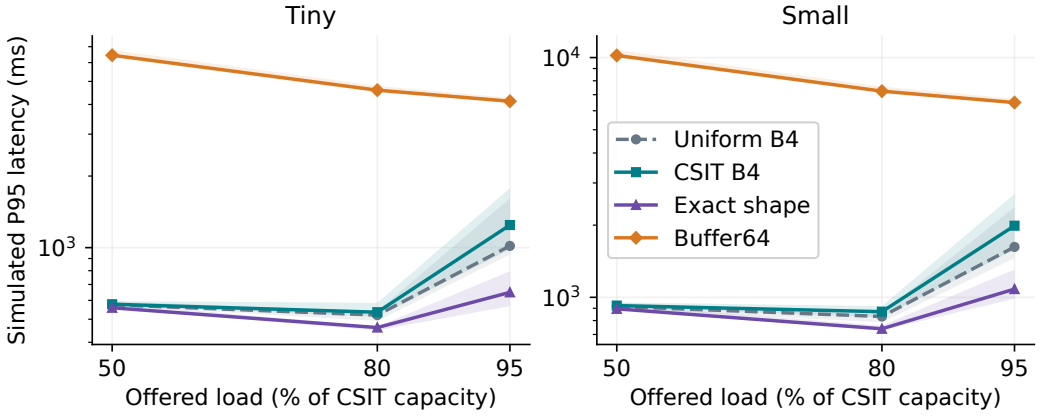

Figure 6: Synthetic Poisson arrivals replayed against measured full-COCO execution times, on logarithmic latency axes. Lines show median P95 latency across five arrival seeds; bands show their minimum and maximum. All policies share arrivals and request order. Buffer64 waits for its window; exact shape partitions the current B4 without extra lookahead. No additional per-call overhead is assumed.

For group g, let a_g be its eligibility time, s_g its measured service time, and d_g its completion time. The replay computes

$$d_g = \max(a_g, d_{g-1}) + s_g, \qquad L_i = d_{g(i)} - a_i^{\text{request}}.$$

We record assembly waiting, queue waiting, total latency, finite-trace output rate, and the queue drain after the final arrival. A separate cumulative-sum implementation verifies the recurrence. Table 11 reports the 80% Poisson case; Figure 6 shows load sensitivity. Shaded ranges describe five arrival draws on one service trace, not repeated GPU measurements or confidence intervals. Decoding, networking, concurrency, and service-time feedback from load are absent; these are simulated latencies, not measured production service levels.

For clustered arrivals at 80% load, median simulated P95 latencies for uniform, CSIT, exact shape, and buffer64 are respectively 2096/2367/1536/5020 ms (Tiny) and 3353/3729/2582/7940 ms (Small). These are a sensitivity check on the same execution trace, not an independent task benchmark. At 80% Poisson load, median finite-trace output rates across the four policies span 18.75–18.88 images/s (Tiny) and 11.74–11.82 images/s (Small); at this load the arrival process limits throughput.

The hypothetical additional per-call costs that equalize mean CSIT and exact-shape service cost are 6.35/8.62 ms (Tiny/Small). The primary comparison adds zero such cost. Sensitivity replays at 5 and 10 ms per call are conditional scenarios, not measured overhead or a deployment speedup claim.

Two elementary calculations clarify the scope. For Poisson requests of rate λ, a fixed cohort of k arrivals has expected per-request assembly wait $(k-1)/(2\lambda)$: position j waits for $k-j$ exponential gaps, and averaging over $j = 1, \ldots, k$ gives the expression. Thus uniform, CSIT, and within-B4 exact grouping all have expected assembly wait $1.5/\lambda$. Buffer64 has $31.5/\lambda$ for complete windows, or $31.4552/\lambda$ averaged over this finite 5,000-request trace including its eight-request tail. This waiting difference is independent of any CSIT mechanism.

Suppose an implementation adds an unmeasured cost h per forward call beyond the measured execution times. With mean group costs $\bar{s}_C, \bar{s}_E$ and mean exact-shape call count $\bar{n}_E$, CSIT has lower mean service cost only if $\bar{s}_C + h < \bar{s}_E + \bar{n}_E h$. This is a sensitivity condition, not evidence that such a cost exists. Neither more calls nor a smaller wait budget establishes a CSIT latency advantage over within-B4 exact grouping.

J Natural-image restoration probe and failed direct transfer

J.1 A different family with a deliberately limited claim

The synthetic-input result in Appendix F motivates checking whether sensitivity survives on natural inputs and whether the same repair transfers. We load the released LOL-v1 TTTIR checkpoint and verify complete state-dictionary coverage after the documented key rename. Sixteen COCO images are selected with a separate SHA-256 salt and resized to 256×256 with bicubic interpolation. These are inputs to a mechanism probe, not low-light input/target pairs. There is no PSNR, SSIM, LOL-test accuracy, or restoration quality claim in this experiment.

For each image we append 32, 64, or 128 pixels of bottom/right reflect padding, run the model, and crop its output back to the original region. We compare three interventions: the released update; the update disabled by returning the incoming transition weight; and a direct support-weighted update. Enabled and support-weighted outputs use the enabled no-padding reference. Disabled outputs use their own disabled no-padding reference, because turning off adaptation changes the function even without padding.

The direct transfer constructs an indicator for the original square, area-resizes it to each inner key/value grid, and divides it by its mean. For the released depthwise product gradient, multiplying values by this map changes the spatial measure before clipping: the raw gradient is linear in these values. The released clipping and state evolution are retained. This construction tests a plausible inner-support intervention; it does not establish that original token features or frequency-derived states are preserved under padding.

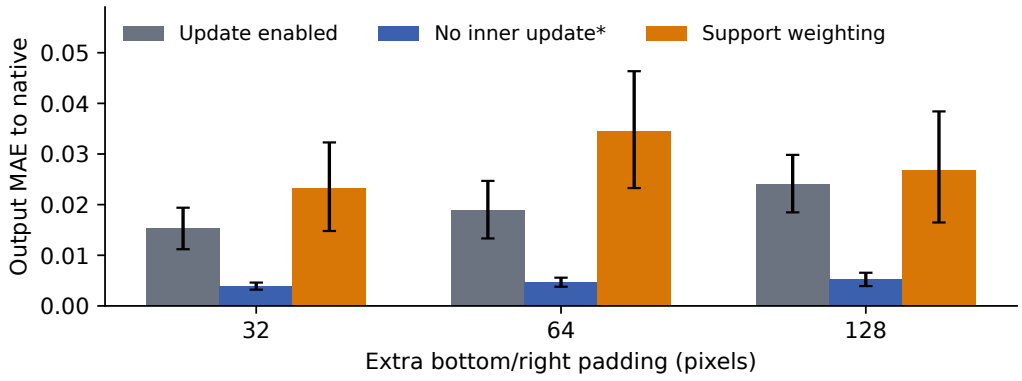

Figure 7: A direct support-weighting transfer fails to restore TTTIR's native output on average. Bars show mean output MAE over 16 natural inputs; intervals are image-bootstrap 95% intervals for each mean. The no-inner arm (*) has its own native reference. These are sensitivity measurements, not restoration quality scores or an additive decomposition.

Pad	Enabled	Disabled	Weighted	Enabled minus weighted 95% CI	Improved
32	0.01530	0.00389	0.02319	$[-0.01678, 0.00059]$	7/16
64	0.01894	0.00467	0.03448	$[-0.02632, -0.00535]$	4/16
128	0.02402	0.00519	0.02674	$[-0.01403, 0.00763]$	8/16

All 16 all-one native comparisons are tensor-exact. The paired intervals resample images 5,000 times and are not corrected across doses. At 64 pixels, the transferred weighting increases average drift, with the paired interval excluding zero in the harmful direction. We retain every dose and image. The evidence strengthens the observation that inner learning can amplify execution sensitivity, while rejecting an unrestricted claim that the detector repair can simply be copied into another architecture.

The patch selects the actual inner-layer class from the instantiated model and asserts nonzero invocation counts: 640 weighted and 640 disabled calls across this probe. An initial implementation smoke accidentally targeted a duplicate Python import of the class; its unchanged outputs were invalidated before analysis. The corrected run uses the same preselected inputs and intervention protocol. This is an instrumentation correction, not a replacement of difficult images. Original input, checkpoint, and runner hashes are recorded. The support and disabled patches are restored after inference, including on failure.

TTTIR builds recurrent state from spatial and frequency pathways. Padding can therefore alter the state supplied to the inner update even where the support indicator remains one. Our finite-measure proposition assumes common token gradients and a common initial state, which this intervention does not guarantee. The negative result is consistent with that limitation, but does not isolate which feature pathway causes the failure. A valid positive transfer would require an architecture-specific support/state correspondence and evaluation on real restoration pairs. We do not count this probe as such a transfer.

K An executable serving contract

K.1 Reference and state ownership

Our inference adapter implements four offline modes: uniform, CSIT, singleton, and exact-shape grouping. The caller supplies model scale, checkpoint path, input paths, batch size, device, and output path. The flattened inference configurations contain the detector architecture and preprocessing and do not inherit a private training-repository path. Released weights, external dataset images, and the MMDetection runtime are separate dependencies. This is a runnable inference adapter, not a network service or a production latency certification.

For this detector, fast weights are derived separately for each image during the forward pass; checkpoint parameters are frozen. The support map and per-stage hooks, however, are mutable Python model state. A concurrent request could overwrite them even though the mathematical learner is per-image. The adapter obtains a model-specific reentrant lock, rejects nested scopes on the same model, clears stale measure state, installs the current support, performs prediction, and removes hooks and measure state even if inference raises an exception. All access to a shared model must use this adapter; an unrelated caller that directly invokes the model can bypass its lock. Parallel workers should own separate model instances in separate processes.

K.2 Request algorithm

1. Apply the released aspect-preserving resize and retain each resized height and width. Form the native canvas by rounding each dimension up to the configured size divisor. Preserve image identifiers and input order.
2. Choose a scheduling policy. Uniform and CSIT preserve the incoming group. Exact-shape mode partitions that group by the pair of native dimensions; singleton mode partitions it into individual requests. Neither searches later arrivals.
3. Acquire the model lock for each forward. For CSIT, create a canonical indicator on the common canvas, resize it with area interpolation to each stage, and apply the mean-one weights to the inner product gradients. Route all-one maps through the literal released operator.

4. Run preprocessing and prediction with the fixed steady-state branch scale. Keep detector post-processing and score thresholds unchanged.
5. Copy predictions in the caller's original order. Restore the uniform measure and original gradient methods even when a forward raises an exception. Release the lock only after cleanup.
6. Record checkpoint hash, mode, batch size, resize shapes, software version, and state-lifetime contract alongside class indices, boxes, and scores.

Why output masking is insufficient. Cropping predictions after a forward cannot undo fast weights learned from extra positions. Masking only a detector loss is likewise irrelevant to a learner whose inner update occurs inside the backbone. The measure must reach the operation that constructs the temporary weights. Conversely, replacing all attention, normalization, and convolution operations with masked versions would define a larger architectural intervention and would require a new singleton-equivalence audit. The current adapter makes only the audited inner-support intervention.

K.3 Checks performed on the adapter

Four focused tests check exception cleanup after an actual stage hook runs, invalid-geometry cleanup and nested-scope rejection, serialization of two threads sharing a model, and stable grouping by canonical rather than raw shape. An end-to-end Tiny CLI smoke runs from an unrelated working directory with four supplied images; its detection-score arrays exactly match the corresponding CSIT benchmark records, and input order and checkpoint hash are verified. These checks complement the operator and experiment-analysis tests; they are not a concurrency load test or a proof of numerical identity across CUDA versions.

Portability and operational limits. The adapter does not promise TorchScript, ONNX, TensorRT, CUDA-graph capture, or dynamic-shape compiler compatibility. The hooks and lock are Python-side implementation choices, and export would require an explicit measure input in the compiled model interface. It does not silently fall back to smaller batches after out-of-memory errors, since that would change the execution being measured. Callers may deliberately select singleton or exact-shape mode, but should log the policy change. The adapter returns boxes and scores; instance masks remain evaluated by the experimental pipeline.

L Distinguishing the contract axes

L.1 An execution record

An inference-time learner needs more than a checkpoint identifier to specify its computation. A useful execution record contains the input transformation, semantic support identities and weights, initial state, state reset rule, update sequence, chunk aggregation rule, clipping/precision, and scheduler wait budget. This is a description of a function, not a demand that all implementations be bitwise identical. A system may intentionally choose a new function; the record makes that choice auditable.

Axis	A change that may look operational	Required comparison
Support	Add padded positions or duplicate tokens	Pushforward mass on canonical identities; fixed-feature gradient control
State lifetime	Reuse a previous request's state	Declared reset/continuation boundary and request-order tests
Chunk boundary	Average unequal partial chunks equally	Common support, initial state, aggregation and update count
Wait budget	Wait for shape-compatible future arrivals	Equal arrival set; request waits and task/reference endpoints
Numerics	Change shape-specific kernels or precision	Same-support duplicate and repeated-forward controls

The support axis has a positive detector repair. The full Watercolor factorial and fixed-position predecessor intervention test state lifetime, including the accuracy cost of complete reset (Appendix V); the chunk probe tests a separate boundary. A wait budget restricts available schedules, rather than entering every learner as a trainable parameter. These are different empirical tests of the contract, not four interchangeable repairs.

L.2 Two small counterexamples

Support changes are not always scalar changes. Take a canonical gradient $(1, 0)$ and an extra-position gradient $(0, 1)$ with equal support masses. Uniform averaging gives $(1/2, 1/2)$, whereas canonical averaging gives $(1, 0)$. No scalar multiple of the uniform vector equals the canonical vector. Radial clipping preserves the distinct directions, so a post-clipping learning-rate scalar cannot generally restore the result. This example motivates the measured same-key/value scalar oracle; it does not claim that every observed detector difference is a direction change.

Equal support does not imply equal update trajectories. Consider scalar losses $\ell_1(w) = (w - 1)^2/2$ and $\ell_2(w) = (w + 1)^2/2$, initialized at zero. One update using their mean gradient leaves $w = 0$. Two sequential updates of size η give $w_1 = \eta$ and $w_2 = -\eta^2$ in this order; reversing the order gives $+\eta^2$. Both sequences see the same examples. Consequently, preserving support and its total mass does not permit changing update frequency, order, or state carry. For unequal chunks, mass-weighted gradient aggregation at a common state can match the full gradient; repeated state updates generally cannot. This is why CSIT is a repair of one contract axis.

L.3 From measure correctness to useful behavior

The preservation theorem is conditional on token-gradient correspondence. In a real model, a larger canvas can change convolutional neighborhoods, positional features, feature normalization, proposal selection, and numerical reductions. The support intervention removes one route but leaves the others. A small fast-weight discrepancy also does not guarantee a small change after thresholding or NMS. A reference-consistency target is therefore an empirical endpoint in addition to the operator identity, not a consequence established by the elementary theorem alone.

An unlabeled geometric observation is particularly useful for auditing whether an inner example is execution-induced. It is less informative about whether retaining that example happens to improve task accuracy. Recent work on identifiability from unlabeled evidence (Jhawar & Wang, 2026) motivates keeping this distinction explicit. We do not apply an impossibility theorem to our detector without checking its assumptions, nor claim that the geometric controller identifies an AP-optimal action. The fixed-square experiment makes the distinction visible without requiring such a theorem.

M Alternative explanations and scope of resolution

Why not ordinary padding masks? Masking is established practice. The contribution is the identification and isolation of execution-induced inner training data, together with a canonical reference that retains native divisor padding. The failed valid-pixel repair shows why the ordinary real-pixel boundary is not automatically the right reference for this checkpoint. We do not claim a new masking primitive.

Why not BatchNorm, GroupNorm, or an attention mask? Group Normalization removes dependence on batch statistics by normalizing within channel groups (Wu & He, 2018). That precedent motivates inspecting which axis an aggregation uses. It does not remove an inner product loss that still averages over extra spatial tokens; spatial normalization itself may also depend on a changed canvas. The released model is evaluated in inference mode, and the identified inner gradients are per-image. A hypothetical normalization swap would alter the checkpoint architecture and is not a matched substitute for the support intervention.

Why not learning-rate tuning or simply disable adaptation? The factorial experiment separates support removal, denominator correction, and post-clipping scalar adjustment on the same targets. The same-key/value oracle tests whether a scalar can recover update direction. Disabling the inner

learner is an attribution control with its own native reference; it changes the learned model rather than preserving the checkpoint's intended TTT behavior. Neither control justifies omitting its utility effects.

Why not crop or bucket inputs? Exact native-shape grouping is a strong measured alternative. Cropping features after they have been mixed with excess context need not reproduce a singleton forward. Re-executing each image on its native canvas does provide that reference, at a scheduling choice and cost that should be measured. Approximate aspect buckets reduce excess area but do not guarantee equal native canvases; full-data buffering experiments already measure this route.

Why not fix the global tensor shape? A fixed canvas can largely stabilize execution under batching. Its correct reference is fixed-canvas singleton. The paired comparison shows that this can still differ substantially from native singleton, without necessarily lowering AP. Choosing fixed shape is legitimate if the revised function and resource cost are acceptable. It cannot establish preservation of a different reference by construction.

Why not deterministic kernels? Batch-dependent floating-point reductions are a separate source of variation. CoRun (Zhao et al., 2026) studies a scheduling approach to deterministic LLM inference. Our duplicated-target controls hold support and canvas fixed to estimate numerical variation, while canvas interventions change the inner empirical problem itself. A deterministic implementation of a changed sum is still a changed sum. Conversely, CSIT does not promise bitwise equality under every kernel, batch size, or precision.

Why not report only AP? AP answers a task-utility question and is reported, including losses. It can remain unchanged while individual decisions move, or improve while the model departs from its declared reference. The fixed-square and buffer counterexamples show why score drift and prediction-set F1 should accompany it. Reference agreement is useful for reproducibility, but is not evidence that the reference is accurate, calibrated, fair, or optimal.

Why not more seeds and significance tests? Different image partitions do not create new independent images or new trained checkpoints. Repeating them can estimate schedule sensitivity, but should not inflate the sample count. Our bootstrap units follow the source of dependence: targets for fixed-target interventions, B4 groups for paired random execution, and sequences/scenes for reconstruction. The 128-image adaptive comparison contains only two buffer windows and cannot sustain a reliable scheduler-level interval. The separate full-data scheduler comparison resamples its 79 arrival windows, preserving the dependence introduced by buffering.

How far does the cross-family evidence extend? CTA-OD provides full COCO support-correction evidence in an independent, variable-size optimizer-based detector (Appendix U). This is not a second fast-weight TTT architecture. VLOD-TTA provides full Watercolor state controls (Appendix V), while native fixed-canvas SAM-TTT is a support null control. TTTIR supplies a real restoration test but fails the support-only transfer claim. These outcomes identify different contract axes; model count alone would obscure their distinct evidential roles.

Why not retrain with padding augmentation? Retraining could teach robustness to a distribution of execution layouts, but it changes the checkpoint and training budget and does not by itself specify which layouts are semantically equivalent. Our question is what existing released checkpoints do under variable-size inference, and what a no-retraining repair can preserve. Robust training and architecture-level support propagation are complementary future directions; no superiority to a retrained baseline is claimed.

N SCOPE OF THE EVIDENCE

The evidence supports three levels of statement. At operator level, a changed empirical measure changes the inner problem, and a scalar correction is not always sufficient. At detector level, controlled interventions and full-data runs show a substantial support-mediated effect and a useful canonical-support repair within one fast-weight family, with independent optimizer-based support

evidence and state controls. At systems level, the results identify alternatives and tradeoffs, but do not establish a universal scheduler or a production SLA. Keeping these levels separate is essential when interpreting the broad title.

Two external validations remain open. A second fast-weight detector must expose native variable-canvas inner support and an explicit canonical state/token correspondence, then pass matched controls and task evaluation. More forward passes on unrelated models do not supply that evidence. A production serving claim requires measured end-to-end latency under a declared workload and wait policy. Our synthetic-arrival replay uses real execution times but omits decoding, networking, and concurrent service effects. It cannot substitute for that deployment measurement.

The reported evidence includes complete 5,000-image COCO execution comparisons for both scales, three full CTA-OD stream orders, full Watercolor state comparisons, and negative transfer tests. Historical controls retain their initialization labels; neither extra seeds nor repeated policies increase the number of unique test images. The appendices specify the reference and scope of each study.

O Natural support exposure across workloads

For each image in the saved Small COCO workloads, define exposure as $d_i = 1 - \rho_i$, using its canonical fraction of the common execution canvas. The response is its mean absolute matched-GT score drift from the same image's native singleton output. Forty-eight empty-GT images are excluded from this endpoint, leaving 4,952 images in each workload; they remain in task AP. Annotation identities are checked before comparing score vectors.

Table 12: Natural support exposure and reference drift for Small on COCO. Ordinary least-squares slopes include an intercept. Paired intervals resample whole execution batches (5,000 draws), preserving all eligible images and both arms in each selected batch. Intervals are exploratory and unadjusted across the six overlapping workloads.

Execution	Images / batches	Uniform slope	CSIT slope	Difference 95% CI
B2	4952 / 2500	0.0664	0.0130	[0.0466, 0.0613]
B4 s0	4952 / 1250	0.0642	0.0102	[0.0488, 0.0597]
B4 s1	4952 / 1250	0.0653	0.0110	[0.0488, 0.0602]
B8	4952 / 625	0.0649	0.0086	[0.0507, 0.0622]
B16	4952 / 313	0.0797	0.0137	[0.0545, 0.0789]
Aspect B4	4952 / 1250	0.0874	0.0487	[0.0119, 0.0706]

We fit ordinary least squares with an intercept separately to uniform and CSIT responses on the same image/exposure pairs. Both fits use the same weights. To estimate uncertainty, each bootstrap draw selects entire execution batches with replacement and retains all their eligible images jointly across both arms. The difference between the two fitted slopes is computed within that draw. The final B16 group contains eight arrivals and is retained with its observed size. Each reported interval uses 5,000 paired whole-batch bootstrap draws; source-stream hashes were recorded before fitting.

All six paired slope-difference intervals exclude zero. This supports an association between exposure and attenuation in these recorded workloads; it is not a within-image causal slope, a monotonicity theorem, or independent replication across six datasets. Exposure co-varies with image aspect ratio, content, and batch composition. Aspect grouping narrows its range to [0,0.2], so its slope should not be extrapolated to the random-batch range. The workloads share the same COCO image pool, and their intervals are not adjusted for multiple comparisons. Causal support comes from the separate fixed-target canvas interventions, which hold target content fixed.

No selector threshold is fitted to these outcomes.

P Fixed-canvas mate-content control

A changed mate may alter either its content or the shared execution geometry. We separate these channels in an additional mechanism check, using 32 annotated near-square COCO targets on each released scale. Targets are ordered by a new fixed SHA-256 salt; the widest and tallest non-target

images supply mate geometries. These targets come from the existing COCO pool, so this is not an independent dataset or a second architecture. The protocol preceded these runs.

For each target, geometry, and uniform/CSIT arm, we preprocess the real pair, then replace only the mate's normalized tensor by its horizontal reflection or by zeros. The target tensor, all metadata, batch index, and canvas stay fixed. These synthetic mate interventions test content coupling; they are not used as task-accuracy data. We assert nonzero mate perturbation and an unchanged target-input hash. We compare the complete target tensors at all four backbone outputs and post-NMS box coordinates, scores, and labels. We do not assert bitwise identity of instance masks, which are outside this control's endpoint.

Table 13: Fixed geometry, changed mate content. Each scale has 32 targets, two geometries, two update arms and two content replacements: 256 comparisons. All feature tensors are bitwise identical. Box predictions are not always bitwise identical; their matched-score differences remain tiny. These counts are diagnostic comparisons, not independent statistical replicates.

Scale	Exact features	Exact box outputs	Maximum score drift
Tiny	256/256	119/256	2.09×10^{-7}
Small	256/256	130/256	1.68×10^{-7}

On the same targets, the original wide/tall mate induces mean drift 0.02959/0.03469 for Tiny and 0.02985/0.02942 for Small relative to native singleton. CSIT reduces those values to 0.00467/0.00626 and 0.00456/0.01117, respectively. The intervention therefore distinguishes backbone content independence at fixed geometry from substantial geometry dependence. Exact feature agreement does not certify the entire detector as bitwise batch-independent; the box output comparison makes this limitation visible. Our evaluation records all 512 content comparisons, tensor hashes, input/checkpoint hashes, and the runner. All failures of exact output equality are retained.

Q Distributional gains and per-image regressions

An average drift reduction need not protect every image. We rejoin the frozen full-COCO B4/B8 prediction streams to each scale's native singleton, verifying image and annotation identities. Each workload contains all 5,000 images; the drift endpoint uses the same 4,952 images with eligible ground truth. Empty-GT images are not assigned zero drift. We report distribution quantiles, all maxima, and increases as well as decreases. An increase exceeds 10^{-6} in image-mean matched-object drift; changes within that tolerance are classified as ties. No samples or outliers are removed.

Table 14: Full-COCO drift tails and regressions. The interval is for the uniform-minus-CSIT difference of the two marginal 95th percentiles, not a percentile of paired differences. It uses 2,000 paired whole-batch bootstrap draws. Intervals are exploratory, unadjusted, and conditional on each saved partition; the four workloads share images.

Scale / batch	P95 uniform	P95 CSIT	Difference 95% CI	Higher drift
Tiny / B4	0.0762	0.0261	[0.0464, 0.0537]	10.42%
Tiny / B8	0.0915	0.0311	[0.0545, 0.0649]	9.94%
Small / B4	0.0692	0.0189	[0.0452, 0.0541]	8.16%
Small / B8	0.0843	0.0228	[0.0574, 0.0658]	7.96%

The 95th percentile falls by 65.8–72.9% across these workloads, with positive paired difference intervals. Between 76.8% and 88.9% of eligible images improve beyond the tolerance, while 8.0–10.4% worsen. The remaining images are ties. This is a distributional reference-consistency result, not an AP improvement or an individual-image guarantee.

The maxima expose a different boundary. Tiny's maximum drift falls from 0.97353 to 0.59326 under both B4 and B8. Small's maximum remains about 0.99153 with CSIT, versus 0.99381/0.99463 with uniform. The largest individual increases occur at COCO image 370818 for Tiny (0.04428 to 0.32728 in B4) and 210915 for Small (0.02710 to 0.46667 in B4); the same images recur under B8. They remain in every aggregate. The saved top-five increases and 99th percentiles enable

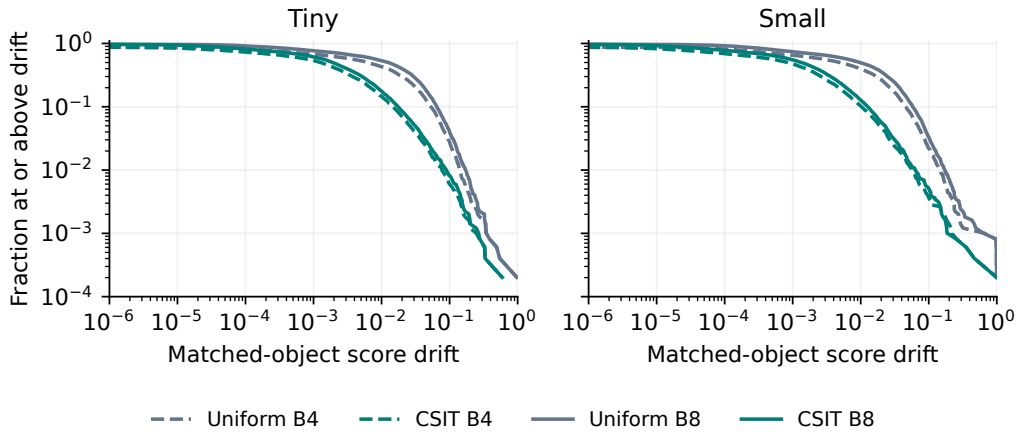

Figure 8: Drift survival curves on full COCO. Both axes are logarithmic; zero-drift observations are omitted only from the plotted log-coordinate, not from the denominator or statistics. Lower curves indicate fewer images above a given drift. No worst-case guarantee follows from the aggregate shift.

inspection without selecting only favorable examples. These outcomes motivate reporting reference distributions alongside means and task metrics.

Compact per-image records allow CPU replay of all tail statistics and the figure. Their provenance includes full-stream and singleton-reference hashes. Three analytic checks cover empty-GT exclusion and image weighting, paired constant shifts, and a two-cluster bootstrap distribution with known endpoints.

R Sensitivity to the object-matching threshold

The drift endpoint selects same-class predictions using a GT IoU cutoff. To test whether its reduction depends on the choice of 0.5, we independently recompute scores from the saved boxes, labels, confidences, and original non-crowd annotations at cutoffs 0.3, 0.5, and 0.7. Reconstruction at 0.5 matches every saved object score to absolute tolerance 10^{-14} before any new result is accepted. The four full workloads each retain all 5,000 images and the same 4,952 drift-eligible images. This follow-up was motivated by the tail audit; it is exploratory, not a predeclared primary analysis.

Table 15: IoU-cutoff sensitivity on full COCO. Differences are uniform minus CSIT; 95% intervals use 2,000 paired whole-batch draws and are unadjusted over the overlapping workloads and cutoffs. No cutoff is selected afterward.

Scale / batch	IoU	Uniform	CSIT	Reduction	Difference 95% CI
Tiny / B4	0.3	0.01610	0.00473	70.6%	[0.01072, 0.01203]
Tiny / B4	0.5	0.01925	0.00615	68.0%	[0.01215, 0.01408]
Tiny / B4	0.7	0.02921	0.01139	61.0%	[0.01584, 0.01973]
Tiny / B8	0.3	0.02149	0.00577	73.2%	[0.01496, 0.01654]
Tiny / B8	0.5	0.02541	0.00745	70.7%	[0.01691, 0.01908]
Tiny / B8	0.7	0.03793	0.01321	65.2%	[0.02265, 0.02683]
Small / B4	0.3	0.01450	0.00358	75.3%	[0.01023, 0.01163]
Small / B4	0.5	0.01741	0.00467	73.2%	[0.01172, 0.01385]
Small / B4	0.7	0.02441	0.00706	71.1%	[0.01582, 0.01885]
Small / B8	0.3	0.01970	0.00446	77.4%	[0.01446, 0.01606]
Small / B8	0.5	0.02325	0.00558	76.0%	[0.01656, 0.01886]
Small / B8	0.7	0.03371	0.00874	74.1%	[0.02314, 0.02686]

All twelve comparisons show lower mean drift, with reductions of 61.0–77.4% and positive paired intervals. Thus the mean conclusion survives these cutoffs. It does not imply threshold-free stability: localization and NMS remain nonlinear, and missed matches contribute zero to this endpoint.

The two largest B4 regressions selected in Appendix Q make that measurement boundary concrete. For Tiny image 370818, a high-confidence same-class detection for annotation 1129098 moves from IoU 0.50248 in singleton to 0.49673 under CSIT, while confidence changes from 0.97992 to 0.98169. For Small image 210915, annotation 646576 moves from IoU 0.50214 to 0.49295, while confidence changes from 0.93321 to 0.93103. Crossing the 0.5 matching boundary creates a large matched-score change without a comparable confidence change. Both adverse cases remain in all aggregates.

S INDEPENDENT TASK PILOTS: SUPPORT VERSUS STATE DOMAIN

S.1 BIOMEDICAL DENOISING WITH OPTIMIZER-BASED TTT

TTT-MIM (Mansour et al., 2024) is a separately trained U-Net that adapts through an explicit optimizer, rather than a fast-weight layer. We use its released checkpoint and all ten supplied fastMRI noisy/GT PNG pairs. These are the authors' selected examples, not the complete fastMRI benchmark. Two slices share one file identifier; there are nine distinct file identifiers, which are not verified patient identifiers. Each image is center-cropped to 224×224 and replicated to three grayscale channels in $[0, 1]$. We use the supplied noise, not regenerated or selected noise. The authors' default fixed-crop pipeline is not claimed to exhibit variable-canvas batching. Our intervention adds 32 or 64 zero-valued pixels right and bottom, holding the canonical image fixed, then evaluates only the original region.

For each image and each of three seeds, we reset both checkpoint and optimizer. We retain the released paired reconstruction/residual losses, AdamW with learning rate 10^{-4}, zero weight decay, eight steps, and coefficient 0.01 on each loss. The learning rate follows the released hyperparameter table. One 1% pixel mask is reused for all eight steps, as in the released online loop. Canonical masked coordinates are coupled across canvas interventions; padding receives separately sampled 1% masked coordinates (rounded up). Uniform and canonical variants have identical inputs, masks and forward GroupNorm; only loss membership and its denominator differ. Ground truth never enters adaptation. No hyperparameter or dose is selected after seeing these results. These are locally specified exploratory pilots, not a held-out confirmation of a medical deployment claim.

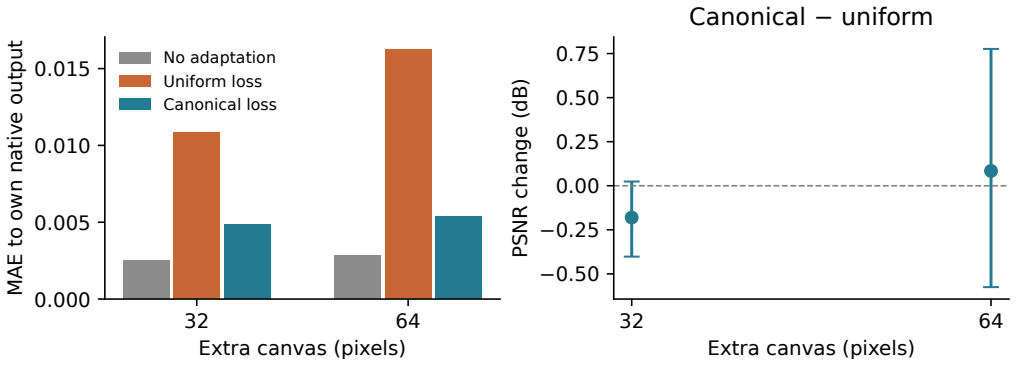

Figure 9: TTT-MIM MRI pilot. Canonical loss support reduces drift, while task utility is mixed. No adaptation uses its own native reference. PSNR changes are canonical minus uniform with 95% file-cluster bootstrap intervals.

Pad	Uniform MAE	Canonical MAE	Reduction [95% CI]	ΔPSNR [95% CI]
32	0.01084	0.00485	55.2% [48.0, 62.8]	-0.181 [-0.402, +0.024]
64	0.01624	0.00536	67.0% [58.3, 74.2]	+0.084 [-0.575, +0.776]

Means average seeds within image. Intervals resample nine file clusters 10,000 times, retaining both slices when their shared file is drawn, and weight the resampled images equally. They are unadjusted exploratory intervals. All 60 image–seed–dose comparisons reduce drift; all 30 native-canvas uniform/canonical checks are tensor-exact with deterministic kernels. Per-seed mean reductions range from 52.5% to 70.4%. PSNR intervals include zero at both doses. Mean SSIM changes are $-0.01005/+0.00623$, also with intervals including zero. Frozen forward drift (0.00253/0.00283) remains below canonical adapted drift: support correction is not full invariance, and GroupNorm and convolutional features still see the enlarged canvas. All arms were evaluated with recorded mask, prediction, and checkpoint hashes.

S.2 REAL LOW-LIGHT PAIRS AND A BROADER STATE-DOMAIN INTERVENTION

We extend the negative TTTIR probe to all 15 LOL-v1 test pairs (Wei et al., 2018), at their native 400×600 resolution. The released enhancement loader and test routine use this resolution directly; we do not import the deraining routine's padding-to-32 rule. State-dictionary coverage is complete. Native predictions agree with the checkpoint package's provided 8-bit result images to at most one intensity level, with mean absolute differences below 0.014 on the 0–255 scale for every image. Those provided outputs serve only as a pipeline check; task scores use the dataset's GT.

We append 32/64/128 reflect-padded pixels right and bottom and crop predictions back to the original region. Five variants are specified before running: uniform; mean-one support weighting of inner values; cropping inner keys and values to also restore the inner convolution boundary; cropping the full state-evolution (STE) module's input to the canonical spatial grid, running its released forward there, and reflect-extending its output; and disabling inner updates. At every resolution the canonical grid is propagated by the known integer scale. No GT, learned parameter, native output cache or additional training is used by a repair. The no-inner diagnostic has its own no-inner native reference. All other variants have the same native reference, verified by 60 tensor-exact checks across four arms and 15 images.

	MAE to own native output			RGB PSNR (dB)		
Variant	32	64	128	32	64	128
Uniform	0.00420	0.00652	0.00766	23.628	23.550	23.394
Support only	0.01145	0.01609	0.02285	23.294	23.088	22.688
Inner crop	0.01156	0.01618	0.02277	23.284	23.072	22.669
STE domain crop	0.00195	0.00206	0.00225	23.806	23.811	23.821
No inner (own ref.)	0.00147	0.00227	0.00320	16.506	16.490	16.457

Native PSNR/SSIM are 23.765/0.85523 with adaptation and 16.504/0.74009 without inner updates. Support-only and inner-crop repairs worsen mean drift at all three doses; their paired improvement intervals are strictly negative. The broader STE-domain intervention reduces drift by 53.7/68.3/70.6%, with image-bootstrap 95% intervals [24.6,70.2]/[47.9,79.3]/[54.5,80.5]%. Its PSNR changes versus uniform are +0.178/+0.260/+0.426 dB, with unadjusted intervals $[-0.074,+0.470]/[-0.097,+0.617]/[+0.063,+0.828]$. The last interval alone excludes zero; we do not select this dose or claim a general restoration-quality improvement. SSIM is 0.85493/0.85501/0.85514 for the STE-domain arm. All statistics retain the full 15-image test set.

What this adds, and what it does not. The loss-only failure remains a result. Cropping an entire state-evolution module restores more semantics than CSIT's inner objective: spatial/frequency target construction, state transitions and their boundaries are all scoped together. It is an architecture-specific diagnostic repair, not unchanged CSIT or a demonstrated efficient heterogeneous-batch implementation. No latency advantage over exact-shape or singleton execution is established. Both pilots use controlled extra canvases on fixed-size inputs, so they expand independent task-level evidence while leaving support repair in a second fast-weight detector family unverified. Their small sample sizes and exploratory status limit generalization.

T SAM-TTT: FULL-SPLIT PROMPTED EXECUTION AUDIT

We audit the independently trained SAM-TTT (Yu et al., 2025) checkpoint (96.3M parameters; all 367 state-dictionary entries match) on complete GrabCut, Berkeley and COCO-MVal splits distributed by RClicks (Antonov et al., 2024). **Both box and boundary prompts use GT**, following the preprocessing referenced by the release. These are prompted transfer scores, not automatic detection or the official click-based benchmark protocol. COCO-MVal is a separate 800-instance benchmark, not full COCO validation. Images are resized to 256^2 then SAM's 1024^2 canvas; every image has 4096 TTT tokens. No natural batch-excess canvas occurs in this pipeline. The TTT update uses image-derived features; the GT prompts enter the separate fusion/decoder path.

Split	Instances/images	Native Dice	Native IoU	No-inner Dice
GrabCut	50/50	97.65	95.61	97.60
Berkeley	100/96	95.73	92.15	95.58
COCO-MVal	800/722	84.25	76.89	83.89

Scores are percentages at original resolution after bilinear probability resizing and threshold 0.5, with ignored pixels excluded. All instances are retained. Per-case prompts and a seeded grouping are held fixed across arms.

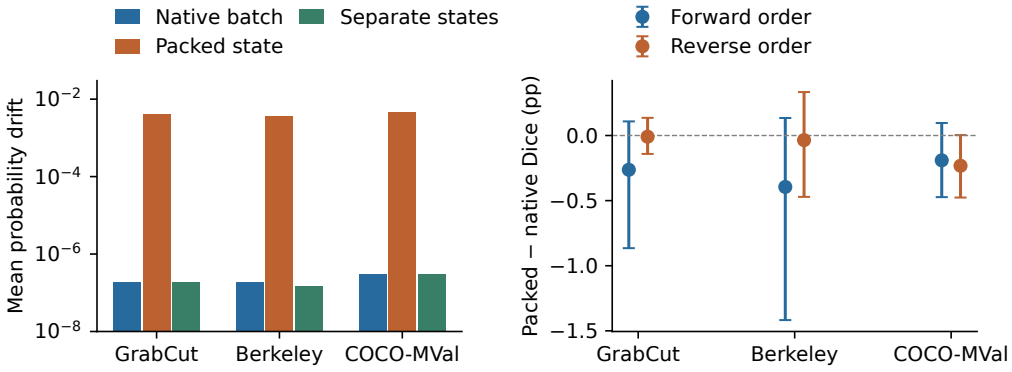

Figure 10: Full-split prompted audit. Native batching preserves reference behavior. Deliberate state packing changes predictions; its Dice effects depend on order. The state-isolated arm uses the same cached image features as the packed arm.

State axis intervention. Native B4 keeps separate state per image. Our *deliberately non-native* stress concatenates the same groups into one state trajectory, preserving per-image Q/K/V convolution boundaries and the released 16-token chunks. Both forward and reverse orders are evaluated. Later images inherit fast weights from earlier images; restoring separate states recovers the reference up to the batch-shape numerical floor. This is not an efficient packing proposal or a fault attributed to the released pipeline. A control additionally joins the convolution boundaries; this is a separate non-native control.

Checks and interpretation. A fixed four-case implementation check matches the actual upstream inference function's binary mask. Disabling inner updates makes its state-packing effect tensor-exactly zero; checkpoint tensors remain unchanged. The full-data task ablation above changes the predictor and is not an inconsistency score. The figure uses 10,000 whole-group bootstrap draws, linking groups that share an identical source image. Intervals are exploratory and conditional on these prompts and this partition; all six Dice-difference intervals include zero. Both orders and all reported arms use the same case IDs and prompts, with input and output hashes recorded during evaluation. This independent family supports the state-lifetime contract and provides a fixed-canvas null control; it does not establish unchanged-CSIT transfer or a second naturally variable-canvas detector replication.

U NATIVE VARIABLE-SIZE DETECTOR TTA EXTENSION

Scope and released model. CTA-OD (Yoo et al., 2024) supplies an independent Faster R-CNN ResNet-50-FPN detector with optimizer-based adaptation. Its released B4 pipeline resizes images while preserving aspect ratio and pads the batch to a common canvas divisible by 32. Its global alignment statistic averages every feature location, including batch-excess padding. This is a naturally variable-canvas detector experiment, but **not a second fast-weight TTT architecture**. We evaluate complete clean COCO val2017, not the original COCO-C robustness task.

Correction and semantic reference. For feature stride s, image size (h_i, w_i) and divisibility $d = 32$, we pool over $\lceil d\lceil h_i/d\rceil/s\rceil \times \lceil d\lceil w_i/d\rceil/s\rceil$ locations, retaining the checkpoint's own singleton padding. Only the global-pooling expression changes; foreground alignment, source statistics, EMA transitions and optimizer settings are retained. CTA-OD intentionally learns jointly from its four images. The reference therefore computes each image's feature/proposal path on its singleton canvas, aggregates all four images' global and foreground statistics, then applies the original loss, EMA transition and one optimizer update. Comparing four independent singleton learners would confound canvas support with learning-data membership. All arms share the same ordered groups, initial weights and adapter initialization.

Complete benchmark. All 5,000 validation images, including empty-GT images, are evaluated in each arm. We use the released SGD settings (learning rate 0.001, momentum 0.9, weight decay 0.0001), the authors' adapter configuration and gradient clipping at 1.0. Predictions precede the update, following the release; adaptation state persists through the stream. Three seeded image orders are evaluated in full, with no validation-AP tuning. A frozen native-batch control is evaluated for order 1. The 10 complete runs total 50,000 image evaluations, not 50,000 distinct images.

Order	Execution	AP	AP50	Confidence drift	Box drift
1	Native	44.289	64.714	0.01375	0.727
1	Canonical	44.294	64.581	0.01108	0.629
1	Reference	44.143	64.606	0	0
1	Frozen	44.165	64.431	0.02124	1.028
2	Native	44.247	64.581	0.01311	0.716
2	Canonical	44.174	64.503	0.01041	0.611
2	Reference	44.148	64.537	0	0
3	Native	44.206	64.605	0.01310	0.706
3	Canonical	44.215	64.589	0.01068	0.613
3	Reference	44.107	64.452	0	0

AP uses the standard COCO evaluator. Confidence drift is absolute score error against reference detections with score ≥ 0.5, with one-to-one class-consistent IoU matching ≥ 0.5; missing matches receive score zero. Box drift is mean absolute coordinate error in original-image pixels, conditional on matching. Reference rows are defined to have zero self-disagreement; their AP need not be the highest. Native AP is already strong and the correction does not materially improve it.

U.1 MECHANISM, CHECKS AND LIMITS

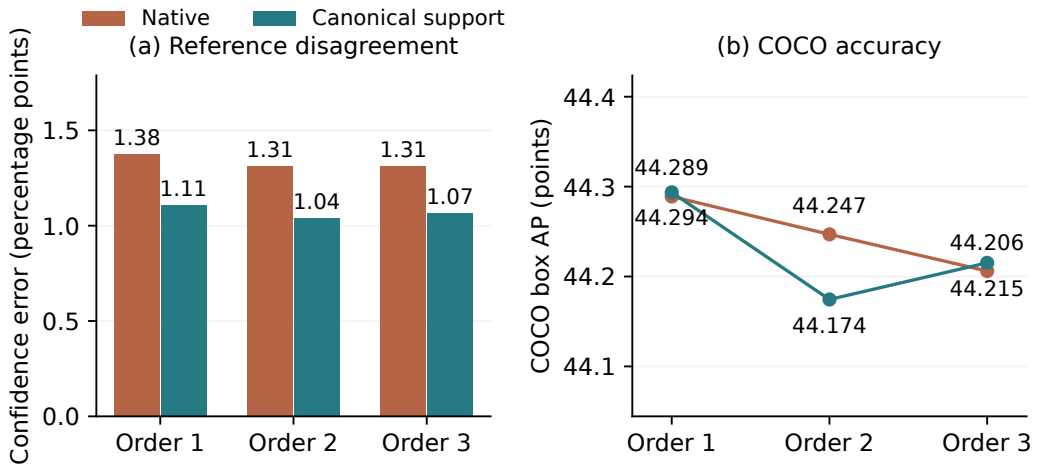

Figure 11: Independent native variable-canvas detector TTA. All three orders use the full 5,000-image COCO validation set and their own semantically matched reference. Missing detections are penalized in confidence error; AP changes remain small. Panel (b) uses a narrow vertical range to make these changes visible.

What changes and what does not. In order 1, confidence drift falls from 0.01375 to 0.01108 (19.5%). The descriptive paired group-bootstrap interval for the absolute reduction is [0.00233, 0.00303] (10,000 resamples). Because the learner carries state, these groups are not independent adaptation trajectories: this interval is conditional on the realized stream and is not a population or independent-seed confidence interval. The complete additional orders provide separate checks of scheduling dependence. Missing-reference fractions are 0.0778% and 0.0808%; their paired interval includes zero. Common-match and conditional-match confidence endpoints are also retained in the evaluation records; no endpoint is substituted after seeing results.

Fixed-state mechanism probe (40 images, separately labeled). The first ten predetermined groups compare global feature means on the batch canvas against singleton feature means at fixed model state. The sum of per-group/per-level error norms falls by 86.3% with canonical pooling. This is a mechanism probe, not a full-dataset quality score. On identical batch features, the correction sets all gradients outside canonical support to zero; best-scalar gradient fits retain a nonzero residual. Thus the change is not just a scalar learning-rate adjustment. The combined foreground-plus-global parameter-gradient distance improves in one implementation-test group and slightly worsens in the other: support repair does not eliminate ordinary feature and proposal-boundary effects.

Implementation and reproducibility. The released checkpoint and statistics are hashed. Every pretrained tensor loads; five serialized anchor buffers equal the currently generated buffers. The only initialization-only tensors are the release's zero-output adapters. The deployed pooling substitution matches the benchmark wrapper's predictions and losses. Singleton predictions and scalar losses remain exact; backward differences are at the repeated-native CUDA RoIAlign numerical floor. Validation labels enter only metric computation. The published source statistics are retained; upstream feature collection switches the loader to the training split. Prediction records, group orders and source hashes were saved for the reported analysis. The pooling patch is reversible. Positive transfer here supports the execution-contract principle in optimizer-based detection TTA; it does not remove the limitation to one positive fast-weight detector family.

V STATE LIFETIME: GRADIENT HISTORY AND OPTIMIZER TIME

VLOD-TTA (Belal et al., 2025) supplies an independent YOLO-World-S prompt/adapter learner. We evaluate all 1,000 official Watercolor test images (1,654 instances, six classes) (Inoue et al., 2018), using the released 640-square B1 pipeline, class/template embeddings, top-600 query objective, AdamW hyperparameters, and one update per image. Labels never enter adaptation. This tests native fixed-canvas state semantics, not CSIT support transfer or a second fast-weight detector.

The release restores adapter/projection parameters after each prediction but retains Adam moments and step counters. Every model buffer is unchanged after every image. We verify the initial adaptive tensors against the actual native evaluation entrypoint, including checkpoint and EMA hooks: their hashes match. Native testing retains constructor-zero adapter outputs and does not invoke the training-time weight-initialization traversal. Eight earlier explicit reinitialization controls are excluded from the native results below.

V.1 COMPLETE-DATA STATE FACTORIAL

We separate carrying gradient moments from carrying the bias-correction count. The four policies below use the same initial model, two frozen orders, and all 1,000 images. The first order is the official split list; the second sorts IDs by a fixed salted hash. All policies restore model parameters after each image. The carry/carry and first reset/reset runs precede this exploratory factorial; the other five full streams were specified before their outputs. No learning rate or threshold is selected using AP. Resetting the counter while retaining moments is an artificial diagnostic, not a recommended optimizer policy.

Moments	Counter	AP, order 1	AP, order 2	Order F1	Score drift
Carry	Carry (release)	29.50	28.46	0.8420	0.07582
Reset	Carry	12.63	12.72	0.8097	0.06717
Carry	Reset	29.10	27.60	0.8653	0.06984
Reset	Reset	18.83	18.83	0.9988	1.04×10^{-7}

AP uses full-split COCO evaluation of the original XML boxes, converted from one-based inclusive coordinates; this split has no difficult objects. The frozen model gives 27.07 AP. The table comprises 8,000 evaluations of the same 1,000 images, plus 1,000 frozen evaluations. It is a controlled audit of released code, not a claim to reproduce the authors' undistributed annotation conversion or exact environment. Order F1 uses class-aware prediction matching at score/IoU 0.5/0.5; score drift averages absolute changes over same-class GT matches within each image, then over all 1,000 images. These are two-trajectory descriptions, not independent-image confidence intervals. Full reset leaves numerical score variation near 10^{-7}; thresholded set F1 need not be exactly one.

Both counter-only and moment-only carryover produce order dependence. Full reset almost removes score drift but lowers AP relative to both frozen and released adaptation. Carrying moments with a reset counter retains much of the released task accuracy in these two orders; that observation neither identifies an optimal state policy nor licenses tuning on this test set. The useful conclusion is that parameter reset does not define an independent episode, and reference independence does not imply task utility.

V.2 What can the counter change without gradient history?

For Adam (Kingma & Ba, 2015) with zero incoming moments, incoming count $t-1$, no weight decay, no AMSGrad, and current gradient g, the next update is coordinatewise

$$\Delta\theta_t = -\eta \frac{A_t g}{B_t|g| + \epsilon} = -\eta \frac{A_t}{B_t} \frac{g}{|g| + \epsilon/B_t}, \quad A_t = \frac{1-\beta_1}{1-\beta_1^t}, \quad B_t = \sqrt{\frac{1-\beta_2}{1-\beta_2^t}}. \tag{4}$$

To see this, one update gives $m_t = (1-\beta_1)g$ and $v_t = (1-\beta_2)g^2$; substituting their bias-corrected values yields Eq. 4. Thus a retained counter changes both the effective step size and epsilon even without predecessor information in the moments. For $(\beta_1, \beta_2) = (0.9, 0.999)$, A_t/B_t is 1 at $t = 1$, 0.744 at $t = 2$, 0.484 at $t = 10$, and 2.515 at $t = 1000$. It is not a monotone attenuation. Float64 AdamW checks at six counts and eight gradient magnitudes agree with this expression to 3.5×10^{-18} maximum absolute error. This is an algebraic diagnostic, not a new convergence result. Comparing a fresh target at step 1 with a target at step 2 cannot, by itself, isolate gradient history.

V.3 Fixed-position predecessor intervention

We therefore hold the target at step 2 and compare two different predecessors at step 1. Six disjoint target/predecessor triples are selected by a salted ID hash from Watercolor, without labels or result-based selection. Before each target, all model tensors are restored to the same initial values; optimizer state alone is retained. Target gradients are bitwise equal across predecessors in all six cases, yet raw-head logits differ by 11.81–18.67. A best nonnegative scalar fit between the two actual parameter-update vectors leaves relative residuals of 0.538–0.999. Changing the predecessor therefore changes more than update magnitude at an identical count and target gradient.

Zeroing both moments while retaining the same counter reduces raw-logit differences to 0–2.00×10^{-4}; identical-history replays give 0–4.90×10^{-4}. Four targets are exactly equal in both controls. The maximum control target-gradient difference is 7.92×10^{-9}, and the largest relative control update difference is 1.19×10^{-5}. These numerical floors are far below the history effect and distinguish gradient-history content from optimizer time in this diagnostic. The six targets are mechanism probes, not an additional task benchmark. Together with the full-split factorial they sharpen the state contract: parameters, gradient memory, and optimizer count have distinct effects and must be specified together.

V.4 Boundary of the Grounding DINO candidate

Using TTAOD-F's released code (Gao et al., 2025) and a local public Obj365+GoldG checkpoint, an eight-image native-canvas diagnostic executes prompt backward and an AdamW update. Aspect-preserving resize is not evidence of a fixed canvas. The relevant observation is that, despite batch-excess tokens, none enters the selected top-900 encoder queries in these probes. Its detection-query loss therefore supplies no demonstrated analogue of the ViT^3 token-mean correction. This is a limited negative diagnostic, not a full corruption benchmark or a claim that all padding effects vanish.

A further feasibility check executes the released teacher–student training loop on the same eight clean images, in two native batches of four. It uses the released weak/strong augmentations, visual deep prompts, AdamW, and mean-teacher update, with instance memory disabled and the same alternative checkpoint. Every released checkpoint tensor is loaded exactly into both models; only the five prompt tensors are additional. Two executions confirm nonzero prompt gradients, student updates, and teacher updates. Their first update has gradient norm 0.0451, student change 39.9, and teacher change 0.0399; the second has gradient norm 0.426, student change 29.0, and teacher change 0.0607 (rounded Euclidean norms; float32 trajectories are not bitwise equal). The teacher remains in evaluation mode while the student trains. Native teacher canvases are 736×1010 and 829×981, each containing differently sized images. These observations establish an executable adaptation loop, not a second CSIT repair or full TTAOD-F benchmark. Neither this diagnostic nor the earlier query audit uses validation labels to train the prompts.