ADAPTIVE OBJECTIVES HAVE CLOCKS: TRANSACTIONAL GRADIENT ACCUMULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Can a training checkpoint preserve its algorithm when memory constraints require a different gradient-accumulation factor? For stateful adaptive objectives, invoking the controller per microbatch changes its observation, state, and possibly random-event sequence. We identify this undeclared coupling in pinned PhysicsNeMo-Sym and PaddleScience integrations and specify an *observe–prepare–realize–commit* contract for an explicit optimizer-step reference. We characterize gradient agreement, establish reference equivalence under stated assumptions, and distinguish controller transactions from statistic reconstruction in a controlled 2×2 study. Paired PaddleScience experiments isolate action-connected clock effects; MultiMNIST checkpoint switches demonstrate prediction changes beyond ordinary accumulation controls. In a five-seed NYUv2 continuation study, Transactional Replay reduces prediction-distance AUC in every seed and task, with median paired reductions of 38–46%, while substantial finite-precision deviation remains. Retained graphs, replay, gradient bases, and an explicitly lagged alternative expose the cost of preserving current-step semantics. Trajectory recovery does not imply better task performance: Replay worsens NYUv2 depth absolute error in four of five seeds. Controller cadence is therefore part of an algorithm's specification, not merely its memory configuration.

1 INTRODUCTION

Suppose training resumes on a smaller-memory device. The checkpoint, effective batch, sample order, optimizer, and learning-rate schedule remain fixed, but the batch is now split into K microbatches. For an additive stateless objective with matched normalization, forward semantics, and randomness, this preserves the accumulated gradient in real arithmetic. Does the same reasoning hold when the objective itself has memory?

Adaptive objectives add a stateful consumer. SoftAdapt and ReLoBRaLo read losses, produce weights, and update histories (Heydari et al., 2019; Bischof & Kraus, 2021). If invoked inside the microbatch loop, the controller advances K times while the optimizer advances once. Neither cadence is intrinsically correct; it requires a declared semantic unit. Accumulation is a physical refinement only when it preserves the observation, action, state, randomness, and commit sequence of that unit.

Pinned PhysicsNeMo-Sym v2.4.0 and PaddleScience paths invoke adaptive aggregators at fragment frequency while documenting accumulation in effective-batch terms; controller cadence is unspecified (NVIDIA Corporation, 2026; PaddlePaddle Authors, 2026). Their SoftAdapt/ReLoBRaLo state mutates per call, exposing an undeclared clock coupling in real independent forward/backward paths.

We ask whether continuation from a checkpoint changes after switching K, how much of that change is attributable to the controller clock, and what it costs to preserve current-step semantics. This matters when reproducing a declared algorithm or resuming it under a new memory budget. When that fidelity is unnecessary, an explicit one-step lag may be a cheaper alternative.

We make three contributions:

1. We identify undeclared controller-clock coupling in a released PhysicsNeMo-Sym path and reproduce it through a pinned PaddleScience `Solver`'s unmodified training loop.

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.

2. We characterize gradient agreement and specify an optimizer-step controller transaction. A 2×2 study separates statistic reconstruction from state/RNG recovery; four executors expose the information and resource choices behind the contract.
3. Paired public-path, MultiMNIST, and NYUv2 studies test causal attribution, fragmentation, and checkpoint portability, separating trajectory recovery from task utility.

2 ONE BATCH, TWO CLOCKS

Suppose a controller declares one optimizer step as its semantic unit. Let z_t denote that unit's statistics, such as the vector of task losses over the effective batch. A stateful controller is

$$(a_t, s_{t+1}) = C(z_t, s_t; \xi_t), \tag{1}$$

where a_t parameterizes the scalarized objective, s_t is persistent controller state, and ξ_t denotes the random-event sequence for one call (empty for a deterministic controller). In the pinned SoftAdapt and ReLoBRaLo implementations, the resulting action is treated as a non-differentiated weight vector. Other declared units, such as one dataloader batch, define different but potentially valid algorithms.

Now partition the same ordered effective batch, with fragment observations $z_{t,1}, \ldots, z_{t,K}$. Native execution instead performs

$$(a_{t,k}, s_{t,k}) = C(z_{t,k}, s_{t,k-1}; \xi_{t,k}), \qquad s_{t,0} = s_t, \tag{2}$$

$$g_t^{\text{NATIVE}} = \sum_{k=1}^{K} B(z_{t,k}, a_{t,k}), \tag{3}$$

where B denotes a fragment's contribution to the accumulated gradient, not a parameter update. If R is the declared effective-batch reduction and $T_{z,\xi}$ is the state component of C, then in general

$$T_{z_{t,K},\xi_{t,K}} \circ \cdots \circ T_{z_{t,1},\xi_{t,1}}(s_t) \neq T_{R(z_{t,1},\ldots,z_{t,K}),\xi_t}(s_t). \tag{4}$$

The left side depends on fragmentation, order, and K; the right side is the transition specified by the optimizer-step declaration.

Definition 1 (Controller-clock mismatch)**.** Let a controller declare one transition per semantic unit. A controller-clock mismatch occurs when a physical execution of that unit invokes, orders, commits, or consumes randomness for a different transition trace. Under gradient accumulation, invoking the controller once per fragment creates K transitions inside a unit declared to contain one.

Definition 2 (Declared-clock consistency)**.** For a controller whose declared semantic unit is one optimizer step, an executor is clock-consistent for C and reduction R if every partition of the same ordered effective batch preserves the reference's observation, action, committed state, and random-event sequence, with one controller commit and the same optimizer-transition result. Exact consistency refers to real arithmetic; numerical agreement requires a separately stated tolerance.

Floating-point reduction can differ even for Sum, but it does not add controller transitions. Stateful calls can change both the current action and the state subsequently observed.

For linear scalarization, the link from controller events to the accumulated gradient is exact. Let $G_k \in \mathbb{R}^{P \times m}$ contain the m normalized task gradients contributed by fragment k. Native uses action $a_k \in \mathbb{R}^m$, whereas the declared whole-step execution uses one action $a^\star$.

Proposition 1 (Accumulated-gradient identity)**.** *For fixed, non-differentiated controller actions, define* $g^{\text{NATIVE}} = \sum_k G_k a_k$ *and* $g^{\text{ref}} = \sum_k G_k a^\star$*. Then*

$$g^{\text{NATIVE}} - g^{\text{ref}} = \sum_{k=1}^{K} G_k (a_k - a^\star). \tag{5}$$

For fixed $G_{1:K}$, the accumulated gradients agree if and only if the sum on the right is zero. Holding these actions fixed, if agreement is required for every independently variable $G_k \in \mathbb{R}^{P \times m}$, it holds if and only if $a_k = a^\star$ for every fragment. Equality of the accumulated gradient alone is not full semantic equivalence: the terminal controller state, consumed randomness, and commit trace must also equal the declared whole-step trace.

Parameter motion is a second layer. Let optimizer state be o_t and write

$$(\theta_{t+1}, o_{t+1}) = \mathcal{U}_t(\theta_t, o_t, g_t; \lambda_t), \tag{6}$$

where λ_t collects the update configuration. Equal $(\theta_t, o_t, g_t, \lambda_t)$ gives the same next pair for a deterministic optimizer transition, including Adam or AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). We do not claim the converse: different gradients can occasionally map to the same finite-precision update.

Cancellation for one batch does not establish algorithmic equivalence. A scalar correction is universally exact only if $\alpha_k a_k = a^\star$ for every fragment, and cannot itself restore state/RNG (Appendix B).

Example: exponential-state dilation. Suppose a controller buffer follows $s^+ = \beta s + (1-\beta)u$ per call. With constant observation u, K Native calls in one optimizer step give

$$s_{t+1}^{\text{NATIVE}} = \beta^K s_t + (1-\beta^K)u,$$

whereas one semantic call gives $s_{t+1} = \beta s_t + (1-\beta)u$. Thus changing only K changes the effective decay from β to β^K.

For this constant scalar case only, per-call $\rho = \beta^{1/K}$ restores the terminal coefficient. With heterogeneous observations, however, it yields $\rho^K s + (1-\rho)\sum_k \rho^{K-k} u_k$, an ordered statistic generally different from one declared reduction; nonlinear histories and RNG add more mismatch. Beta-root compensation is therefore a controller-specific approximation, not transaction semantics.

3 Released paths expose an undeclared clock coupling

PhysicsNeMo-Sym v2.4.0 invokes its loss aggregator inside the `grad_agg_freq` loop before each backward; its guide describes the option as changing effective batch size. Its SoftAdapt and ReLoBRaLo mutate per call (NVIDIA Corporation, 2026). PaddleScience commit `d57561b` likewise invokes ReLoBRaLo each iteration but gates the optimizer by documented `update_freq` (PaddlePaddle Authors, 2026). Neither declares controller cadence, so we call this an *undeclared coupling*, not a violated maintainer contract. A matched litmus records 1/4/8 calls per update at $K1/K4/K8$ in both paths. Exact source evidence and the fixed-panel audit—a composition-boundary analysis, not a prevalence estimate—are in Appendix D.

4 An optimizer-step reference executor

To make an optimizer-step reference executable, we express one logical step as a four-phase controller transaction:

1. **Observe**: compute sufficient statistics for every physical fragment without committing controller state or randomness;
2. **Prepare**: reduce the observations and evaluate $(a_t, \tilde{s}_{t+1}) = C(R(z_{t,1:K}), s_t; \xi_t)$ exactly once;
3. **Realize**: obtain fragment gradients under the single prepared action a_t;
4. **Commit**: after the optimizer update succeeds, publish $\tilde{s}_{t+1}$ and its logical RNG position once.

The resulting accumulated gradient is

$$g_t^{\text{TR}} = \sum_{k=1}^{K} B(z_{t,k}, a_t), \qquad (a_t, s_{t+1}) = C(R(z_{t,1:K}), s_t; \xi_t). \tag{7}$$

An aborted observation or realization leaves controller state and logical RNG unchanged; tentative gradients must be discarded before retry. This is a controller transaction, not an atomic optimizer or process transaction. Failures during an optimizer update require separate recovery machinery.

Why one cannot simply move the call. Consider two tasks and two fragments. The current whole-batch losses determine $a^\star = C(R(z_1, z_2), s; \xi)$, but z_2 is unknown while fragment 1 is processed. If the executor already backpropagates $G_1 a_1$ and frees its graph, calling the controller once after seeing z_2 cannot retroactively replace that contribution by $G_1 a^\star$. It must preserve, recompute, or retain enough information to realize the later action, or explicitly change the algorithm to use an action known in advance.

Table 1: Ways to make the action available to all fragments. The lagged transaction executor (LTX) uses an explicitly lagged algorithm rather than the original current-step controller.

Executor	How the action becomes available	Principal trade-off
Retained	keep forward graphs until statistics are complete	activation memory
Replay	collect statistics, then recompute forward	extra forward work
Gradient Basis	retain task gradients for later combination	gradient compute/storage
LTX	use the preceding logical step's action	a different, lagged algorithm

Transactional Replay. TRANSACTIONAL REPLAY implements Eq. (7) with two passes. The observation pass computes detached per-task losses and discards its activation graphs. After one controller prepare, the realization pass repeats the same fragment forwards with the prepared weights and performs ordinary accumulated backward calls. The optimizer is invoked once after realization; tentative controller state and its logical RNG position are published only after that update succeeds. Hereafter, Replay denotes this transactional executor. It preserves the streaming activation memory advantage at the cost of extra forward computation.

Proposition 2 (Reference refinement equivalence)**.** *Start from identical model, optimizer, controller, and RNG states. Assume R reconstructs the whole-step statistic, fragment contributions use effective-batch normalization, and stochastic operators replay the same logical-sample random stream. Assume also that the detached prepared action has additive fixed-action derivatives and that replay neither changes forward semantics nor commits extra model-buffer updates. In real arithmetic, Transactional Replay produces the same controller transition and accumulated gradient as the unfragmented $K{=}1$ execution. If Eq. (6) also has identical inputs and is deterministic, it produces the same next model and optimizer state.*

The proof is in Appendix B. In floating point, reduction and kernel topology leave a measured one-step residual. Long-run trajectory recovery is a separate empirical question: a small local residual does not guarantee that two nonlinear training trajectories remain at the Sum floor.

Without retained graphs or recomputation, a same-step executor needs a derivative summary sufficient for every reachable action. A linear sketch over arbitrary $P{\times}m$ gradients requires $d \geq Pr$ coordinates for action-span rank r; a task-gradient basis attains mP at full rank. This worst-case linear bound excludes structured/approximate alternatives (Proposition 3, Appendix B).

LTX fixes the previous step's action over all fragments and prepares the next action from one whole-step observation. Given identical initial lagged action, controller/optimizer state, and the conditions of Proposition 2, induction gives refinement equivalence for this lagged algorithm in real arithmetic. Finite-precision trajectory deviations require separate measurement.

5 CAPABILITY BOUNDARY AND PAIRED STUDIES

Two independent failure axes. Existing accumulation repairs often reconstruct a whole-batch statistic. That can be sufficient for a stateless nonseparable objective, but not for a controller whose observation also advances state or RNG. We isolate these axes in a float64 2×2 matrix with three fixed seeds, $K{=}8$, and 48 steps. Numerical matching uses tolerance 10^{-10} for gradient, action, state, and parameter residuals, plus exact RNG traces. Statistic replay is used as a capability oracle in this controlled matrix; it is not a verbatim reimplementation or empirical evaluation of GradCache or DeePM.

In this matrix, naive GA fails for either nonseparability or controller clocking. Statistic-only replay repairs only the former. Replay matches the reference in all four cells and passes controller rollback and checkpoint-resume tests in every seed. Appendix E reports the measured residuals and validation contract.

Public-path clock attribution study. The 10,000-update Euler–Bernoulli beam study uses the PaddleScience path, five paired seeds, a fixed validation grid, and Native ReLoBRaLo, Transactional Replay, Sum, stateful-null, and decay-root conditions at $K1/K8$. The stateful-null advances

Table 2: Controlled capability boundary, not a comparison of published implementations. Match means 3/3 seeds meet the numerical and trace checks; fail means 0/3. Statistic replay reconstructs the whole-batch statistic but keeps fragment-clocked controller transitions. Transactional Replay reconstructs the statistic and commits state/RNG once.

Objective/controller	Full reference	Naive GA	Statistic replay	Transactional Replay
Stateless + additive	reference	match	match	match
Stateless + nonseparable	reference	fail	match	match
Stateful/RNG + additive	reference	fail	fail	match
Stateful/RNG + nonseparable	reference	fail	fail	match

state/RNG but applies equal weights; decay-root uses $\alpha^{1/8}$. Protocol, estimands, and all task-space directions were fixed before execution. A descriptive transfer adds Native/Transactional $K4$.

Non-PINN fragmentation-transfer study. MultiMNIST uses a shared GroupNorm CNN (Wu & He, 2018), two classification heads, five fresh paired seeds, effective batches of 256, AdamW, and 2,340 updates. Sum, Native SoftAdapt, statistic replay, and Transactional Replay cross $K \in \{1, 2, 4, 8\}$ and the unequal partition $32 + 96 + 48 + 80$. A second matrix branches identical midpoint checkpoints into uninterrupted, resume-same, and bidirectionally switched $K1/K8$ runs. The validation panel has 1,024 examples; all 10,000 test examples serve descriptive endpoints.

A baseline extension adds LTX $K1/K8$, comparing its own fragmentation consistency and its fidelity to current-step Native $K1$ separately.

For the preceding studies, parameters θ and fixed validation predictions or logits y define the whole-trajectory distances

$$A_i^z = \frac{1}{T}\int_0^T \frac{\|z_{i,t}^{\mathrm{ref}} - z_{i,t}^{q}\|_2}{\max(\|z_{i,t}^{\mathrm{ref}}\|_2, \|z_{i,t}^{q}\|_2)}\, dt, \qquad z \in \{\theta, y\}, \tag{8}$$

using a fixed trapezoidal rule. We also report action/state distances, one-step residuals, commits, all seed-level endpoints, and matched Sum controls. "Floor" denotes this empirical control, not a lower bound. Medians and seed ranges are descriptive; five paired seeds do not establish population-wide significance. Earlier downstream studies are unpooled (Appendix H).

NYUv2 checkpoint continuation. A five-seed NYUv2 (Silberman et al., 2012) integration follows the LibMTL recipe (Lin & Zhang, 2023) with a pretrained dilated ResNet50, three ASPP heads, batch 8, Adam, and 2,000 updates. At update 1,000, algorithm-matched checkpoints branch into SoftAdapt $K1$, Native/Replay with $K = 8, 2, 4, 8$ in 250-update windows, and Sum controls. BatchNorm statistics are frozen; dropout is keyed by logical sample occurrence, not fragment index. Task-wise prediction AUC uses a fixed 16-image, stride-4 probe and the *reference* norm as denominator; all 654 images serve full-resolution endpoints. This is an internal integration, not the unmodified Trainer or a convergence study (Appendix A).

Executor and resource protocol. A shared API implements all four executors with explicit state, RNG, abort, and checkpoint behavior. Float64 oracle checks precede a 24-run GPU benchmark: SoftAdapt/ReLoBRaLo, a six-layer three-task convolutional regression model, and $K \in \{1, 4, 8\}$. We record CUDA event time, peak allocated memory, semantic residuals, controller calls, VJPs, and gradient storage.

6 Results

6.1 MultiMNIST: changing fragmentation at a checkpoint

At update 1,170, each serialized checkpoint is restored either with the same K or with $K1 \leftrightarrow K8$. Resume-same parameter and logits arrays are bitwise identical to uninterrupted runs in all seeds. A Native switch changes at least one final predicted digit for median 1.17%/1.05% of test examples in the two directions; Transactional changes 0.20%/0.25%, versus Sum's 0.17%/0.24% (Figure 1).

These post-hoc descriptive prediction changes do not imply degradation: median accuracy changes are below 0.06 percentage points and have mixed signs.

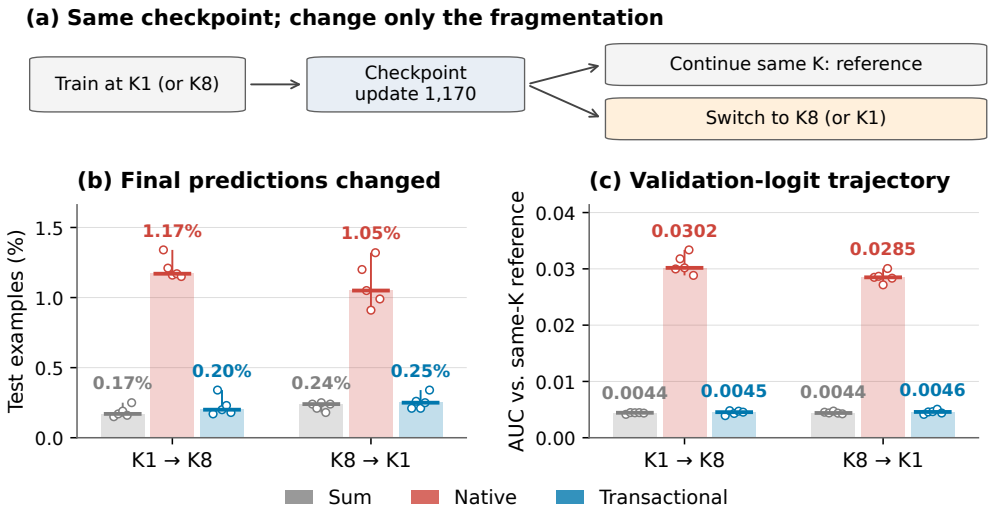


Figure 1: Checkpoint portability. (a) Each execution branches at the same midpoint checkpoint. (b) Fraction of test examples with either predicted digit changed at update 2,340; this endpoint analysis is post-hoc and descriptive. (c) Validation-logit trajectory AUC relative to the uninterrupted same-K execution. Bars/lines are medians, circles are all five seeds, and whiskers span seeds. Resume-same controls are zero. Prediction change is not degradation.

Median logits AUC is 0.03017/0.00443/0.00454 for Native/Sum/Transactional under $K1 \to K8$, and 0.02849/0.00440/0.00458 under $K8 \to K1$. Native exceeds Sum in both trajectory spaces in 5/5 seeds and both directions; Transactional medians remain close to Sum. Parameter AUCs are in Appendix F.

6.2 Checkpoint continuation on NYUv2

All 25 branches complete the fixed horizon. Native prediction AUC exceeds the matched Sum control in all five seeds for each task; Replay reduces Native AUC in all 15 seed–task comparisons (Figure 2). Median Native/Replay/Sum AUCs are 0.220/0.143/0.146 for segmentation logits, 0.103/0.054/0.064 for depth, and 0.240/0.131/0.147 for normal outputs. Median paired raw recovery is 38.2%/45.6%/46.3%, respectively; parameter recovery is 48.0%. Lower Replay medians than Sum are descriptive, not evidence of a universal numerical bound or exact long-run invariance.

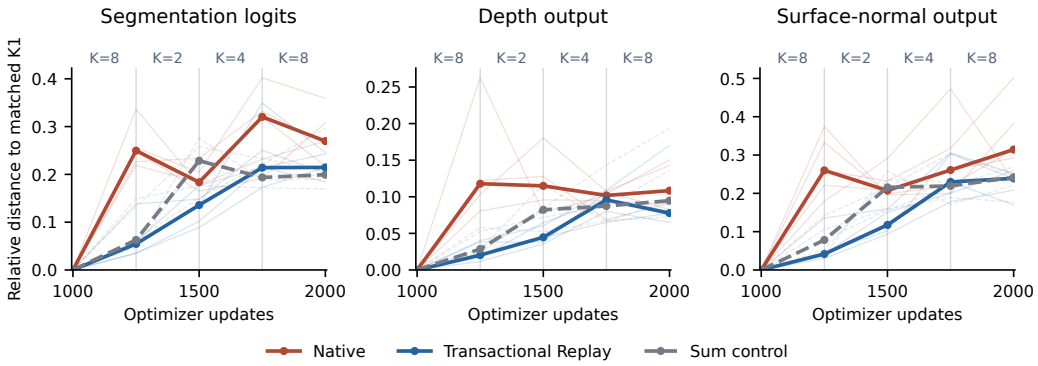


Figure 2: NYUv2 checkpoint continuation. All arms load their algorithm's $K1$ checkpoint at update 1,000; references remain $K1$, while switched arms follow the displayed schedule. Thin curves show all five seeds; thick curves are pointwise medians. Distances use a fixed spatial prediction probe. Replay reduces Native deviation but does not eliminate the sizable Sum-like background. Prediction difference is not prediction degradation.

Endpoint quality is mixed: Replay improves segmentation mIoU in 3/5 seeds and mean normal angular error in 4/5, but worsens depth absolute error in 4/5. These complete endpoints, including all losses and auxiliary metrics, are retained; trajectory recovery is not an accuracy-superiority criterion.

6.3 Clock attribution over 10,000 PaddleScience updates

All eight executions complete for each of the five paired seeds. Median prediction AUC is 0.05892 for Native $K8$, compared with 0.00932 for the stateful-null and 0.00750 for Sum; parameter AUC is 0.10138, compared with 0.00034 and 0.00036. Native exceeds both controls in every seed. These controls support attribution to action-connected clock changes rather than additional state/RNG transitions alone.

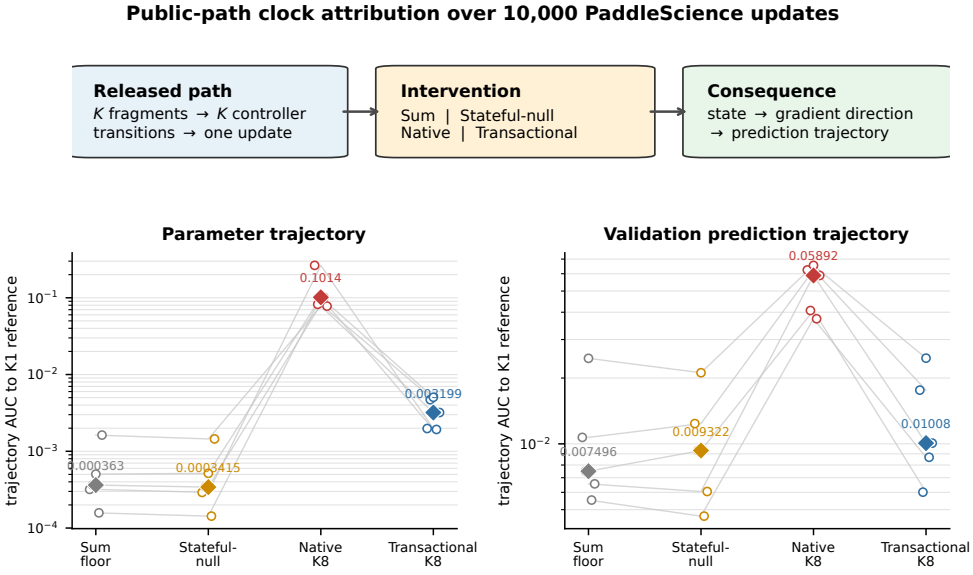


Figure 3: Clock attribution on the released PaddleScience path. Native $K8$ exceeds both ordinary accumulation and an action-disconnected stateful-null; Transactional Replay returns prediction and parameter trajectories toward the $K1$ reference. Open circles are paired seeds and diamonds are medians.

Transactional Replay has median prediction/parameter AUCs of 0.01008/0.00320, with every absolute result shown in Figure 3. Decay-root compensation does not repair this setting: its median prediction/parameter AUCs are 0.09870/0.16294, both larger than Native's. Task utility is separate: Native has higher final solution error than $K1$ in 4/5 seeds, Transactional is lower than Native in 3/5, and PDE and boundary directions are mixed. A follow-up $K4$ transfer reports Native/Transactional prediction AUCs of 0.0403/0.0143, but supports neither a seed-wise monotonicity nor matched-$K4$-floor claim.

6.4 Same-step recovery across fragmentation

The MultiMNIST float64 one-step oracle has Transactional gradient residual at most 5.83×10^{-16}, including unequal partitions, with one commit; Native and statistic replay retain K transitions and large residuals. Over 2,340 updates, Native action/state AUC rises strictly with K in 5/5 seeds, but parameter AUC is monotone in none and logits AUC in only 3/5. Transactional median model-side AUCs remain at or below the matched Sum control across tested partitions, without systematic accuracy gains. Appendix F retains the complete dose figure, absolute effects, raw recovery, and endpoint directions.

6.5 Choosing current-step fidelity or explicit lag

All 24 SoftAdapt/ReLoBRaLo executions at $K \in \{1, 4, 8\}$ are valid. At $K8$, Replay uses about 32% of Retained peak memory but 1.14–1.37$\times$ its time (Table 3). Basis retains task gradients; no same-step implementation minimizes all three resource costs.

Table 3: Controlled executor resource comparison at K=8. Ranges span SoftAdapt and ReLoBRaLo. Time is median CUDA event time per step; memory is peak allocated MiB. "Current" denotes the real-arithmetic reference, not bitwise agreement; numerical residuals are reported below. LTX uses the previous action.

Executor	Time (ms)	Memory (MiB)	Semantics
Retained	26.74–28.32	493.7–495.2	current
Replay	32.33–36.66	158.7	current
Basis	45.42–51.54	215.8	current
LTX	29.84–30.19	158.7	lagged

A subsequent MultiMNIST extension trains LTX $K1/K8$ *from scratch* under the same five seeds and 2,340-update horizon; it does not test an LTX mid-training K-switch. Table 4 separates its own fragmentation deviations from fidelity to the original current-step controller.

Table 4: LTX comparisons: five-seed medians from fixed-K, from-scratch runs. "Changed" is the test-example fraction with either predicted digit changed relative to the stated reference.

Run vs. reference	Param. AUC	Logit AUC	Changed
Within-algorithm fragmentation deviations			
LTX $K8$ vs. LTX $K1$	0.0442	0.0371	1.20%
Sum $K8$ vs. Sum $K1$	0.0831	0.0584	1.30%
Fidelity to the original current-step algorithm			
LTX $K1$ vs. Native $K1$	0.1153	0.0919	2.21%
Transactional $K8$ vs. Native $K1$	0.0462	0.0421	1.32%

LTX's median parameter/logit deviations are below Sum's, each in 4/5 seeds; this supports its design without proving numerical equivalence. Action/state AUCs are 0.0194/0.0213. Against current-step Native $K1$, median mean accuracy changes by +0.02 percentage points (3/5 improvements; mixed task CE). LTX is cheaper than Replay here; Replay buys current-step fidelity, not a demonstrated accuracy advantage.

Replay residual is at most 2.01×10^{-7}; Basis reaches 2.69×10^{-5}, while a float64 diagnostic gives 1.61×10^{-16}, consistent with sensitivity to numerical reduction.

7 IMPLICATIONS

Execution choice depends on the reference one wishes to preserve. When reproducing a current-step controller or continuing its checkpoint, Replay retains that algorithm at an extra-forward cost. When a one-step lag is acceptable, LTX avoids replay and is cheaper in our controlled benchmark. Its favorable within-algorithm results do not establish LTX checkpoint-switch portability, which remains untested. Neither choice promises better task utility.

Given user-supplied integration metadata, `ClockGuard` warns about stateful microbatch calls, enforces controller commits, and restores controller state/logical RNG on abort. It neither detects integrations automatically nor intercepts optimizer updates. The tested PaddleScience retries additionally rewind the deterministic data cursor (Appendix G).

8 RELATED WORK

Adaptive weighting rules such as SoftAdapt and ReLoBRaLo specify how losses control weights (Heydari et al., 2019; Bischof & Kraus, 2021); we study when that rule is executed. Batch normalization and token-loss normalization expose other accumulation boundaries (Ioffe & Szegedy, 2015; Hugging Face, 2024). GradCache and DeePM reconstruct information for nonseparable losses (Gao et al., 2021; Wood et al., 2026). We do not claim replay itself as new: our contract additionally specifies controller state, random events, and commits. The controlled comparison in Table 2 isolates this distinction, not those methods' empirical performance. Appendix C relates the executors to gradient bases, set consistency, and rematerialization.

9 LIMITATIONS

Our source audit establishes existence in two pinned public paths, not prevalence across training systems. Replay requires detached actions, reconstructible statistics, additive fixed-action derivatives, and replayable forward behavior. MultiMNIST and NYUv2 are internal non-PINN implementations, not external replications. NYUv2 freezes BatchNorm, keys dropout, subsamples its trajectory probe, and stops after 2,000 updates without a convergence claim. Nonmonotone model-side dose, substantial Sum deviations, and mixed endpoints limit extrapolation beyond the tested settings. Controller rollback does not cover optimizer failures, AMP scalers, or the whole process.

10 CONCLUSION

Stateful fragment calls can change the algorithm. A declared transaction preserves current-step semantics under its stated assumptions and reduces checkpoint-switch trajectory deviations in the tested tasks; it does not guarantee floating-point trajectory identity. Explicit lag is cheaper when a different algorithm is acceptable. Task utility remains a separate question.

AI USE STATEMENT

Generative AI tools, including OpenAI Codex, assisted research discussions, hypothesis and experiment design, development of the conceptual framework, formulation and refinement of mathematical claims and proofs, implementation and execution of experiments, synthetic test-data generators, data-processing and analysis code, result interpretation, literature checking, figure and table preparation, translation, and manuscript drafting and editing. Measurements were obtained from executed code. Verification includes the recorded oracle tests, trace checks, and separate mechanical validators; these checks are not independent scientific replication or a claim of complete human verification. The authors are responsible for the final text, claims, citations, code, and results, including all AI-assisted contributions.

REPRODUCIBILITY STATEMENT

Sections 2–4 state the execution assumptions, with proofs in Appendix B. Section 5 and Appendices A, E, and F describe the paired designs and estimands; Appendices D and G identify pinned source paths and retry scope. The accompanying artifact contains code, protocols, manifests, traces, seed-level metrics, and mechanical validators. Large NYUv2 and MultiMNIST portability raw arrays are inventoried but not included; raw-array revalidation requires separate local archives (see the artifact availability inventory and Appendix A). Summary checks do not remove this availability limitation.

ETHICS STATEMENT

This work evaluates training executions using numerical problems and existing image benchmarks. It does not establish the suitability of the resulting models for decisions about people or safety-critical physical systems. Benchmark images may contain personal or private-scene information; downstream use and redistribution require attention to the original data terms and privacy constraints. We report mixed and unfavorable outcomes alongside trajectory recovery and do not attribute intent or ecosystem-wide prevalence to the two audited software integrations.

REFERENCES

Rafael Bischof and Michael Kraus. Multi-objective loss balancing for physics-informed deep learning. *arXiv preprint arXiv:2110.09813*, 2021.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP*, pp. 316–321, 2021. doi: 10.18653/v1/2021.repl4nlp-1.31. URL https://aclanthology.org/2021.repl4nlp-1.31/.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Ali A. Heydari, Craig A. Thompson, and Asif Mehmood. SoftAdapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions. *arXiv preprint arXiv:1912.12355*, 2019.

Hugging Face. Fixing gradient accumulation. Technical blog, 2024. URL https://huggingface.co/blog/gradient_accumulation.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, 2015.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Baijiong Lin and Yu Zhang. LibMTL: A python library for multi-task learning. *Journal of Machine Learning Research*, 24(209):1–7, 2023.

Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1871–1880, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

NVIDIA Corporation. PhysicsNeMo Symbolic, version 2.4.0. Software release, 2026. Commit acaeb6dc38ecda58559b5286d3cb743e8cf930d3.

PaddlePaddle Authors. PaddleScience. Open-source scientific machine-learning software, 2026. Pinned commit d57561b139df953a77a2e134b7646048e445e731.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision*, 2012. URL https://cs.nyu.edu/~fergus/datasets/nyu_depth_v2.html.

SimplexLab. TorchJD, version 0.11.0. Software release, 2026. Commit af321be.

Jeffrey Willette, Seanie Lee, Bruno Andreis, Kenji Kawaguchi, Juho Lee, and Sung Ju Hwang. Scalable set encoding with universal mini-batch consistency and unbiased full set gradient approximation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 37008–37041, 2023.

Kieran Wood, Stephen J. Roberts, and Stefan Zohren. DeePM: Regime-robust deep learning for systematic macro portfolio management. *arXiv preprint arXiv:2601.05975*, 2026. URL https://arxiv.org/abs/2601.05975.

Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision*, 2018.

Yijia Zhang, Yibo Han, Shijie Cao, Guohao Dai, Youshan Miao, Ting Cao, Fan Yang, and Ningyi Xu. Adam accumulation to reduce memory footprints of both activations and gradients for large-scale DNN training. *arXiv preprint arXiv:2305.19982*, 2023.

A NYUv2 FIXED-HORIZON CONTINUATION DETAILS

This separate follow-up uses five calendar-derived seeds 2026091701–2026091705 without outcome screening. Protocol, runner, analyzer, data-manifest digest, weight digest, and sample indices were locally frozen before execution; no external timestamp or future-randomness claim is made. A user-requested pause of the initial launch was followed by container exit before any durable checkpoint. Its partial directory is preserved. The completed launch restarts identical seeds and orders with identical source/protocol hashes, not an additional replicate or outcome-selected replacement.

Public LibMTL dataset, dilated ResNet50 (He et al., 2016), and ASPP (Chen et al., 2017) components operate at 288×384 resolution. Adam uses learning rate 10^{-4} and weight decay 10^{-5} with no scheduler, AMP, TF32, or augmentation. BatchNorm statistics are fixed but affine parameters remain trainable. Dropout masks are keyed to seed, logical update, sample occurrence, and operator. Every backward contribution uses full-effective-batch valid-pixel denominators. Native controller observations use fragment means; Replay uses whole-step means. Empty task denominators and nonfinite losses fail the cell.

Ten 1,000-update prefixes and 25 1,000-update continuations execute 35,000 updates. Branches load model, Adam, controller, logical RNG, CPU/CUDA RNG, and logical position from their own algorithm's prefix. Sum and adaptive prefixes are not interchanged. SoftAdapt consumes no controller randomness, so this experiment does not test a stochastic controller. The K schedule is prescribed, not selected from GPU load. Four processes share one RTX 5090; their timings are contended operational measurements, not an isolated resource comparison. The completed launch took 6.8 hours including evaluation and analysis.

Task-wise primary AUC uses a fixed trapezoidal rule at updates 1,000, 1,250, 1,500, 1,750, and 2,000, divided by 1,000, with zero initial distance. The first 16 validation images and every fourth spatial pixel are fixed; raw segmentation logits, depth, and normal-head outputs are kept separate rather than pooled across units. Intermediate quality metrics use 32 images, final quality metrics all 654. Only the prediction probe has an unchanged cohort. The denominator is the reference norm, not Eq. 8's symmetric normalization. Raw recovery is $1 - A^{\mathrm{Replay}}/A^{\mathrm{Native}}$, without clipping or subtraction of the Sum control.

A separate post-run script imports neither the runner nor the main analyzer. It recomputes distances from saved parameter/probe arrays and checks checkpoint hashes, loaded parameters, every sample ID, update count, physical K, controller calls, action counts, and endpoint reporting: all 296 checks pass. This is separate mechanical validation, not independent scientific replication or an independent training implementation. The artifact includes the validator, all trajectory summaries and endpoint metrics; large checkpoint/parameter/probe payloads remain in the local raw archive with an explicit availability inventory.

Table 5: All NYUv2 endpoint seeds at update 2,000 (suffixes 01–05). Lower trajectory deviation does not imply better quality: Replay depth absolute error is worse than Native in four seeds. The artifact also reports all three losses, pixel accuracy, relative depth error, median angular error, and three angular-threshold fractions.

Seed	Execution	mIoU ↑	Depth abs. ↓	Normal degrees ↓
01	Current-step K1	0.5091	0.4519	26.02
01	Native switched	0.5026	0.4324	26.38
01	Replay switched	0.4786	0.4474	26.57
01	Sum K1	0.4989	0.4177	26.01
01	Sum switched	0.4707	0.4421	26.69
02	Current-step K1	0.4904	0.4851	25.92
02	Native switched	0.4836	0.4569	26.23
02	Replay switched	0.4811	0.5005	25.80
02	Sum K1	0.4848	0.4339	25.69
02	Sum switched	0.4916	0.4393	25.76
03	Current-step K1	0.5080	0.4850	26.37
03	Native switched	0.4476	0.4661	27.80
03	Replay switched	0.5098	0.5028	26.19
03	Sum K1	0.4924	0.5312	26.06
03	Sum switched	0.4891	0.4709	26.14
04	Current-step K1	0.4978	0.5209	25.84
04	Native switched	0.4460	0.5728	27.13
04	Replay switched	0.4910	0.5052	25.91
04	Sum K1	0.5013	0.5161	25.54
04	Sum switched	0.5030	0.4251	25.62
05	Current-step K1	0.4937	0.4582	26.20
05	Native switched	0.4582	0.4528	26.96
05	Replay switched	0.4922	0.6129	26.29
05	Sum K1	0.4941	0.5122	25.93
05	Sum switched	0.4877	0.4488	25.91

B PROOF DETAILS AND BOUNDARY CASES

Definition 3 (Prefix action completeness)**.** Fix a processed prefix and let $\mathcal{A}_{\mathrm{pre}} \subseteq \mathbb{R}^m$ be the final actions reachable by admissible suffixes. For cumulative prefix derivative $G \in \mathbb{R}^{P\times m}$, a summary $q(G)$ is action-complete if a decoder satisfies $D(q(G), a) = Ga$ for every admissible G and $a \in \mathcal{A}_{\mathrm{pre}}$.

Proposition 3 (Action-information requirement)**.** *If a same-step executor discards a processed prefix's reverse-mode information and never recomputes it, exactness for all admissible suffixes requires an action-complete summary. If $r = \dim \mathrm{span}(\mathcal{A}_{\mathrm{pre}})$ and a linear sketch $q(G) = S\,\mathrm{vec}(G) \in \mathbb{R}^d$ is action-complete for every $G \in \mathbb{R}^{P\times m}$, then $d \geq Pr$.*

Proof of Proposition 1. Linear scalarization gives $g^{\textsc{Native}} = \sum_k G_k a_k$ and $g^{\mathrm{ref}} = \sum_k G_k a^\star$; subtraction proves Eq. (5) and the fixed-gradient condition. For the universal statement, sufficiency is immediate. Conversely, if $v = a_j - a^\star \neq 0$ for some fragment j, set all other gradient matrices to zero and choose $G_j = uv^\top$ for any nonzero $u \in \mathbb{R}^P$. Then $G_j v = u\|v\|_2^2 \neq 0$, contradicting universal agreement. Gradient equality does not determine controller state, RNG consumption, or commits, so those trace components must be checked separately. The universal statement fixes the action vectors while ranging over arbitrary gradient matrices. It is a worst-case characterization, not an assertion that loss-derived actions and gradients vary independently in every neural network.

Proof of Proposition 2. The observation reduction supplies the same z_t to the same state and controller RNG event, so Eq. (1) yields the same a_t and s_{t+1}. With a_t fixed, normalized fragment derivatives sum to the derivative of the whole effective-batch objective. Thus the accumulated gradient is the same. Under the additional conditions stated after Eq. (6), the optimizer transition is also the same.

Proof of Proposition 3. All suffixes following the fixed prefix leave the retained prefix state identical before the suffix is seen, but can induce different actions in $\mathcal{A}_{\mathrm{pre}}$. Exactness requires the retained state to yield Ga for every such completion. If that state cannot decode Ga for some reachable action and admissible G, at least one completion is wrong. Place r linearly independent reachable actions in $B \in \mathbb{R}^{m\times r}$. Exact decoding recovers GB from the sketch. The linear map $\mathrm{vec}(G) \mapsto \mathrm{vec}(GB)$ has rank Pr, because B has rank r. Equal sketches must give equal GB, so

$\ker S$ is contained in the kernel of this map and $\operatorname{rank}(S) \geq Pr$. Since $\operatorname{rank}(S) \leq d$, the result follows.

Order dependence. For the exponential update $T_u(s) = \beta s + (1-\beta)u$, two fragments give

$$T_v(T_u(s)) = \beta^2 s + \beta(1-\beta)u + (1-\beta)v.$$

Swapping u and v changes the result by $(1-\beta)^2(v-u)$. Thus an ordered Native fragment stream can change controller state even when its mean observation is fixed.

Why Sum is a null control. When C is stateless and its action is fixed, Eq. (3) reduces to a sum of fragment derivatives. Any remaining discrepancy is attributable to floating-point reduction order or non-replayed model stochasticity, both explicitly checked by the Sum and trace controls. Sum measures ordinary accumulation sensitivity under its own objective, not a universal lower bound on another algorithm's finite-precision trajectory deviation.

An action-disconnected stateful null with fixed output and namespaced RNG reduces to this same calculation; any excess gap diagnoses shared-RNG or execution-path leakage, not adaptive-action cadence. Call/state/RNG traces therefore complement rather than redefine the Sum floor.

Randomized controllers. Clock consistency applies to the logical RNG sequence, not only its seed. Consuming K calls' draws from a seeded stream generally differs from consuming one call's draws. Replay evaluates one call against tentative state and publishes its RNG post-state only at commit. Abort restores the pre-step RNG position; deterministic controllers consume no draws.

C Extended related-work distinctions

History-based weighting makes cadence part of the method: SoftAdapt tracks recent loss change, Dynamic Weight Average uses epoch ratios, and ReLoBRaLo combines moving history with randomized lookback (Heydari et al., 2019; Liu et al., 2019; Bischof & Kraus, 2021). Our question is not which weighting rule is best, but which declared event advances such state when an optimizer step is physically fragmented.

Known gradient-accumulation discrepancies often arise inside the objective or model, including batch normalization and mean-of-microbatch token reduction (Ioffe & Szegedy, 2015; Hugging Face, 2024). Scalar normalization can repair the latter. It cannot undo extra history transitions or random events in an external adaptive controller.

GradCache caches representations for large contrastive batches, and DeePM's Sharpe-ratio training reconstructs a full-batch statistic before a second differentiated pass (Gao et al., 2021; Wood et al., 2026). These methods motivate the stateless, nonseparable capability cell; our statistic-replay row is a capability oracle, not a verbatim reimplementation of either system. A stateful controller additionally requires one logical action, one state/RNG transition, abort behavior, and checkpoint-resume reproducibility.

Universal mini-batch consistency studies partition-invariant set functions and full-set gradient approximations (Willette et al., 2023). Jacobian systems such as TorchJD instead expose a task-gradient basis that can realize an action once it is known (SimplexLab, 2026). Retained graphs, replay, gradient bases, and lagged execution are therefore implementation families for the transaction contract, not four new optimization methods.

Finally, Adam Accumulation and rematerialization study optimizer-state or recomputation/memory tradeoffs (Zhang et al., 2023; Chen et al., 2016), while large-batch work studies scaling and generalization (Goyal et al., 2017; Keskar et al., 2017). Our paired experiments instead hold effective batch and update count fixed and intervene on controller event order.

D Source-attested mechanism details

Table 6 records the specific pinned call boundaries used in the source audit. Line numbers identify the frozen snapshots in the artifact; they are not claims about later releases.

Table 6: Pinned released paths that place adaptive-controller calls inside an accumulation loop.

System	Fragment/optimizer boundary	Stateful controller boundary
PhysicsNeMo-Sym v2.4.0	`physicsnemo/sym/trainer.py`, lines 67–87	`physicsnemo/sym/loss/aggregator.py`, lines 415–472
PaddleScience `d57561b`	`ppsci/solver/train.py`, lines 146–181	`ppsci/loss/mtl/relobralo.py`, lines 97–125

Table 7: Complete fixed-panel disposition. "N/A" means the first-party path lacks GA or an in-scope stateful objective; it is not evidence of safe co-integration. Both positive rows were disclosed anchors before this audit.

Project	Stratum	GA	Stateful	Classification
Hugging Face Transformers	engine	yes	no	N/A
PyTorch Lightning	engine	yes	no	N/A
Hugging Face Accelerate	engine	yes	no	N/A
DeepSpeed	engine	yes	no	N/A
MosaicML Composer	engine	yes	no	N/A
PhysicsNeMo-Sym	objective-enriched	yes	yes	undeclared clock coupling
PaddleScience	objective-enriched	yes	yes	undeclared clock coupling
LibMTL	objective-enriched	no	yes	N/A
FAMO	objective-enriched	no	yes	N/A
TorchJD	objective-enriched	no	yes	N/A
Nash-MTL	objective-enriched	no	yes	N/A
MultiObjectiveOptimization	objective-enriched	no	no	N/A

Audit amendment and validation scope. The locked protocol requested a two-update $K{=}1/K{=}4$ litmus, whereas the first report reused prior $K{=}1/K{=}8$ traces. The original timestamped results are retained. An append-only, externally timestamped amendment supplies the missing $K4$ runs: each anchor records 2 controller transitions over 2 optimizer updates at $K1$, and 8 over 2 at $K4$. PhysicsNeMo uses its pinned native CUDA runner rather than the protocol's preferred CPU setting; this is disclosed because invocation count, not numerical quality, is the estimand. A separate mechanical validator checks commits, hashes, coverage, call ratios, and table aggregation without importing the builder. It cannot independently adjudicate the human interpretation of controller semantics.

Table 8 gives the compact PhysicsNeMo-Sym mechanism result. All rows use seed 64101; values are medians over steps 1–31 of a 32-step run.

Table 8: Source-attested mechanism study. Residual and state error are relative ℓ_2 distances to our author-specified optimizer-step reference.

Execution	K	Calls	Grad. residual	Angle (°)	State error
Sum Native	8	8	1.12×10^{-7}	6.3×10^{-6}	0
SoftAdapt Native	1	1	0	0	0
SoftAdapt Native	8	8	0.6157	19.97	0.8740
SoftAdapt norm-matched	8	8	0.6219	19.66	0.8737
SoftAdapt Transactional Replay	8	1	1.73×10^{-7}	7.2×10^{-6}	6.1×10^{-8}
ReLoBRaLo Native	8	8	0.0356	0.637	0.2334

Norm matching reduces update-norm error to 1.80×10^{-8} but leaves a $19.66°$ direction error. Transactional Replay reaches final-parameter distance 6.18×10^{-9} from the reference. The separately instrumented PaddleScience mechanism run likewise records 64 ReLoBRaLo calls for eight optimizer updates at $K{=}8$, versus eight calls under controller-once execution.

E CAPABILITY-BOUNDARY VALIDATION

The shared test model has two encoders, each mapping six-dimensional synthetic inputs through two 12-unit tanh layers to a scalar regression head and a normalized five-dimensional embedding. The effective batch is 64. Two MSE losses, or one MSE and an in-batch contrastive term (temperature

0.35), supply the objective-structure axis; fixed Sum and stochastic ReLoBRaLo supply the controller axis. ReLoBRaLo uses $\alpha = 0.95$, $\beta = 0.99$, and $\tau = 1$. SGD uses learning rate 0.015 and momentum 0.2. The three seeds are 1015145730, 1395231199, and 1168112978; checkpoint restoration is tested after step 16. The frozen numerical tolerance is 10^{-10}; RNG and lifecycle traces use equality checks. These small constructed cases establish a capability distinction, not the frequency or size of failures in downstream workloads.

The formal 2×2 matrix contains 48 steps at $K{=}8$ for each of three fixed seeds. Across every execution labeled match in Table 2, the maximum gradient, state, and one-step parameter residuals are 3.38×10^{-16}, 8.98×10^{-17}, and 1.89×10^{-17}. In the cells expected to fail, gradient residuals range from 3.61×10^{-4} to 0.452 and trajectory parameter AUC from 6.17×10^{-4} to 0.0452. Stateless nonseparability preserves the RNG trace, whereas every naive or statistic-only stateful execution changes it in 3/3 seeds.

Transactional Replay matches gradient, action, state, RNG trace, and parameter update in all 12 cell–seed pairs. It also passes abort rollback and checkpoint-resume equality in all pairs. A read-only validator that does not import the runner or primary analyzer independently reconstructs the declared exactness pattern and these lifecycle checks.

F MULTIMNIST FRAGMENTATION-TRANSFER DETAILS

The float64 Native gradient residuals are 0.199, 0.300, and 0.321 at $K2/K4/K8$, and 0.324 for unequal $K4$. Transactional residual is at most 5.83×10^{-16}. Over 2,340 updates, Native action AUCs are 0.267/0.401/0.494 and state AUCs 0.186/0.292/0.387. Native model-side AUCs exceed matched Sum in 5/5 seeds at $K2$, 4/5 at $K4$, and 5/5 at $K8$ and unequal $K4$. Median raw recovery spans 48.7–68.2% for parameters and 51.7–61.2% for logits. Median mean accuracy is about 96%; Transactional beats matched Native in only 2/5 seeds for each uniform K and 4/5 for the unequal partition. All endpoint task cross-entropies remain descriptive.

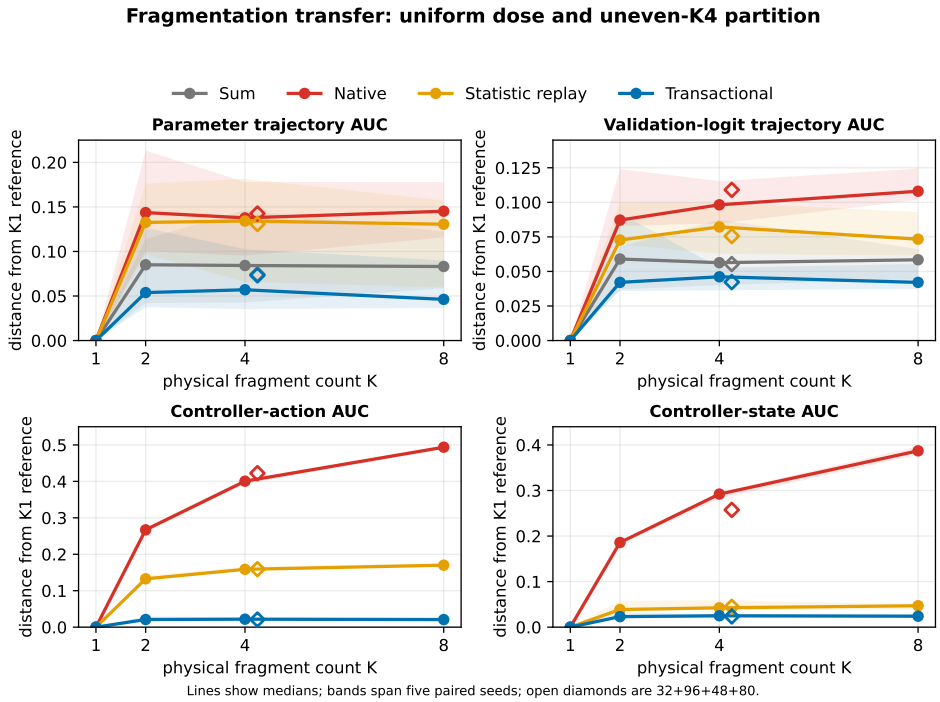


Figure 4: MultiMNIST fragmentation transfer over five paired seeds. Native has strict controller-level dose but nonmonotone model-side distance. Transactional execution remains near the ordinary Sum control across partitions; statistic reconstruction without a single commit does not.

Supporting checkpoint parameter distances. For $K1 \rightarrow K8$, median parameter AUCs are 0.01203/0.00236/0.00235 for Native/Sum/Transactional; for $K8 \rightarrow K1$, they are 0.01063/0.00235/0.00252. The main figure instead prioritizes final prediction changes and validation-logit trajectories.

Table 9 reports the paired controller distances behind Figure 4. Each row is strictly ordered from $K2$ to $K4$ to $K8$ for both action and state. The formal seed set is fresh relative to the earlier development study.

Table 9: MultiMNIST controller-distance AUC by seed.

Seed	Action K2	Action K4	Action K8	State K2	State K4	State K8
2067972338	.263750	.395061	.491809	.183396	.287288	.387101
1092477176	.267012	.400535	.493777	.185814	.292282	.386801
1739058698	.262663	.392441	.490016	.182151	.281736	.378329
1827573681	.274769	.404495	.497525	.187151	.295728	.397019
693220259	.268467	.403608	.494145	.188547	.292129	.394995

Table 10 gives report-only endpoint accuracies for three representative SoftAdapt arms. Cross-entropies and all 38 fragmentation/switch executions remain machine-readable; no endpoint direction is a success criterion.

Table 10: MultiMNIST endpoint accuracy by seed. "Worst" is the lower of the two task accuracies; "mean" averages them.

	Worst-task accuracy			Mean accuracy		
Seed	K1	Native K8	Tx K8	K1	Native K8	Tx K8
2067972338	.9533	.9525	.9541	.96075	.95965	.96105
1092477176	.9555	.9551	.9554	.96170	.96110	.96100
1739058698	.9536	.9549	.9536	.96065	.96105	.96040
1827573681	.9545	.9530	.9541	.96020	.96055	.96025
693220259	.9547	.9542	.9552	.96035	.95915	.96020

Across $K2/K4/K8$ and unequal $K4$, Transactional raw recovery is 48.7–68.2% for parameter AUC and 51.7–61.2% for logits AUC. Its median distance is at or below the matched Sum floor in every partition, although this does not hold seed-wise in every cell. Reporting raw values beside the floor prevents numerical adjustment from being mistaken for literal trajectory identity.

Lagged baseline extension. The append-only LTX extension reuses the same five paired seeds, sample order, initialization, schedule, horizon, and already frozen Native/Transactional/Sum references. It adds only LTX $K1/K8$, for ten new training units. The action used at update t is the action prepared from update $t-1$; the current weighted whole-batch observation is committed exactly once for use at $t+1$. A separate validator imports neither runner nor core and passes 66/66 checks, including the exact one-step lag relation. All seed-level AUCs, endpoint logits, CE/accuracy directions, runtime, and allocation records are in the artifact. LTX $K8/K1$ changes at least one final predicted digit for 0.0111–0.0132 of examples across seeds; LTX $K1$ versus current-step Native $K1$ changes 0.0199–0.0277.

G PUBLIC-PATH TRANSACTION INTEGRATION DETAILS

The PaddleScience prototype changes zero upstream lines. Its base adapter is 230 physical lines (194 nonblank, non-comment), and the deterministic DataLoader-cursor rollback shim is 45 physical lines (31 nonblank, non-comment). It exposes controller `snapshot`, `restore`, `prepare`, and `commit` around the released Solver forward helper, constraints, ReLoBRaLo, and Adam path. The short mechanism study fixes K=4, 16 logical updates, failure/checkpoint update 8, and three seeds.

All 12 scenario cells complete and a separate validator that imports neither the runner, adapter, ClockGuard, Paddle, nor PaddleScience passes 31/31 checks. Observation-abort retry, realization-abort retry, and checkpoint-resume have zero parameter and prediction residual and identical con-

troller/RNG hashes in all seeds. A retained pre-cursor-rollback variant preserves state/RNG and makes no optimizer update during the injected pre-update failures, but has parameter residuals 8.45×10^{-10}–4.00×10^{-9} and prediction residuals 4.41×10^{-8}–1.72×10^{-7} after observation retry. Thus exact transaction replay requires the framework executor to restore input position in addition to controller state and randomness. The artifact also retains two earlier implementation failures involving Tensor serialization and lazy optimizer-accumulator identity; neither changed the scientific protocol.

H SUPPORTING AND HISTORICAL DOWNSTREAM STUDIES

The studies below are retained separately, not pooled with the later attribution or checkpoint experiments. Internal continuation rules describe research decisions, not truth tests of the semantic propositions. The public-path clock attribution study is an exploratory robustness extension: its protocol and estimands were fixed before execution, but it does not replace the earlier downstream decision.

Other unfavorable checks. The earlier 30-run CelebA study did not satisfy its fixed clock-effect and median-reduction criteria; median Replay reduction was 37.8% on test and 36.4% on validation, below the specified 50%. A separate 512-update CelebA LTX check missed action/state invariance, gap-reduction, and validation-stability criteria. Neither is superseded by the later MultiMNIST LTX comparison. The executor benchmark met six of seven checks: two float32 Basis runs exceeded the 10^{-5} residual criterion. Its later float64 diagnostic does not retroactively change that outcome. Complete records remain under `evidence/negative_gates` and `evidence/gates/resource`.

An earlier fixed-horizon study used a deliberately conjunctive reporting rule. That rule was not met because Exact solution-relative $\ell_2 \leq 0.05$ held in only 4/5 seeds; the fifth frozen 10,000-update endpoint is 0.092163. Integrity, oracle, Native-effect, parameter-recovery, and prediction-effect conditions were met. The later attribution and portability studies are analyzed separately and do not replace this outcome. Table 11 reports the complete primary quantities.

Table 11: Earlier fixed-horizon study. "Exact final" is the frozen Exact solution-relative ℓ_2; the fixed quality rule required all five values to be at most 0.05.

Seed	Sum AUC	Native AUC	Exact AUC	Recovery	Pred. Native/thresh.	Exact final	Quality
1457709116	.000381	.04807	.004948	89.71%	.02797/.01000	.005389	pass
1238287026	.000332	.02664	.004034	84.86%	.04211/.01415	.000762	pass
1833727779	.000386	.16455	.001822	98.89%	.04398/.01649	.092163	fail
104490350	.000235	.11666	.001841	98.42%	.04784/.01143	.002918	pass
83400384	.000635	.13884	.015062	89.15%	.03007/.03011	.004106	pass

Prediction AUC in this study has medians 0.04211 (Native), 0.002830 (Sum), and 0.007545 (Exact). Exact reduces the paired Native prediction distance by 56.8–90.7% across the five seeds. Exact final solution error is lower than Native in 3/5 seeds and PDE-residual MSE is lower in 1/5; neither direction was a gate.

Table 12 gives the fresh-seed Wave study at its development-selected horizon. Here P_N, P_E, P_S are final parameter distances for Native, Exact, and Sum against their matched $K1$ references, and $R_P = 1 - P_E/P_N$. With M_N, M_E the absolute differences in solution error from the adaptive $K1$ endpoint, $R_M = 1 - M_E/M_N$; neither is an AUC.

Table 12: Development-selected 449-update Wave study. Solution-relative ℓ_2 is lower-is-better; PDE columns are executor/Native ratios.

Seed	K1	Native	Exact	LTX	P_N / P_S	R_P	R_M	PDE E/N	PDE L/N
288112	.73788	.62588	.76512	.82908	6.33	82.9%	75.7%	8.78	7.39
554389	.68930	.63353	.73074	.75223	7.57	82.8%	25.7%	.317	4.29
487754	.69474	.62646	.69013	.87632	4.83	70.3%	93.2%	1.49	5.27

Table 13 preserves the non-pooled primary endpoints preceding the supporting Wave study. Wave-129/Wave-513 recovery is heterogeneous, while Spring's larger recovery coexists with weak absolute solution quality.

Table 13: Historical prospective solution-relative ℓ_2 endpoints. R reduces the absolute solution-error gap to K1, not parameter distance. The Wave-513 PDE column is Exact/Native; "–" denotes an endpoint not shown in this compact historical table.

Horizon	Seed	K1	Native K8	Exact K8	R	PDE E/N
Wave-129	265358	.85999	.70452	.80382	63.9%	–
	577215	.75881	.66299	.71383	53.0%	–
	866025	.80591	.71154	.81314	92.3%	–
	112358	.75975	.74417	.74627	13.5%	–
	314160	.78199	.68921	.76947	86.5%	–
Spring-513	271829	.98077	.94685	.97592	85.7%	–
	618034	.98731	.95342	.98398	90.2%	–
	999983	1.01196	.97225	.99418	55.2%	–
Wave-513	909639	.86859	1.08791	.92187	75.7%	1.72
	404707	1.09268	.90981	.90128	-4.7%	1.04
	733755	.78275	.95150	.92280	17.0%	3.11

Reproducibility and evidence binding. The reviewer artifact hash-binds the mechanism traces, supporting and historical downstream records, capability-boundary studies, MultiMNIST runs, their frozen inputs, and read-only validators. The earlier fixed-horizon study retains its timestamp, randomness record, protocol, and unmet-rule ledger; later studies retain separate labels and complete reporting and do not replace that decision.