TRAINING MEMORY IN DEEP NEURAL NETWORKS: MECHANISMS, EVIDENCE, AND MEASUREMENT GAPS

Vasileios Sevetlidis*
Athena RC
Xanthi, GR67100, Greece
vasiseve@athenarc.gr

George Pavlidis
Athena RC
University Campus Kimmeria
Xanthi, GR67100, Greece
gpavlid@athenarc.gr

ABSTRACT

Modern deep-learning training is not memoryless. Updates depend on optimizer moments and averaging, data-order policies (random reshuffling vs with-replacement, staged augmentations and replay), the nonconvex path, and auxiliary state (teacher EMA/SWA, contrastive queues, BatchNorm statistics). This survey organizes mechanisms by source, lifetime, and visibility. It introduces seed-paired, function-space causal estimands; portable perturbation primitives (carry/reset of momentum/Adam/EMA/BN, order-window swaps, queue/teacher tweaks); and a reporting checklist with audit artifacts (order hashes, buffer/BN checksums, RNG contracts). The conclusion is a protocol for portable, causal, uncertainty-aware measurement that attributes how much training history matters across models, data, and regimes.

***K*eywords** Training memory · Optimizer state · Sampler state · Data ordering · Path dependence · Causal interventions · Calibration · Reproducibility

1 Introduction

Training a deep neural network is *not* memoryless. By *training memory* we mean that what the learner does next depends not only on its current parameters and the current minibatch, but also on *how it arrived there*—the recent sequence of updates and data. Specially, training memory can arise from (i) *optimizer state* (e.g., momentum buffers or adaptive moments that carry summaries of past gradients), (ii) *sampler state* (e.g., the order in which examples are presented or priorities that make some examples more likely to be seen), and (iii) the *parameter path* itself (the route through the loss landscape) which, in nonconvex problems, makes the order of small updates matter.

Classical and modern optimization analyses formalize these ideas. Momentum and averaging explicitly carry information forward across steps, and in nonconvex settings stochastic updates do *not* commute—applying update A then B can lead to a different point than applying B then A—so outcomes can depend on path and ordering [1, 2, 3, 4, 5]. In practice, stateful mechanisms are used deliberately at scale: adaptive optimizers (e.g., Adam/AdamW), exponential moving averages of weights (EMA) and stochastic weight averaging (SWA), and sharpness-aware updates introduce controlled history dependence to stabilize or improve generalization [6, 7]. In distributed and federated settings, server-side adaptivity further exposes the role of carried-over state across rounds [8].

Memory also lives in the data pipeline. Even without explicit priorities, *random reshuffling* (processing each example once per epoch in a random order) behaves differently from with-replacement sampling; theory and experiments show distinct convergence properties and, at times, performance [9, 5, 4]. When sampling is *stateful*—for example via online

or prioritized selection that revisits high-loss or high-influence examples—training can speed up, but attribution becomes entangled between data order and optimizer history [10, 11, 12, 13]. Beyond loss curves, studies of representation similarity (e.g., SVCCA and CKA) document that learned features can *drift* across runs or phases, even when top-line metrics look similar [14, 15, 16]. These tools describe *that* change occurs, but they are not causal diagnostics: they do not attribute changes to optimizer state, sampler state, or path effects, and they are rarely paired with effect sizes and uncertainty.

Despite broad recognition that DNN training has memory, the community lacks a *portable*, *causal*, and *standardized* way to quantify *how much* that memory matters across architectures, data sets, and training regimes. Typical reports emphasize final accuracy or loss and often leave protocols under-specified (e.g., whether momentum was reset between phases, the exact data-order policy, or augmentation schedules). Order- and state-driven phenomena then become difficult to reproduce or compare. Related areas reinforce this picture at longer time scales: in continual learning, forgetting and transfer depend on task order and replay [17, 18]; in curriculum learning, pacing and example ordering affect stability and sample efficiency [19]. Yet across these settings there is still no consensus diagnostic that cleanly attributes effects to optimizer state, sampler memory, or parameter-path dependence.

We synthesize mechanisms and evidence for training memory across optimizers, samplers, and parameter paths, and we surface protocol pitfalls that hinder attribution. This article does not propose a new algorithm or a single diagnostic. Instead, it contributes (i) a taxonomy (source–lifetime–visibility), (ii) a synthesis of theory and evidence, (iii) causal estimands with portable perturbation primitives, and (iv) a reporting checklist that make attribution auditable. We then articulate solution-agnostic *desiderata* for future diagnostics: isolate sources via controlled perturbations; report effect sizes in function space alongside standard metrics; track representation drift with appropriate caveats; and emphasize early indicators that predict late generalization. Our aim is to motivate principled, causal measurement of training memory—without committing to a specific methodology in this study.

2 Relation to Prior Surveys

Several survey families intersect with what we term *training memory*—persistent traces of optimization history, algorithmic state, and data ordering that shape the final solution. Geometry-oriented overviews on loss landscapes and model merging illuminate path dependence and mergeability (e.g., linear/curvilinear connectivity and permutation-aware alignment), but typically abstract away the concrete sources of *state* accumulated during training [20, 21, 22]. Work on sampler order shows that random reshuffling (without replacement) can materially alter convergence and final solutions relative to with-replacement SGD, yet these analyses are not usually framed as a unifying "memory" mechanism nor tied to broader reporting practice [4, 23, 24]. Reviews of normalization emphasize that running statistics and batch coupling are consequential sources of hidden state—with recent evidence of task-specific pitfalls—again treated largely in isolation from other mechanisms [25, 26]. In self-supervised and contrastive learning, surveys document explicit memory structures such as queues and banks (e.g., MoCo-style dictionaries), but scope is confined to SSL rather than integrated with optimizer/geometry/order effects [27]. Complementary lines synthesize representational and functional similarity tools for auditing training trajectories [28], ensembling surveys that touch SWA/SWAG as temporal averaging along a path [29], federated learning surveys that catalog server/client momentum as cross-round state [30], and calibration/uncertainty surveys that focus on reliability outcomes rather than the upstream memory mechanisms that might drive them [31]. Documentation artifacts such as Model Cards, Datasheets, and NeurIPS reproducibility checklists advance general reporting, but they do not yet target memory-sensitive details (e.g., sampler semantics, EMA/SWA configuration, BN/SyncBN regimes, queue refresh rules) [32, 33, 34].

Our survey differs by (i) unifying these strands under a single taxonomy of *training memory* that treats optimizer momentum/EMA/SWA, sampler order/reshuffling, normalization running statistics, explicit buffers (queues/replay), geometric path dependence, and federated server/client state as first-class, interacting carriers of history; (ii) mapping cross-mechanism interactions (e.g., how order shapes EMA trajectories or how BN state mediates mergeability); and (iii) proposing *measurement protocols* and *reporting checklists* specific to memory. Concretely, we standardize trajectory-level similarity profiling, interpolation/merging and barrier tests, order-sensitivity ablations, state ablations for EMA/SWA/BN/queues, and calibration under controlled memory manipulations, alongside a minimal set of fields for declaring and stress-testing memory effects. This integrated perspective complements existing surveys by making the memory mechanisms explicit, comparable, and reportable across training pipelines.

3 Taxonomy of Training Memory

A learner has *training memory* when its update at step t depends on more than the current weights and minibatch—it also depends on how we got here (optimizer state, data-order decisions, or the specific path taken through parameter

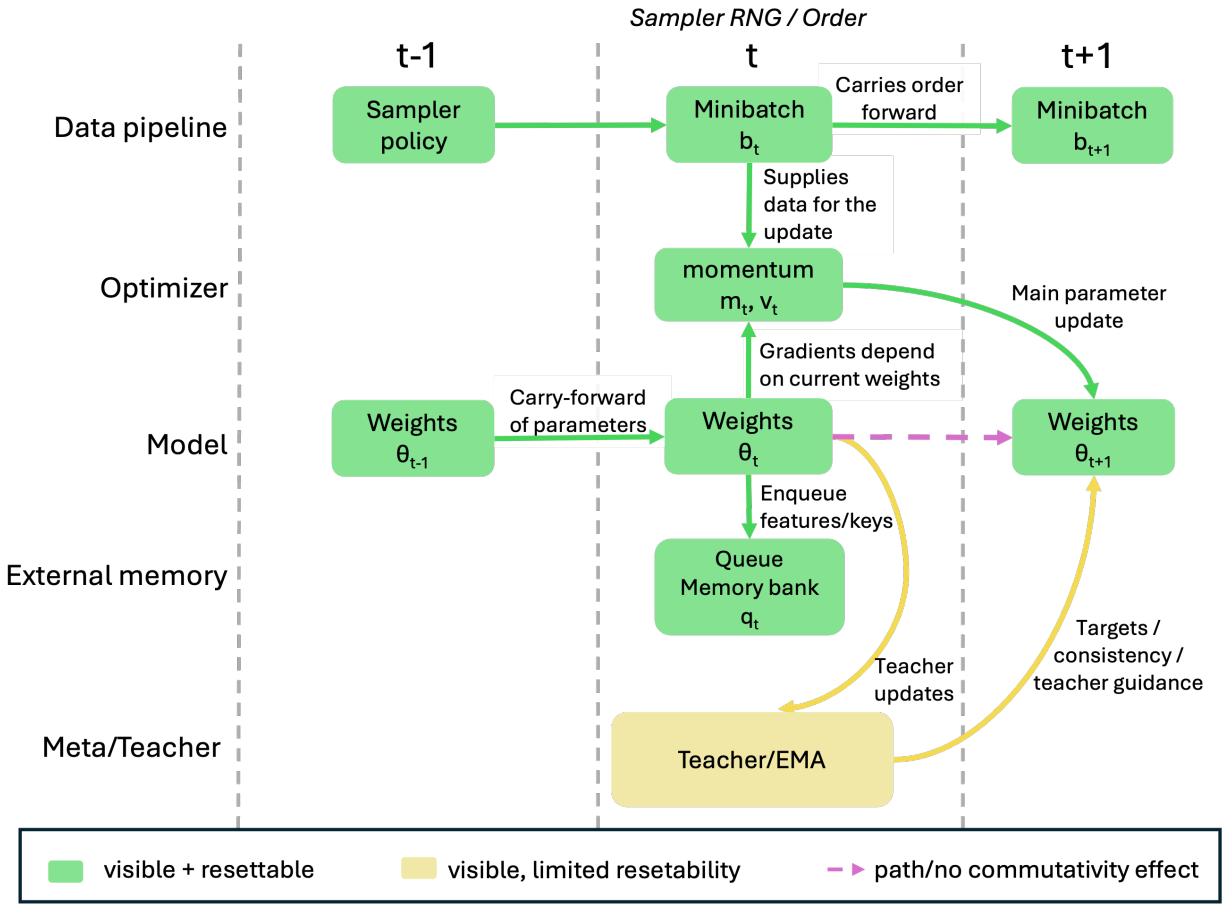

Figure 1: Schematic of the training loop with explicit and implicit state. Boxes: weights θ_t, optimizer buffers (e.g., momentum/Adam), sampler RNG/order, external queues/memory banks, and teacher/EMA. Arrows indicate what carries across steps and phases; coloring encodes visibility/resetability.

space). Formally, relative to the visible interface (θ_t, b_t) i.e., the state the outer training loop exposes: current weights θ_t and current minibatch b_t, we say the procedure has memory at time t if the distribution of the next update depends on history beyond that interface:

$$\mathbb{P}(\Delta\theta_t \,|\, \theta_t,\, b_t,\, \mathcal{H}_t) \;\neq\; \mathbb{P}(\Delta\theta_t \,|\, \theta_t,\, b_t)\,,$$

where θ_t are the current weights, b_t is the current minibatch, and $\mathcal{H}_t$ aggregates prior randomness, batches, and internal states up to time t. This reconciles the intuition with a Markov view: if we augment the state S to

$$S_t \;=\; \big(\theta_t,\ \text{optimizer buffers}_t,\ \text{sampler RNG/order}_t,\ \text{external queues/banks}_t, \dots\big),$$

then $\mathbb{P}(\Delta\theta_t \mid S_t)$ can be history-independent by construction. "Having memory" thus means effects persist when those augmented components are not controlled or are only partially observed.

To orient newcomers, we organize the space along three axes: **source** (where the memory comes from), **lifetime** (how long it persists), and **visibility** (can we see or reset it). Figure1 illustrates the explicit and implicit states we consider in the training loop.

3.1 Axis 1 — Source (where memory comes from)

(S1) Optimizer / trajectory state. Modern optimizers carry running summaries of the past. Momentum accumulates a smoothed direction of travel; Adam/AdamW keep first/second moments; EMA/SWA average weights along a trajectory; sharpness-aware methods adjust steps using recent curvature; second-order/preconditioned methods keep structured curvature statistics. Each mechanism makes the *next* step a function of *many previous* steps. If we reset or perturb the optimizer's internal state (e.g., zero momentum buffers; clear EMA), downstream behavior can shift even when weights and data are unchanged.

Examples. Momentum/Nesterov keep an exponential moving average (EMA) of gradients; the decay β induces a practical "half-life" (how many steps a gradient still matters) [2, 3]. *Adam/AdamW* maintain biased/unbiased moment estimates; AdamW decouples weight decay from the loss gradient, explicitly injecting history via moments while controlling norm growth [35, 36]. *Averaging* (Polyak averaging; SWA) smooths iterate noise and tends to land in wider basins [3, 6]. *SAM* alters steps to avoid sharp regions, making updates depend on recent local geometry [7]. *Limited-memory second-order* (e.g., K-FAC, Shampoo) maintain layer- or tensor-structured curvature state that persists across steps [37, 38].

(S2) Sampler / data-order state. The data pipeline is not memoryless. Changing *which* examples appear (and *when*) changes the gradient noise and the path. Random reshuffling (each example once per epoch) behaves differently from with-replacement sampling; curricula/pacing alter difficulty over time; prioritized sampling and replay make exposure frequencies stateful. Holding optimizer fixed, changes in order, pacing, or priority can move training toward different regions of the landscape.

BatchNorm keeps running means/variances that persist across updates and are used at evaluation [39]. Re-estimating these statistics after a change in weights or domain measurably shifts performance: SWA explicitly requires recomputing BN stats for the averaged model [6, 40]; domain-adaptation methods replace source BN statistics with target-domain estimates (AdaBN), improving accuracy [41]; and test-time adaptation often updates BN statistics on the fly [42]. Thus BN stats are explicit, resettable state that should be logged/manipulated alongside momentum/Adam buffers. By contrast, LayerNorm normalizes per example and uses no running averages, providing a useful foil [43]. Many widely used augmentation policies are explicitly time-varying and thus act like *stateful sampling*: because the transformation distribution changes across epochs, the effective minibatch distribution drifts even when the dataset is fixed. AutoAugment learns stage-specific policies, while RandAugment provides magnitude and count parameters that are often scheduled during training. In attribution studies, these schedules should be treated on the same footing as order policies—frozen or perturbed in isolation—since they alter which *views* of examples are seen when [44, 45].

Examples. Random reshuffling vs. with-replacement have distinct convergence behaviors and often different empirical performance [9, 5, 4]. *Curricula/pacing* implement long-horizon ordering (easy→hard; staged augmentation) [19]. *Prioritized/importance sampling* upsamples "informative" examples (high loss/gradient norm, TD-error), trading speed for entanglement between sampler and optimizer state [11, 12, 13]. *Replay/coresets* retain subsets over time, making the sampler explicitly stateful across many steps or tasks [17].

(S3) Parameter-path dependence. In nonconvex landscapes, small updates need not commute: taking step A then B can end somewhere else than B then A. Even with a "stateless" optimizer and IID minibatches, the *route* conditions the *destination*. This shows up as preference for flatter minima or distinct but mode-connected solutions. If two runs start from the same initialization but follow different orderings or schedules, they can converge to functionally different solutions.

A practical way to *make path effects visible* is via mode-connectivity diagnostics: fit linear or low-curvature paths between solutions found under different orders/schedules and probe predictions along the path. When there are low-loss connectors, we can test whether the behavior of the function space varies smoothly or exhibits 'kinks' that reveal qualitatively different solutions; lack of connectivity suggests truly distinct basins [46, 47, 48]. In short, connectivity provides an operational visibility tool for (S3): it does not only assert that the path matters; it lets us *measure* how endpoints relate in the landscape.

Examples. Loss landscape geometry and *mode connectivity* document low-loss paths between solutions and relate flatter regions to generalization [49, 46]. *Solution anchoring* in transfer: L2-SP pulls fine-tuning towards the pre-train weights; EWC constrains movement along important directions - both bake history into the objective [50, 51].

(S4) Architectural / external memory. Some training recipes add *extra state* beyond model weights: feature queues, memory banks, or plastic traces. Updates then depend on this side memory as well as the current batch. Clearing the queue or bank changes gradients immediately; longer queues imply longer memory.

In contrastive pipelines with queues or memory banks, the queue length sets an explicit lifetime: with a FIFO dictionary of size K and minibatch size B, a stored key typically persists for $\approx K/B$ updates before eviction; ablations in momentum-contrast systems show that varying K materially changes both optimization and downstream performance [52, 53].

Examples. Contrastive queues/memory banks (MoCo; instance discrimination) maintain evolving representations outside the main weights [52, 53]. *Hebbian/plastic traces* introduce additional, training-updated state that shapes learning dynamics [54].

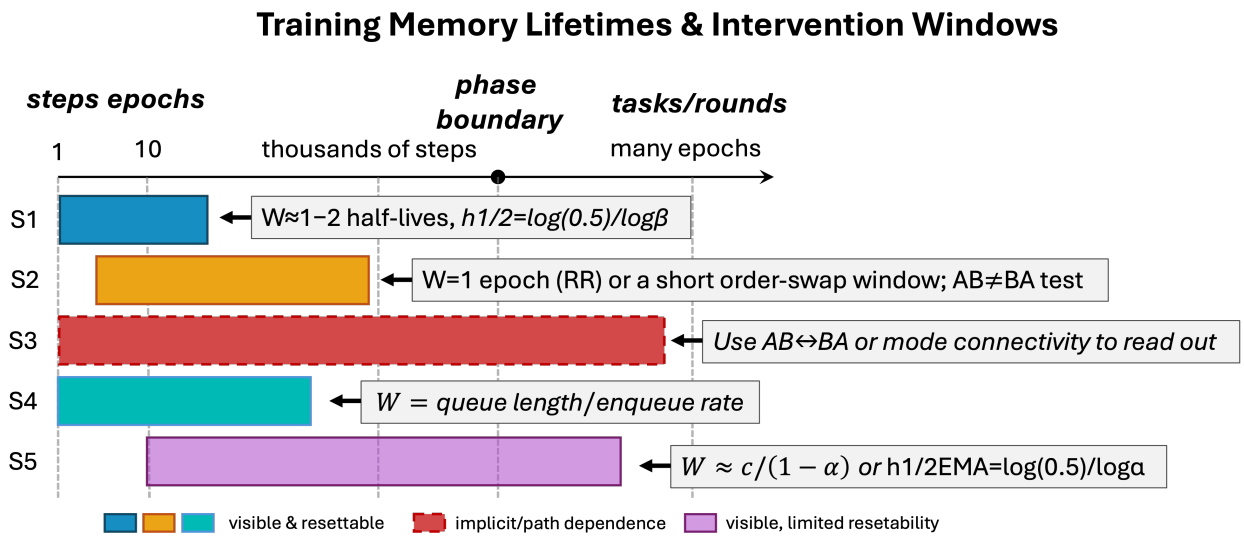

Figure 2: Typical memory lifetimes by source (S1–S5), from steps → epochs → phases → tasks/rounds. Each band shows a suggested intervention window W suited to that source: step-scale state (e.g., momentum/Adam, EMA) use $W \approx 1$–2 half-lives; epoch-scale order use one full reshuffled epoch or a fixed minibatch window under with-replacement sampling; phase-scale effects probe the first k epochs post-boundary; external queues use the queue *turnover* (queue length divided by enqueue rate); short AB≠BA order swaps at boundaries expose non-commutativity.

(S5) Meta-state (teachers / learned optimizers). A teacher network (often an EMA of the student) or a learned optimizer evolves over time and steers updates across many steps. Resetting or slowing the teacher/outer-loop often changes stability and final performance.

Examples. Teacher–student EMA ("Mean Teacher") provides slowly moving targets that encode long-range history [55]. *Learned/slow-fast optimizers* (learned optimizers; Lookahead) accumulate cross-step state and modulate inner updates [56, 57]. *Federated training* exposes server/client accumulators as explicit cross-round memory [58, 8]. Concrete, source-specific perturbations that isolate (S1)–(S5) are listed in Table 4 (see §6.2).

3.2 Axis 2 — Lifetime (how long memory persists)

As summarized in Figure 2, not all forms of training memory last the same amount of time. Some mechanisms fade within a few dozen updates; others persist across an epoch, a phase boundary (pretraining→fine-tuning), or even entire task sequences. Thinking in terms of *lifetimes* explains why two runs that look identical locally can diverge globally.

Step scale. Optimizer statistics that decay every update create short-horizon memory. Momentum and the first/second moments in Adam/AdamW retain influence over tens to hundreds of steps depending on the decay; their practical half-life is $h_{1/2} = \log(0.5)/\log\beta$ (see Table 1) [2, 35, 36]. Weight averaging (EMA/SWA) integrates a trailing tail of iterates and thus also encodes recent history [3, 6, 40]. These mechanisms make warm restarts behave differently from cold starts and explain why clearing buffers or freezing EMA for a short window can measurably alter early dynamics and calibration.

Epoch scale. The data pipeline dominates at the epoch horizon. With the dataset fixed, *order* (random reshuffling vs. with-replacement), batching, and staged augmentations change the gradient noise and steer the path through parameter space [9, 5, 4]. Curricula/pacing and replay policies induce persistence across many updates because the composition of minibatches co-varies over time [19, 59, 60]. These effects often survive small hyperparameter tweaks precisely because their lifetime is longer than step-scale statistics.

Phase scale. Explicit program boundaries—e.g., pretraining→fine-tuning, or schedule restarts—carry history across many thousands of steps. The initialization checkpoint anchors fine-tuning; penalties such as L2-SP/EWC make this anchoring explicit [50, 51]. Meta-state like teacher EMA/SWA can further strengthen it if carried across the boundary [55, 6]. Comparisons across fine-tunes should therefore report whether optimizer buffers and EMA/SWA were *carried*

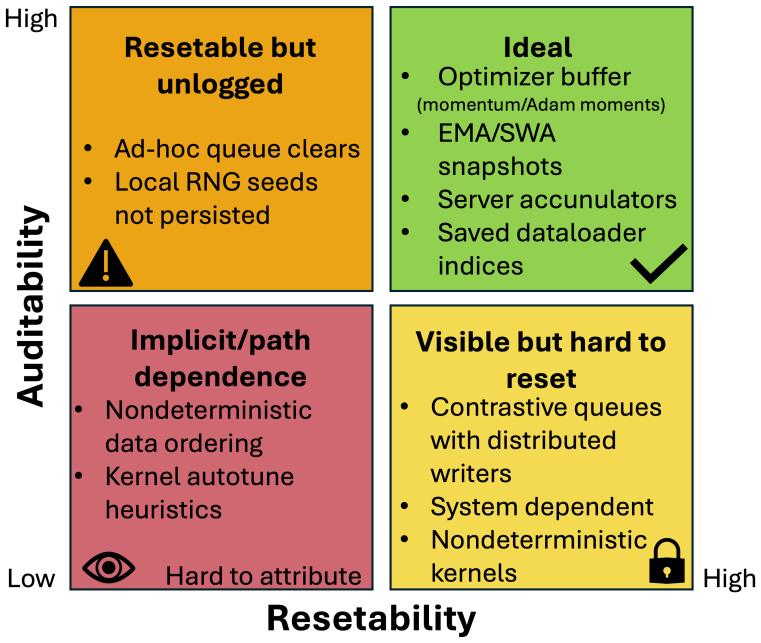

Figure 3: Resetability × auditability matrix with examples: ideal (visible, resetable, auditable) vs. visible-but-hard-to-reset, resetable-but-unlogged, and implicit/path dependence.

or *reset*, and whether BatchNorm statistics were recalibrated [39, 6, 40, 41, 42]. To expose phase-scale memory, branch at the boundary and apply the intervention for the first k epochs of the new phase (typically $k \in [1, 5]$).

Task/round scale. In continual and federated settings, memory spans tasks or rounds. Replay buffers and consolidation penalties retain information across tasks [17, 18], while client sampling and server-side accumulators (momentum/Adam variants) retain cross-round state [58, 8, 61, 62]. Here, outcomes depend on the *sequence* of tasks/clients as well as the optimizer trajectory.

Choosing W. In practice, W is selected with reference to the characteristic lifetime of the perturbed source. For step-scale statistics, a practical choice is $W \approx$ 1–2 half-lives. For order interventions, one reshuffled epoch under random-reshuffling (RR), or a fixed window under with-replacement sampling, is often sufficient. Following a phase boundary, it can be informative to confine the intervention to the first k epochs. For external-memory mechanisms, taking W commensurate with the queue turnover (queue length divided by enqueue rate) reflects the horizon over which contents refresh [52, 53]. Short AB$\neq$BA swaps at natural boundaries may illuminate non-commutativity while limiting interference with the overall training program [9, 4, 5].

3.3 Axis 3 — Visibility (can we see or reset it?)

Some forms of memory are *visible* as concrete state we can inspect or zero (e.g., momentum/Adam buffers, EMA/SWA snapshots, contrastive queues), while others are only detectable through behavior and geometry (path dependence in nonconvex landscapes). Two practical dials matter for protocol design: *resetability* (can we deterministically clear or reinitialize the state in this framework?) and *auditability* (is the state *logged* or checksummed so others can verify what was carried across phases?). Resetability is imperfect in common stacks due to nondeterministic kernels and ops [1], so protocols should state what was reset and how it was verified. Auditability is helped by reproducibility checklists and reporting artifacts (e.g., order hashes, buffer-state logs, model/dataset documentation) advocated in community reports and documentation frameworks [34, 32, 33]. Figure3 positions common examples on the resetability–auditability plane.

Explicit memory includes any state materialized in the training loop. Momentum and Adam buffers, EMA/SWA snapshots, feature queues or memory banks used by contrastive methods, and server accumulators in federated training

[1] Reproducibility — PyTorch Documentation: `https://pytorch.org/docs/stable/notes/randomness.html`, Accessed 2025-09-21

are all examples [35, 6, 52, 58]. Because these states are tangible, they create clear levers for experimentation—one can log, reset, or ablate them—and clear responsibilities for reporting: whether such state is carried across epochs or phase boundaries often determines reproducibility.

Implicit memory leaves no named buffer to read off. In overparameterized, nonconvex models, small stochastic updates need not commute; taking step A then B can land in a different region than B then A. The history is thus written into the *path* itself: which basin was entered, how flat the surrounding region is, and whether distinct solutions are mode-connected by low-loss corridors [49, 46]. When explicit state is controlled yet outcomes still diverge, the culprit is typically this path dependence—an interaction between geometry and ordering that is invisible unless one probes function space or representations, not just losses.

Two protocol–critical dimensions make the visibility axis actionable. *Resetability* asks whether a state can be deterministically zeroed or rewound (e.g., momentum/Adam buffers; EMA/SWA snapshots; dataloader order via saved index streams). Frameworks expose determinism toggles and seeded RNG streams, but nondeterministic kernels and backend heuristics still require explicit handling for faithful resets [63]. *Auditability* asks whether the state is logged so downstream analysis and reproduction are possible: per-epoch order hashes, buffer norms/checkpoints, and versioned configuration artifacts. Community guidance (NeurIPS Reproducibility Checklist; model cards; datasheets) emphasizes that audit logs are as important as code for post-hoc attribution [34, 32, 33]. Taken together, resetability (can we intervene?) and auditability (can we verify?) turn visibility into protocol design: visible but not resetable (hard to attribute); resetable but not audited (hard to trust); visible, resetable, and auditable (ideal).

Most practices blend the aforementioned. A run may use sharpness-aware updates and random reshuffling while maintaining a contrastive queue. The optimizer and sampler expose explicit dials; the landscape supplies the implicit backdrop on which those dials act. Any measurement or diagnostic that claims to attribute "training memory" needs to respect this composition: accounting for visible state while recognizing that part of what persists is the route the model took to get where it is. Any diagnostic that *attributes* training memory should however: (i) separate optimizer vs. sampler vs. path effects; (ii) respect their timescales; and (iii) report effect sizes with uncertainty, not just point estimates.

4 Theory & Established Results

This section explains *why* training memory should exist in modern learning pipelines and *what* the main theoretical lenses say about it. We focus on families of results that are widely cited and useful for reasoning about history-dependence, without committing to any particular diagnostic.

4.1 Stochastic approximation and momentum: why recent history matters

Classical stochastic approximation (SA) views training as a noisy recursion that tracks a moving target under sampling noise [64, 1]. Once we introduce exponential smoothing, the recursion becomes explicitly history-dependent. In momentum SGD, a velocity $v_t = \beta v_{t-1} + (1-\beta) g_t$ accumulates gradients g_t with geometric weights; Adam extends this idea with first and second moments [2, 35]. The influence of a past gradient decays like $\beta^{\,h}$ after h steps, so the *effective half-life* of memory is

$$h_{1/2} \;=\; \frac{\log 0.5}{\log \beta}\,.$$

To ground this quantity, Table 1 translates common β values into half-lives. We will use these numbers to set the short intervention window W for optimizer-state perturbations (typically $W \approx 1$–2 half-lives; see §6.2). For example, $\beta{=}0.99$ implies $h_{1/2} \approx 69$ steps, so a temporary momentum reset or EMA freeze for $W \in [70, 140]$ steps exposes the intended memory without long-term interference.

Polyak–Ruppert averaging makes a similar point at the *iterate* level: averaging a trailing tail of parameters reduces variance and nudges the solution toward wider, flatter regions [3]. Modern weight averaging (SWA) can be seen as a practical incarnation of this principle [6]. Taken together, SA and averaging formalize an obvious practitioner lesson: even if we freeze data and hyperparameters, changing the optimizer's internal state (buffers, EMAs) changes what happens next because we have changed the *recent past* the algorithm is carrying.

A complementary view treats SGD as a stochastic sampler whose noise level steers *path selection*. In the small–step limit, SGD can be approximated by an SDE whose effective "temperature" grows with the learning rate and shrinks with batch size, biasing trajectories toward wider basins; with explicit noise (SGLD), this interpretation becomes literal [65, 66, 67]. Practically, this ties optimizer and sampler parameter—step size, batch size, and even order-induced noise—to where the run goes in parameter space, motivating our emphasis on *function-space* readouts (not just loss) when comparing memoryful training policies.

Momentum β	0.90	0.95	0.99	0.999
Half-life $h_{1/2} = \log(0.5)/\log(\beta)$ (steps)	6.58	13.53	68.97	692.82

Table 1: Optimizer memory half-life as a function of β. We use these values to pick the perturbation window W for optimizer-state interventions ($W \approx 1$–2 half-lives; see §6.2).

Lower noise (e.g., very large batches at fixed step size) can bias training toward *sharper* minima, whereas higher noise explores and can escape them—consistent with reports that large-batch training often finds sharper solutions with worse test performance [68]. This sampler-as-thermostat view dovetails with SGLD and Bayesian perspectives that tie batch size and step size to an effective temperature and predictive uncertainty [67, 66].

4.2 Order dependence and non-commutativity: why the route matters

In nonconvex objectives, small stochastic updates generally do *not* commute. Applying minibatch A and then B yields a different point than B then A because the curvature encountered in between is different. Theory detects this even in convex baselines through the comparison of *with-replacement* sampling and *random reshuffling* (RR, without replacement): the two procedures induce different noise structures and can have different convergence rates, with RR often enjoying tighter guarantees [9, 5, 4]. More recent analyses treat single-shuffle and arbitrary-order schedules, reinforcing that "data order" is a real algorithmic choice, not an implementation detail [69]. For our purposes, these results justify treating the sampler (and its ordering across epochs) as a bona fide *source of memory*: by fixing or perturbing order, we change the path—and in nonconvex problems, changing the path changes the endpoint.

4.3 Importance sampling and prioritization: memory in the sampler

If examples are sampled with nonuniform probabilities p_i, unbiased risk estimates require reweighting by $1/p_i$. This can reduce gradient variance and accelerate optimization, particularly when losses or gradient norms are heavy-tailed [11, 70]. At scale, robust or approximate prioritization schemes trade exactness for speed [12]. Beyond convergence speed, however, deep models exhibit more nuanced behavior: importance weighting interacts with loss curvature and with adaptive optimizers, and its effect on *generalization* depends on when and how it is applied (e.g., early vs. late training, separable vs. nonseparable regimes) [71]. The conceptual takeaway is simple: once the sampler's probabilities depend on the evolving state (losses, errors, features), the sampler acquires *its own memory*. Which examples are repeatedly emphasized becomes part of the history that shapes the model.

4.4 Continual learning and the stability–plasticity trade-off: intended long-term memory

The stability–plasticity dilemma asks the learner to acquire new information (plasticity) without erasing useful prior knowledge (stability) [72]. Modern continual-learning methods make long-term memory explicit via three broad strategies: (i) regularization that keeps solutions near previous optima (e.g., EWC), (ii) replay buffers or coresets that preserve a working memory of past data, and (iii) architectural expansion that isolates parameters for new tasks [51, 17, 18]. These mechanisms are intentionally stateful: their purpose is to carry information across task boundaries. They therefore provide concrete testbeds for studying training memory at the longest timescales.

4.5 Scope and limitations: where current theory stops

The results above are informative but not exhaustive. Many guarantees rely on convexity, smoothness, or effectively IID sampling; where nonconvex analyses exist, they often target simplified models. Algorithmic stability connects optimization dynamics to generalization and helps explain when faster training can also generalize better [73], but it does not yet yield tight, predictive statements for modern overparameterized networks across the range of practices used in the field [74, 75]. For this reason, careful *measurement* remains necessary: we should expect theory to suggest which parameters matter (state, order, path), but not to tell us a priori how large their effects will be in a given regime. In overparameterized regimes, these stability bounds are often too loose to discriminate neighboring training policies that diverge in function space, so we use stability as a qualitative lens rather than a standalone diagnostic [73, 74, 75].

Implication for this survey. The families above justify treating optimizer state, sampler state, and path as *separate* sources of training memory, each with its own timescale. They also motivate protocols that (i) perturb one source at a time and (ii) read out effects in function space, not only in loss/accuracy, because noncommutativity and averaging influence what the model *does*, not just what loss it attains.

5 Empirical Evidence & Practitioner Heuristics

This section collects recurring empirical findings that make "training memory" inapplicable. We group results by where memory arises (optimizer state, sampler/order, distributed settings, and representation drift), emphasize *what was reported* in the original studies, and extract the simple heuristics practitioners rely on.

5.1 Optimizer state effects

Warm restarts and momentum resets visibly change outcomes even when the model and data are fixed. Cosine schedules with warm restarts improve anytime performance and reach strong accuracy with fewer epochs, illustrating that carrying (or clearing) velocity state across cycles matters [36]. Exponential weight averaging—from classical Polyak averaging to SWA—consistently improves test accuracy and tends to sharpen *calibration* (lower ECE/NLL) by smoothing jagged late-stage trajectories [6, 40]. Sharpness-aware steps (SAM) shift solutions toward flatter neighborhoods and raise test accuracy across CIFAR/ImageNet and fine-tuning regimes, again showing that the per-step memory encoded by the optimizer alters generalization [7]. Independently, calibration studies report that modern deep nets are often overconfident, and that ensembling/averaging is a strong practical fix [76].

Heuristics used. (i) When restarting or phase-shifting, explicitly specify momentum/EMA carry-over vs. reset; (ii) SWA/EMA late in training often improves stability and calibration; (iii) Sharpness-aware updates mitigate sharp-minima overfitting and brittle validation curves.

5.2 Order and curriculum effects

Order within and across epochs is not an implementation detail. Comparisons of with-replacement vs. random reshuffling (RR) show different noise structures; RR often converges faster or to better plateaus even in simple baselines, and practice reflects this preference in large-scale codebases [4, 5]. Pacing policies and curricula stabilize early training and can reduce variance across seeds; surveys document gains across vision and NLP tasks, with improvements measured in accuracy and sample efficiency rather than exotic surrogates [19]. In continual settings, small *replay buffers* mitigate catastrophic forgetting and improve average accuracy/backward-transfer metrics, underscoring that sampler memory (what is rehearsed, how often) is decisive [59, 60].

Heuristics used. (i) Single-epoch RR is generally preferred for SGD; (ii) staged augmentations or curricula can alleviate early instability; (iii) in sequential-task settings, a small, well-chosen replay set reduces forgetting.

5.3 Distributed/federated effects

Across rounds, federated optimization exhibits *server-side memory*: FedAvg maintains a running aggregate; adding server momentum/Adam (FedOpt variants) changes both speed and final accuracy, while control-variate methods (e.g., SCAFFOLD) specifically target cross-round *drift* due to heterogeneous clients [58, 8, 61, 62]. Empirically, reported metrics are validation accuracy (global and per-client), stability across rounds, and fairness measures; sensitivity to client sampling/ordering is a practical concern when participation is sparse or non-IID. A round-level view with server accumulators and client sampling is shown in Figure 4.

Heuristics used. (i) Server momentum/adaptivity should be logged and tuned; (ii) drift-correction stabilizes training under high heterogeneity; (iii) client sampling/ordering should be documented because it interacts with cross-round memory.

5.4 Representation drift (what changes, even when accuracy looks fine)

SVCCA/CKA and related tools show that intermediate representations evolve substantially across epochs, phase boundaries, and runs with different orders or seeds [14, 15]. Studies typically report *similarity curves* (SVCCA/CKA values over time) alongside accuracy; many also note that lowered similarity across phases does not necessarily imply worse performance, cautioning against causal claims from similarity alone. Still, these measures are informative descriptors of how much history the model "keeps" at the feature level.

Heuristics used. A lightweight similarity signal (SVCCA/CKA or stitching) across phases provides context for accuracy changes; interpretation should remain qualitative rather than a stand-in for generalization.

Table 2 summarizes recurring empirical patterns, what was reported, and the practical takeaway. Across optimizers, samplers, and distributed settings, *state that persists across steps/epochs/rounds* changes both the path and the endpoint. Practically, the strongest and most portable levers are: (i) make optimizer state handling explicit (carry vs. reset), (ii)

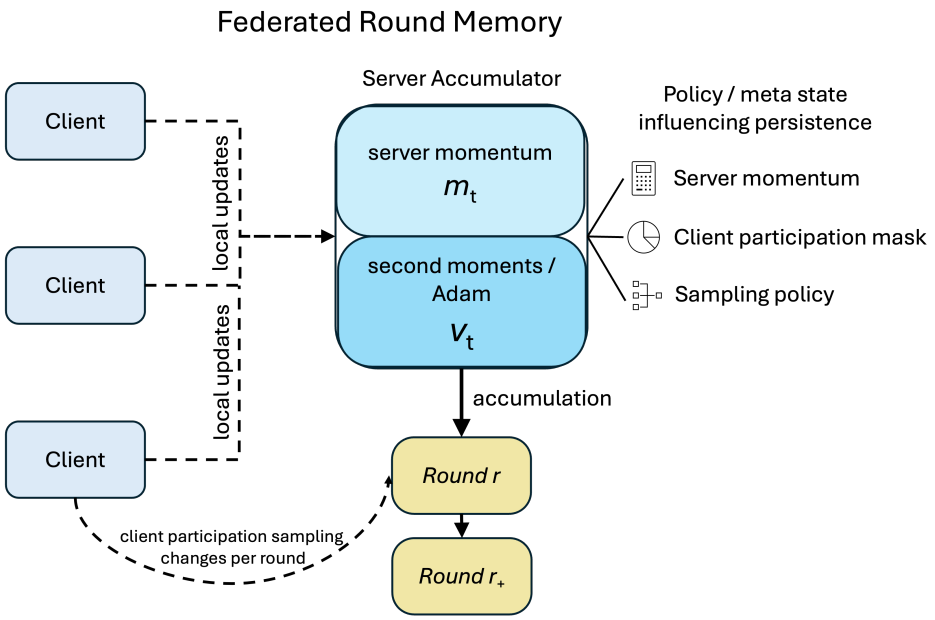

Figure 4: Federated learning rounds with server-side memory: client local updates aggregate into server accumulators (e.g., momentum m_t, second moments v_t), which persist across rounds; participation masks and sampling policy contribute additional round-scale memory.

Technique	Setup (typical)	Reported	Empirical takeaway
Warm restarts / momentum resets[10]	CIFAR/ImageNet; cosine LR with restarts	Top-1; epochs-to-target	LR restarts (and velocity handling) alter trajectory; better anytime performance.
EMA/SWA weight averaging[6, 40]	CIFAR/ImageNet; late-phase averaging	Acc.; ECE; NLL	Averaging improves generalization; often improves calibration; smoother endpoints.
SAM (sharpness-aware)[7]	Standard vision benchmarks	Acc.; robustness	Flatter-neighborhood solutions generalize better across models/data sets.
Order: RR vs. replacement[4, 5]	Supervised image tasks	Train/val loss; acc.	RR shows different (often faster) convergence and plateaus than replacement.
Curricula/pacing[19]	Vision/NLP; staged difficulty/augs	Acc.; sample efficiency	Early stability and sample efficiency improve with pacing.
Replay (continual)[59, 60]	Class-incremental CIFAR/ImageNet	Avg. acc.; BWT/forgetting	Small rehearsals reduce forgetting; stabilize representation drift.
Federated server adaptivity[58, 8, 61, 62]	FedAvg/FedOpt variants	Global acc.; per-round stability	Server momentum/Adam and drift correction matter under heterogeneity.
Representation similarity[14, 15]	Across seeds/phases	SVCCA/CKA curves	Large representational changes can coexist with similar accuracy; descriptive, not causal.

Table 2: Common empirical findings about training memory (*compact*). Metrics are those reported in the cited papers; the rightmost column records the qualitative takeaway as stated by authors.

treat data order and replay as algorithmic choices, and (iii) document cross-round state in federated runs. Representation-similarity tools add context but should be read as descriptors rather than causal attributions.

6 Measurement Limitations in the Literature

Training *memory*—history dependence from optimizer state, sampler state, and the path a model takes through parameter space—is widely acknowledged. But when we read closely how papers *measure* its impact, a pattern emerges: several choices that plausibly carry history are changed together, the evaluation lens is narrow, and the procedural details that would enable clean attribution are under-described. The result is that improvements are easy to claim and hard to ascribe.

A first source of confusion is attribution. Modern recipes typically combine momentum or adaptive moments in the optimizer with per-epoch random reshuffling or prioritized sampling in the data pipeline, all while operating in

Algorithm 1: Branch-and-Hold: single-source interventional diagnostic

Input: model $\mathcal{M}$ with params θ, optimizer $\mathcal{O}$, sampler policy $\mathcal{S}$, schedule Λ, probe $\mathcal{P}$, distance D or scalar metric M, window W, branch time t_0, horizon T, intervention ϕ (perturbs exactly one source), seeds $\mathcal{R}$, log sink $\mathcal{L}$.

Output: Early effect Δ^{early} (at $t_0{+}W$) and final effect Δ^{final} (at T), each with 95% CI, plus audit logs.

foreach *seed* $r \in \mathcal{R}$ **do**
- Set global RNG to r; record and name RNG streams for: init, sampler/order, augmentation, model
- Train a **root** run to step t_0 under $(\mathcal{O}, \mathcal{S}, \Lambda)$
- Persist: $(\theta_{t_0}, \text{optimizer buffers}_{t_0}, \text{teacher/EMA/SWA}_{t_0}, \text{BN stats}_{t_0}, \text{sampler state}_{t_0})$
- Pre-fetch and store the next W minibatch *IDs* and augmentation RNG states; compute *order hash* for $[t_0, t_0{+}W)$
- Deep-copy the full state into two branches: CONTROL, TREAT
- `// Apply the single intended perturbation in Treat at` t_0`; keep all else equal for` W
- **Treat:** apply ϕ at t_0
- **Both branches:** replay the recorded W minibatch IDs and augmentation RNGs in lockstep; advance schedules identically
- `// Early readout at the end of the hold window`
- Evaluate both on probe $\mathcal{P}$; **if** D *is provided* **then** set $z_r^{\text{early}} \leftarrow \frac{1}{|\mathcal{P}|}\sum_{x\in\mathcal{P}} D(P_{\text{ctrl}}(x), P_{\text{treat}}(x))$
- **else** set $z_r^{\text{early}} \leftarrow M(f_{\text{treat}}) - M(f_{\text{ctrl}})$ on $\mathcal{P}$
- `// Resume normal training to` T `with the same high-level policies; RNGs free to evolve`
- Resume standard dataloading from $t_0{+}W$ with identical *policies* $(\mathcal{S}, \Lambda)$ in both branches; do not enforce identical micro-order beyond the recorded window
- Optionally recalibrate BN before final eval if SWA/EMA is used
- Train both branches to horizon T; evaluate on $\mathcal{P}$ to get z_r^{final} via D or M as above
- Log to $\mathcal{L}$: order hash, buffer norms (e.g., $\|m\|$, $\|v\|$), EMA/teacher decay, BN checksum, queue fingerprints (if any)

Compute $(\Delta^{\text{early}}, \text{CI}_{95})$ by calling Alg. 2 on $\{z_r^{\text{early}}\}$; same for Δ^{final} using $\{z_r^{\text{final}}\}$

return $\Delta^{\textit{early}}$, $\Delta^{\textit{final}}$, *CIs, and audit logs*

a nonconvex landscape where the path itself can matter. Theory and empirics already tell us that these levers can *independently* alter dynamics—without-replacement (reshuffled) SGD does not behave like with-replacement SGD [9, 5, 4]. Yet gains are often reported as a single top-line improvement, with little clarity about which parameter did the work. Reproducibility reports have flagged the same issue from another angle: missing information about data order, augmentation stages, or buffer handling makes it difficult to reconstruct what was actually varied [34].

Ablation studies help, but they are rarely designed as *interventions*. To identify a source of memory, one must vary that source while holding the others fixed. In practice, we seldom see protocols that, for example, reset momentum buffers but keep the exact sampler and order intact, or swap sampler policies while preserving optimizer state. Adjacent fields have shown why this matters: variance across seeds and configurations can flip conclusions and demands statistical controls, not just point estimates [63]. Supervised training papers have adopted some of the reporting hygiene, but true causal contrasts remain uncommon.

Concretely, a minimal *branch-and-hold* design (Algorithm 1) can be instantiated as follows. (i) At a pre-specified step t, the full training state S_t is snapshotted—current weights θ_t; optimizer buffers (e.g., momentum/Adam moments); the sampler's order state (or a record of the index sequence for the next window); and any external or teacher state. (ii) Two branches, A and B, are then initialized from this snapshot and differ only in the targeted source (e.g., CARRY vs. RESET for optimizer state; RR vs. WR for order), with all other ingredients of π held constant. (iii) Over the subsequent W updates, data order and randomization streams are matched across branches so that only the intended source is perturbed; training then proceeds to the horizon T. Function-space metrics $M(\cdot)$ are evaluated on a fixed probe, and paired ATEs are summarized as in §6.1, with W chosen to reflect the source's characteristic timescale (e.g., half-life for momentum/EMA, one reshuffled epoch for RR vs. replacement, or queue turnover for memory banks).

The measurement lens is also narrow. Accuracy and loss dominate reporting, even though they can obscure calibration errors and instability under shift. A model can improve top-1 while becoming more overconfident [76], and methods with

similar accuracy can diverge sharply in predictive uncertainty when distributional conditions change [77]. Benchmarks that probe subpopulations and domains (e.g., WILDS) repeatedly show uneven behavior that a single aggregate metric hides [78]. Representation-similarity tools such as SVCCA or CKA add welcome visibility into *change* across runs or phases, but the literature cautions that different indices need significance testing or resampling before they support claims about causality [14, 15, 16, 79].

Procedure matters as much as metrics, and here too details are thin. Seemingly small choices—whether sampling is with- or without-replacement, how and when augmentations are staged, whether optimizer buffers are carried across warm restarts—alter trajectories in ways both theory and experiments have documented [9, 5, 4]. Program-level reviews keep finding that such choices are under-specified [34]. Beyond reproducibility concerns, this connects to *underspecification*: pipelines with indistinguishable held-out accuracy can encode very different behaviors under shift [80].

Finally, the community underuses early signals and underreports uncertainty. There is rich work on learning-curve and scaling-law extrapolation [81, 82, 83, 84, 85], but those threads rarely ask whether *early* differences induced by optimizer or sampler state *predict late* generalization across policies. When such signals are reported, they often appear without uncertainty, and single-seed results remain common despite evidence that they can reverse conclusions [63, 86]. Heterogeneous probe sets and mixed statistical practices further complicate synthesis across papers. A practical default is to run at least five seeds on small/medium benchmarks and at least three on costlier regimes, pairing seeds across branches when estimating $\widehat{\mathrm{ATE}}$. Report the mean and 95% confidence intervals *and* print CI width next to each point estimate; show seed-level scatter when space allows. When asserting "no material difference," pre-declare an equivalence margin ε and use an equivalence test rather than relying on overlapping CIs. As a pragmatic convention, resources permitting, one may use at least five seeds on small/medium benchmarks and at least three on costlier regimes, pairing seeds across branches when estimating $\widehat{\mathrm{ATE}}$. Reporting the mean with 95% confidence intervals—along with the CI width next to each point estimate—and, where space allows, seed-level scatter plots can aid interpretation. Claims of "no material difference" are more defensible when accompanied by a pre-specified equivalence margin ε and an equivalence procedure (e.g., TOST), rather than relying solely on overlapping confidence intervals.

When we assert "no material difference," we propose to conduct an *equivalence* test rather than rely on a non-significant difference test. Let δ_i be the paired, seed-matched effect (e.g., for seed i, the mean Δ under condition B minus condition A) and a practically negligible margin $\varepsilon > 0$ in the same units as δ_i (e.g., total-variation units on the probe). Equivalence is then assessed with the *Two One-Sided Tests* (TOST) procedure on the mean paired effect $\bar{\delta}$: then reject the composite null $|\bar{\delta}| \geq \varepsilon$ if and only if both one-sided t tests (lower and upper) are significant at level α with $df = n - 1$ degrees of freedom; equivalently, the $(1 - 2\alpha)$ two-sided confidence interval for $\bar{\delta}$ lies entirely within $(-\varepsilon, \varepsilon)$. Note that δ_i needs formed at the seed level (averaging repeats within seed) to avoid pseudoreplication. Alongside the TOST decision report $(\bar{\delta}, s, n)$, the chosen ε with rationale, both one-sided p-values, and the $(1 - 2\alpha)$ CI. For small n or uncertain normality, a bootstrap CI check for containment within $(-\varepsilon, \varepsilon)$ can be added. See **(author?)** [87, 88, 89].

In short, the literature establishes that history enters through optimizer state, sampler state, and path, and that protocol choices can be causal. What is missing is a *portable, causal, and statistically principled* way to quantify *how much* each source contributes under transparent, reproducible protocols, and to report those effect sizes with uncertainty.

Table 3 summarizes common failure modes and pragmatic controls. These failure modes share a common root: there is lack of *explicit targets of estimation* that isolate optimizer state, sampler policy, or path effects while holding everything else fixed. To make attribution portable and testable, we now formalize source-specific *causal estimands* that every diagnostic should report. These estimands turn the informal notion of "training memory matters" into concrete, seed-averaged effect sizes with uncertainty.

6.1 Causal Estimands for Attribution

Fix a training recipe π (architecture, optimizer hyperparameters, augmentation, evaluation protocol) and a horizon T. Let $f_T^{(i,s)}$ denote the predictor obtained after T updates when we apply intervention i and random seed s, with *all other* ingredients of π held fixed. Let $M(\cdot)$ be a scalar *function-space* metric on a fixed evaluation distribution $\mathcal{D}_{\mathrm{eval}}$ (e.g., accuracy ↑, ECE/NLL ↓, or pairwise disagreement on a probe set). We define source-specific average treatment effects (ATEs) as expectations over seeds:

$$\mathrm{ATE}_{\mathrm{opt}} = \mathbb{E}_s\Big[\, M\Big(f_T^{(\textsc{carry},\, s)}\Big) - M\Big(f_T^{(\textsc{reset},\, s)}\Big)\Big], \qquad \mathrm{ATE}_{\mathrm{order}} = \mathbb{E}_s\Big[\, M\Big(f_T^{(\textsc{rr},\, s)}\Big) - M\Big(f_T^{(\textsc{wr},\, s)}\Big)\Big].$$

Here, CARRY/RESET differ only in optimizer-state handling at a predeclared branch point (e.g., momentum/Adam buffers, EMA), while RR/WR differ only in sampling policy (random reshuffling vs. with-replacement). Finite-sample estimates use S *paired* seeds, $\widehat{\mathrm{ATE}} = \frac{1}{S}\sum_{j=1}^{S}\big[M(f_T^{(i_1,s_j)}) - M(f_T^{(i_0,s_j)})\big]$, with 95% bootstrap confidence intervals. The same pattern yields $\mathrm{ATE}_{\mathrm{teacher}}$ (EMA decay $\alpha \to \alpha'$), $\mathrm{ATE}_{\mathrm{queue}}$ (contrastive-memory clear vs. carry), or *order-*

Limitation	Typical manifestation	Why it hinders attribution	What to control/report
Attribution ambiguity	Change optimizer *and* sampler; report one headline gain	Multiple stateful sources move; effects mixed	Hold two sources fixed, perturb one; log carry-over of buffers across phases
Causal gap	Ablations without explicit resets or swaps	No interventional contrast to isolate optimizer vs. sampler vs. path	Include reset/swap interventions; pre-declare attribution tests and seeds
Metric myopia	Only top-1/loss curves are reported	Calibration, OOD reliability, and function-space movement are hidden; rep-metrics lack uncertainty	Add calibration/shift stress tests; function-/rep-space deltas with CIs; equivalence tests when claiming "no difference"
Under-specification	Sampler policy, augmentation schedule, buffer handling omitted	Pipelines look "equivalent" in accuracy but behave differently under shift	Specify order policy, augmentation phases, and state handling at restarts; share configs and seeds
Early-phase blindness	No early leading indicators	Predictive signal about later generalization is unused	Report early-window indicators (with uncertainty) and correlate with final outcomes across policies
Reporting heterogeneity	Single seed; no CIs or statistical tests	High variance can flip conclusions; hard to synthesize across papers	Multi-seed summaries; paired/bootstrap CIs; appropriate tests and declared equivalence margins

Table 3: Failure modes that obscure measurement of training memory and pragmatic controls to make results interpretable—without prescribing a specific diagnostic.

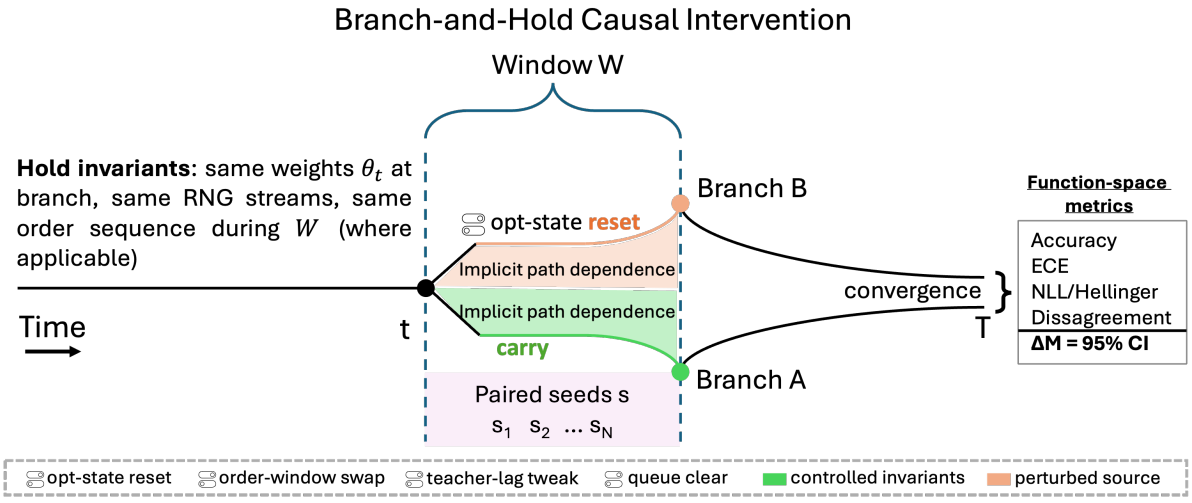

Figure 5: Branch-and-hold design at time t: fork runs that differ only in a targeted source (e.g., CARRY vs. RESET optimizer state, or order window swap) for a window W, then continue to horizon T. Report paired seed effect sizes ΔM in function space with CIs.

window ATEs where only a window of W minibatches is permuted. Our estimator and uncertainty summary are given in Algorithm 2, and the branching design is illustrated in Figure 5.

6.2 Portable Interventions (Perturbation Primitives)

The estimands in §6.1 require targeted perturbations that change exactly one source of training memory while holding others fixed. Table 4 lists minimal, portable interventions that map directly to the taxonomy in §3 (S1–S5) and to typical lifetimes (step, epoch, phase, task). They are architecture- and dataset-agnostic and pair naturally with paired-seed ATEs and bootstrap confidence intervals. We implement these single-source perturbations via Algorithm 3 (optimizer-state reset), Algorithm 4 (order-window swap), Algorithm 5 (phase-boundary policy), and Algorithm 6 (external-memory interventions).

Algorithm 2: Paired ATE and Bootstrap CI

Input: Per-seed paired measurements $\{z_r\}_{r\in\mathcal{R}}$ (e.g., mean probe distance or metric delta for seed r); bootstrap rounds B.
Output: Point estimate $\widehat{\mathrm{ATE}}$ and 95% CI.

Compute $\widehat{\mathrm{ATE}} \leftarrow \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} z_r$
for $b \leftarrow 1$ **to** B **do**
  Sample with replacement a multiset $\mathcal{R}^{(b)}$ from $\mathcal{R}$
  Set $\widehat{\mathrm{ATE}}^{(b)} \leftarrow \frac{1}{|\mathcal{R}^{(b)}|}\sum_{r\in\mathcal{R}^{(b)}} z_r$
Let CI_{95} be the 2.5th–97.5th percentiles of $\{\widehat{\mathrm{ATE}}^{(b)}\}_{b=1}^{B}$
return $\widehat{\mathrm{ATE}}$, CI_{95}

Algorithm 3: Optimizer-state reset (S1): zero/rewarm buffers; sampler fixed for W

Input: As in Alg. 1; momentum/AdamW hyperparams (β_1, β_2); optional EMA/SWA handle; optional LR rewarm length K.
Output: $\Delta_{\text{opt}}^{\text{early}}$, $\Delta_{\text{opt}}^{\text{final}}$ + CIs; buffer-state provenance.

Define ϕ_{opt} at t_0:
- For SGD+momentum: set velocity to zero.
- For Adam/AdamW: zero first and second moments; keep weight parameters identical; preserve weight decay state (decoupled).
- Optionally pause EMA/SWA updates for the next W steps (freeze teacher / averaging buffers).
- If using LR rewarm, apply the same K-step warmup schedule in *both* branches to avoid confounding.

Choose $W \approx$ 1–2 half-lives where $h_{1/2} = \log(0.5)/\log(\beta_1)$ for momentum/Adam first moment; note Adam's second moment has half-life $\log(0.5)/\log(\beta_2)$
Call Alg. 1 with $\phi \leftarrow \phi_{\text{opt}}$, enforcing identical minibatch IDs and augmentation RNGs only on $[t_0, t_0{+}W)$
return Δ_{opt}^{early}, Δ_{opt}^{final} + *CIs*

The intervention window W is the horizon over which a perturbation is applied (or held) so that its influence is visible but not obscured by unrelated drift. A good rule is to match W to the characteristic *memory* of the source that is being probed. For momentum/Adam, the relevant timescale is the half-life of the exponential smoothing, $h_{1/2} = \log(0.5)/\log(\beta)$: with β=0.9 the signal halves in roughly $6-7$ steps, while β=0.99 stretches this to about 69 steps (and β=0.999 to hundreds). Choosing $W \approx$1–2 half-lives typically balances power and interference: too short and the effect is underresolved; too long and other factors creep in. For order interventions, portability argues for one full epoch under random reshuffling, or a fixed minibatch window when sampling with replacement. Teacher–student EMA follows an analogous logic: the effective averaging horizon is on the order of $1/(1-\alpha)$, so α=0.99 exposes change over $\sim 10^2$ steps and α=0.999 over $\sim 10^3$. External memory (e.g., contrastive queues) suggests W comparable to the queue's turnover (queue size divided by enqueue rate), long enough for the bank to "forget" past contents without permanently altering the training distribution.

Causal contrasts live or die by the guarantee that the two branches differ *only* in the intended perturbation. In practice, this can be facilitated by adopting a single root seed and deriving separate, documented streams for the sampler/order, augmentations, and model-side randomness; persisting the exact sequence of example indices (e.g., per-epoch "order hashes") allows order to be replayed verbatim. Potential sources of nondeterminism may be constrained by fixing framework and toolkit versions, enabling deterministic kernels where available (e.g., cuDNN determinism flags), and pinning dataloader parameters (number of workers, prefetching, sharding), while being mindful of operators that are nondeterministic on the target hardware. At branch points, snapshotting and logging optimizer buffers and any teacher/EMA or external-memory state makes CARRY and RESET policies auditable. Taken together, these controls (see Table 5) help isolate the intervention and align with the protocol outlined in §8.

Algorithm 4: Order-window swap (S2): permute a recorded window; optimizer state fixed

Input: As in Alg. 1; window size W (one epoch for RR; fixed steps for WR).
Output: $\Delta^{\text{early}}_{\text{order}}$, $\Delta^{\text{final}}_{\text{order}}$ + CIs; order/augmentation hashes.

At t_0, record the next W minibatch *IDs* and augmentation RNG states under policy $\mathcal{S}$; compute order and augmentation hashes

Define ϕ_{order}:

- CONTROL: replay recorded order and augmentation RNGs for the window.
- TREAT: replay the same *multiset* of W minibatches under a fresh permutation; reuse the recorded augmentation RNG per-example (or re-seed deterministically to isolate *order* only).

Ensure optimizer buffers, EMA/SWA, BN stats, and schedule Λ are identical at t_0 across branches
Call Alg. 1 with $\phi \leftarrow \phi_{\text{order}}$
return $\Delta^{\textit{early}}_{\textit{order}}$, $\Delta^{\textit{final}}_{\textit{order}}$ + *CIs*

Algorithm 5: Phase-boundary policy (S1): carry/reset/rewarm at pretrain→finetune

Input: Checkpoint $(\theta^\star, \mathcal{O}^\star, \text{EMA/SWA}^\star, \text{BN}^\star)$ from pretraining; finetune data; options {CARRY, RESET, REWARM}; probe $\mathcal{P}$; early window size k epochs.
Output: Early and final effects Δ_{phase} with CIs; calibration deltas.

Construct three branches at the phase start t_0:

1. CARRY: load $(\theta^\star, \mathcal{O}^\star, \text{EMA/SWA}^\star, \text{BN}^\star)$ unchanged.
2. RESET: load $\theta^\star$; zero optimizer moments (SGD velocity; Adam m, v); reinit EMA/SWA and BN running stats.
3. REWARM: as RESET, plus K-step LR warmup (same Λ thereafter).

Hold sampler policy, seeds, and augmentation pipelines identical across branches; match target LR at the phase start
Compute early effects over the first k epochs; continue to convergence; report accuracy and calibration (ECE/NLL) alongside function-space Δ
return $\{\Delta^{\textit{early}}_{\textit{phase}}, \Delta^{\textit{final}}_{\textit{phase}}\}$ + *CIs*

7 Desiderata for Future Diagnostics

A useful diagnostic should not merely confirm that training has memory; it should separate *where* that memory comes from, quantify *how much* it matters for what the model does, and do so *early* and *reproducibly*. The points below state properties a diagnostic ought to have, without prescribing a particular method.

D1. Attribution. A useful diagnostic isolates optimizer state, sampler/data-order state, and path/geometry as distinct sources of history. The organizing principle is orthogonal perturbation: one source is altered while the others are held fixed and the control is verified. Examples include resetting momentum/Adam buffers at a phase boundary while keeping weights and the exact data order unchanged; freezing the sampler policy across runs while varying whether EMA/SWA is carried over; or swapping the order of two short segments (*A then B* vs. *B then A*) to expose non-commutativity. Perturbations are minimal in duration and scope and are chosen on the appropriate timescale (step, epoch, phase, task) so that observed effects are attributable to the intended source rather than collateral changes. Portable interventions that operationalize these single-source perturbations appear in §6.2 and Table 4.

D2. Function-space sensitivity. Differences in *function space*—the model's predictive distributions or outputs on a held-out *probe set*—are reported alongside accuracy and loss. Task-aligned distances (e.g., total variation, Jensen–Shannon, Hellinger for classification; calibrated error measures for regression/probabilistic models) provide interpretable signals even when top-line accuracy is unchanged. Effect sizes are summarized with uncertainty (e.g., bootstrap confidence intervals), and, where appropriate, equivalence tests support "no material difference" claims rather than relying on point estimates.

D3. Representation tracking. Representation-level readouts (e.g., SVCCA/CKA or stitching-based probes) are used to contextualize feature drift across phases or interventions and are treated as descriptive rather than causal. Layers/splits are predefined to avoid cherry-picking. Similarity curves are accompanied by stability checks (resampled

Algorithm 6: External-memory intervention (S4): freeze or clear queue/bank for W

Input: Queue/bank size K; enqueue rate m per step; window $W \approx K/m$; probe $\mathcal{P}$.
Output: $\Delta^{\text{early}}_{\text{queue}}$, $\Delta^{\text{final}}_{\text{queue}}$ + CIs; queue fingerprints.

Define treatments at t_0:

1. FREEZE: stop enqueue/dequeue for W steps (read-only snapshot).
2. CLEAR: empty queue, then repopulate under normal policy (stronger perturbation).

Hold optimizer state, sampler order for the window, and schedule identical across branches; hash queue contents at t_0 and $t_0{+}W$
Invoke Alg. 1 with $\phi \in \{\text{FREEZE}, \text{CLEAR}\}$
return $\Delta^{\textit{early}}_{\textit{queue}}$, $\Delta^{\textit{final}}_{\textit{queue}}$ + *CIs; note stability differences between treatments.*

Target (S#)	Intervention (what you change; ***hold fixed***)	Window W	Estimand
S1 Optimizer / trajectory	**Opt-state reset**: at a predeclared step/epoch, zero momentum/Adam moments; optionally freeze EMA/SWA for W steps *(hold: weights, LR schedule, sampler policy & seeds, augmentations)*	≈1–2 half-lives ($h_{1/2} = \log 0.5/\log \beta$)	ATE_{opt}
S2 Sampler / data order	**Order-window swap**: record the next W minibatches (IDs), replay them once under a fresh permutation, then resume *(hold: weights, optimizer state, LR schedule, augmentation RNG streams)*	1 epoch (RR) or a fixed window (10^3–2×10^3 steps)	$\text{ATE}_{\text{order}}$
S1+S4 Phase boundary	**Phase policy**: at checkpoint (e.g., pretrain→finetune), branch into CARRY/RESET/REWARM for momentum/Adam and EMA/SWA *(hold: data order, seeds, eval protocol; equal target LR at phase start)*	First k epochs of the new phase (k=1–5)	ATE_{opt} (phase)
S5 Meta-state (teacher)	**Teacher-lag tweak**: temporarily change teacher EMA decay $\alpha \to \alpha'$ for W steps, then restore *(hold: student optimizer state, sampler policy, weights at branch)*	$\approx c/(1-\alpha)$ for small $(1-\alpha)$	$\text{ATE}_{\text{teacher}}$
S4 External memory	**External-memory ablation**: clear or *freeze* contrastive queues/memory banks for W steps (no enqueue/dequeue), then resume *(hold: optimizer state, sampler order, LR schedule, seeds)*	≈ queue length / enqueue rate	$\text{ATE}_{\text{queue}}$

Table 4: Minimal interventions for source-specific attribution of training memory. Each perturbs exactly one source, uses paired seeds, and is summarized with seed-averaged effect sizes and bootstrap CIs (cf. §6.1).

probes, noise injections) and are interpreted alongside function-space effects so that disagreements are informative rather than misleading.

D4. Early predictivity. Leading indicators are measured during early epochs to forecast final generalization under varied momentum schedules, sampler policies, or phase decisions. Emphasis is on cross-policy predictivity—signals that maintain rank or calibration across the interventions under study—rather than tuned heuristics for a single setting. Predictivity is quantified (e.g., rank correlation, calibrated regression) with uncertainty, and failures of early indicators are reported explicitly.

D5. Protocol clarity. Portability is achieved by specifying datasets and model families; random seeds and run counts; the *exact* data-order policy (with- or without-replacement, reshuffle rule and seed); augmentation regimes and their phase changes; optimizer-state handling across epochs/phases (carry vs. reset for momentum/Adam/EMA/SWA); probe construction and reuse across interventions; compute budget; and the full statistical treatment (confidence intervals, tests, equivalence margins). Parameters are predeclared where feasible to limit degrees of freedom. The objective is that another lab can replay the same memory perturbations and obtain comparable effect sizes.

8 Benchmarks & Reporting Checklist

The persuasiveness of a diagnostic depends on the testbeds that host it. Selection is guided by three criteria. **Accessibility:** runs complete on commodity hardware with multiple seeds and interventions (§7), enabling uncertainty reporting beyond single seeds. **Controllability:** each testbed exposes the parameters that create memory—optimizer state across phase boundaries, sampler policies and order, augmentation or preprocessing stages—so that attribution is meaningful. **Sensitivity:** the task reacts to step-, epoch-, and phase-scale perturbations; if order or state changes never move the needle, the testbed is uninformative for this purpose.

Control	Practice	Purpose
Randomness contract	Identify all randomness sources (initialization, data order, augmentation, batching, parallel execution). Derive named streams from a single root key per run and record those keys.	Reproducibility; enables paired A/B branches with identical noise.
Freeze the stack	Fix code, configuration, and numerical options into an immutable artifact; record dependency manifests and a lightweight hardware fingerprint.	Prevents environment drift from masquerading as treatment effects.
Persist data order	Store the exact sequence of example identifiers per step/epoch; emit a compact per-epoch "order checksum" for quick verification.	Allows verbatim replay and equality checks across branches.
Snapshot hidden state at branch points	Save weights and all state that can carry history (optimizer summaries, teacher/EMA or weight-averaging state, external queues/banks, scheduler and sampler state). Declare the policy (CARRY/RESET/REWARM) for each component.	Makes the intervention auditable; ensures only the intended source differs.
Hold invariants across branches	Keep batch boundaries, order policy, augmentation pipeline, schedules, and parallelism layout identical. Validate post-branch invariants (matching order checksums, schedule values, buffer-norm sanity).	Isolates the perturbation; reduces interference.
Account for residual nondeterminism	When exact determinism is unattainable, use paired seeds and repeated runs; report paired/bootstrap confidence intervals and CI widths.	Quantifies remaining noise; guards against overclaiming.

Table 5: Isolation and determinism controls (summary). Platform-agnostic practices that ensure branches differ only in the intended perturbation; complements the protocol guidance in §8.

One encoder family per modality typically suffices; horizons are capped to prioritize seeds and probes. A single well-instrumented setting per modality is more informative than a broad but under-specified suite. Small, controllable tasks are preferred so that single-source perturbations, paired seeds, and probe-based readouts are feasible. *Compact exemplars* that satisfy these constraints are summarized in Table 7, together with the memory sources they expose and convenient intervention windows W.

The *minimum information* required to enable attribution and replay is listed in Table 8. Items are domain-agnostic; modality-specific notes can be appended as needed. Multiple seeds are used where feasible; summaries include a central tendency (e.g., mean) and uncertainty (e.g., 95% CI). Paired designs are preferred; uncertainty is estimated via paired resampling/bootstrap over the probe and/or seeds, or via analytic intervals when appropriate. For "no material difference" claims, a practical equivalence margin ε is predeclared and an equivalence test (e.g., TOST) is applied; otherwise, confidence intervals are reported without dichotomous accept/reject language. When many interventions are tested, a false discovery rate is controlled or the number of tests is disclosed, avoiding selective reporting.

9 Related Areas & Transferable Insights

Training memory does not live only inside the supervised learning loop. Several nearby communities—continual learning, curriculum and pacing, data selection and coresets, reinforcement learning with replay, and federated optimization—have been explicitly managing (or struggling with) state that persists across updates for years. Reading these areas through the lens of our taxonomy (source, lifetime, visibility) yields concrete lessons for how to *measure* memory in standard deep learning pipelines.

Desideratum	Minimal properties to satisfy	Anti-patterns to avoid
D1 Attribution	Single-source perturbations; verify controls (buffer norms, order hashes); choose perturbation timescale to match source	Changing multiple sources at once; unverified "we reset X"; long perturbations that obscure effects
D2 Function-space	Task-aligned distances on a fixed probe; effect sizes with CIs and (when relevant) equivalence tests	Only reporting accuracy/loss; distances without uncertainty; ad-hoc probe selection
D3 Representation	Predefined layers/splits; pair similarity with function-space results; stability checks via resampling/noise	Treating similarity as causal; cherry-picked layers or runs; no robustness analysis
D4 Early predictivity	Early-window measures tested across policies; rank/fit reported with CIs; failures documented	Tuning on one policy and claiming universality; reporting single numbers without uncertainty
D5 Protocol clarity	Explicit sampler policy, augmentation schedule, state handling across phases; seeds/runs; reusable configs	Omitted order/augmentation details; unclear buffer handling at restarts; single-seed reports

Table 6: Solution-agnostic checklist for diagnostics that measure training memory in a way that is attributable, sensitive in function space, representation-aware, predictive early, and reproducible.

Modality	Compact testbed	Memory sources exposed & convenient perturbations (W)
Vision	CIFAR-10/100 + ResNet-18/MobileNetV2	Epoch-scale order (RR vs. WR; $W=1$ epoch), step-scale optimizer (momentum/Adam/EMA; $W\approx$1–2 half-lives), phase boundary (pretrain→finetune; first k epochs), staged augs.
Language	SST-2/AGNews/IMDb + DistilBERT	Length bucketing/packing (order), tokenization choices (phase), AdamW moments/EMA (step); probe on a fixed corpus slice.
Audio	SpeechCommands/ESC-50 + small CNN	Preprocessing stages (STFT/mel) as phase changes; clip order and speaker balancing; accuracy saturates early ⇒ calibration & function-space distances.
Graphs	Cora/CiteSeer/PubMed	Neighborhood sampling (order), train/val/test edge splits; schedules/regularizers across phases; fixed node/edge probe.
Recsys	MovieLens-100K/1M + MF/NeuMF	Negative-sampling policy (sampler memory), refresh cadence; metrics flat while predictive distribution shifts ⇒ probe-based distances.
Time series	ETTh1/h2, small M4 splits	Sliding-window construction and scaling (phase), EMA/momentum (step); evaluate forecast distributions and interval coverage.
RL	CartPole/MountainCar/A2C; DQN	Replay size/priority (sampler memory), target-network lag (server-like memory); episode order/environment seeds as data-order policy.
Federated	FEMNIST/Shakespeare; CIFAR/AGNews (Dirichlet)	Client sampling, server momentum/Adam (FedOpt), control variates; per-client dispersion and probe-based distances on fixed clients.

Table 7: Compact testbeds that expose step/epoch/phase/round memory with modest compute. W denotes the intervention window suggested by the source's characteristic lifetime.

9.1 Continual learning: explicit long-term memory and its costs.

Continual learning makes memory the main character: the system must acquire new tasks without catastrophically overwriting old ones. The classic picture is that naive fine-tuning destroys previously learned competencies, motivating mechanisms that *stabilize* parts of the model or *rehearse* past data [17, 18]. Elastic Weight Consolidation (EWC) penalizes movement along parameters deemed important to prior tasks using a Fisher-based quadratic, an explicit, inspectable memory of past curvature that persists over task scales [51]. Gradient Episodic Memory (GEM) stores exemplars and enforces per-step constraints to avoid increasing loss on earlier tasks, turning a replay buffer into a causal lever on forgetting [60]. iCaRL mixes exemplar replay with prototype-based classification, highlighting that the *form* of memory (raw samples vs. prototypes) changes both compute and measurement [59]. Large surveys synthesize

Category	Specification (to be recorded in paper or artifact)
Datasets	Name and version/split; resizing/filtering; federated partition scheme (if applicable).
Architectures	Model family/size; heads/tokenizer (NLP); feature front-end (audio).
Seeds & randomness	Initialization, data-order, augmentation/probe RNG; environment/client seeds (RL/FL); number of runs; naming of multiple RNG streams.
Sampler policy	With- vs. without-replacement and reshuffle rule/seed; bucketing/packing; negative sampling; neighborhood sampling; client sampling.
Optimizer & meta-state	Optimizer family; momentum/EMA/SWA/teacher; carry vs. reset at each boundary; BN handling (recalibration if applicable).
Schedules	LR schedule and parameters; warmup/restarts; interaction with state (e.g., whether restarts reset or carry buffers).
Transforms / preprocessing	Vision augs; tokenization/truncation; STFT/mel; normalization/windowing; whether stages change over time.
Compute budget	Max epochs/steps/episodes; batch size; gradient accumulation; evaluation cadence; hardware (context only).
Probe (if used)	Size; construction; reuse policy; whether fixed across branches; domain specifics (states for RL, clients for FL).
Metrics	Task metrics; at least one calibration/probabilistic metric when relevant; function-/representation-space distances; effect-size definitions.
Uncertainty	CI method (e.g., bootstrap with B resamples); equivalence margin ε for "no material difference"; multiple-comparison policy.
Artifacts	Configs (e.g., YAML); order hashes; buffer/BN-state logs; code commit/containers; evaluation scripts.

Table 8: Memory-sensitive reporting checklist (minimum information to enable attribution and replay).

these families—regularization, replay, parameter isolation/expansion—and repeatedly warn that reported gains depend strongly on protocol details: task ordering, buffer budgets, and how optimizer state is handled across task boundaries [17, 18, 90]. For our purposes: (i) rehearsal-style methods provide ready-made interventions for attribution (toggle, size, and sampling within buffers); (ii) quadratic anchoring (EWC/L2-SP) shows how "implicit" path memory can be made explicit via auxiliary state; and (iii) the community's metrics (backward/forward transfer, forgetting curves) are exemplars of task-aligned evaluation with uncertainty.

9.2 Curriculum and pacing: order as a first-class control parameter.

Curriculum learning and self-paced variants turn data order into policy: start with "easy" examples or high-confidence regions, then expand difficulty [91, 92]. Modern surveys catalog manual and automatic curricula across vision, NLP, and RL, including progress signals, teacher–student schemes, and environment-generated pacing [19, 93]. The central insight is that reordering the same multiset of examples can change stability, sample efficiency, and final generalization—an *epoch-scale* memory whose lifetime outlasts transient optimizer noise. That community also surface practical pitfalls that map directly to our gap analysis: curricula are often under-specified (how is "difficulty" scored? how often is the schedule recomputed?), and ablations rarely separate sampler effects from optimizer state (e.g., momentum buffers that retain pre-curriculum statistics). For diagnostics, curricula offer testbeds where order is deliberately structured; causal designs can swap curricula while freezing optimizer state (or vice versa) and measure function-space deltas with confidence intervals.

9.3 Data selection & coresets: sampler memory as principled selection.

A rich line of work formalizes sample *prioritization* and subset selection. Influence functions and data valuation quantify how training points move parameters and predictions, revealing that not all points are equally useful and that their value depends on the current state—an inherently stateful notion of sampling [94, 95]. Empirically, example "forgetting events" and early-loss proxies (e.g., EL2N) show that example hardness and learnability evolve across training [96, 97]. On the *algorithmic* side, importance sampling for deep nets proposes loss- or gradient-magnitude–weighted sampling to reduce variance and accelerate optimization [11, 12, 10]. Coreset methods such as CRAIG (gradient matching via submodular surrogates), GLISTER (bi-level generalization-driven selection), and GRAD-MATCH (explicit gradient matching) make the sampler's state visible and controllable through selected subsets [98, 99, 100]. These works provide *measurement* scaffolding: gradient-matching gaps, validation lift vs. subset size, and per-iteration selection diagnostics. For our study, they suggest straightforward interventions (freeze the subset while varying momentum, or keep optimizer state fixed while live-updating the subset) to attribute observed memory to the sampler vs. trajectory.

9.4 RL replay: mature design space for prioritized, stateful sampling.

Experience replay predates deep RL, framing memory as a buffer of transitions reused for sample efficiency [101]. In deep RL, replay is indispensable (e.g., DQN) and entails design choices that *shape* learning dynamics: buffer size (an effective half-life), prioritization by TD-error with importance-sampling correction, and distributed pipelines that decouple acting from learning [102, 13, 103]. Systematic studies show performance can swing with buffer size and prioritization schedules—clear, quantifiable manifestations of sampler memory and its lifetime [104, 105]. Distributed agents (Ape-X, R2D2) surface further issues: parameter lag and representational drift between actors and the learner, showing that *who* writes to the buffer and *when* experiences are consumed are causal factors [103, 106]. The RL toolkit thus offers off-the-shelf levers (priority exponent, IS annealing, buffer turnover) and reporting patterns (return with CIs across seeds, ablations over buffer hyperparameters) that translate directly to supervised settings: a prioritized sampler is simply PER without TD-errors, and its "age" distribution is a measurable lifetime parameter.

9.5 Federated optimization: cross-round memory via server/client state.

Federated learning bakes memory into the protocol: local steps on non-IID client data, then global aggregation. FedAvg ineffectively introduces long *round-scale* memory because models evolve locally before being mixed, and the server may accumulate its own momentum or adaptive moments [58, 8]. Client drift under heterogeneity motivated explicit control variates (SCAFFOLD), making cross-round corrections an inspectable state that reduces variance [107]. FedProx tethers local objectives to the global iterate, an explicit *path* constraint that stabilizes long-lived memory across rounds [62]. MIME shows how to mimic centralized momentum/Adam by sharing server-side statistics so that client updates track a target trajectory, again turning trajectory memory into an explicit, portable object [107]. Methodologically, the federated literature is unusually disciplined about reporting across seeds, client participation, and heterogeneity settings, and often provides ablations that flip server momentum, control-variates, or local step counts one at a time—precisely the kind of causal diagnostics our section advocates.

9.6 What transfers back.

Across these areas, three themes recur. First, *make memory visible*: record and expose the state that carries over (buffers, moments, control variates, selected subsets), so it can be reset or swapped. Second, *treat lifetime as a parameter*: buffer size in RL, number of local steps in FL, and curriculum pacing in supervised learning are all tunable half-lives with measurable effects on endpoints. Third, *evaluate with task-aligned, uncertainty-aware metrics*: continual learning's backward/forward transfer, RL's returns across seeds with CIs, and coreset generalization gaps all go beyond single-run accuracy. These are templates for the causal, function-space–aware measurements we argue are missing in standard training reports.

10 Open Problems

The case for *training memory* is clear: optimizer moments, sampler policies, and the path a model takes through parameter space all leave measurable traces. What is missing is a way to study these traces that scales across modalities, survives replication, and speaks to both theory and practice. We articulate below a set of open problems that, if addressed, would turn scattered evidence into cumulative science. Each problem is framed to be solution-agnostic but empirically actionable.

OP1. Causal attribution *at scale* with standardized protocols. Small, carefully controlled studies can disentangle optimizer state from data order or augmentation phases, but we lack *portable* protocols that make those attributions auditable across labs and modalities. The underspecification problem [80] shows that pipelines with identical headline accuracy can encode very different histories. Reproducibility programs have improved documentation practices [34], yet there is no community standard for memory-specific artifacts: e.g., *order hashes* for each epoch, *buffer-state logs* for momentum/Adam/EMA/SWA, and *pre-registered perturbations* that toggle exactly one memory source at a time (cf. §7). An open problem is to define such protocol cards and checksums so that one group's "AB$\neq$BA" claim is straightforward for another to replay and verify on different hardware, data sets, and architectures.

OP2. Early diagnostics that *predict across policies*, not just within them. We often observe early divergence in loss, calibration, or function-space distances, but when do these signals *generalize* across momentum schedules, augmentation phases, or sampler policies? Learning-curve extrapolation and freeze–thaw approaches [81, 82, 83] forecast eventual accuracy, and data-centric early signals (forgetting events, EL2N) identify influential examples [96, 97]. Flatness-oriented training (SAM) [7], large-batch generalization behavior [68], and double-descent phenomenology [108] suggest that early geometry interacts strongly with later outcomes. The open question is to define early-window readouts—with uncertainty—that retain *rank consistency and calibration* when we vary optimizer half-life, order policy,

or augmentation schedule, and to formalize equivalence margins that support "no meaningful difference" conclusions when predictions do not transfer.

OP3. Stateful sampling under label noise and distribution shift. Prioritizing by loss, gradient norm, or "learnability" can accelerate training [11, 12], but under label noise or spurious correlations, such feedback can entrench errors. Robust learning-from-noise methods (e.g., MentorNet) [109] and surveys [110] document mitigations; coreset/subset selection gives principled alternatives (CRAIG, GLISTER, GRAD-MATCH) [98, 99, 100]. Still lacking is a *unified diagnostic* that (i) estimates the bias/variance introduced by adaptive sampling with confidence intervals, (ii) stress-tests samplers under controlled class-conditional noise and realistic shifts (cf. WILDS) [78], and (iii) triggers *on-the-fly* corrections (importance weights, annealed priorities) when early signals indicate drift toward mislabeled or out-of-distribution pockets.

OP4. Deep-specific theory for non-commutativity and path dependence. We know stochastic updates need not commute in nonconvex objectives; order and path matter. There is geometric evidence—loss-landscape shape and mode connectivity [49, 46]—and stronger nonconvex SGD analyses [4], but we lack a theory that *quantifies* AB$\neq$BA effects *in function space* under realistic deep-training ingredients: momentum/EMA half-lives, heavy augmentation, and stateful sampling. Open directions include linking measurable half-lives (momentum β, EMA decay, replay turnover) to bounds on non-commutativity; characterizing when different orderings remain linearly connected vs. diverge into isolated basins; and extending analyses from IID/reshuffle regimes to curricula and prioritized policies that evolve with the learner. Connections to implicit bias in deep models [111, 112, 113] and flatness-aware updates [7] remain to be made precise.

OP5. Privacy and safety of explicit vs. implicit memory. Explicit memory (replay buffers, exemplar sets, server momentum snapshots) sharpens attribution but raises privacy and safety concerns: membership inference [114], training-data extraction [115], and gradient inversion [116]. Federated learning adds cross-round state and system-level visibility; secure aggregation and DP help but interact subtly with stateful optimization [117, 58, 118]. Open questions: how to design memory interventions that are *privacy-aware by construction* (e.g., DP-safe buffer turnover, per-sample accounting for prioritized draws); how to quantify the trade-off between stateful gains and leakage risk under realistic threat models; and how to extend unlearning guarantees to both explicit buffers and path-dependent implicit memory.

OP6. Robustness of memoryful training under shift. Distribution-shift benchmarks reveal gaps that accuracy alone hides [78, 77]. Do curricula, prioritized replay, or momentum half-life tuning improve or degrade calibration and OOD reliability? The open problem is a protocol that attributes OOD behavior to *specific* memory sources, using probe-based function-space distances with uncertainty alongside standard metrics, and that reports *per-subpopulation* effects so that "wins" are not averaged away.

OP7. System-level memory in federated and distributed regimes. Client ordering, local step counts, and server optimizers imprint cross-round memory; control variates (SCAFFOLD) and proximal terms (FedProx) make that memory explicit [61, 62, 8]. We lack measurement tools that separate client sampler effects (non-IID order, augmentation) from server-side trajectory memory, and that expose compute–privacy–convergence trade-offs. Recent analyses (e.g., non-parametric views of FedAvg) [119] motivate diagnostics that are agnostic to parametric assumptions yet sensitive to round-scale history.

OP8. Standardized uncertainty for training-dynamics claims. Finally, the community needs norms for uncertainty in *training-memory* studies: multi-seed summaries; paired/resampled bootstrap over fixed probes; equivalence tests with declared margins for "no meaningful difference"; and transparent pre-registration of which parameters will be varied (LR, β, batch size, order policy, state carry/reset) [86, 34]. Without these, memory effects remain fragile to researcher degrees of freedom.

Progress will come from paired advances: protocol artifacts that make source isolation checkable at scale; early diagnostics validated across families of optimizers and samplers; robust, debiased stateful sampling; deep-theory links between half-lives and non-commutativity; and privacy-aware memory accounting. Each thread is orthogonal to any single diagnostic, but together they enable results that travel across data sets, labs, and modalities.

11 Scope, Limitations, and Threats to Validity

11.1 Scope & non-claims.

This article advances a *measurement protocol* for attributing observed training effects to specific memory sources via minimal interventions and paired average treatment effect (ATE) estimands. It does *not* propose a new learning algorithm, optimizer, loss, or architecture; it is *not* a new benchmark; and it does *not* introduce a novel statistical estimator beyond standard paired designs and nonparametric uncertainty quantification. We intentionally avoid prescriptive thresholds for

what constitutes a "material" effect. Instead, we recommend reporting effect sizes with uncertainty (e.g., bootstrap confidence intervals over seeds/runs) and leaving materiality judgments to downstream users and domains. Our results are interventional within the studied training stacks and timescales; they do not claim causal transportability to unseen regimes, nor do they preclude unmeasured nuisance factors.

11.2 Search Strategy and Survey Methodology

This work is an *expert-guided scoping survey* with a structured search protocol. The initial corpus was assembled from field knowledge and canonical seed papers, then expanded via database queries and forward/backward passes. We queried multiple scholarly indexes to mitigate source bias:

- **Computer science indices:** ACM Digital Library, IEEE Xplore, DBLP, arXiv (cs.LG, cs.CV, cs.AI), and Google Scholar.
- **General indices:** Scopus and Web of Science (for cross-disciplinary coverage).

Time window: January 1, 2010 through September 23, 2025 (inclusive). We include pre-2010 optimization classics when directly relevant (e.g., heavy-ball momentum, Nesterov acceleration).

We grouped queries by our taxonomy's *source* axis; each block was executed with field restrictions (Title/Abstract/Keywords when supported) and then as a full-text fallback. Representative queries follow (parentheses indicate OR-lists; wildcards as supported by the engine):

S1: Optimizer/trajectory state.

```
("momentum" OR "Nesterov" OR "heavy-ball" OR "Polyak")
AND ("deep neural" OR "neural network" OR "deep learning")
AND (training OR optimization)

("Adam" OR "AdamW" OR "adaptive moment" OR "decoupled weight decay")
AND ("deep" OR "neural") AND (generalization OR calibration OR "test error")

("exponential moving average" OR EMA OR "stochastic weight averaging" OR SWA)
AND (neural OR deep) AND (generalization OR calibration)

("sharpness-aware" OR SAM) AND (generalization OR robustness)

("K-FAC" OR "Kronecker-factored" OR Shampoo OR "second-order" OR precondition*)
AND (neural OR deep)
```

S2: Sampler/data-order state.

```
("random reshuffling" OR "without replacement" OR "with replacement"
 OR "data order" OR "minibatch order")
AND (stochastic gradient* OR SGD) AND (deep OR neural)

(curriculum OR pacing OR "self-paced learning" OR "staged augmentation")
AND ("deep learning" OR "neural network")

("importance sampling" OR "prioritized sampling" OR "priority sampling")
AND (deep OR neural OR SGD)

(replay OR "experience replay" OR coreset OR "memory buffer")
AND (continual OR incremental OR "class-incremental" OR deep)
```

S3: Parameter-path dependence.

```
("mode connectivity" OR "loss landscape" OR "flat minima" OR "flatness"
 OR "path dependence" OR noncommutativ* OR "AB!=BA")
AND ("deep neural" OR "neural network")
```

S4: Architectural/external memory.

```
("memory bank" OR queue OR "feature bank" OR "momentum encoder" OR MoCo
 OR "instance discrimination" OR Hebbian OR "plasticity trace")
AND ("self-supervised" OR contrastive OR "deep learning")
```

S5: Meta-state.

```
("mean teacher" OR "teacher EMA" OR "temporal ensembling" OR "lookahead optimizer"
 OR "learned optimizer" OR "meta-optimizer")
AND ("deep learning" OR "neural network")
```

Cross-cutting measurement terms (combined with blocks above).

```
AND (generalization OR calibration OR "representation similarity" OR SVCCA OR CKA
 OR "function space" OR "prediction distribution" OR "causal" OR "intervention")
```

11.3 Eligibility criteria

We followed a two-stage screen with de-duplication:

1. **Title/abstract screen** by one reviewer; borderline items retained.
2. **Full-text assessment** for relevance to at least one source axis (S1–S5) and at least one lifetime (step/epoch/phase/task).

De-duplication used DOI and normalized titles across databases. **Forward/Backward passes** added forward citations and backward references from included "seed" papers, capped when no new items passed the following (Inclusion)–(Exclusion) criteria for two consecutive waves.

Inclusion:

1. Peer-reviewed conference/journal papers or widely adopted preprints (clear community uptake) in ML/AI/vision/NLP.
2. The work *introduces, analyzes, or empirically studies* mechanisms that carry state across updates/epochs/phases (optimizers, samplers, path/geometry, external memory, meta-state) *or* measures their effects (e.g., order dependence, averaging, representation drift).
3. For privacy/robustness (e.g., membership inference, data extraction), included when the phenomenon is tied to training history or state retention in the model.

Exclusion:

1. Theses, blogs, patents, and purely tutorial/introductory pieces without original empirical/theoretical contribution.
2. Works on reinforcement learning where replay is used solely for off-policy control without claims about training-memory mechanisms relevant to supervised/self-supervised training.
3. Papers whose only use of "memory" is architectural (e.g., LSTMs/Transformers) without updates to external state during training beyond standard weights/optimizer (unless they explicitly act as training-time memory banks).

11.4 Data extraction and coding schema

For each included paper we recorded:

- **Bib/venue/year** and **area**.
- **Taxonomy labels:** source (S1–S5), lifetime (step/epoch/phase/task), visibility (explicit/implicit).
- **Setting:** supervised, self-supervised, continual, federated; dataset/task family.
- **Mechanism studied:** e.g., momentum/EMA/SWA; RR vs. replacement; curriculum; prioritization; replay; mode connectivity; queues/banks; teacher EMA; learned optimizer.

Threat	Failure Mode	Impact	Mitigation
Numerical nondeterminism	Non-reproducible kernel paths or small numeric drift across environments.	Higher within-branch variance; spurious cross-branch deltas; weaker paired ATEs.	Record a substrate manifest (versions/digests); prefer deterministic modes; disable kernel autotuning; use named PRNG streams with fixed seeds; note which parts are best-effort.
Order drift (data-order mech.)	Parallel loading/partitioning/resume silently changes example-index sequences across epochs/branches.	Unintended S2/S3 perturbations masquerade as the intended intervention.	Derive per-epoch permutations from explicit (seed, epoch) contracts; log short digests per worker/partition to verify equality; freeze worker seeding/resume semantics; log sampler config verbatim.
Probe cross-talk	Probes (held-out batches, cached activations, retrieval keys, feature buffers) shared via caches/global state.	Contaminated measurements; shrunk or reversed deltas.	Materialize branch-local probe copies; isolate caches and PRNGs for eval; include probe content digests; treat retrieval/teacher/external-memory buffers as branch-scoped.
Normalization miscalibration	Interventions change features while normalization layers with running state retain stale stats.	Apparent gains/losses due to stale normalization, not the intended source.	Re-estimate running stats on a held-out calibration pass *per branch* before final eval, or explicitly freeze and state policy; for EMA/SWA, declare calibration data and passes.

Table 9: Threats framed as components.

- **Measurement:** metrics (accuracy, loss, calibration, SVCCA/CKA, function-space distances), whether *causal interventions* were performed (carry vs. reset, order swaps), and whether uncertainty was reported (CIs, tests).
- **Findings (qualitative)**.

11.5 Surveyal Limitations

Terminology around "memory" varies (e.g., implicit bias vs. flatness vs. order effects), which risks false negatives under strict keywording; we mitigated this with snowballing and cross-area seeds. We prioritize works that either analyze state carry-over or *measure* its impact; adjacent but orthogonal topics (e.g., architecture-only memory without training-time state changes) are not exhaustively covered.

11.6 Threats to validity

We phrase threats and safeguards in terms of abstract components that any implementation provides: the *execution substrate* (numerical kernels and schedulers), *pseudo-random number generators* (PRNGs), the *data-order mechanism* (how example indices are sequenced), *evaluation probes* (fixed inputs for function-space readouts), and *normalization layers with running state*. These abstractions keep the guidance independent of programming languages, libraries, and hardware (see Table 9).

12 Conclusion

Training memory is a feature, not a bug, of modern pipelines. Momentum and adaptive moments, weight and teacher averaging, data-order policies and staged augmentations, architectural side memory (queues, replay), normalization statistics, and nonconvex path dependence all carry history across updates. Our synthesis argues that these sources should be treated as *distinct*, with characteristic lifetimes and with varying degrees of visibility and resetability. The practical consequence is methodological: claims about training dynamics and generalization are persuasive only when they (i) perturb one source at a time, (ii) verify controls, and (iii) read out effects in function space with uncertainty.

We offered four structures to make this routine. The *source–lifetime–visibility* taxonomy clarifies where memory comes from and how long it lasts. The consolidation of theory and empirical patterns explains why order and optimizer state can independently move both trajectory and endpoint. The *causal estimands* and *perturbation primitives* turn attribution into seed-paired experiments with portable interventions and principled uncertainty (bootstrap CIs and equivalence testing). Finally, the reporting checklist enumerate what must be fixed, logged, and shared—order hashes, optimizer/EMA/BN snapshots, queue state, RNG stream contracts, and configuration artifacts—to make results auditable and replayable.

We see three near-term payoffs. First, sharper *attribution*: studies can separate optimizer vs. sampler vs. path effects rather than bundling them into a single headline gain. Second, better *predictivity*: early, function-space signals can be tested for rank stability *across* policies, not only within them. Third, improved *transfer and safety*: memory-aware protocols clarify how phase boundaries, replay, and server-side adaptivity affect calibration, robustness, and privacy.

Limitations remain. Our recommendations do not remove all nondeterminism, and they do not resolve deep-theory gaps in noncommutativity or the privacy risks of explicit memory (replay, server accumulators). They also ask for modest but real engineering discipline. Still, the bar is attainable on commodity budgets: small, well-instrumented testbeds with seed pairing and audit logs provide outsized clarity.

The path forward is communal. If authors routinely release audit artifacts alongside code, adopt single-source perturbations with seed-paired ATEs, and report function-space deltas with uncertainty, "training memory" will transition from folklore to cumulative science. We hope this survey accelerates that transition by supplying the vocabulary, contrasts, and minimal tooling to make memory *measurable* and *auditable*.

References

[1] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

[2] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, Atlanta, USA, 2013. pmlr.

[3] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[4] Konstantin Mishchenko, Ahmed Khaled, and Peter Richtárik. Random reshuffling: Simple analysis with vast improvements. *Advances in Neural Information Processing Systems*, 33:17309–17320, 2020.

[5] Mert Gürbüzbalaban, Asu Ozdaglar, and Pablo A Parrilo. Why random reshuffling beats stochastic gradient descent. *Mathematical Programming*, 186(1):49–84, 2021.

[6] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew G. Wilson. Averaging weights leads to wider optima in deep learning. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

[7] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[8] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[9] Ohad Shamir. Without-replacement sampling for stochastic gradient methods. *Advances in neural information processing systems*, 29, 2016.

[10] Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[11] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534, Stockholm, Sweden, 2018. PMLR.

[12] Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. *Advances in Neural Information Processing Systems*, 31, 2018.

[13] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[14] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[15] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529, Vancouver, Canada, 2019. PMlR.

[16] Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding representation similarity through statistical testing. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1556–1568, Virtual Only, 2021. Curran Associates, Inc.

[17] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

[18] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[19] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

[20] Damien Ferbach, Baptiste Goujaud, Gauthier Gidel, and Aymeric Dieuleveut. Proving linear mode connectivity of neural networks via optimal transport. In *International Conference on Artificial Intelligence and Statistics*, pages 3853–3861. PMLR, 2024.

[21] Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey. *arXiv preprint arXiv:2309.15698*, 2023.

[22] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.

[23] Itay Safran and Ohad Shamir. How good is sgd with random shuffling? In *Conference on Learning Theory*, pages 3250–3284. PMLR, 2020.

[24] Hengxu Yu and Xiao Li. High probability guarantees for random reshuffling. *arXiv preprint arXiv:2311.11841*, 2023.

[25] Lei Huang. *Normalization Techniques in Deep Learning*. Springer, 2022.

[26] Dominik Rivoir, Isabel Funke, and Stefanie Speidel. On the pitfalls of batch normalization for end-to-end video learning: a study on surgical workflow analysis. *Medical Image Analysis*, 94:103126, 2024.

[27] Tobias Uelwer, Jan Robine, Stefan Sylvius Wagner, Marc Höftmann, Eric Upschulte, Sebastian Konietzny, Maike Behrendt, and Stefan Harmeling. A survey on self-supervised methods for visual representation learning. *Machine Learning*, 114(4):1–56, 2025.

[28] Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. *ACM Computing Surveys*, 57(9):1–52, 2025.

[29] Yongquan Yang, Haijun Lv, and Ning Chen. A survey on ensemble learning under the era of deep learning. *Artificial Intelligence Review*, 56(6):5545–5589, 2023.

[30] Huiming Chen, Huandong Wang, Qingyue Long, Depeng Jin, and Yong Li. Advancements in federated learning: Models, methods, and privacy. *ACM Computing Surveys*, 57(2):1–39, 2024.

[31] Cheng Wang. Calibration in deep learning: A survey of the state-of-the-art. *arXiv preprint arXiv:2308.01222*, 2023.

[32] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 220–229, New York, NY, USA, 2019. Association for Computing Machinery.

[33] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[34] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Lariviere, Alina Beygelzimer, Florence d'Alche Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research(a report from the neurips 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164):1–20, 2021.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[37] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[38] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[39] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[40] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[41] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

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

[43] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[44] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2019.

[45] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624, Virtual Only, 2020. Curran Associates, Inc.

[46] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[47] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318. PMLR, 10–15 Jul 2018.

[48] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR, 13–18 Jul 2020.

[49] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[50] LI Xuhong, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International conference on machine learning*, pages 2825–2834. PMLR, 2018.

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

[52] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[53] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[54] Manu Srinath Halvagal and Friedemann Zenke. The combination of hebbian and predictive plasticity learns invariant object representations in deep sensory networks. *Nature Neuroscience*, 26(12):1906–1915, 2023.

[55] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[56] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29:3988 – 3996, 2016.

[57] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[58] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[59] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[60] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[61] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020.

[62] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

[63] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[64] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[65] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

[66] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent, 2018.

[67] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress.

[68] 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, 2017.

[69] Anastasiia Koloskova, Tao Lin, and Sebastian U Stich. An improved analysis of gradient tracking for decentralized machine learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11422–11435. Curran Associates, Inc., 2021.

[70] Dominik Csiba and Peter Richtárik. Importance sampling for minibatches. *Journal of Machine Learning Research*, 19(27):1–21, 2018.

[71] Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 872–881, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[72] Stephen Grossberg. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, 37:1–47, 2013. Twenty-fifth Anniversay Commemorative Issue.

[73] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1225–1234, New York, New York, USA, 20–22 Jun 2016. PMLR.

[74] Ruoyu Sun. Optimization for deep learning: theory and algorithms, 2019.

[75] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2017.

[76] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 06–11 Aug 2017.

[77] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[78] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR, 18–24 Jul 2021.

[79] MohammadReza Davari, Stefan Horoi, Amine Natik, Guillaume Lajoie, Guy Wolf, and Eugene Belilovsky. Reliability of cka as a similarity measure in deep learning, 2022.

[80] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu, Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61, 2022.

[81] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization, 2014.

[82] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3460–3468. AAAI Press, 2015.

[83] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. In *International Conference on Learning Representations*, 2017.

[84] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[85] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

[86] Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[87] Donald J Schuirmann. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of pharmacokinetics and biopharmaceutics*, 15(6):657–680, 1987.

[88] Esteban Walker and Amy S Nowacki. Understanding equivalence and noninferiority testing. *Journal of general internal medicine*, 26(2):192–196, 2011.

[89] Daniël Lakens. Equivalence tests: A practical primer for t tests, correlations, and meta-analyses. *Social psychological and personality science*, 8(4):355–362, 2017.

[90] Martin Mundt, Yongwon Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 160:306–336, 2023.

[91] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[92] M. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[93] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[94] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 06–11 Aug 2017.

[95] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR, 09–15 Jun 2019.

[96] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning, 2019.

[97] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20596–20607. Curran Associates, Inc., 2021.

[98] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6950–6960. PMLR, 13–18 Jul 2020.

[99] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):8110–8118, May 2021.

[100] Krishnateja Killamsetty, Durga S, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5464–5474. PMLR, 18–24 Jul 2021.

[101] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8(3–4):293–321, May 1992.

[102] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[103] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay, 2018.

[104] Shangtong Zhang and Richard S. Sutton. A deeper look at experience replay, 2018.

[105] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3061–3071. PMLR, 13–18 Jul 2020.

[106] Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

[107] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning, 2021.

[108] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: where bigger models and more data hurt*. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, dec 2021.

[109] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them, 2019.

[110] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. Early Access; arXiv:2007.08199.

[111] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

[112] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[113] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming, 2020.

[114] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[115] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 2021)*, pages 2633–2650, Virtual Only, 2021. USENIX Association.

[116] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[117] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[118] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17455–17466, Virtual Only, 2021. Curran Associates, Inc.

[119] Lili Su, Jiaming Xu, and Pengkun Yang. A non-parametric view of fedavg and fedprox: Beyond stationary points, 2022.