

Which Fine-Tuning Failed? The Reported Failure and Recovery of Fine-Tuning in Model Editing Are Findings About the Edit Protocol, Why the Breadth-First Recovery Changes What the Editor May Remember, and the Matched Comparisons That Would Assign the Credit

Pranay Mahendrakar

SONYTECH · ORCID 0009-0003-7224-029X

pranaymahendrakar@sonytech.in · mahendrakarpranay@gmail.com · +91 6361723454

Abstract. Fine-tuning has been treated as the weak baseline of model editing since the field's first benchmarks. A 2025 study argues that the weakness was an artefact of implementation: when edits are optimised in epochs over mini-batches instead of one at a time to convergence, and when a single, empirically selected module is the only thing tuned (an MLP down-projection for two of the three models, a whole layer for the third), fine-tuning beats every specialised editor it was compared with on editing success, and holds up at 100,000 edits on a 72-billion-parameter model. In this analysis the author reviews and synthesises published results to show that both the failure and the recovery are findings about the edit protocol rather than about fine-tuning as a method. The paper separates five axes on which editing comparisons silently differ: what the editor may keep of earlier edits, how hard each edit is optimised, which parameters move, which auxiliary data are used, and how the edited model is scored. The recovery moves on at least three axes at once. Its breadth-first pipeline revisits every earlier edit in every epoch, which in continual-learning terms is full replay. Its scorer is autoregressive decoding, under which plain one-at-a-time fine-tuning was already the strongest sequential editor in an earlier study from the same group. Its tuning location was chosen on three of the five benchmarks that make up its capability score. Read along these axes, the record since 2020 shows fine-tuning losing under single-pass, teacher-forced, single-example protocols and tying or winning under batch, augmented or replayed ones. The paper consolidates 24 published results in one table, states an attribution procedure, and specifies the matched comparisons, chiefly a streaming test with a bounded replay buffer, that would show what fine-tuning contributes on its own.

Keywords: model editing; knowledge editing; fine-tuning; sequential editing; lifelong editing; continual learning; experience replay; evaluation methodology

SONYTECH ORCID: 0009-0003-7224-029X

1. Introduction

Model editing asks for something that sounds modest. A deployed language model holds a wrong or outdated fact, and the editor should change that fact, make the change hold under rephrasing, and leave everything else alone (Yao et al., 2023; Wang et al., 2024c). Since 2021 the field has built three families of specialised methods for the job: hypernetworks that turn a gradient into a targeted weight change (De Cao et al., 2021; Mitchell et al., 2022a), locate-then-edit methods that solve for a low-rank update to a chosen MLP matrix (Meng et al., 2022; Meng et al., 2023; Fang et al., 2025), and methods that add parameters or memories beside the model (Mitchell et al., 2022b; Hartvigsen et al., 2023; Wang et al., 2024b). Through most of that period, ordinary fine-tuning appeared in result tables as the baseline the new method beat. The framework paper that much of the field uses states the comparison in its abstract: in its experiments on LLaMA-2, editing methods beat "traditional fine-tuning in terms of reliability and generalization" (Wang et al., 2024a).

A study accepted at ICLR 2026 reverses that verdict (Yang et al., 2026). Its authors examine how fine-tuning had been implemented inside editing codebases and find a procedure that they call depth-first: each edit request is optimised alone until the model has memorised it, and only then does the procedure move to the next request, with a batch size of one and a single pass. They contrast it with the breadth-first procedure of ordinary supervised training, which cycles through the whole set of edits for several epochs in mini-batches. Changing only the pipeline, with batch size held at one, raises the editing success of FT-M on ZsRE with LLaMA-3-8B-Instruct from 75.30 to 99.70 percent. Adding mini-batches and a sweep over every layer and five module types (160 locations for LLaMA-3-8B) yields LocFT-BF, which in their lifelong-editing comparison on 3,000 edits attains the highest reliability in all nine dataset-model combinations, an average of 33.72 points above the second-best method. It then sustains 100,000 ZsRE edits on Qwen2.5-72B at 99.80 percent reliability while keeping capability at 62.54 percent against the unedited 64.84. The authors conclude that the long-reported failure arose "not from the inherent limitations of fine-tuning" but from implementations ill-suited to it.

This paper accepts nearly every number in that study and disputes the inference drawn from them. The claim that fine-tuning was always capable and merely misimplemented has the same shape as the claim it overturns: both attribute to a method a result that was produced by a protocol. An editing comparison fixes, usually without saying so, at least five things that are not the method. These are: what the editor is allowed to keep of the edits it has already made; how many optimisation steps each edit receives and when they stop; which parameters are allowed to move; which auxiliary data enter training; and how the edited model is decoded and scored. The depth-first fine-tuning baseline lost under one setting of these five. LocFT-BF won under a different setting, and it differs on at least three of them, not only on the pipeline its name records.

The first difference matters most. A breadth-first pipeline over a stream of edits must, by construction, keep every earlier edit and train on it again. The anchor states this directly for the real-time case: each new request is added to the cumulative dataset, which triggers a fresh fine-tuning run. In the vocabulary of continual learning that is full experience replay, a regime that the continual-learning literature has long found to reduce catastrophic forgetting substantially (Rolnick et al., 2019; van de Ven and Tolia, 2019). The depth-first baseline kept nothing. The specialised sequential editors it is compared with keep something in between. AlphaEdit carries forward a fixed-size statistic of earlier edit keys (Fang et al., 2025). UltraEdit maintains running feature statistics across turns (Gu et al., 2025). GRACE and WISE keep side memories (Hartvigsen et al., 2023; Wang et al., 2024b). The headline comparison therefore sets an editor that retains every past edit verbatim against editors that retain a compressed summary or none. That is a comparison of memory budgets as much as of update rules.

The second difference is the scorer. The anchor evaluates with the WILD framework, autoregressive decoding with natural stopping and an LLM judge (Yang et al., 2025). The same group introduced it a year earlier. Under it, the average success of six editors on Llama-2-7b-chat, on QAEdit questions the model had failed before editing, was 38.5 percent against the 96.8 percent typical of earlier reports. In sequential editing of 1,000 QAEdit facts, one-at-a-time FT-M reached 40.5 percent average reliability. The average over all six methods, FT-M included, was 9.3 percent, and without FT-M the other five averaged under 3 percent (the author's arithmetic on the source's Table 9) (Yang et al., 2025). Under an autoregressive scorer, then, depth-first fine-tuning was already the strongest of the six editors Yang et al. (2025) tested (FT-M, MEND, ROME, MEMIT, GRACE and WISE) before the pipeline was changed. None of those six is AlphaEdit, RLEdit or UltraEdit, the newer editors in the anchor's comparison, and whether depth-first FT-M leads them under WILD has not been reported. Part of what

the anchor reports as a rehabilitation of fine-tuning was delivered by a change of scorer that had already reordered those six editors.

The third difference is selection. LocFT-BF's tuning location was picked from a sweep in which capability was measured on GSM8K, MMLU and WMT16. Three of the five tasks averaged into the capability score of the main comparison are those same three. The text does not say whether the 1,000 ZsRE samples used for the sweep are disjoint from the 3,000 used in the comparison.

None of this makes the anchor's result wrong. It makes it a result about a cell of a protocol grid. The paper's thesis is this: neither the field's long verdict against fine-tuning nor the 2026 verdict for it can be read as a property of fine-tuning until comparisons match the five axes. On the axis that matters most for deployment, memory of past edits, no published comparison matches them. The literature is closer to settling the question than either side suggests. Gangadhar and Stratos (2024) had shown fine-tuning matching MEMIT on edit score in batch mass-editing two years earlier, Hewitt et al. (2024) had shown LoRA beating MEMIT under canonical-example editing, and Thede et al. (2025) had shown continual LoRA fine-tuning with weight merging matching specialised editors at the scale of hundreds of thousands of real edits. The fine-tuning baseline failed in some cells and won in others well before 2026. The useful question is which cell a deployment occupies.

The genre is analysis. In this analysis the author reviews and synthesises published results to show which protocol settings each verdict on fine-tuning depends on, and which comparison would decide between them. Each figure quoted below names the study that measured it, and anything the paper computes from reported values is marked as the author's arithmetic. The paper's own contributions are the five-axis decomposition of an editing protocol (Section 4), the account of breadth-first training as replay and of what each sequential editor retains (Section 5), the consolidated evidence in Table 1 and the retention comparison in Table 2, the attribution procedure in Algorithm 1, and the matched design in Section 11.

An objection should be met at once. A reader may say that the anchor's authors already handle the replay point in their Section 2.5, where they argue that retraining on a cumulative dataset is cheap because editing workloads are small next to pretraining. They do argue this. The argument concerns cost, not attribution. Cheap replay is still replay, and a comparison in which one arm replays and the others do not cannot say how much of the gap the update rule accounts for. A second objection is that this paper restates an earlier analysis by the author of what importance-based selective forgetting can guarantee. It does not. That paper concerned removal of knowledge and the guarantees the word unlearning carries. This one concerns insertion of facts, and how the

ranking of insertion methods depends on the protocol that ranks them.

The paper proceeds as follows. Section 2 traces where the verdict against fine-tuning came from and shows that it was never unanimous. Section 3 sets out what the anchor's design holds fixed. Section 4 defines the five axes. Sections 5 to 9 take the axes in turn: retention, optimisation, scope, auxiliary data and scoring. Section 10 consolidates the evidence and states the attribution procedure. Section 11 specifies the comparisons that would settle the open part. Sections 12 to 14 give the strongest case against this paper, what is not known, and its limitations.

2. Where the verdict against fine-tuning came from

The verdict did not start as one. The paper usually cited for it argued the opposite. Zhu et al. (2020) posed the task of modifying specific facts in a transformer while preserving the rest, and benchmarked natural baselines. Their main finding was that fine-tuning only on the modified facts, with an l-infinity constraint keeping weights near their original values, was a simple and effective method, and that it worked better when applied to particular layers than to the whole model. Their unconstrained fine-tuning baseline did destroy the unmodified facts, and that half of the result is what later papers carried forward.

The hypernetwork papers that followed framed the problem as editing from one example. MEND's abstract states that when given only a single input and its desired output, fine-tuning approaches "tend to overfit" (Mitchell et al., 2022a). KnowledgeEditor presented itself as a way to fix a fact without expensive re-training or fine-tuning (De Cao et al., 2021). In this framing the baseline is one example, a few gradient steps, no augmentation and no constraint, and its failure on paraphrase and locality is a fact about that baseline.

The locate-then-edit papers kept fine-tuning in the table and showed it trading one desideratum against another. ROME's abstract reports that on CounterFact it keeps specificity and generalisation together while other methods give one up (Meng et al., 2022). MEMIT's own results complicate the picture. In mass-editing 10,000 zsRE facts in GPT-J, its weight-decay fine-tuning baseline FT-W scored 42.1 against MEMIT's 50.7. It beat both MEND (20.0) and sequentially applied ROME (2.6), which the MEMIT authors put down, tentatively, to fine-tuning applying its objective once (Meng et al., 2023). The MEMIT paper also records that the weight-decay setting was "highly sensitive to the number of edits" and was tuned only at the 10,000-edit point, with at most 25 optimisation steps at a learning rate of $5e-4$ and early stopping once the loss fell below 0.01. That is one tuning decision, and it fixed the baseline's position for the field.

From 2023 the framework papers standardised the baselines. The editing survey of Yao et al. (2023) and the EasyEdit framework implement FT-L, a fine-tuning baseline that updates one MLP layer chosen by causal tracing (Wang et al., 2024a), and KnowEdit adds FT-M, which supervises the full target answer with the prompt masked (Zhang et al., 2024). The anchor's audit finds the depth-first procedure in the fine-tuning editors of this lineage. It also reports that FT-L supervises only the final token of the target answer, which helps explain why FT-L scores near zero in every cell of its controlled comparison (Yang et al., 2026, Table 1).

Running alongside this lineage is a set of results in which fine-tuning variants do well. Gangadhar and Stratos (2024) mass-edited 10,000 facts in GPT-J with LoRA, optimising the likelihood of the target conditional on the prompt and adding paraphrase and unrelated-fact augmentation. Their edit score was 62.0 on zsRE against MEMIT's 50.8 as they report it, and 86.5 on CounterFact against 85.8. The gain is on edit score only. On CounterFact the same variant's fluency was 352.0 and its consistency 5.2, against MEMIT's 619.9 and 40.1, and adding a language-modelling loss on Wikipedia text brought fluency back to 609.2 but consistency only to 29.2, still below MEMIT's (Gangadhar and Stratos, 2024, Table 4). Hewitt et al. (2024) defined editing from canonical examples with strictly out-of-distribution evaluation and a bound on deviation from the initial model, and found LoRA ahead of both full fine-tuning and MEMIT on Pythia models. Thede et al. (2025) built WikiBigEdit from more than 500,000 real Wikidata changes and found that at a fixed inference budget, continual fine-tuning matched or beat the specialised editors at scale, especially when each step's adapter was merged into the previous ones. These are not marginal venues: Findings of ACL, and ICML 2025.

So the record before 2026 does not show fine-tuning failing. It shows one implementation of fine-tuning failing under one family of protocols, and other implementations succeeding under others. Taken alone, the anchor's claim to be the first to revisit the failure of fine-tuning-based editing is too strong. Its own related-work section cites Gangadhar and Stratos, but classes that work as data augmentation rather than as a re-examination.

3. What the anchor did and what its design holds fixed

The anchor's argument has three steps. Each is supported by a controlled comparison, and each fixes more than its label says.

Step one: pipeline. With batch size fixed at one, the anchor runs four fine-tuning editors depth-first and breadth-first on ZsRE and CounterFact, on LLaMA-3-8B-Instruct, Mistral-7B-v0.1 and Qwen2.5-7B-Instruct (Yang et al., 2026, Table 1). The anchor states 1,000 ZsRE edits

for the matching FT-M setting on LLaMA in its Figure 1, but gives no count for the table itself. The reliability gains are large for the two editors that neither supervise only the last answer token, as FT-L does, nor restrict training to a very sparse parameter set, as RoseLoRA does (the anchor's own explanation of the two exceptions). On ZsRE, FT-M rises from 75.30 to 99.70 percent on LLaMA, 41.10 to 100.00 on Mistral and 58.70 to 99.80 on Qwen. AdaLoRA rises from 3.80 to 90.50, 2.50 to 88.50 and 3.40 to 90.60. FT-L stays at or near zero, and RoseLoRA stays below 5 percent. A shard experiment shows the mechanism. Under the depth-first pipeline the first fifth of the edits starts high and decays as later fifths are written. Under breadth-first, all fifths rise together across epochs.

The capability column of the same table moves less uniformly than the text's summary that the change "improves performance across all methods". On ZsRE, FT-M's capability falls from 18.14 to 15.64 on Mistral under the breadth-first pipeline. AdaLoRA's falls from 53.47 to 51.21 on Qwen, and on CounterFact AdaLoRA's falls from 49.88 to 39.27 on LLaMA. Every FT-M capability value in the table, under either pipeline, lies far below the unedited models' 57.26, 44.82 and 58.52. The pipeline change fixes forgetting of edits. On its own it does not fix damage to the model.

Step two: granularity. Holding the breadth-first pipeline, the anchor increases batch size for FT-M and AdaLoRA and reports that FT-M's capability improves substantially, to the point of exceeding AdaLoRA. AdaLoRA gains little, which the authors attribute to its low-rank structure already regularising the update. Generalisation falls somewhat as batch size grows, an effect the authors relate to the large-batch generalisation gap of Keskar et al. (2017) and propose to offset with augmentation.

Step three: location. The anchor fine-tunes each of five module types (whole layer, attention, MLP, MLP up-projection, MLP down-projection) at each layer, 160 locations for LLaMA-3-8B. Each is trained independently on 1,000 ZsRE edits and evaluated on reliability, generalisation and capability, with capability measured on GSM8K, MMLU and WMT16 (Yang et al., 2026, Section 3). Reliability is near-perfect almost everywhere. Generalisation favours coarse modules and early layers, and capability favours fine modules and late layers. The selected locations are MLP down-projection at layer 22 for LLaMA, the whole of layer 23 for Mistral and MLP down-projection at layer 6 for Qwen.

The main comparison (Table 3 of the anchor) then sets LocFT-BF against MEMIT, RECT, AlphaEdit, WISE, RLEdit and UltraEdit on 3,000 edits from each of ZsRE, CounterFact and WikiBigEdit. Reliability and generalisation are scored under WILD. Capability is the mean accuracy over MMLU, Natural Questions, SST-2, WMT16 and GSM8K. LocFT-BF's reliability lies between

98.87 and 99.73 percent in all nine cells. Its generalisation is best in six of nine. Its capability stays within 3.4 points of the unedited model in every cell (the author's arithmetic on the table; the largest gap is Mistral with CounterFact, 41.46 against 44.82). On Qwen with CounterFact its generalisation is 11.77 percent, against MEMIT's 40.90. The authors trace this to CounterFact's paraphrase prompts, which prepend unrelated text, and to MEMIT's and AlphaEdit's use of random-prefix augmentation. With the same five-prefix augmentation, LocFT-BF's generalisation on that cell rises to 44.90 (Yang et al., 2026, Table 7).

Four further features of the design matter for attribution. First, the breadth-first pipeline iterates over the entire edit set across epochs. In the main comparison LocFT-BF therefore has simultaneous access to all 3,000 edits, while the sequential baselines, as their protocol requires, see them in turns. The anchor reports that baselines' batch sizes were "empirically tuned" within the available compute. The text does not give the values, so the reader cannot tell whether MEMIT, which Yang et al. (2025) found to work best when all requests are edited in one batch, was run that way. Second, the anchor appends an end-of-sequence token to every target for every method. It notes that earlier implementations omitted it and that standard fine-tuning requires it. That is a correction, but one made on the fine-tuning convention, applied to editors whose official configurations were developed without it, and the anchor does not report whether it helps or hurts them. Third, the reported time per edit is an average over a single breadth-first run. It does not include the retraining that the real-time version of the pipeline would trigger on each arrival. Fourth, a LlamaFactory reimplementation of LocFT-BF reaches different generalisation values: for Qwen with CounterFact, 41.80 percent instead of 11.77. The authors attribute such gaps to low-level differences such as token-level against sample-level loss averaging (Yang et al., 2026, Table 8). A 30-point swing from framework-level implementation differences, of which loss averaging is the one the authors name, is itself evidence for the anchor's own thesis that implementation details move editing results.

The same appendix compares full-parameter fine-tuning in LlamaFactory. The authors describe its effect on capability as catastrophic, and on two of three models the numbers bear this out: on ZsRE, 40.44 against the unedited 57.26 on LLaMA, and 16.07 against 44.82 on Mistral. On Qwen it is 59.05 against 58.52, no loss at all, though on CounterFact Qwen's capability does fall, to 53.53. Whether full fine-tuning damages the model depends on the model and, for Qwen, on the dataset.

4. Five axes of an editing protocol

Every editing result is produced by an editor operating under a protocol, and the protocol has at least five settings that are not part of the editor's update rule. The

axes below are the author's construction; the literature reports each in isolation, never as a set.

A. Retention: what the editor may keep of earlier edits. At one extreme the editor keeps nothing and sees each request once (A0). Next comes a compressed sufficient statistic of past edits, such as a key covariance or a running mean and variance (A1). Next is an explicit memory of past edits, consulted at inference but not trained on (A2). At the other extreme the editor retrains on every past edit verbatim (A3, replay). Batch editing of a fixed set of known requests is the limiting case of A3 with a single arrival.

B. Optimisation: how hard each edit is pushed. This covers the number of steps, the stopping rule (a fixed count, a loss threshold, a probability threshold), the learning rate and the batch size. Depth-first editing to convergence and breadth-first training for a fixed number of epochs differ on this axis as well as on retention.

C. Scope: which parameters may move. The options are a single matrix, a module, a layer, a set of layers, a low-rank adapter or the whole network, together with the rule that selected them.

D. Auxiliary data: what besides the requests enters the update. Candidates are paraphrase or prefix augmentation of the request, unrelated facts included to anchor locality, and statistics of preserved knowledge estimated from a reference corpus. A meta-training set for a hypernetwork is another case.

E. Scoring: how the edited model is read. The choices are teacher-forced or autoregressive decoding; truncation at the target length or natural stopping; token-match ratio, exact match or a judge; a locality probe or a battery of general tasks; and whether consequences of the edit, such as multi-hop entailments, are tested.

The rest of this paper takes the axes in order and asks, for each, what published evidence says about how much it moves a ranking. One point should be stated before the evidence. The three method families are not as distinct on these axes as their names suggest. Gupta et al. (2025) show that a locate-then-edit method can be written as a two-step fine-tuning procedure. A gradient-descent step finds a target activation, and a closed-form least-squares step writes it into an MLP matrix. ROME, MEMIT and AlphaEdit are therefore fine-tuning with a particular scope (C), a particular auxiliary statistic (D) and a particular optimiser for the second step (B). MEND transforms the fine-tuning gradient (Mitchell et al., 2022a). The contest between fine-tuning and specialised editors is mostly a contest among settings of B, C and D, with A and E held differently on each side.

5. Retention: the breadth-first recovery is a replay result

5.1 Breadth-first over a stream is replay

Consider the anchor's own description of how the breadth-first pipeline handles requests that arrive one at a time: each new request is added to the cumulative edit set and starts a new fine-tuning run. After k arrivals the model has been trained on all k edits for some number of epochs. Edit 1 has been revisited in every run since it arrived. This is the definition of an unbounded replay buffer. Replay of all past data brings a sequential learner close to joint training on everything it has seen, which is why continual-learning studies tend to treat it as a reference point rather than as one method among others. Rolnick et al. (2019) found that replay buffers over all past experience substantially reduced catastrophic forgetting in sequential reinforcement learning. A randomly pruned buffer of limited size did almost as well. van de Ven and Tolia (2019) found that when task identity must be inferred at test time, regularisation methods such as elastic weight consolidation (Kirkpatrick et al., 2017) failed, and that replaying representations of earlier experience seemed necessary.

The anchor's shard experiment (Section 3) is, read this way, a replay ablation. Depth-first, earlier shards decay as later ones are written. Breadth-first, all shards are trained together and none decays. The explanation the anchor gives, that the depth-first pipeline lets later edits overwrite earlier ones, is the standard description of catastrophic forgetting in the absence of replay. It is correct. But it identifies the decisive variable as access to past edits, not as the fine-tuning update rule. A depth-first editor that also replayed past edits would be expected to show the same recovery. So would a breadth-first editor denied them.

The cost of unbounded replay is simple arithmetic. If each run trains for a fixed number of epochs E over the cumulative set, the k -th run processes E_k examples, and the total after N arrivals is $E N(N+1)/2$ example-passes. That is quadratic in the number of edits, where depth-first editing is linear. At the anchor's 100,000-edit scale, retraining from the cumulative set on every arrival would cost about 50,000 times the work of a single pass in total ($N(N+1)/2$ divided by N , with $N = 100,000$, on the author's arithmetic). The anchor's alternative, buffering requests into small batches before each retraining run, divides this by roughly the buffer size but keeps the quadratic form. Its argument that this is affordable because editing data are small next to pretraining corpora may well hold at 100,000 short facts. It is an argument about price, and it does not bear on whether the comparison was matched.

5.2 What the other editors retain

Table 2 sets out, from each method's own description, what a sequential editor carries forward from one round to the next. The point of the table is that no two arms of the anchor's main comparison sit at the same level of axis A.

The A1 entries deserve emphasis, because they show the specialised field moving toward retention on its own. AlphaEdit's sequential objective adds a term that penalises disturbance of the keys written in previous rounds, and its closed-form solution carries their covariance forward (Fang et al., 2025; the objective is restated in Suresh and Hariharan, 2026). UltraEdit's lifelong normalisation keeps running statistics over every past editing feature (Gu et al., 2025). Ma et al. (2026) find the same module, lifelong normalisation, which normalises value gradients with running statistics, in both UltraEdit and RLEdit, and argue that it is what lets these two editors survive long horizons. They report that removing it causes immediate collapse, and that with it early edits can make later ones easier. The editors that held up best in the anchor's comparison other than LocFT-BF, RLEdit and UltraEdit, are both editors that carry state forward. What MEMIT carries depends on the implementation. Fang et al. (2025) write its sequential solution with the same previous-edit key term as AlphaEdit's, differing only by AlphaEdit's projection, while Yang et al. (2025) describe MEMIT as relying on "covariance matrices derived from original LLMs"; the anchor does not say which form it ran. MEMIT sits mid-table: on CounterFact reliability it is second or third of the seven methods, and on ZsRE and WikiBigEdit fourth to sixth (the author's count from the anchor's Table 3).

The ordering is suggestive, not clean. GRACE and WISE keep explicit memories and still fail under WILD scoring: Yang et al. (2025) report both falling below 5 percent on reliability and generalisation in their single-edit comparison. In the anchor's comparison WISE's reliability ranges from 4.17 to 38.03 percent. AlphaEdit, an A1 editor, is a second exception. On Mistral and Qwen it falls below MEMIT in all six reliability cells, to between 2.70 and 6.67 percent in five of them, where MEMIT has 12.47 to 39.57 (Yang et al., 2026, Table 3). Retention is necessary for a sequential editor to keep its edits. It is not sufficient for those edits to be read out by an autoregressive scorer. The retention axis is therefore a confound to control, not a single explanation of the ranking.

5.3 Two results that conflict until retention is named

The same group published two results about batch size for fine-tuning editors that point in opposite directions. In Yang et al. (2025), FT-M edited QAEdit facts sequentially in batches. It did best at a batch size of one and degraded drastically as the batch grew, while MEMIT did best only when all requests were edited in one batch. All methods stayed below 30 percent at 1,000

edits. In Yang et al. (2026), FT-M's capability improved with larger batches. The two are reconciled by axis A. The 2025 batches were single-pass and sequential: as the sequential protocol requires, each batch was edited once and not revisited, so a larger batch meant fewer, larger updates with no replay. The 2026 batches sit inside a breadth-first loop over the whole set, so a larger batch means lower-variance gradients over data that will be seen again next epoch. "Batch size" names a different intervention in each.

WikiBigEdit supplies the middle case (Thede et al., 2025). There, continual LoRA fine-tuning is trained per timestep on that timestep's edits, which is breadth-first within an arrival batch but without replay of earlier timesteps. It matches specialised editing at moderate scale but forgets as timesteps accumulate. Merging each timestep's adapter into the running adapter with an interpolation weight of 0.25 removes most of that forgetting. Thede et al. report that the merged variant overtakes plain continual fine-tuning, and through it every specialised editor, after roughly 100,000 edits, with at most marginal forgetting and some positive backward transfer. Merging is a form of A1 retention: the past survives as a weighted average of weights, not as data. This is the one published result in which fine-tuning without replay holds up at deployment scale, and it does so by adding a retention mechanism.

5.4 What the retention reading predicts

If the recovery is mainly a replay result, three predictions follow. Each could fail. First, LocFT under a strictly streaming protocol with no replay should lose most of its advantage in edit retention, as FT-M lost it in the anchor's own depth-first condition. Second, a bounded replay buffer of m past edits should recover most of the reliability even at small m , as Rolnick et al. (2019) found in reinforcement learning that buffers of 5 and 50 million frames, out of 900 million, performed almost as well as one of 450 million. Third, giving the A0 and A1 editors the same replay should narrow the gap. MEMIT run as one batch over all 3,000 edits is the obvious case, since that is the setting its authors designed it for. None of these comparisons has been published. Section 11 states them as a design.

6. Optimisation: over-optimising an edit is a shared failure

The anchor's diagnosis of the depth-first pipeline is that driving each edit to convergence over-optimises it, and that batch-size-one gradients are noisy enough to damage general capability. The diagnosis is not peculiar to fine-tuning. The locate-then-edit literature reached the same diagnosis for its own methods at about the same time.

Gupta et al. (2025), having recast locate-then-edit as two-step fine-tuning, located two causes of degradation in sequential locate-then-edit methods. The first is over-

Table 2. What each sequential editor retains of earlier edits, from each method's own description (A0 none, A1 statistic, A2 explicit memory, A3 replay).

Editor	Retained across rounds	Level
FT-M, depth-first (Zhang et al., 2024)	nothing beyond the weights	A0
MEMIT, sequential batches (Meng et al., 2023)	preserved-knowledge covariance; keys of previous edits as Fang et al. (2025) state it, none as Yang et al. (2025) describe it	A0/ A1
AlphaEdit (Fang et al., 2025)	keys of previous edits in the objective	A1
UltraEdit (Gu et al., 2025)	running mean and variance of edit features	A1
REdit (Li et al., 2025)	hypernetwork trained on edit sequences; lifelong normalisation (Ma et al., 2026)	A1/D
GRACE (Hartvigsen et al., 2023)	codebook of edited keys and values	A2
WISE (Wang et al., 2024b)	side memory with a router	A2
LocFT-BF (Yang et al., 2026)	every past edit, retrained each epoch	A3

optimisation of the target activation in the gradient step. Unedited Llama-2-7B and Llama-3-8B recall their own facts with average probability of 52 and 49 percent, while facts written by MEMIT are predicted at 78 to 79 percent. The second is continuous growth in the norm of edited matrices. Stopping the gradient step once the target becomes the most probable token, and adding a Frobenius-norm penalty, let them scale to 10,000 sequential edits while keeping downstream performance. Zhang et al. (2025) describe the same phenomenon as editing overfit: edited models give the new target disproportionate probability, and this damages its use in multi-hop questions. They also report that standard anti-overfitting measures do not cure it. Ma et al. (2025) trace the loss of general ability under sequential editing to the growing condition number of edited matrices. Zhang et al. (2026) trace it to the disruption of dominant singular directions of the pretrained weights.

Each of these is a finding about axis B (how hard and how far an update is pushed) or about its interaction with axis C. The family of the update rule is not what matters. Gu et al. (2024) reach a version of the same conclusion for four editors: side effects on general ability come from changing the weights too much, and a regulariser on relative weight change reduces them. The anchor's breadth-first pipeline and mini-batching are, on this reading, one more way to stop over-optimising an edit. Training to a fixed epoch budget over many edits replaces training each edit to zero loss.

The implementation lesson the anchor draws for fine-tuning holds for the specialised methods too. Gupta et al. (2024b) show that the edits that made ROME collapse a model in a single step were an artefact of irregularities in its implementation, and that a corrected r-ROME made large-scale sequential edits without collapse. Yang et al. (2024b) trace the same collapse to inconsistent handling of prefixed and unprefixed keys in ROME's update equation. The fix is to use prefixed keys throughout. The anchor's title says fine-tuning was done wrong. The record says editors were done wrong in both families, that in each case a corrected implementation removed the failure the method had been known for, and that published baselines were rarely re-run with the corrections.

This is a familiar pattern outside editing. Melis et al. (2018) found that standard LSTMs, properly regularised and tuned, beat a series of newer language-model architectures. Henderson et al. (2018) showed that reported results in deep reinforcement learning varied with random seeds and implementation details enough to make many claimed improvements hard to judge. Musgrave et al. (2020) found methodological flaws across metric-learning papers and concluded that four years of reported advances amounted to little once those flaws were removed. Dacrema et al. (2019) reproduced only 7 of 18 neural recommendation methods with reasonable effort, and found that 6 of the 7 could often be beaten by simple heuristics. In each case, part of the reported progress came from comparisons in which the baseline had been run less carefully than the proposal. Model editing has the same structure, with one addition: the baseline was mistuned in a way, depth-first with no replay, that the field's task definition encouraged.

7. Scope: location decides the damage, not the success

The anchor's location sweep reproduces, with a fine-tuning editor and an autoregressive scorer, a finding first made for locate-then-edit methods. Hase et al. (2023) asked whether the layer that causal tracing identifies as storing a fact is the best layer at which to edit it. For ROME on GPT-J, they found the correlation between tracing effect and edit success at layer 6 to be slightly negative ($\rho = -0.13$). Across editing problems and methods, adding the tracing effect to a regression that already contained the choice of layer increased explained variance in rewrite score by at most 3.2 percent. The choice of layer itself predicted far more. The anchor reports that near-perfect reliability is reachable by tuning almost any module at almost any layer. It cites, as consistent, work arguing that knowledge is not confined to a few neurons (Chen et al., 2025).

What the sweep adds is that location matters a great deal for capability and generalisation even though it barely matters for reliability. In the anchor's sweep, capability favours fine modules in late layers, and generalisation favours coarse modules in early layers. In the anchor's

LlamaFactory comparison, full-parameter fine-tuning reaches the highest generalisation of any variant, at a large cost in capability on two of three models (Section 3). Biderman et al. (2024) report the same trade in ordinary domain adaptation: LoRA learns less than full fine-tuning and forgets less, and mitigates forgetting better than weight decay or dropout. Hewitt et al. (2024), with deviation from the initial model bounded, found LoRA ahead of full fine-tuning. Axis C is therefore a dial between learning the edit broadly and leaving the model alone. The anchor's choice to prioritise capability is a position on that dial, not a discovery that fine-tuning is benign.

Scope interacts with architecture, and it does so for both families. Suresh and Hariharan (2026) reproduced AlphaEdit's results on its original models. On newer architectures they found it failing: for Phi-3-3.8B, whose fused projection matrices they suggest may break the subject-token localisation the method assumes, and for Gemma-2-2B, where after a tokenisation fix ZsRE rewrite accuracy was still 0.43 percent. They attribute the latter, tentatively, to a post-MLP normalisation that may undo the linearity the closed-form update relies on. In the anchor's comparison AlphaEdit's ZsRE reliability is 64.50 percent on LLaMA-3-8B but 3.30 on Mistral and 2.70 on Qwen. A method whose scope is fixed by a mechanistic assumption transfers only as far as the assumption does. LocFT-BF's scope is chosen by empirical sweep instead, which transfers further: the same rule worked on a mixture-of-experts model and on Qwen models up to 72B in the anchor's tests. But it brings in a selection step, taken up next.

Selection on the reported benchmarks. LocFT-BF's location was selected using capability measured on GSM8K, MMLU and WMT16, and the capability score reported in the main comparison averages MMLU, Natural Questions, SST-2, WMT16 and GSM8K. Three of the five tasks overlap. The sweep used 1,000 ZsRE samples, and the main ZsRE comparison uses 3,000 randomly sampled ones. The text does not say whether the two sets are disjoint. The baselines' locations were taken from their official configurations, chosen on their authors' own criteria. This does not show that LocFT-BF's capability margin is inflated. Its locations were chosen per model, not per benchmark, and its capability also holds on CounterFact and WikiBigEdit, which played no part in the sweep. It does mean that the capability comparison is between one method tuned on the reported metric and others that were not. The matched version would select every method's scope on held-out tasks, or tune none.

8. Auxiliary data: generalisation tracks augmentation

Axis D explains most of the generalisation differences in the record, and it cuts against fine-tuning's critics and its defenders alike.

The augmentation the specialised methods rely on is often invisible in comparisons. ROME and MEMIT optimise their target vectors over a set of random-prefix contexts, ten of length 5 and ten of length 10 in the ROME configuration used for MEMIT's GPT-J baselines. They also estimate the covariance of preserved keys from 100,000 Wikitext samples (Meng et al., 2023, Appendix B). AlphaEdit's null-space projection is computed from the same kind of preserved-knowledge statistic (Fang et al., 2025). MEND trains its hypernetwork on a separate set of edits (Mitchell et al., 2022a; Gangadhar and Stratos, 2024), and the anchor groups RLEdit with it as a method that needs extra labelled data (Li et al., 2025; Yang et al., 2026). The fine-tuning baselines these methods were compared with typically trained on the single request alone.

When fine-tuning receives comparable data, the comparison changes. Gangadhar and Stratos (2024) gave each edit 15 model-generated pseudo-paraphrases and 20 unrelated facts from the training split, fine-tuning on 360,000 sequences in all for 10,000 requests. Their ablation isolates the steps. Masking the prompt alone lifted zsRE edit score from 44.8 to 58.3, ahead of MEMIT. Paraphrase augmentation raised generalisation. Random-fact augmentation raised CounterFact locality from 38.0 to 72.0 and the overall edit score to MEMIT's level, at the cost in fluency and consistency noted in Section 2. The anchor's CounterFact generalisation deficit closed when LocFT-BF received MEMIT's prefix augmentation: on Qwen from 11.77 to 44.90 percent, on LLaMA from 33.23 to 50.93, on Mistral from 39.53 to 49.70 (Yang et al., 2026, Table 7).

Work on knowledge acquisition outside editing explains why augmentation should matter. Allen-Zhu and Li (2024) found in controlled pretraining that a fact is reliably extractable by question only if it appeared in sufficiently varied forms during training. Without that variation it may be memorised but not extractable. Ovadia et al. (2024) found that unsupervised fine-tuning learns new facts poorly and that exposure to many variations of a fact helps. Mecklenburg et al. (2024) found that generating synthetic training data fact by fact covered new knowledge more evenly than scaling it by token count, when injecting knowledge by supervised fine-tuning. Augmentation has limits. Berglund et al. (2024) report that the reversal curse, a model trained on "A is B" failing to answer for B, survives data augmentation. Gekhman et al. (2024) report that facts new to a model are learned more slowly than familiar ones and, once learned, raise its tendency to hallucinate linearly. So augmentation decides much of what is called generalisation in editing benchmarks. It does not decide whether an edit propagates to what it entails.

The consequence for attribution is direct. A generalisation gap between a specialised editor and fine-tuning, measured when only one of them is augmented, belongs to axis D, not to the editor. The anchor shows

this for its own method on CounterFact, and Gangadhar and Stratos showed it for mass-editing. Neither shows how much of the gap remains when both arms use the same augmentation and the same locality data.

9. Scoring: the ranking moved when the scorer moved

Axis E has the largest documented effects in the record, and it is the axis on which the anchor's recovery most depends.

Yang et al. (2025) divide a conventional editing evaluation into four modules: input, generation, truncation and metric. The conventional settings were a bare question with no context, teacher forcing (feeding ground-truth tokens at each step), truncation at the length of the target, and a token-match ratio. Replacing them with task-style prompts, autoregressive decoding, natural stopping and an LLM judge lowered single-edit success sharply: from about 96 percent to 43.8 on ZsRE and 38.9 on QAEEdit, averaged across the methods and the three models tested (a different average from the 38.5 percent of Section 1, which is for Llama-2-7b-chat alone). Teacher forcing and target-length truncation produced most of the inflation. Both leak the ground truth, its content and its length, into a test that should not have it. The methods that did worst under realistic scoring were the ones most inflated by teacher forcing. Li et al. (2025b) report a similar gap between teacher-forced success and behaviour under autoregressive generation, and add that most methods they studied degraded pre-trained capabilities.

The reordering is the important part. In sequential editing of 1,000 facts under WILD, FT-M, run depth-first at batch size one, had 40.5 percent average reliability. The other methods were MEND, ROME, MEMIT, GRACE and WISE. The source's average of 9.3 percent reliability is taken across all six methods including FT-M, and so is its average locality of 21.3 percent (Yang et al., 2025, Table 9); without FT-M the other five average under 3 percent reliability (the author's arithmetic). The paper attributes FT-M's relative stability to its optimising the current parameters at each step, where the other methods rely on hypernetworks or covariance matrices computed from the original model. Under realistic scoring, then, fine-tuning was the strongest of the six editors Yang et al. (2025) tested before the anchor changed its pipeline. None of those six is AlphaEdit, RLEdit or UltraEdit, and whether depth-first FT-M leads those three under WILD has not been reported. Against the older editors, much of the "long-standing view of fine-tuning as a weak baseline" had already been undercut by the scorer. The anchor's pipeline change took FT-M from strongest of six weak editors to strongest of its own comparison by a wide margin.

The same axis contains a second scoring change: the move from a locality probe to a battery of general tasks.

The anchor measures capability on five benchmarks instead of the conventional neighbour-prompt locality, citing evidence that editing damages general ability in ways locality probes miss (Yang et al., 2024a; Gupta et al., 2024a). Yang et al. (2024a) found that a single edit can collapse a model on benchmark tasks, and that nearly all methods they examined collapsed after a few sequential edits on hard cases. Gupta et al. (2024a) found sequential ROME and MEMIT forgetting both earlier edits and downstream skills, gradually and then abruptly. Suresh and Hariharan (2026) found AlphaEdit stable through its original 3,000-edit horizon but degrading beyond it. On CounterFact with Llama-3-8B, efficacy was 99.25 percent at 2,000 edits and 49.02 at 5,000, then 69.35 at 10,000, and MMLU fell from about 0.6 to about 0.1 F1 by 10,000 edits. The choice of capability battery and of horizon decides which methods look stable.

Two disagreements in the current literature come down to axis E. UltraEdit's authors report that their method sustains up to 2 million edits with high accuracy, using exact match as the primary metric, WILD as a secondary one on ZsRE, and evaluation after all turns are complete (Gu et al., 2025). The anchor reports that RLEdit's and UltraEdit's capabilities "decline sharply" past 20,000 ZsRE edits and that their editing success stays low (Yang et al., 2026, Section 5.1). The protocols differ in dataset, metric, the EOS convention imposed on all methods, turn size and model, and neither paper reports the other's setting. The two claims cannot be reconciled from the published text, and neither can be preferred on it. The anchor's priority claim, to be the first method sustaining 100,000 edits while preserving capability, is a claim inside its own scorer.

Finally, the standard scorers do not test what an edit should change beyond itself. MQuAKE found that editors which recall edited facts accurately fail badly on multi-hop questions whose answers should change as a consequence (Zhong et al., 2023). RippleEdits found that methods fail to make consistent changes to related facts, and that a simple in-context baseline scored best (Cohen et al., 2024). In-context editing more generally reached competitive success with fewer side effects than gradient methods on GPT-J (Zheng et al., 2023). Li et al. (2024) report conflicts and distortions in the model's knowledge structure after editing. Hase et al. (2024) argue that the editing problem is underspecified as belief revision, and set out twelve open problems in defining and benchmarking it. Cheng et al. (2026) add a downstream axis no editing benchmark scores: across 254 configurations, edits decayed substantially when the model was later fine-tuned. In the worst case, 25.27 percent of AlphaEdit's previously successful zsRE edits in GPT-J were lost. The anchor's comparison, like its baselines', scores the edit and its paraphrase. Whether LocFT-BF's edits propagate, or survive later adaptation, better than other methods' is not known.

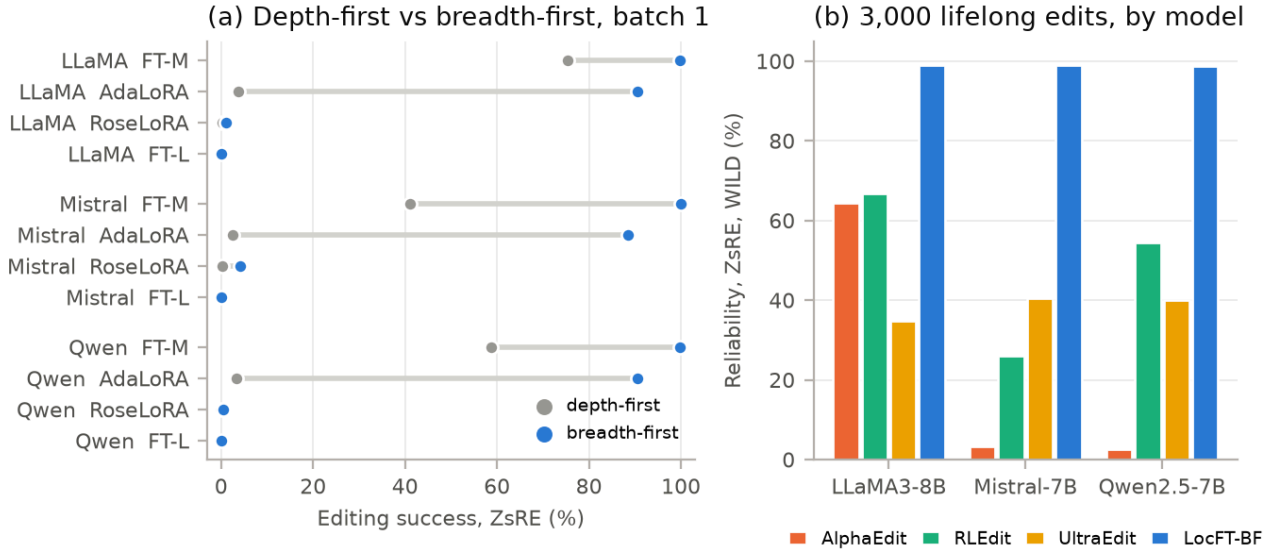


Figure 1. Published reliability values from Yang et al. (2026), ZsRE. (a) Table 1: editing success of four fine-tuning editors under the depth-first (grey) and breadth-first (blue) pipelines, batch size 1 (the anchor states 1,000 edits for the matching setting of its Figure 1). (b) Table 3: WILD-scored reliability of four editors on 3,000 lifelong edits by model; the unequal heights of the AlphaEdit, RLEdit and UltraEdit bars across models are the cross-model instability the anchor reports.

Figure 1 draws two of the anchor's own tables. Panel (a) shows that the pipeline change is decisive for FT-M and AdaLoRA and inert for FT-L and RoseLoRA. It is not a property of fine-tuning in general but of the fine-tuning editors that supervise the whole answer and train a non-trivial parameter set. Panel (b) shows the main comparison by model. The scale of LocFT-BF's lead depends on the model mostly because the baselines' reliability depends on it.

10. The evidence in one table, and an attribution procedure

Table 1 consolidates the published results the argument rests on, one row per result. The axis codes are the author's reading of each design, using Section 4. They are not labels the sources use. The result column gives the source's own numbers.

Read down the retention column, the table sorts itself. With one exception whose retention setting the author has not classified (Hewitt et al., 2024, n/s), every result in which fine-tuning beats or ties a specialised editor has fine-tuning in batch mode or at A3, or with merging (Thede et al., 2025), or under autoregressive scoring (Yang et al., 2025). Among editors that supervise the full answer, every result in which fine-tuning loses badly has it at A0 with no augmentation, or unaugmented against an augmented editor; FT-L and RoseLoRA, which stay near zero even under the breadth-first pipeline (Section 3), are the exceptions. Where retention is matched, D decides: in batch mode unaugmented FT-W loses to MEMIT on edit score (Meng et al., 2023) and augmented LoRA fine-tuning beats it on edit score (Gangadhar and Stratos, 2024), though not on fluency or consistency. No

result compares the two families at the same retention level, with the same auxiliary data and under the same scorer, and none compares them at A3 against A3 in a streaming protocol. That empty cell is the paper's main finding.

Algorithm 1 states the decision structure this paper applies to a reported ranking of editors. It is meant to be run on a methods section before a headline is quoted.

Applied to the anchor's main comparison, Algorithm 1 labels LocFT-BF (A3; fixed epoch budget; scope selected on three of five capability tasks; no augmentation; AR plus CAP at 3,000 edits) and the baselines (A0 to A2; official settings with tuned batch size; scope from official configurations; prefixes and covariance for MEMIT and AlphaEdit; same scorer). The labels differ on A and C, so the ranking is a joint effect of the update rule, replay and selection. Applied to MEMIT's FT-W comparison, it labels both arms batch-mode, with MEMIT augmented and FT-W not, under a teacher-forced scorer. The labels differ on D, so the gap is first a data gap. Applied to Yang et al. (2025), it labels FT-M, MEND, ROME and MEMIT A0 (MEMIT as those authors describe it) and GRACE and WISE A2, under the same autoregressive scorer. Among the four A0 editors that is the one sequential comparison in the record matched on retention, and fine-tuning wins it; FT-M also beat the two A2 editors.

11. What would settle the open part

The open question is not whether fine-tuning can edit facts. It can, under the right protocol, as the record has shown since 2020. The question is how much of fine-tuning's advantage in the anchor's comparison belongs to its update rule, and how much to replay, scope selection

Table 1. Published results on fine-tuning and specialised editors, with retention (A0-A3), auxiliary data (D) and scorer (TF teacher-forced, AR autoregressive, CAP capability battery) as read by the author; n/s means not classified.

Source	Setting	A	D	Scorer	Reported result
Zhu 2020	BERT, ALBERT; constrained FT	batch	none	accuracy	l-inf constrained layer FT beats unconstrained
Meng 2023	GPT-J, 10k zsRE, mass edit	batch	prefixes, cov.	TF argmax	MEMIT 50.7; FT-W 42.1; MEND 20.0; ROME 2.6
Gangadhar 2024	GPT-J, 10k, LoRA FT	batch	15 para + 20 facts	TF	edit score zsRE 62.0 vs MEMIT 50.8; CF 86.5 vs 85.8
Gangadhar 2024	same, ablation	batch	none	TF	mask alone: 44.8 to 58.3 on zsRE
Hewitt 2024	Pythia, canonical examples	n/s	none	OOD eval	LoRA beats full FT and MEMIT
Hase 2023	GPT-J, ROME, layer sweep	single	prefixes	TF	tracing vs success $\rho = -0.13$; added R^2 at most 3.2%
Thede 2025	WikiBigEdit, over 500k edits	A1 (merge)	none	QA	LoRA-merge beats editors after about 100k edits
Yang 2025	6 editors, single edit	single	per method	TF to AR	about 96% to 43.8% (ZsRE), 38.9% (QAEdit)
Yang 2025	1k sequential, QAEdit	A0, A2	per method	AR	FT-M 40.5%; all six 9.3% (others under 3%); locality 21.3% (all six)
Yang 2025	batch-size sweep, 1k edits	A0	per method	AR	FT-M best at batch 1; MEMIT best all-in-one
Yang 2026	ZsRE, LLaMA, FT-M (1k in its Fig. 1)	A0 to A3	none	AR	75.30 to 99.70% reliability
Yang 2026	Table 1, ZsRE, LLaMA, AdaLoRA	A0 to A3	none	AR	3.80 to 90.50% reliability
Yang 2026	Table 1, ZsRE, Mistral, FT-M	A0 to A3	none	CAP	capability 18.14 to 15.64 (pre 44.82)
Yang 2026	3k lifelong, 9 cells	A3 vs A0-A2	none (LocFT)	AR, CAP	LocFT 98.87-99.73%; + 33.72 pts mean
Yang 2026	CF, Qwen, prefix aug.	A3	prefixes	AR	generalisation 11.77 to 44.90%
Yang 2026	full FT (LlamaFactory), ZsRE	A3	none	CAP	40.44 vs 57.26 (LLaMA); 59.05 vs 58.52 (Qwen)
Yang 2026	Qwen2.5-72B, 100k ZsRE	A3	none	AR, CAP	99.80% rel.; capability 62.54 vs 64.84
Gu 2025	UltraEdit, up to 2M edits	A1	none	EM, AR	high accuracy claimed to 2M edits
Gupta 2025	MEMIT, AlphaEdit, 10k	A0, A1	prefixes, cov.	TF, CAP	edited facts $p = 0.78-0.79$ vs $0.49-0.52$ natural
Suresh 2026	AlphaEdit, Llama-3-8B, CF	A1	prefixes, cov.	TF	99.25% at 2k, 49.02% at 5k, 69.35% at 10k
Suresh 2026	AlphaEdit, Gemma-2-2B	A1	prefixes, cov.	TF	ZsRE rewrite 0.43%
Gupta 2024b	r-ROME, sequential	A0	prefixes, cov.	TF, CAP	disabling edits were an implementation artefact
Zhong 2023	MQuAKE, multi-hop	single, batch	per method	multi-hop	recall high; multi-hop fails
Cheng 2026	edits then later FT	n/a	per method	TF	up to 25.27% of edits lost (AlphaEdit, GPT-J)

Algorithm 1. Attributing a reported ranking between two editors X and Y.

```

Input: a study reporting  $X > Y$  on edit success, generalisation or capability
Step 1: RETENTION. For each arm, what is kept of earlier edits?
    none (A0), statistic (A1), explicit memory (A2), replay or batch (A3).
    If levels differ: the ranking is a joint effect of rule and retention;
    require a matched-retention arm before attributing it to the rule.
Step 2: OPTIMISATION. Steps, stopping rule, learning rate, batch size, epochs
    per arm; were all arms tuned with the same budget on the same split?
Step 3: SCOPE. Which parameters move in each arm, and how were they chosen?
    If chosen on the reported benchmarks for one arm only: mark the
    affected metric as selected.
Step 4: AUXILIARY DATA. Paraphrases, prefixes, locality facts, reference-corpus
    statistics, meta-training edits, per arm. If unequal: attribute the
    generalisation and locality gaps to data before method.
Step 5: SCORER. Teacher-forced or autoregressive; truncation; metric;
    locality probe or capability battery; horizon; propagation tests?
    Compare only rankings produced under the same scorer.
Step 6: OUTPUT. The ranking, labelled (A, B, C, D, E) per arm. Two rankings
    contradict each other only if their labels match.

```

and the scorer. A design that would answer it has five features. No published study has all five.

A retention factorial. Cross the editor (LocFT, MEMIT, AlphaEdit, UltraEdit) with retention: none, a bounded replay buffer of m past edits for several values of m , and full replay. For MEMIT and AlphaEdit, full replay means re-solving over all past edits, which for MEMIT is its

native batch mode. Arrivals should be streamed one at a time, or in fixed small turns, with evaluation after every turn on all past edits. The retention reading of Section 5 predicts that LocFT without replay falls toward the anchor's depth-first FT-M numbers. It also predicts that a small buffer recovers most of the gap, and that MEMIT with full replay closes much of its deficit. If LocFT without replay keeps near-perfect retention at 3,000

streamed edits, the retention reading is wrong and the anchor's attribution to the update rule stands.

Matched cost accounting. Report, for every arm, the total compute to process N arrivals, not the average time per edit within a single run. Full replay grows quadratically in N (Section 5.1). Bounded replay and A1 statistics grow linearly. The comparison of interest is accuracy at matched total cost.

Held-out scope selection. Choose every method's tuning location or editing layers by the same rule, on edits and capability tasks disjoint from those reported. Or take every method's location from a single fixed heuristic, such as the anchor's late MLP down-projection or its proportional-depth rule.

Matched auxiliary data. Give every arm the same augmentation: the same prefixes or paraphrases and the same locality facts, or none. Report generalisation separately for prefix-style and paraphrase-style probes, since CounterFact's are the first kind.

A scorer that reads consequences. Score reliability and generalisation under autoregressive decoding with natural stopping. Add a multi-hop or ripple probe on the edited facts (Zhong et al., 2023; Cohen et al., 2024), a capability battery disjoint from any selection set, and a retention test after a round of ordinary fine-tuning (Cheng et al., 2026). Run the horizon past the point where the A1 editors are known to degrade, at least 10,000 edits (Suresh and Hariharan, 2026).

The design needs no new method and no new benchmark. Its components are published methods and benchmarks with public code links: LocFT, EasyEdit, AlphaEdit, UltraEdit, MQuAKE, and the WILD protocol as described by Yang et al. (2025). The anchor's own shard experiment is a small instance of the first feature with $m = 0$ and $m = \text{all}$. Extending it to intermediate m , and to the specialised editors, would answer most of the question.

12. The strongest case against this paper

The strongest objection is that replay is not a confound but the method. On this view LocFT-BF is precisely the proposal to edit by retraining on the cumulative set, and the anchor's authors say so. Faulting the comparison for giving one arm replay is like faulting a comparison of a database with a cache for giving the database persistence. The objection has force. If the claim is "a practitioner who can afford to retrain on all past edits should use LocFT-BF rather than AlphaEdit", the anchor's evidence supports it well, across three models, three datasets, a medical benchmark, a mixture-of-experts model and scale to 72B. This paper does not dispute that claim. It disputes the stronger one in the anchor's title and conclusion, that the long-reported failure of fine-tuning is explained by its implementation rather than by anything inherent to it. If replay is doing the work, then

the depth-first baseline failed because it lacked replay, which is a protocol property. And any sequential editor given replay might recover similarly. The anchor did not test that.

A second objection is that the scoring argument cuts both ways. If the field's earlier numbers were inflated by teacher forcing, then the anchor's switch to WILD is a correction, and a ranking under the corrected scorer is the one that counts. This paper agrees that WILD is the better scorer. The point of Section 9 is narrower. Under WILD, depth-first FT-M was already the strongest of the six editors Yang et al. (2025) tested (FT-M, MEND, ROME, MEMIT, GRACE and WISE). None of them is AlphaEdit, RLEdit or UltraEdit, and whether depth-first FT-M leads those three under WILD has not been reported. So against the older editors the scorer, not the pipeline, carried part of what the anchor reports as a reversal; against the newer ones the record does not say. The pipeline's own contribution is what lies beyond that, for instance FT-M's rise from 75.30 to 99.70 percent on ZsRE with LLaMA in the anchor's controlled comparison, and that step coincides with replay. (The 40.5 percent in Yang et al. (2025) and the anchor's numbers come from different datasets and models and should not be subtracted from each other.)

A third objection is that the selection concern of Section 7 is minor. LocFT-BF's capability holds on datasets not used in the sweep, and its locations generalised across model sizes by a fixed rule. This is fair, and the paper says so in Section 7. The concern is about the margin in the capability column, not about whether LocFT-BF preserves capability. The latter is well supported.

13. What is not known

It is not known how LocFT performs in a strictly streaming protocol without replay, or with bounded replay. It is not known how MEMIT, AlphaEdit or UltraEdit perform when given the replay LocFT-BF receives. No study has compared fine-tuning and a specialised editor at the same retention level, with the same auxiliary data and under an autoregressive scorer. No study reports total compute at matched accuracy across retention regimes. It is not known whether LocFT-BF's edits propagate to entailed facts better or worse than other methods' edits, or whether they survive later fine-tuning. The disagreement between UltraEdit's 2-million-edit result and the anchor's report of UltraEdit degrading past 20,000 edits is unresolved.

14. Limitations

The axis codes in Tables 1 and 2 are the author's reading of each source's design. Where a source does not state a setting, such as the baselines' batch sizes in the anchor's comparison, the paper says so and does not infer one. The quadratic-cost argument of Section 5.1 is arithmetic on the anchor's description of its real-time variant, not a

measured cost. Several sources are recent and unrefereed preprints, notably the AlphaEdit reproduction (Suresh and Hariharan, 2026), and their results are reported as they state them. The retention reading makes predictions, set out in Section 5.4, that the published record cannot yet test. If those predictions fail, the paper's central attribution fails with them. The paper concerns factual editing of decoder language models on QA-style benchmarks. It does not address editing of behaviours, of multimodal models or of reasoning.

15. Conclusion

Fine-tuning did not fail at model editing, and it was not rescued from a misconception. One implementation of it was run under a protocol in which it kept nothing of earlier edits, was pushed to convergence on each one, trained on a single example, and was scored by a test that fed it the answer. Under that protocol it lost. Under protocols that let it see all edits together, augmented its data, merged its updates, or scored it by its own decoding, it tied or won, and it did so in 2020, 2023, 2024 and 2025 before it did so in 2026. The 2026 result is the most thorough of these, and its location sweep and scaling tests are useful. Its central attribution, though, moves on retention, selection and scoring along with the update rule, so it cannot assign the credit to the rule. The matched comparison has not been run. Until it is, "fine-tuning done right" means fine-tuning with replay, under a fair scorer, at a location chosen for the metric. Those are three facts about the protocol, and none of them is a fact about fine-tuning.

References

- Allen-Zhu, Z., & Li, Y. (2024). Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. ICML 2024. arXiv:2309.14316. doi:10.48550/arXiv.2309.14316.
- Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., & Evans, O. (2024). The Reversal Curse: LLMs trained on A is B fail to learn B is A. ICLR 2024. arXiv:2309.12288. doi:10.48550/arXiv.2309.12288.
- Biderman, D., Portes, J., Gonzalez Ortiz, J. J., Paul, M., Greengard, P., Jennings, C., et al. (2024). LoRA Learns Less and Forgets Less. Transactions on Machine Learning Research. arXiv:2405.09673. doi:10.48550/arXiv.2405.09673.
- Chen, Y., Cao, P., Chen, Y., Liu, K., & Zhao, J. (2025). Knowledge Localization: Mission Not Accomplished? Enter Query Localization! ICLR 2025. arXiv:2405.14117. doi:10.48550/arXiv.2405.14117.
- Cheng, Y., Youssef, P., Seifert, C., Schlotterer, J., & Zhao, Z. (2026). Can Fine-Tuning Erase Your Edits? On the Fragile Coexistence of Knowledge Editing and Adaptation. KDD 2026. arXiv:2511.05852. doi:10.48550/arXiv.2511.05852.
- Cohen, R., Biran, E., Yoran, O., Globerson, A., & Geva, M. (2024). Evaluating the Ripple Effects of Knowledge Editing in Language Models. Transactions of the Association for Computational Linguistics. arXiv:2307.12976. doi:10.48550/arXiv.2307.12976.
- Dacrema, M. F., Cremonesi, P., & Jannach, D. (2019). Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. RecSys 2019. arXiv:1907.06902. doi:10.48550/arXiv.1907.06902.
- De Cao, N., Aziz, W., & Titov, I. (2021). Editing Factual Knowledge in Language Models. EMNLP 2021. arXiv:2104.08164. doi:10.48550/arXiv.2104.08164.
- Fang, J., Jiang, H., Wang, K., Ma, Y., Jie, S., Wang, X., He, X., & Chua, T.-S. (2025). AlphaEdit: Null-Space Constrained Knowledge Editing for Language Models. ICLR 2025. arXiv:2410.02355. doi:10.48550/arXiv.2410.02355.
- Gangadhar, G., & Stratos, K. (2024). Model Editing by Standard Fine-Tuning. Findings of ACL 2024. arXiv:2402.11078. doi:10.48550/arXiv.2402.11078.
- Gekhman, Z., Yona, G., Aharoni, R., Eyal, M., Feder, A., Reichart, R., & Herzig, J. (2024). Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations? EMNLP 2024. arXiv:2405.05904. doi:10.48550/arXiv.2405.05904.
- Gu, J.-C., Xu, H.-X., Ma, J.-Y., Lu, P., Ling, Z.-H., Chang, K.-W., & Peng, N. (2024). Model Editing Harms General Abilities of Large Language Models: Regularization to the Rescue. EMNLP 2024. arXiv:2401.04700. doi:10.48550/arXiv.2401.04700.
- Gu, X., Huang, Z., Gu, J.-C., & Zhang, K. (2025). UltraEdit: Training-, Subject-, and Memory-Free Lifelong Editing in Language Models. Transactions on Machine Learning Research. arXiv:2505.14679. doi:10.48550/arXiv.2505.14679.
- Gupta, A., Rao, A., & Anumanchipalli, G. (2024a). Model Editing at Scale leads to Gradual and Catastrophic Forgetting. Findings of ACL 2024. arXiv:2401.07453. doi:10.48550/arXiv.2401.07453.
- Gupta, A., Baskaran, S., & Anumanchipalli, G. (2024b). Rebuilding ROME : Resolving Model Collapse during Sequential Model Editing. EMNLP 2024. arXiv:2403.07175. doi:10.48550/arXiv.2403.07175.
- Gupta, A., Prateepamornkul, P., Lu, M., Alaa, A., Hartvigsen, T., & Anumanchipalli, G. (2025). Lifelong Knowledge Editing requires Better Regularization. arXiv:2502.01636. doi:10.48550/arXiv.2502.01636.
- Hartvigsen, T., Sankaranarayanan, S., Palangi, H., Kim, Y., & Ghassemi, M. (2023). Aging with GRACE: Lifelong Model Editing with Discrete Key-Value Adaptors. NeurIPS 2023. arXiv:2211.11031. doi:10.48550/arXiv.2211.11031.
- Hase, P., Bansal, M., Kim, B., & Ghandeharioun, A. (2023). Does Localization Inform Editing? Surprising Differences in Causality-Based Localization vs. Knowledge Editing in Language Models. NeurIPS 2023. arXiv:2301.04213. doi:10.48550/arXiv.2301.04213.
- Hase, P., Hofweber, T., Zhou, X., Stengel-Eskin, E., & Bansal, M. (2024). Fundamental Problems With Model Editing: How Should Rational Belief Revision Work in LLMs? arXiv:2406.19354. doi:10.48550/arXiv.2406.19354.
- Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning that Matters. AAAI 2018. arXiv:1709.06560. doi:10.48550/arXiv.1709.06560.
- Hewitt, J., Chen, S., Xie, L. L., Adams, E., Liang, P., & Manning, C. D. (2024). Model Editing with Canonical Examples. arXiv:2402.06155. doi:10.48550/arXiv.2402.06155.
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. ICLR 2017. arXiv:1609.04836. doi:10.48550/arXiv.1609.04836.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. PNAS. arXiv:1612.00796. doi:10.48550/arXiv.1612.00796.
- Li, R., Wang, Y., Zhu, W., Li, C., Zhang, J., Li, C., Yan, J., & Wang, J. (2025b). EtCon: Edit-then-Consolidate for Reliable Knowledge Editing. arXiv:2512.04753. doi:10.48550/arXiv.2512.04753.
- Li, Z., Jiang, H., Chen, H., Bi, B., Zhou, Z., Sun, F., Fang, J., & Wang, X. (2025). Reinforced Lifelong Editing for Language Models. ICLR 2025. arXiv:2502.05759. doi:10.48550/arXiv.2502.05759.
- Li, Z., Zhang, N., Yao, Y., Wang, M., Chen, X., & Chen, H. (2024). Unveiling the Pitfalls of Knowledge Editing for Large Language Models. ICLR 2024. arXiv:2310.02129. doi:10.48550/arXiv.2310.02129.

- Ma, J.-Y., Wang, H., Xu, H.-X., Ling, Z.-H., & Gu, J.-C. (2025). Perturbation-Restrained Sequential Model Editing. ICLR 2025. arXiv:2405.16821. doi:10.48550/arXiv.2405.16821.
- Ma, X., Chen, W., Liu, Q., Xu, D., Zheng, Z., Xu, T., & Chen, E. (2026). More Edits, More Stable: Understanding the Lifelong Normalization in Sequential Model Editing. ICML 2026. arXiv:2605.11836. doi:10.48550/arXiv.2605.11836.
- Mecklenburg, N., Lin, Y., Li, X., Holstein, D., Nunes, L., Malvar, S., et al. (2024). Injecting New Knowledge into Large Language Models via Supervised Fine-Tuning. arXiv:2404.00213. doi:10.48550/arXiv.2404.00213.
- Melis, G., Dyer, C., & Blunsom, P. (2018). On the State of the Art of Evaluation in Neural Language Models. ICLR 2018. arXiv:1707.05589. doi:10.48550/arXiv.1707.05589.
- Meng, K., Bau, D., Andonian, A., & Belinkov, Y. (2022). Locating and Editing Factual Associations in GPT. NeurIPS 2022. arXiv:2202.05262. doi:10.48550/arXiv.2202.05262.
- Meng, K., Sharma, A. S., Andonian, A., Belinkov, Y., & Bau, D. (2023). Mass-Editing Memory in a Transformer. ICLR 2023. arXiv:2210.07229. doi:10.48550/arXiv.2210.07229.
- Mitchell, E., Lin, C., Bosselut, A., Finn, C., & Manning, C. D. (2022a). Fast Model Editing at Scale. ICLR 2022. arXiv:2110.11309. doi:10.48550/arXiv.2110.11309.
- Mitchell, E., Lin, C., Bosselut, A., Manning, C. D., & Finn, C. (2022b). Memory-Based Model Editing at Scale. ICML 2022. arXiv:2206.06520. doi:10.48550/arXiv.2206.06520.
- Musgrave, K., Belongie, S., & Lim, S.-N. (2020). A Metric Learning Reality Check. ECCV 2020. arXiv:2003.08505. doi:10.48550/arXiv.2003.08505.
- Ovadia, O., Brief, M., Mishaeli, M., & Elisha, O. (2024). Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs. arXiv:2312.05934. doi:10.48550/arXiv.2312.05934.
- Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., & Wayne, G. (2019). Experience Replay for Continual Learning. NeurIPS 2019. arXiv:1811.11682. doi:10.48550/arXiv.1811.11682.
- Suresh, A. K., & Hariharan, A. (2026). Reproducibility Study of "AlphaEdit: Null-Space Constrained Knowledge Editing for Language Models". arXiv:2606.26783. doi:10.48550/arXiv.2606.26783.
- Thede, L., Roth, K., Bethge, M., Akata, Z., & Hartvigsen, T. (2025). WikiBigEdit: Understanding the Limits of Lifelong Knowledge Editing in LLMs. ICML 2025. arXiv:2503.05683. doi:10.48550/arXiv.2503.05683.
- van de Ven, G. M., & Tolia, A. S. (2019). Three scenarios for continual learning. arXiv:1904.07734. doi:10.48550/arXiv.1904.07734.
- Wang, P., Zhang, N., Tian, B., Xi, Z., Yao, Y., Xu, Z., et al. (2024a). EasyEdit: An Easy-to-use Knowledge Editing Framework for Large Language Models. ACL 2024 System Demonstrations. arXiv:2308.07269. doi:10.48550/arXiv.2308.07269.
- Wang, P., Li, Z., Zhang, N., Xu, Z., Yao, Y., Jiang, Y., Xie, P., Huang, F., & Chen, H. (2024b). WISE: Rethinking the Knowledge Memory for Lifelong Model Editing of Large Language Models. NeurIPS 2024. arXiv:2405.14768. doi:10.48550/arXiv.2405.14768.
- Wang, S., Zhu, Y., Liu, H., Zheng, Z., Chen, C., & Li, J. (2024c). Knowledge Editing for Large Language Models: A Survey. ACM Computing Surveys. arXiv:2310.16218. doi:10.48550/arXiv.2310.16218.
- Yang, W., Sun, F., Ma, X., Liu, X., Yin, D., & Cheng, X. (2024a). The Butterfly Effect of Model Editing: Few Edits Can Trigger Large Language Models Collapse. Findings of ACL 2024. arXiv:2402.09656. doi:10.48550/arXiv.2402.09656.
- Yang, W., Sun, F., Tan, J., Ma, X., Su, D., Yin, D., & Shen, H. (2024b). Understanding the Collapse of LLMs in Model Editing. Findings of EMNLP 2024. arXiv:2406.11263. doi:10.48550/arXiv.2406.11263.
- Yang, W., Sun, F., Tan, J., Ma, X., Cao, Q., Yin, D., Shen, H., & Cheng, X. (2025). The Mirage of Model Editing: Revisiting Evaluation in the Wild. ACL 2025. arXiv:2502.11177. doi:10.48550/arXiv.2502.11177.
- Yang, W., Tang, R., Zang, H., Su, D., Cao, Q., Wang, J., Shen, H., Cheng, X., & Sun, F. (2026). Fine-tuning Done Right in Model Editing. ICLR 2026. arXiv:2509.22072. doi:10.48550/arXiv.2509.22072.
- Yao, Y., Wang, P., Tian, B., Cheng, S., Li, Z., Deng, S., Chen, H., & Zhang, N. (2023). Editing Large Language Models: Problems, Methods, and Opportunities. EMNLP 2023. arXiv:2305.13172. doi:10.48550/arXiv.2305.13172.
- Zhang, C., Zhang, M., Ye, X., Cheng, R., Zhou, Z., Zhou, Y., Ren, P., & Chen, Z. (2026). Spectral Characterization and Mitigation of Sequential Knowledge Editing Collapse. ACL 2026. arXiv:2601.11042. doi:10.48550/arXiv.2601.11042.
- Zhang, M., Ye, X., Liu, Q., Ren, P., Wu, S., & Chen, Z. (2025). Uncovering Overfitting in Large Language Model Editing. ICLR 2025. arXiv:2410.07819. doi:10.48550/arXiv.2410.07819.
- Zhang, N., Yao, Y., Tian, B., Wang, P., Deng, S., Wang, M., et al. (2024). A Comprehensive Study of Knowledge Editing for Large Language Models. arXiv:2401.01286. doi:10.48550/arXiv.2401.01286.
- Zheng, C., Li, L., Dong, Q., Fan, Y., Wu, Z., Xu, J., & Chang, B. (2023). Can We Edit Factual Knowledge by In-Context Learning? EMNLP 2023. arXiv:2305.12740. doi:10.48550/arXiv.2305.12740.
- Zhong, Z., Wu, Z., Manning, C. D., Potts, C., & Chen, D. (2023). MQuAKE: Assessing Knowledge Editing in Language Models via Multi-Hop Questions. EMNLP 2023. arXiv:2305.14795. doi:10.48550/arXiv.2305.14795.
- Zhu, C., Rawat, A. S., Zaheer, M., Bhojanapalli, S., Li, D., Yu, F., & Kumar, S. (2020). Modifying Memories in Transformer Models. arXiv:2012.00363. doi:10.48550/arXiv.2012.00363.