# CAN OLD DOGS LEARN NEW TRICKS? CAPABILITY DYNAMICS ALONG PRETRAINING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

When teams continue training a language model, they have to choose which data to train on, which checkpoint to start from, and how much old data to mix back in. These choices are usually guided by the training loss and by benchmark scores based on answer likelihood. We test whether these signals lead to the right choices. We take public OLMo-2 checkpoints and continue each one several times. Each run changes a single setting and keeps all training batches identical. We judge the model by the answers it writes. Likelihood scores can pick the wrong data. Training a 1B model on math data raises its accuracy on word problems by 47 percentage points while code data does not, yet a likelihood score computed on the final answer alone, with no written reasoning, favors code. The starting checkpoint also matters. At 7B, with our fine-tuning recipe, checkpoints from before mid-training learn a new task far more slowly than later ones. Finally, runs that look the same on loss and benchmark averages differ in which questions they get right. On GSM8K the difference is mostly a missing stop token.

## 1 INTRODUCTION

Watch OLMo-2-1B learn word problems that take two to four steps (Figure 1). For four trillion tokens of stage 1, its held-out loss falls steadily and its answers stay wrong. It solves at most 0.050 of the problems. In the model's own reasoning the arithmetic becomes correct, from 0.34 of its statements true early in the run to 0.92 at the end. Then, within the first 3B tokens of mid-training, the answers start to come out right, and accuracy rises from 0.042 to 0.252. Over those same tokens the web loss goes up, and the accuracy of a likelihood score on the same problems does not change significantly. Neither the loss nor a likelihood score shows these changes.

Every pretraining decision a team makes relates data with a specific checkpoint, and teams make these decisions from the loss and a benchmark panel (OLMo Team, 2025; Grattafiori et al., 2024; Magnusson et al., 2025). Along OLMo-2-1B, grammar forms in the first 21B tokens, and multi-step math forms only in mid-training (Appendix E, Figure 11a). A decision made from the loss is therefore partly blind to the capability it is about. We measure capabilities along the run from generated answers and count, in tokens, what data and checkpoint choices cost (Table 1).

Measuring the effect of a choice requires two training histories, and a public run releases one. Cheap scores computed on every checkpoint can miss the capability (Figure 1c). Benchmark scores also mix a capability with the format of its answer, and the format can change while the capability does not (Section 3).

Runs that start from one checkpoint and share every training batch differ only in the variable we change. We fork OLMo-2 checkpoints at 1B and 7B in this way (Section 2). We score each capability per example from generated answers. We repeat the scoring comparison on Open-1B (Donaghy et al., 2026), a model trained by another group. We describe four important settings:

- **Data selection (Section 3).** Across ten 1B mid-training checkpoints, generated accuracy on multi-step word problems rises from 0.042 to 0.279 while likelihood scoring does not change significantly. When we anneal a 1B checkpoint, math data raises generated one-step accuracy by +0.465, and code data leaves it unchanged (+0.000). A likelihood score read without written reasoning still prefers code (one run per source). SVAMP agrees with the generated answers.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

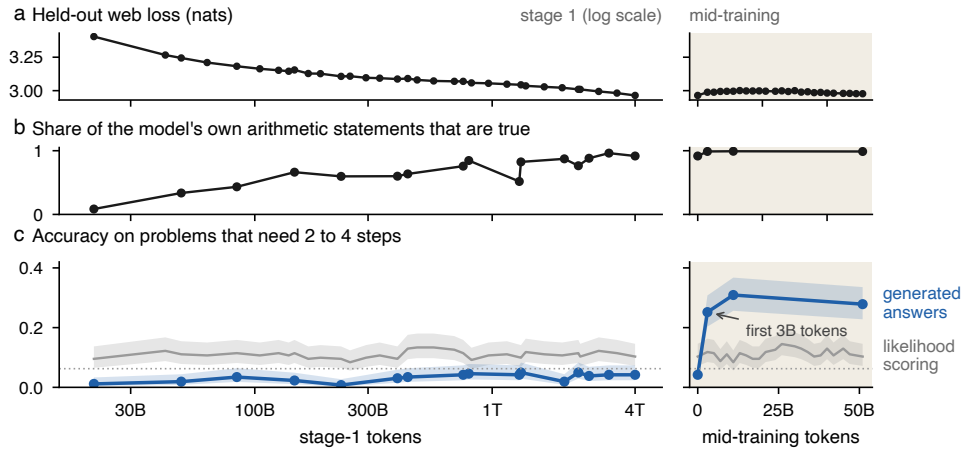



Figure 1: **Loss, arithmetic and accuracy along OLMo-2-1B.** OLMo-2-1B through stage 1 (left, log scale) and the first 51B tokens of mid-training (right, shaded; the first of three mid-training mixes). (a) Held-out web loss. (b) In the model's generated reasoning on word problems that need two to four steps, the share of its own arithmetic statements that are true. (c) Accuracy on the same 262 problems, from generated answers (blue) and by likelihood scoring among 16 candidate answers (grey), with Wilson 95% bands; dotted line: chance for likelihood scoring.

- **Checkpoint choice (Section 4).** At one 7B recipe and learning rate, no stage-1 checkpoint reaches 0.3 on a new task within 50M fine-tuning tokens. Mid-training checkpoints reach it in 23.9M to 41.7M. At 1B, a larger learning rate closes most of that gap. There, the stage mainly determines how small a rate still works. The fine-tuning loss after the first 2M to 5M tokens predicts the level at 50M across 20 runs from four models (Spearman $-0.858$). Within one 7B model it orders stage-1 checkpoints but not mid-training checkpoints.
- **Replay and optimizer state (Section 5).** On OLMo-2-7B, we replay one percent of a capability's own data, spread uniformly. At our learning rate, on capabilities fine-tuned to a moderate level, this leaves lower minima than no replay on 13 of 13 non-tied parents. The final level is not resolved. The sign reverses on a fully formed Qwen2.5-7B parent. A fork that changes $\beta_2$ on inherited Adam state takes steps 1.77 times smaller. Retention studies built on forks must control for this.
- **Per-item evaluation (Section 6).** Across ten 7B forks, training loss and seven benchmarks do not separate the conditions. Per-item scores identify the condition of every run and the stage of 11 of 11 public OLMo-2-7B revisions. On GSM8K the change is a lost stop token. A rank-eight change in the early residual stream reproduces it, and one update to layers 0 to 7 largely restores it.

We propose that capabilities form in a model's reasoning before they appear in its answers. The model's own reasoning is therefore a better readout than a likelihood score or a benchmark average. Capability acquisition, forgetting, performance and replay across pretraining thus become a new axis to optimize over.

## 2 Methods and Experimental Setup

**Forked models.** We take a released checkpoint, the parent, and continue it several times. Each continuation is a fork. Forks of one parent share every training batch and differ in one thing: the checkpoint, the replay fraction, its placement, its format, or the optimizer's $\beta_2$. OLMo-2 releases checkpoints from stage 1 and from the mid-training stage, in which the learning rate decays to zero on a mix dense in problems and worked solutions (OLMo Team, 2025). At 7B we continue each checkpoint for 25M to 200M tokens on a background task. For most runs this is a different question about the same synthetic programs. The GSM8K runs and one replication use C4 web text instead. The learning rate is $2 \times 10^{-5}$, with 128 sequences of 1,024 tokens per update. Weights, gradients

| decision | what the forks show | scale, sample |
|---|---|---|
| which data to anneal on (§3) | math data raises generated one-step accuracy by +0.465 and code data leaves it unchanged (+0.000), yet a likelihood score without written reasoning prefers code; SVAMP agrees (+0.080 against +0.007) | 1B, 262M tokens; one run per source |
| which checkpoint to fine-tune (§4) | at one recipe none of 5 stage-1 checkpoints reaches 0.3 within 50M tokens; the first mid-training checkpoint needs 41.7M, later ones 23.9M to 26.1M | 7B; one recipe, one learning rate |
| whether to pay for the rest (§4) | the fine-tuning loss after 2M to 5M tokens ranks the level at 50M (Spearman $-0.858$) | four models, 20 runs; not among mid-training checkpoints |
| how much replay, and where (§5) | one percent uniform replay: lower minima than none on 13 of 13 non-tied parents, final level unresolved; where replay falls predicts retention (Spearman 0.365), how much does not ($-0.097$) | 7B OLMo-2; 14 parents |
| which optimizer setting (§5) | a $\beta_2$ change on inherited Adam state shrinks each step 1.77-fold; without replay 0.999 kept 3 of 3 and 0.95 lost 3 of 3 over 200M tokens, at twice the rate both lost, and 0.95 at the matched rate kept it (one run each); in runs with replay the step alone gives 0.262 to 0.293, the second moment alone 0.164 to 0.184; no separation on GSM8K | 7B synthetic task; 3 runs each |
| whether a run has drifted (§6) | training loss and seven benchmarks do not separate the runs; the per-item pattern identifies each run, and 100 TriviaQA items suffice, at about 0.02 percent of the compute of a 25M-token run | 7B OLMo-2; 10 GSM8K runs |

Table 1: **Decisions and what the forks show.** Rows follow the order of the sections. Every value is computed from a stored run record (Appendix I).

and Adam (Kingma & Ba, 2015) states are held in bfloat16 (Kalamkar et al., 2019). The effective step is the size of each update after Adam's normalisation. At 1B we read 107 public checkpoints of OLMo-2-1B, 34 from stage 1 and 73 from the mid-training stage. We also run controlled cooldowns of stage-1 checkpoints. These separate the learning-rate decay from the change of data. Qwen2.5 (Qwen Team, 2024) and Mistral (Jiang et al., 2023) add other families, and OLMo-2-13B a second scale.

**Tasks and scoring.** The synthetic task shows a short generated Python program. It asks for the final value of one variable, as a one-key JSON object such as {`"batch": -197`} (a full item is in Appendix D). It has a difficulty setting, and no public checkpoint solves it before fine-tuning (greedy accuracy 0 and pass@64 at most 0.012 across twelve models). Its level is the exact-match accuracy on held-out items. GSM8K (Cobbe et al., 2021) is scored on an answer-only probe and five-shot, with GSM-Plus (Li et al., 2024) and SVAMP (Patel et al., 2021) as controls. MathGAP, code and closure problems are scored by likelihood and from generated answers. For a generated answer, the model writes its reasoning after three worked examples, and we read the number on the answer line, the line where it states its final answer. A pair of checkpoints is decisive when their generated accuracies differ with a paired 95 percent interval that excludes zero. Closure problems ask for a relation implied by a small stated graph or family tree, such as reachability or kinship, at three levels of composition. Every result is red per example, never from the training loss (sources of each readout: Appendix D).

## 3 Likelihood scores versus generated answers

Reading a capability from generated answers is slow, because the model has to write out its reasoning. A cheaper score ranks candidate answers by their likelihood and generates nothing. This is the form of the proxies used to compare data recipes at small scale (Magnusson et al., 2025; Gu et al., 2025). If the likelihood score tracked the capability, it would rise with generated accuracy along a run and rank checkpoints and data the same way. On multi-step word problems it does not. The two readouts differ in two ways. One is the scoring rule, and the other is whether the model writes out its reasoning. Scoring after reasoning, below, separates the two. The failure comes from the missing reasoning, not from likelihood scoring itself. This matters because the proxies used for data selection score the final answer alone.

**Items and scoring.** We generate word problems with MathGAP (Opedal et al., 2025), which fixes how many reasoning steps an answer needs, and drop the 85 items whose template appears in the OLMo-2 mid-training mix. This leaves 86 one-step items and 262 items that need two to four steps, such as this two-step item: "Mia has 10 couches. Then Mia buys 7 more couches from Ava. Then Mia donates 5 couches to John. How many couches does Mia have?" (answer 12). Likelihood

scoring ranks 16 candidate final answers by the summed log-probability of each, written as the JSON value in {"answer": $c$}, directly after the question, with no written reasoning. Chance is $1/16$ (distractors in Appendix A). Every likelihood result below is about this form. Generated answers are read from the answer line (Section 2).

**Along a run.** The likelihood score stays flat while generated accuracy rises. At the last stage-1 checkpoint of OLMo-2-1B, likelihood-scored accuracy on the multi-step items is 0.103, or 0.043 above chance after correcting for the 16 candidates. Through the mid-training stage, it changes significantly at none of the 10 checkpoints we read. Its largest change is 0.038. Over the same checkpoints, generated accuracy rises at every one, from 0.042 to 0.279 (Figure 1c). The failure is specific to multi-step word problems. On two other item types, likelihood accuracy is above generated accuracy at all 14 checkpoints of the 1B data comparison below (Appendix H). These are relational items, where generated answers often lack the answer line, and code items that need two or three steps. The same split recurs at a larger scale and in another group's model. Through OLMo-2-7B stage 1, from 110B to 1,443B tokens, likelihood scoring stays between 0.130 and 0.160 while generated accuracy climbs from 0.034 to 0.294 at 915B (Figure 2a). Open-1B was trained by another group on other data. There, 93B tokens of mid-training raise generated one-step accuracy from 0.058 to 0.535. Likelihood scoring stays at 0.058 and 0.035, both consistent with chance (3 of 86 against $1/16$, one-sided binomial $p = 0.21$; Figure 2b). In OLMo-2-1B, likelihood scoring does rise on one-step items (Appendix H).

**Checkpoint ranking.** The two scores rank checkpoints differently. We compare them on decisive pairs of OLMo-2-1B checkpoints (Section 2). On the simplest code problems and on one-step word problems, likelihood scoring orders the pairs the way generation does (0.93 and 0.89 of pairs). On two-step and three-step problems, its agreement (0.34 and 0.50) is not distinguishable from chance (Figure 2c; pair counts and intervals in Appendix A).

**Data selection.** They also pick different training data. We annealed the 4,001B OLMo-2-1B checkpoint for 262M tokens, once on web text alone and once on each of four sources (math, code, instruction and knowledge data). Each source was mixed half and half with web text, with one run per source. Against the web-only run, math data raises generated one-step accuracy by +0.465 (+0.523 on a second data draw) and leaves the likelihood score unchanged ($-0.035$). Code data does the reverse, raising the likelihood score by +0.093 and leaving generated accuracy unchanged (+0.000; Figure 2d). Chosen by this likelihood score, code data would be picked to improve math, and none of the 10 likelihood metrics of Magnusson et al. (2025) picks math (Appendix F). SVAMP, a separate problem set, agrees with the generated answers, with gains of +0.080 from math data and +0.007 from code data. At four times the budget, 1.05B tokens, math still wins on SVAMP (+0.160 against $-0.010$). The 65.5M generated choice picks this winner and the 65.5M likelihood choice, by point estimate, does not. Instruction data lowers generated accuracy only because it removes most answer lines. Math data also forms the multi-step skill (0.210 against 0.031 on two to four steps; intervals and the other data sources in Appendix A). The intervals here are over evaluation items. Each source is one training run, so they do not include run-to-run variation. Second draws of the same data moved gains by 0.06 (math, 262M) and 0.12 (cooldown, 4,001B). That is far below the math gain of +0.465, but comparable to the 0.13 by which the likelihood score prefers code over math (Table 6). Generation picks math by a wide margin, and the likelihood score does not pick math. Whether it strictly prefers code is within run noise. The disagreement is not specific to one target. Across 10 targets, the two readouts pick different sources on 5, with the same sign in both the 65.5M and 262M anneals, which are separate runs. On three MathGAP targets the likelihood score picks code and generation picks math. On a code target and a closure target it picks math and generation picks code (Appendix A). Only one of the five has intervals excluding zero for both readouts at both budgets, so this is a consistent sign, not five resolved effects.

**JSON formatting.** One reading is that code data teaches the JSON wrapper. The opening {"answer": is part of the prompt and is not scored, so each candidate's score covers only its number and the closing brace. A gain on the wrapper that all candidates share cannot change which candidate wins. It also cannot change the correct candidate's share of the probability. Both of those metrics still prefer code (Table 6). Scored after the model's own reasoning, as bare numbers with no wrapper, the same candidates prefer math (below). That read also adds the reasoning, so rescoring

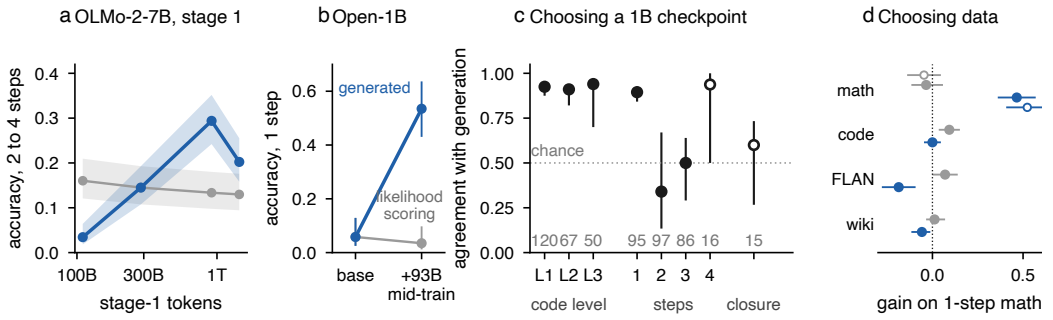



Figure 2: **Likelihood scores and generated answers.** (a) OLMo-2-7B stage 1, MathGAP items that need two to four steps ($n = 262$): likelihood scoring (grey) and generated answers (blue), Wilson 95% bands. (b) Open-1B before and after 93B mid-training tokens, one-step items ($n = 86$). (c) Share of decisive pairs of OLMo-2-1B checkpoints that likelihood scoring orders the same way as generated accuracy, by problem type; bars are item-bootstrap 95% intervals, grey numbers count decisive pairs, hollow markers have fewer than 30, and the dotted line is chance. (d) One-step MathGAP gain of 262M-token anneals of the 4,001B OLMo-2-1B checkpoint over a web-only anneal, by likelihood score (grey) and generated answers (blue); paired bootstrap 95% intervals, one run per source (hollow: second math draw); FLAN is instruction data, wiki knowledge data, and FLAN loses the answer line.

without the wrapper remains the direct test. The reversal on code and closure targets, where the likelihood score prefers math, is also hard to reconcile with a general wrapper gain from code data.

**Scoring after reasoning.** Scored after the model's own written reasoning, the same likelihood score tracks generation. Over the ten mid-training checkpoints its accuracy on two to four steps correlates with generated accuracy (Spearman 0.70), and it prefers math data over code by +0.430 (generation: +0.465; Appendix A). The failure is specific to scoring without written reasoning.

**Answer format.** Many scorers require the answer in a particular format, such as an answer line, a stop token after the number or a JSON wrapper. A model can lose the format while keeping the skill. Table 2 (Appendix A) lists settings where we scored one model twice on the same items, with and without a format requirement. The two scores differ in size in every row and in sign in two rows. Every result below is therefore read under the scoring that does not require the format.

## 4 Learning a new capability

We use generated answers to date when capabilities form along a run and to measure what a new capability costs at each checkpoint. Along OLMo-2-1B stage 1 the web loss falls smoothly (Figure 1a), yet capabilities form at different times (Figure 11a). Grammar reaches half its stage-1 gain by 21B tokens and code output by 1,322B. Generated accuracy on word problems does not rise through 4T tokens, and the multi-step skill appears only in the first few billion tokens of mid-training (Figure 1c).

**Arithmetic and composition.** The model gets the arithmetic steps right during stage 1 and combines them correctly only in mid-training. We check the arithmetic statements ("$a$ op $b = c$") that the 1B model writes on the multi-step items. The share that are true rises from 0.34 at 49B tokens to 0.92 at 4,001B. Over the same span, accuracy stays at 0.019 to 0.042 (Figure 1b). By the end of stage 1, most wrong answers contain only true arithmetic. The error lies in which quantities are combined (Figure 11c). This resembles the compositionality gap of Press et al. (2023), where a model gets each calculation right and combines the wrong quantities. Given "Lucy has 5 towels. Then Lucy sells 3 towels to John. Mila has 9 more towels than Lucy. How many towels does Mila have?" (answer 11), the last stage-1 checkpoint writes `5 + 3 = 8, 8 + 9 = 17`. Both statements are true, and the first adds the towels Lucy sold. We also apply an onset rule calibrated on simulated

flat curves. It finds no onset of generated multi-step accuracy in stage 1. It finds one at the first mid-training checkpoint we read (3B tokens) in each of the 3 mixes. Likelihood scoring has none in any of the 4 series (Appendix C, which also checks the arithmetic against a judge of step validity). The answer line is in place by 84B tokens (0.33 at 21B, 0.99 at 84B), long before the arithmetic is reliable.

**Step difficulty.** Mid-training accuracy falls faster with the number of steps than a constant per-step success rate $p^k$ predicts (Schaeffer et al., 2023; Dziri et al., 2023) (Figure 11b; model comparison in Appendix C), so steps are not all alike. Synthetic word problems of one to six steps separate the length of a chain from what each step does. At 7B, when each step adds a stated number to the running result, accuracy stays at 0.95 or above through six steps. When each step instead combines the two previous quantities, accuracy falls from 0.94 to 0.57 at the second step and to 0.11 at six. Length alone does not make a problem hard. The combining chains already fail at the second step. That step, like each later stated-number step, uses only one computed result. So the comparison does not isolate which feature of the step causes the drop (more in Appendix C).

**Learning-rate decay.** Mid-training changes two things at once. The learning rate decays to zero, and the data turns dense in problems and worked solutions. We separate them with three 65.5M-token cooldowns of one stage-1 checkpoint. The first decays the rate on stage-1 data. The second uses mid-training data at a constant rate. The third uses mid-training data with the decay. At the 4,001B checkpoint, the mid-training data adds 0.430 to one-step accuracy over the decay on stage-1 data, which is above the noise band. The decay adds 0.000 over the constant rate, which is inside it. The constant-rate run has a higher mean learning rate, so this contrast does not isolate decay at a matched rate. The same holds earlier in the run. We ran the cooldown at 7 checkpoints from 49B to 4,001B tokens. Read by the last number, it adds +0.20 to +0.44 to one-step accuracy at each one. All are above noise in single runs (4 with the answer line required; Figure 3a). At 4,001B a second draw of the same data alone moves the gain from 0.442 to 0.326, so smaller differences between checkpoints are not resolved. On the multi-step items, which were the primary readout fixed before these runs, the outcome is mixed (Appendix A). What these 65.5M-token cooldowns form is the one-step skill. The tokens a skill needs grow with its number of steps.

**Fine-tuning cost at 7B.** At 7B, the number of tokens a new task costs depends on the stage of the checkpoint. We fine-tune public OLMo-2-7B checkpoints on the synthetic task with one recipe and one learning rate ($2 \times 10^{-5}$) for 50M tokens (Figure 3b). None of the 5 stage-1 checkpoints, from 412B to 3,532B tokens, reaches 0.3. A larger rate does not close the gap. At 412B, $5 \times 10^{-5}$ reaches at most 0.14 in three runs and $10^{-4}$ reaches 0.04. The last stage-1 checkpoint reaches 0.2 after 47.7M tokens. The first mid-training checkpoint, 5B tokens into mid-training, reaches 0.3 after 41.7M tokens. Checkpoints after 50B mid-training tokens reach it after 23.9M to 26.1M on each of the three mixes.

**Stage-1 checkpoints.** Three factors, each measured in a different setting, explain part of the stage difference, and none accounts for it alone (Appendix D). The first is the step size. At 1B one rate, $4 \times 10^{-5}$, is best for all four stage-1 checkpoints and brings each to between 0.785 and 0.883. Smaller rates separate them (Figure 3c). At $5 \times 10^{-6}$, the 210B checkpoint reaches 0.023 and the 4,001B checkpoint 0.695 (one seed per rate). The second is the answer template: 25M tokens in the fine-tune's template close 0.36 of the gap, about half of it by the extra training alone. The third is a head start. On GSM8K the released checkpoint keeps a lead of 0.248 at the best of three learning rates at 7B and of 0.212 (0.164 to 0.262) at 13B.

**Early fine-tuning loss.** The fine-tuning loss at the first evaluation, 2M to 5M tokens in, predicts the level at 50M across 20 runs from four models and two targets (Spearman $-0.858$; Figure 3d). Much of that is between models. Within OLMo-2-7B it holds across stage-1 runs and not across mid-training runs, whose levels sit close together. The loss of the untouched checkpoint orders stage-1 checkpoints backwards (Appendix A). Once a capability has formed, the next question is what keeps it.

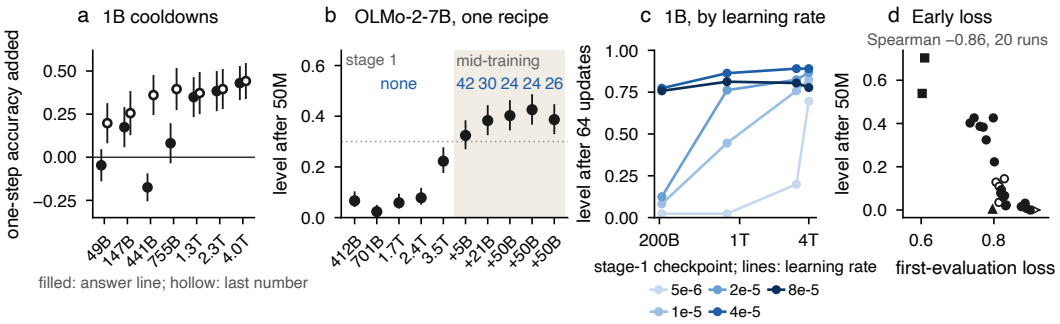



Figure 3: **Cost of a new capability by checkpoint and learning rate.** (a) OLMo-2-1B cooldowns of 65.5M tokens: one-step accuracy that mid-training data adds over a decay on stage-1 data, at seven parent checkpoints, under the answer-line rule (filled) and by last number (hollow); bars are the noise half-widths; single runs. (b) OLMo-2-7B, one fine-tuning recipe: held-out level on the synthetic task after 50M fine-tuning tokens, by checkpoint (stage 1 white, mid-training shaded; the three +50B points are the three mid-training mixes); Wilson 95% intervals over 256 items; blue numbers are the fine-tuning tokens (M) to reach 0.3, the dotted line, and *none* means no stage-1 checkpoint reaches it within 50M. (c) OLMo-2-1B: level after 64 fine-tuning updates by stage-1 checkpoint, one line per learning rate; one seed each. (d) Fine-tuning loss at the first evaluation against the level at 50M, OLMo-2 (circles), Qwen2.5 (squares), Mistral (triangles); rate controls hollow; one Mistral control lies off the axis (arrow).

## 5 Retention and Replay

A 7B model learns a capability, and training then continues on other data. The other data is a second synthetic task after the synthetic task, or C4 web text after GSM8K. The level of the capability falls. The usual remedy is to mix a few percent of the old data back in (Ibrahim et al., 2024; Gupta et al., 2023; Scialom et al., 2022). From formed 7B parents we vary the amount, placement and content of replay, and then the optimizer, with every other batch held fixed (Figure 4).

**Early loss.** Without replay, a capability formed over 50M tokens loses 0.67 of its level in the first 5M tokens of the background task. The rest decays with a time constant of about 132M tokens. One formed over 200M tokens loses 0.33 in the same 5M (Figure 4a; intervals in Appendix D). Replay that starts after a few million tokens therefore comes after most of the loss.

**One percent replay.** We compare one percent uniform replay with no replay, parent by parent, at our learning rate and on capabilities fine-tuned to a moderate level. One percent leaves a lower minimum on 13 of 13 non-tied OLMo-2-7B parents (sign test $p = 1.2 \times 10^{-4}$). It leaves a lower mean level on 12 of 14. The final level does not resolve. Five percent keeps the capability, so retention is not monotone in the dose. The sign reverses on a fully formed Qwen2.5-7B parent and on the one OLMo-2-13B parent (Appendix D). Over 284 forks, where replay falls predicts the minimum on configurations left out of the fit (Spearman 0.365) and the dose does not ($-0.097$). The damage also needs the capability's own data. The same amount of other data lands in the no-replay band (Appendix D).

**Adam $\beta_2$.** A fork that changes $\beta_2$ also changes its step size. Our forks inherit the parent's Adam state, including its step counter ($t = 382$), accumulated at $\beta_2 = 0.95$. A fork run at 0.999 applies that value's bias correction to it, so its effective step is 1.77 times smaller. Without replay, at our learning rate, 0.999 keeps the capability over 200M tokens in all three runs and 0.95 loses it in all three (Figure 4c). The step accounts for most of this. At the matched, smaller learning rate, 0.95 also keeps the capability. At twice the rate, both settings lose it. In runs with replay, the step alone recovers more of the minimum than the second moment alone (one to four runs per setting; Appendix D). The number format may account for part of the rest. Our trainer keeps Adam's second

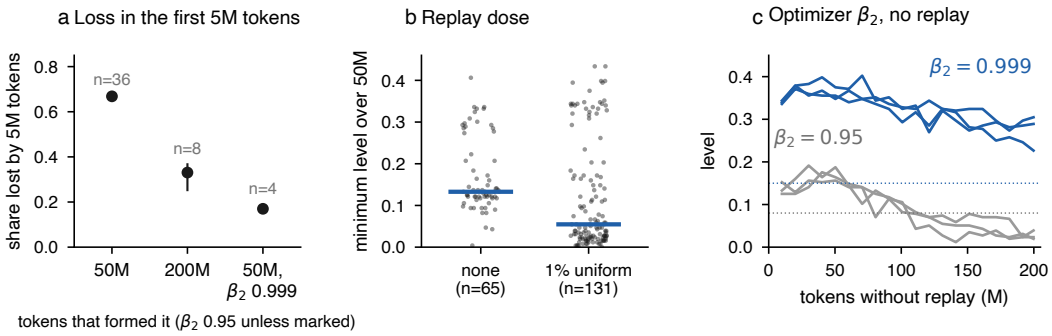



Figure 4: **Losing and keeping a capability at 7B.** OLMo-2-7B. (a) Share of the formed level lost in the first 5M tokens without replay, by formation budget and $\beta_2$; bars are fork-bootstrap 95% intervals. (b) Minimum of the synthetic score over a 50M-token window without replay and with one percent uniform replay, every distinct fork we ran, across learning rates and families (the parent-level test in the text uses 56 and 43 OLMo-2-7B forks); blue bars are medians over forks, which share batches within a parent (the parent-level test is in the text). (c) 200M tokens of a synthetic background task without replay at $\beta_2$ 0.95 (grey) and 0.999 (blue), three runs each; dotted lines are the kept and lost thresholds.

moment in bfloat16, where multiplying by 0.999 leaves every value unchanged. At 0.999 the second moment can therefore rise but not decay (Appendix D). We verified this rounding on CPU only and have not tested whether it drives retention. In a fork, a $\beta_2$ change on inherited state is therefore also a learning-rate change, which affects forgetting (Ibrahim et al., 2024). Section 6 asks whether standard training metrics can tell such forks apart.

## 6 Per-item evaluation

We use the ten OLMo-2-7B GSM8K runs of Section 5, which differ only in replay. Four have no replay, four have one percent uniform replay, and two have replay in one block at the end. Training loss and seven accuracy benchmarks do not separate them. Their answers on the same items do, and on GSM8K the change is a missing end-of-sequence token. We read the runs at five points up to 200M tokens, and each null is exact, over every relabelling of the runs. The capability probe itself separates the conditions, and held-out language-model loss does too, but it moves by only 0.0015 nats (statistics in Appendix D).

**Per-item scores.** The pattern of which items a model gets wrong, which an average discards, separates the runs. A nearest-centroid decoder on per-item correctness identifies every held-out run. The per-item log-likelihood of the gold answer identifies 10 of 10 runs on GSM8K and 10 of 10 on TriviaQA. Chance, the mean over every relabelling of the runs, is 3.1 of 10. One hundred random TriviaQA items suffice. Each test sits at the floor of its exact permutation null. Arithmetic tokens alone still identify every run, so the pattern is not in the answer format. Three benchmarks with no shared items order the runs alike, so the pattern is a property of the checkpoint. Applied to the eleven public OLMo-2-7B revisions, it identifies the stage of 11 of 11 (Figure 12; statistics in Appendix D).

**The end-of-sequence token.** On GSM8K, what the exact-match score records as forgetting is the loss of the end-of-sequence token after a correct answer. A no-replay fork at 150M tokens reads 0.020 with that token required, while the right number stays on 0.113 of items against the parent's 0.133. On one held-out item whose answer is 4, the parent writes `4` and then the end-of-sequence token. The fork writes `4 logs.` and then starts a new problem (`Problem: A 3.5 foot tall person is standing on a 2 foot tall box.`). If the parent receives a no-replay fork's residual stream after layer 8, it stops emitting the token (Meng et al., 2022). A single-layer sweep on the first fork placed the change in layers 3 to 7, over the question tokens. The depth profile varied across forks, and it did not replicate on two of three (Appendix D). A rank-eight reconstruction of the fork's change reproduces the full effect where random subspaces do not, and

the effect holds on a disjoint item set (Appendix D, Figure 6). The parent fed a fork's residual stream also fails on the fork's own items. The Spearman correlation is 0.83 to 0.87 against a permutation null whose 95th percentile is 0.10 (three forks, 256 items). One update on target data restricted to layers 0 to 7 restores the exact-match score from 0.016–0.020 to 0.113–0.125, while eight updates restricted to layers 24 to 31 reach 0.031–0.059. These interventions locate the change in this one behavior.

## 7 Related work

One line of work compares proxy scores with downstream accuracy. Loss and downstream accuracy diverge (Du et al., 2024; Schaeffer et al., 2023), and likelihood and generated answers disagree at a single checkpoint (Wang et al., 2024; Lyu et al., 2024). We measure that disagreement along a run and on the data choice that small-scale likelihood scores inform (Magnusson et al., 2025; Gu et al., 2025). Downstream metrics have been compared across checkpoints of runs up to 67B parameters (Yang et al., 2024). Our forks share every batch, so curves differ by one variable.

A second line studies what each training stage adds. Mid-training data raises benchmark scores, most for math and code (OLMo Team, 2025; Blakeney et al., 2024; Hu et al., 2024; Liu et al., 2025). Heavily pretrained or decayed checkpoints can fine-tune worse (Springer et al., 2025; Yano et al., 2026). Larger weight decay in pretraining raises the gain from fine-tuning (Han et al., 2026). Fine-tuning OLMo checkpoints reveals gains that the base model hides (Sun & Dredze, 2024). Fine-tuning updates transfer within the two stages of OLMo 2 but not across them (Lin et al., 2025). We count the tokens each stage needs to form a capability and separate the stage from the learning rate. Those studies fine-tune toward capabilities the model already has, where in our runs a later stage raises the starting level and leaves the rate of gain unchanged (Appendix D). Our 7B target is new to every checkpoint. Across released models, benchmark capability tracks one latent axis of compute (Zhang et al., 2026), and we follow capabilities along one run instead. A third line studies forgetting and replay. Replay doses are usually chosen on loss (Ibrahim et al., 2024; Gupta et al., 2023). Uniform replay of old-domain data can keep less than no replay at a 20 percent budget (Atreya et al., 2026). Forgetting has been timed in the magnitude of optimizer updates (Feng et al., 2026), and Adam has been studied as a source of forgetting (Ashley et al., 2021). We vary where replay falls and split the $\beta_2$ effect (Appendix B).

## 8 Discussion and limitations

**Practical rules.** We recommend the following practices, each scoped in Table 1. (1) Choose data for a checkpoint from generated answers, not from a likelihood score read without written reasoning. (2) As a cheap screen before committing a post-training budget, fine-tune for 2M–5M tokens and read the loss. In our runs this ranked models and stage-1 checkpoints (Spearman $-0.89$ within OLMo-2-7B stage 1) but not mid-training checkpoints ($-0.20$), whose final levels were close together. (3) Near a capability's margin, replay early, and when a fork changes the optimizer, hold its effective step fixed. (4) Track per-item scores on about a hundred items rather than a benchmark average. At our scale this costs about 0.02 percent of the compute of a 25M-token run.

**Limitations.** Our models span 1B to 13B parameters: 1B for readout and data choice, 7B for formation cost, retention and repair, and OLMo-2-13B in two contrasts. Apart from one Qwen2.5-7B contrast, which reverses the one-percent result, every 7B retention result is on OLMo-2 under one trainer. That trainer keeps Adam states in bfloat16 and passes the step counter of the parent to each fork. Part of the $\beta_2$ result may depend on that setup. All but the GSM8K runs use a fine-tuned synthetic capability. Eleven of fourteen OLMo-2-7B parents have one fork per condition, and each 1B data source is one run. The one-percent deficit holds on the minimum and mean. The final level does not resolve. We did not run the one-percent comparison at a small matched learning rate, which is the direct test of the overshoot explanation (Appendix H). Appendix H explains the unexpected results.

**Conclusion.** In Figure 1, the multi-step skill formed within the first 3B tokens of mid-training while the held-out web loss rose. The loss curve pointed the wrong way as the capability appeared.

### AI USE STATEMENT

We used generative AI assistants (Claude Code and Codex) to write and run analysis and figure scripts, launch and monitor training and evaluation jobs, search the literature, and edit text under the authors' direction for grammatical and readability help as well as density of text. The results came from the stored runs and scripts named in the paper. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI. Everything was manually verified by the authors.

### REPRODUCIBILITY STATEMENT

All models we train start from public OLMo-2, Qwen2.5 and Mistral checkpoints. The run record of each run in the supplementary material names its checkpoint revision. Training settings are in Section 2 and Appendix D. The supplementary material contains our fine-tuning and evaluation scripts, the MathGAP item files, and the scripts that produce the numbers and figures in the paper. Appendix I lists which script and data file each number comes from. The synthetic task generator with its held-out items and the GSM8K item sets will be released with the code.

## REFERENCES

Dylan R. Ashley, Sina Ghiassian, and Richard S. Sutton. Does the Adam optimizer exacerbate catastrophic forgetting?, 2021. URL https://arxiv.org/abs/2102.07686.

Alankar Atreya, Devesh Batra, Yoages Kumar Mantri, Geremy Bantug, Greig A. Cowan, and Raad Khraishi. When to review: Spaced repetition for continual pre-training of language models, 2026. URL https://arxiv.org/abs/2608.17530.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2304.01373.

Cody Blakeney, Mansheej Paul, Brett W. Larsen, Sean Owen, and Jonathan Frankle. Does your data spark joy? performance gains from domain upsampling at the end of training, 2024. URL https://arxiv.org/abs/2406.03476.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. URL https://arxiv.org/abs/1905.10044.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

John Donaghy, Brian Wilcox, Oğuzhan Ersoy, Shikhar Rastogi, Adam St Arnaud, Alexey Titov, Jordan Greenberg, Ben Fielding, and Harry Grieve. OPEN-1B: A fully auditable training run, 2026. URL https://arxiv.org/abs/2609.17380.

Aleksandr Dremov, Alexander Hägele, Atli Kosson, and Martin Jaggi. Training dynamics of the cooldown stage in warmup-stable-decay learning rate scheduler, 2025. URL https://arxiv.org/abs/2508.01483.

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. Understanding emergent abilities of language models from the loss perspective. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2403.15796. arXiv:2403.15796.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang,

Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*, volume 36, pp. 70293–70332. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/deb3c28192f979302c157cb653c15e90-Abstract-Conference.html.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Advances in Neural Information Processing Systems*, volume 36, pp. 70757–70798. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/dfc310e81992d2e4cedc09ac47eff13e-Abstract-Conference.html.

Yujie Feng, Hao Wang, Jian Li, Xu Chu, Zhaolu Kang, Yiran Liu, Yasha Wang, Philip S. Yu, and Xiao-Ming Wu. FOREVER: Forgetting curve-inspired memory replay for language model continual learning, 2026. URL https://arxiv.org/abs/2601.03938.

Aaron Grattafiori et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5020–5048, 2025. URL https://aclanthology.org/2025.findings-naacl.282/. arXiv:2406.08446.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to (re)warm your model? In *ICML 2023 Workshop on Efficient Systems for Foundation Models (ES-FoMo)*, 2023. URL https://arxiv.org/abs/2308.04014. arXiv:2308.04014.

Tessa Han, Sebastian Bordt, Hanlin Zhang, and Sham Kakade. Weight decay improves language model plasticity, 2026. URL https://arxiv.org/abs/2602.11137.

Shengding Hu et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.06395. arXiv:2404.06395.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=DimPeeCxKO. arXiv:2403.08763.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, et al. Mistral 7B, 2023. URL https://arxiv.org/abs/2310.06825.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, et al. A study of BFLOAT16 for deep learning training, 2019. URL https://arxiv.org/abs/1905.12322.

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

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, et al. Measuring faithfulness in chain-of-thought reasoning, 2023. URL https://arxiv.org/abs/2307.13702.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2402.19255.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking spell on base LLMs: Rethinking alignment via in-context learning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2312.01552. arXiv:2312.01552.

Pin-Jie Lin, Rishab Balasubramanian, Fengyuan Liu, Nikhil Kandpal, and Tu Vu. Efficient model development through fine-tuning transfer, 2025. URL https://arxiv.org/abs/2503.20110.

Emmy Liu, Graham Neubig, and Chenyan Xiong. Midtraining bridges pretraining and posttraining distributions, 2025. URL https://arxiv.org/abs/2510.14865.

Emmy Liu, Kaiser Sun, Millicent Li, Isabelle Lee, Lindia Tjuatja, Jen-tse Huang, and Graham Neubig. What do language models learn and when? the implicit curriculum hypothesis, 2026. URL https://arxiv.org/abs/2604.08510.

Chenyang Lyu, Minghao Wu, and Alham Fikri Aji. Beyond probabilities: Unveiling the misalignment in evaluating large language models. In *Proceedings of the 1st Workshop on Towards Knowledgeable Language Models (KnowLLM 2024)*, pp. 109–131, 2024. URL https://aclanthology.org/2024.knowllm-1.10/. arXiv:2402.13887.

Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena D. Hwang, Luca Soldaini, et al. DataDecide: How to predict best pretraining data with small experiments. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2504.11393. arXiv:2504.11393.

Aleksandar Makelov, Georg Lange, Atticus Geiger, and Neel Nanda. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Ebt7JgMHv1.

Sohir Maskey, Philipp Scholl, Jonas Knupp, Pit Neitemeier, and Sascha Wirges. Good pretraining, bad SFT: Checkpoint quality across the training stack, 2026. URL https://arxiv.org/abs/2609.08966.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2202.05262. arXiv:2202.05262.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

OLMo Team. 2 OLMo 2 furious. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2501.00656. Shorter version accepted to COLM 2025; arXiv:2501.00656.

Andreas Opedal, Haruki Shirakami, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. MathGAP: Out-of-distribution evaluation on problems with arbitrarily complex proofs. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.13502. arXiv:2410.13502.

Sailesh Panda, Pritam Kadasi, Abhishek Upperwal, and Mayank Singh. When LLMs stop following steps: A diagnostic study of procedural execution in language models, 2026. URL https://arxiv.org/abs/2605.00817.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021. URL https://arxiv.org/abs/2103.07191.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023. URL https://aclanthology.org/2023.findings-emnlp.378/. arXiv:2210.03350.

Qwen Team. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Laura Ruis, Maximilian Mozes, Juhan Bae, Siddhartha Rao Kamalakara, Dwarak Talupuru, Acyr Locatelli, Robert Kirk, Tim Rocktäschel, Edward Grefenstette, and Max Bartolo. Procedural knowledge in pretraining drives reasoning in large language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2411.12580. arXiv:2411.12580.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2304.15004. arXiv:2304.15004.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, 2022. URL https://aclanthology.org/2022.emnlp-main.410/. arXiv:2205.12393.

Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2503.19206. arXiv:2503.19206.

Kaiser Sun and Mark Dredze. Amuro and char: Analyzing the relationship between pre-training and fine-tuning of large language models, 2024. URL https://arxiv.org/abs/2408.06663.

Lintang Sutawika, Leo Gao, Hailey Schoelkopf, Stella Biderman, Jonathan Tow, Baber Abbasi, et al. EleutherAI/lm-evaluation-harness: Major refactor (v0.4.0), 2023. URL https://doi.org/10.5281/zenodo.10256836. GSM8K task configuration: https://github.com/EleutherAI/lm-evaluation-harness/blob/main/lm_eval/tasks/gsm8k/gsm8k.yaml.

Xinpeng Wang, Bolei Ma, Chengzhi Hu, Leon Weber-Genzel, Paul Röttger, Frauke Kreuter, Dirk Hovy, and Barbara Plank. "my answer is C": First-token probabilities do not match text answers in instruction-tuned language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7407–7416, 2024. URL https://aclanthology.org/2024.findings-acl.441/. arXiv:2402.14499.

Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D. Hwang, et al. The generative AI paradox: "what it can create, it may not understand". In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2311.00059. arXiv:2311.00059.

Chen Yang, Junzhuo Li, Xinyao Niu, Xinrun Du, Songyang Gao, Haoran Zhang, Zhaoliang Chen, Xingwei Qu, Ruibin Yuan, Yizhi Li, Jiaheng Liu, Stephen W. Huang, Shawn Yue, and Ge Zhang. The fine line: Navigating large language model pretraining with down-streaming capability analysis, 2024. URL https://arxiv.org/abs/2404.01204.

Kazuki Yano, Shun Kiyono, Sosuke Kobayashi, Sho Takase, and Jun Suzuki. Pre-training LLM without learning rate decay enhances supervised fine-tuning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2603.16127. arXiv:2603.16127.

Hanlin Zhang, Jikai Jin, Vasilis Syrgkanis, and Sham Kakade. Prescriptive scaling reveals the evolution of language model capabilities. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.15327. arXiv:2602.15327; oral.

# A DETAILS MOVED FROM THE MAIN TEXT

**Math data at an early checkpoint.** With 262M tokens of math data and the answer line intact, the 441B checkpoint gains +0.419 on one-step and +0.111 on multi-step items over its own web-only anneal, against +0.465 and +0.179 from 4,001B.

**Checkpoint ranking at four steps (Section 3).** At four steps only 16 pairs of OLMo-2-1B checkpoints are decisive ([0.50, 1.00]).

**Likelihood candidates (Section 3).** Each MathGAP item has 16 candidates: the gold answer and 15 integers from the range of the pool's answers (offset from the gold -56 to +71). Of the distractors, 0.081 equal a number stated in the question, and most are larger than the gold (mean magnitude rank of the gold 0.33). Each candidate is scored as the continuation `<c>}` after the question and `{"answer":`; per-token scores were not stored, so the closing brace cannot be removed after the fact.

**Likelihood after the model's own reasoning (Section 3).** For each item we append the model's stored trace, cut at its answer line (or followed by one), to the generation prompt, and score the 16 candidates after it. This runs on the parent, the 10 mid-training checkpoints and the math, code and web-only anneals at both budgets. The score picks the model's own generated answer on 0.9996 of 2,356 items where that answer is a candidate, and with traces shuffled across items its accuracy falls to 0.101 (chance 0.0625). Over the 10 mid-training checkpoints its accuracy correlates with generated accuracy at Spearman 0.70 on two to four steps ($p = 0.023$) and 0.39 on one step. One checkpoint writes an answer line on only 0.047 of one-step items, so its generated accuracy is near zero while the conditioned score is not. On one-step items the score prefers math over code by +0.430 at 262M and +0.349 at 65.5M. The failure in Section 3 therefore belongs to the form without written reasoning. Outputs: `data/review_2026-09-25/r4/`.

**Template overlap in the math data (Section 3).** Of 20,017 documents from the math run's own draws, 20 (0.10 percent, Wilson 95% 0.06 to 0.15) contain both a MathGAP opener (*Name has n objects.*) and a MathGAP question form; the same pattern matches all 600 MathGAP items. Most sampled documents (81 percent, from TinyGSM) are grade-school word problems in other templates, so this scan rules out template copying but does not separate the math gain from word-problem practice in general.

**Data choice after 65.5M tokens, and knowledge data (Section 3).** After 65.5M tokens of annealing the choices are the same as after 262M by point estimate; the likelihood preference for code over math is +0.093 at 65.5M (95% interval $-0.012$ to $+0.198$, $n = 86$), against +0.128 ($+0.035$ to $+0.221$) at 262M, so at 65.5M it is not resolved (likelihood: code +0.174, math +0.081; generated: math +0.395, code $-0.047$). Knowledge data changes last-number accuracy by $-0.058$ ($-0.116$ to $-0.012$).

**The early-loss forecast within one model (Section 4).** Much of that is between models. Within OLMo-2-7B it holds across the 7 stage-1 runs ($-0.89$) and not across the 6 mid-training runs ($-0.20$), whose levels sit close together. The untouched checkpoint's loss is no substitute: it orders stage-1 checkpoints backwards (+0.66).

**Four quantities, in tokens (Section 2).** The level after a fixed fine-tuning budget, or the tokens to reach a level (post-trainability, Springer et al., 2025); the minimum level over a window at a given replay of the capability's own data; the share lost in the first 5M tokens without replay and the time constant after; and the tokens of own data that regain 80 percent of a repair's gain.

**The seven-checkpoint cooldowns (Section 4).** The cooldown works at earlier checkpoints too (Figure 3a). Read by the last number, it adds +0.20 to +0.44 at each of 7 checkpoints from 49B to 4,001B tokens, all above noise (single runs); with the answer line required, the gain clears noise at 4 of them. Four of these checkpoints were annealed on a second draw of the same data, and at 4,001B the draw alone moves the gain from 0.442 to 0.326, so smaller differences between checkpoints are not resolved. On the multi-step items, the primary readout fixed before these runs, the outcome

is mixed: the mid-training data raises the right answer's likelihood margin above noise at 5 of the seven checkpoints and generated accuracy at 0, with no clear trend in tokens (rank correlation 0.571). What these 65.5M-token cooldowns form is the one-step skill. The tokens a skill needs grow with its number of steps.

**What replay costs on GSM8K (Section 6).** Replay keeps the token and costs elsewhere: the one-percent forks sit 0.34 to 0.58 nats below the parent on the gold answer.

**Format first, then repair (Section 5).** On GSM8K what goes first is the end-of-sequence token after a correct answer, not the answer (Section 6). Repair returns the capability to one level from every history; whether its cost grows with delay is unresolved (Appendix D).

**The data choice at 1.05B tokens (Section 3).** The rule was fixed before the 1.05B runs: the winner is the source with the larger SVAMP gain over the web-only anneal of the same length (written reasoning, lenient, $n = 300$). Math data gains +0.160 (+0.110 to +0.210) and code data $-0.010$ ($-0.033$ to +0.010); one run per source. On one-step MathGAP the pattern of the 262M anneals repeats: math data changes generated accuracy by +0.384 (+0.267 to +0.500) and the likelihood score by $-0.081$, and code data by +0.058 and +0.058. The SVAMP answers for these runs were generated with Hugging Face greedy decoding instead of vLLM, with the same prompts, stop strings and token limits; rerun this way, the 262M math anneal agrees with its vLLM run on 0.99 of items (accuracy 0.257 against 0.267).

**Checkpoint ranking (Section 3).** Take any two of the OLMo-2-1B checkpoints we read (18 of 20 for one-step problems; the others yield too few answers), and call the pair decisive when generated accuracy separates them, that is, when its paired 95 percent interval excludes zero. Likelihood scoring orders a decisive pair the way generation does on 0.93 of 120 pairs for the simplest code problems and on 0.89 of 95 for one-step word problems (Figure 2c). On two-step and three-step problems it agrees on 0.34 of 97 and 0.50 of 86 pairs, and both item-bootstrap intervals include one half ([0.13, 0.67] and [0.29, 0.64]). On these problems, the likelihood score's choice of the better checkpoint is not distinguishable from chance. Four-step problems give too few decisive pairs to read (Appendix A).

**The data choice (Section 3).** We annealed the 4,001B OLMo-2-1B checkpoint for 262M tokens on web text alone and on four sources (math, code, instruction and knowledge data), each mixed half and half with web text, one run each. Against the web-only run, math data raises generated one-step accuracy by +0.465 (95% interval +0.360 to +0.570, $n = 86$) and moves the likelihood score by $-0.035$ ($-0.116$ to +0.058); a second draw of the data gives +0.523 and $-0.047$. Code data does the reverse: it raises the likelihood score by +0.093 (+0.035 to +0.151) and leaves generated accuracy unchanged (+0.000; Figure 2d). Chosen by this likelihood score, code data would be picked to improve math. None of the 10 likelihood metrics of Magnusson et al. (2025), recomputed on our saved scores, picks math for one-step items (Appendix F). On SVAMP, a separate problem set ($n = 300$), 262M tokens of math data raise accuracy with written reasoning by +0.080 (+0.043 to +0.120) and code data by +0.007 ($-0.010$ to +0.027). After 65.5M tokens both readouts choose as they do at 262M (by point estimate), and only the generated choice picks the winner at 1.05B, where math data raises SVAMP accuracy by +0.160 (+0.110 to +0.210) and code data by $-0.010$ (Appendix A). Instruction data lowers generated accuracy only through the format: it removes most answer lines and leaves last-number accuracy unchanged ($-0.012$; knowledge data in Appendix A). Math data also forms the multi-step skill that a 65.5M-token cooldown did not (Section 4): generated accuracy on two to four steps is 0.210 against 0.031 with web text alone.

**Ten targets (Section 3).** We read the math and code anneals of the 4,001B OLMo-2-1B checkpoint on ten targets, at 65.5M and 262M tokens, one run per source and budget. For each target we compare the math-minus-code difference under the likelihood score and under generated answers. On five targets the two readouts pick different sources with the same sign at both budgets. On one-step, multi-step and two-width MathGAP, the likelihood score picks code and generation picks math. On code level 3 and closure level 1, the likelihood score picks math and generation picks code. On closure level 1 the generated difference is $-0.002$ at both budgets. Code data raises the one-step likelihood score by 0.174 at 65.5M and 0.093 at 262M, while generated accuracy does

not rise ($-0.047$ and 0.000). Only two-width MathGAP has paired intervals that exclude zero with opposite signs in both readouts at both budgets. Files: `data/final_pass/C_transfer.json` and `C_robustness.json`; all five cells that overlap the main-text anneal match its numbers.

**Six results changed when only the scoring changed.** The GSM8K formation gap vanished under an answer-only fine-tuning recipe and returned under chain of thought (Section 4). An offline re-probe that did not require the end-of-sequence token read a real GSM8K collapse as flat. The GSM8K–TriviaQA trade-off vanished under containment. A 7B onset and a 1B null vanished without the JSON wrapper. A 1B plasticity test read by likelihood scoring found no difference. The readout is itself a variable of the axis, and we report each quantity under a readout that sees it.

| setting (model, items) | readout that requires the format or allows no steps | readout that does not |
|---|---|---|
| 7B GSM8K fork, 150M web tokens, no replay | stop required after the answer: 0.133 → 0.020 | right number: 0.133 → 0.113 |
| 7B released, 8.4M GSM8K fine-tuning tokens | answer-only data, five-shot: 0.700 → 0.144 | worked solutions, five-shot: 0.69 → 0.646, 0.668 |
| 7B stage 1 at 1,443B, program arithmetic | told to answer in JSON: 0.04 | program as the prompt: 1.00 |
| 1B cooldown on anneal data, 441B parent, one-step | answer line required: −0.174 | last number: +0.360 |
| 1B cooldown on own data, 4,001B, multi-step | likelihood margin: +0.177 (above noise) | generated accuracy: +0.015 (inside noise) |
| 1B mid-training, multi-step MathGAP | likelihood-scored accuracy: changes at 0 of 10 checkpoints | generated accuracy: rises at 10 of 10 |
| 7B GSM8K replay forks, 100M, against TriviaQA | exact match: $r = -0.915$ | containment: $r = 0.103$ |
| 1B end of stage 1, correct GSM8K answers | final answer right: 1.00 | every step valid (judge): 0.35 |

Table 2: **All eight same-model, same-item pairs in our runs.** Files in Appendix I.

## B Related work in full

**Two further neighbours.** The cooldown's loss drop has a loss-landscape account (Dremov et al., 2025). Accuracy falls with procedure length and look-back (Panda et al., 2026); at matched length, we vary how a step uses earlier quantities (Section 4).

Late-stage upsampling of math and code raises GSM8K (Blakeney et al., 2024), and in MiniCPM a decay on pretraining data gave far lower GSM8K than a decay mixed with high-quality data (Hu et al., 2024); our cooldowns add the constant-rate run and read the base model. Adding math to OLMo-2's mid-training mix raises its GSM8K by more than twenty points (OLMo Team, 2025; see also Hu et al., 2024; Grattafiori et al., 2024). We measure in tokens what the stage installs and separate its data from its decay. Catastrophic overtraining varies the checkpoint by token count and finds later checkpoints harder to fine-tune (Springer et al., 2025), and in their controlled runs, which anneal every budget fully, the penalty still grows with tokens. Along the public OLMo-2 run at a fixed recipe we find the stage separating checkpoints more than the count, and at 1B a matched learning rate removes most of the count effect. The two findings differ in recipe and target, and a rate-matched comparison at 7B is the test that relates them. The implicit curriculum hypothesis reads what a checkpoint can already do (Liu et al., 2026); early fine-tuning loss reads what it can still be taught. Loss and downstream accuracy are known to diverge (Du et al., 2024), and the choice of metric can make a smooth change look abrupt (Schaeffer et al., 2023); we read the same items by two scoring rules along one run. DataDecide ranks data recipes from small runs with likelihood proxies (Magnusson et al., 2025), and OLMES standardises cloze and multiple-choice formats (Gu et al., 2025); on multi-step problems we find cloze-style likelihood scoring flat where generation moves. Likelihood scores and generated answers disagree at a single checkpoint (Wang et al., 2024; Lyu et al., 2024). Replay fractions are usually chosen on loss (Ibrahim et al., 2024; Gupta et al., 2023; Scialom et al., 2022), and spaced replay schedules help at a 20 percent budget (Atreya et al., 2026); we vary where a fixed one-percent fraction falls and whose data it is. Adam has been studied as a source of forgetting (Ashley et al., 2021); we find that a $\beta_2$ change on an inherited Adam state separates survival without replay, largely through a smaller effective step. Alignment tuning changes mostly stylistic tokens (Lin et al., 2024); continued pretraining removes the answer-format conditional first while the answer stays. Generation and discrimination are known to come apart in

capable models (West et al., 2024); we find the likelihood and generation readouts diverge along one pretraining run, more so as the answer needs more steps. Pretraining documents that carry procedures shape reasoning (Ruis et al., 2025); our cooldowns attribute a capability to a stage's data at the level of the whole mix.

**A cooldown that removes the answer line.** From the 441B checkpoint the 65.5M-token cooldown on mid-training data adds +0.360 to one-step accuracy if we read the last number, but $-0.174$ if we require the answer line, because the new data removes the answer line from 0.68 of the answers (against 0.035 for the decay alone).

## C Reasoning traces along the run

All numbers in this section come from two exploratory scripts run on the stored generations, each with a planted self-test that passed before any real read: `cpujobs/v3_cot_dev.py` (output `data/v3/cot_dev.json`) and `cpujobs/v3_proof_quality.py` (`data/v3/proof_quality.json`). MathGAP excludes the 85 toy-car items; GSM8K numbers at mid-training checkpoints are contaminated and are marked as such in the files.

**Milestones in tokens.** On one-step MathGAP items (86 items), the answer line appears in 0.33 of traces at 21B tokens and 0.99 at 84B, and stays there. The share of true arithmetic statements reaches its stage-1 level only between 2,559B and 4,001B, and its curve is not monotone (it dips at 1,301B). We therefore report the order of the two milestones and not a precise token count for the second. On GSM8K, exact match stays between 0.017 and 0.035 across ten stage-1 checkpoints. From 231B on, the largest class of wrong answers is the one in which every checkable arithmetic statement is true and the answer is still wrong (35 to 52 percent of wrong traces).

**Early intermediate-step onset underperforms difficulty.** On 1244 GSM8K items with a reference intermediate step, we recorded the first stage-1 checkpoint from which that value appears in the trace and at two later checkpoints. This onset predicts correctness after mid-training at AUROC 0.563, below the AUROC 0.668 of the reference solution's step count alone (`data/v3/trajectories.json`; `cpujobs/item_trajectories.py`). The onset is a weaker predictor than the reference step count.

**Arithmetic check against a judge.** The arithmetic check of Section 4 reads arithmetic only, so it disagrees with a judge of step validity by design ($\kappa = 0.16$, $n = 969$). It passes 336 of the 340 traces the judge accepts and 489 of the 629 it rejects, so most traces the judge rejects have correct arithmetic and faulty reasoning steps.

**Step models.** The comparison of Section 4 fits each model per checkpoint by maximum likelihood over the step counts whose answer line appears on at least half the items (`analysis_timing/compose/compose.py`). On planted data with the same items and checkpoints, the test of equal steps rejects the true model on 4% of checkpoints and a constant accuracy on 94%, and the pooled test of $r = 1$ fires on 5% of equal-step draws and on 95% of draws with $r = 0.87$. In stage 1, models with a guessing floor fit better than equal steps (summed AIC 168.7 against 197.6 over 9 checkpoints), which is why no step reliability is reported there.

**The likelihood score is not frozen.** The margin of the right answer grows at 7 of the 10 checkpoints, but it stays negative, so the right answer is never ranked first.

**What mid-training changes.** From the last stage-1 checkpoint to the released 1B model, accuracy on the multi-step items rises by 0.305 (95 percent interval 0.242 to 0.366, 262 paired items), while the answer-line rate ($-0.012$) and the rate of copying a number from the question ($-0.004$) do not change. The same kind of trace becomes correct more often. In the 65.5M-token cooldowns from the 4,001B parent, the anneal data at a constant rate raises accuracy on the multi-step items by 0.065 (0.025 to 0.107) and removes the answer line from 0.210 more traces. The decay on stage-1 data moves neither. Open-1B's mid-training repeats the pattern under another group's run: the share of true arithmetic statements rises by 0.302 (0.237 to 0.372), accuracy by 0.057 (0.019 to 0.099), and the answer-line rate falls by 0.252.

**Scale.** At matched stage-1 tokens OLMo-2-7B leads OLMo-2-1B on the multi-step items by up to 0.248 (0.195 to 0.301; 915B against 797B) and on GSM8K by up to 0.117 (1,443B against 1,301B). The two are indistinguishable at the earliest matched pair (110B against 84B).

**Faithfulness.** The early-answering test of Lanham et al. (2023) gives an area over the curve of 0.48 to 0.56 at four stage-1 checkpoints and 0.66 at the first mid-training checkpoint.

**Length and proof quality.** Within items, longer traces are more often correct on GSM8K at stage 1 (point-biserial $r = +0.088$) and less often on MathGAP ($-0.161$) and closure ($-0.492$). On MathGAP the within-item sign turns positive after mid-training ($+0.196$), while an item-clustered logistic coefficient stays negative at both stages, so length is not a stable marker of quality. Among stage-1 GSM8K answers scored correct, 21 of 90 (0.23) have every judged step valid, against 102 of 108 (0.94) after mid-training; wrong answers with every step valid are rare at every stage (0 to 4.6 percent). A predictor of correctness from trace features fitted on stage-1 traces transfers to mid-training traces on MathGAP (AUROC 0.871) and code (0.840) but not on closure (0.593) or GSM8K (0.620).

**Two further notes on Section 4.** With 2 percent of the tokens of AI2's first public mid-training checkpoint, the 4,001B cooldown reaches 0.558 one-step accuracy against that checkpoint's 0.686. Under standard complexity assumptions, bounded-depth transformers cannot compute compositions of growing length that way, and can when they write the intermediate steps (Feng et al., 2023; Merrill & Sabharwal, 2024).

**Step reliability model.** If every step of a $k$-step answer succeeded with the same probability $p$, accuracy would be $p^k$ (Schaeffer et al., 2023; Dziri et al., 2023). Our mid-training checkpoints fall faster than that (Figure 11b). Over the 8 with three or more step counts, a model whose step reliability falls by a factor $r$ per step fits better (summed AIC 164.8 against 182.5; median $r$ 0.87; $r = 1$ rejected at $p = 5 \times 10^{-5}$), and so does a straight line (163.6); stage-1 accuracy is too near its floor to fit.

**Chains after the second step, and at 1B (Section 4).** After the second step the decline per step is not steeper: the per-step reliability factor $r$ of the step model above is lower in only 2 of 7 models. At 1B even stated-number chains decline ($r$ from 0.88 to 0.98).

**Likelihood on multi-step items (Section 4).** A likelihood score ranks the final answer with no written steps, so every combination must happen inside one forward pass; on two to four steps it stays between 0.06 and 0.19 throughout.

**Onset rule.** We date a change with an onset rule: the first checkpoint whose accuracy exceeds the first checkpoint's by more than the sum of their Wilson half-widths at $z = 2.33$. On planted flat curves the rule fires in at most 1.5% of 200 draws.

## D Additional 7B evidence

**Section 6 statistics.** Training loss does not separate them on any panel (separation $D = 0.73$, permutation $p = 0.343$). Seven accuracy benchmarks separate them less than a pure-noise monitor does (mean $D = 0.65$), while the capability probe reaches $D = 6.67$. Held-out language-model loss is the one exception (uncorrected for the nine metrics). It separates the conditions at the exact-null floor at four of the five read points ($D = 2.50$, $p = 0.0286$), yet moves only 0.0015 nats. The pattern of which items a model gets wrong, which an average discards, separates the runs. Within-condition agreement on which items fail exceeds between-condition agreement at rank 1 of the 1,575 distinct partitions of the runs, on 5 tasks at 200M. A leave-one-run-out nearest-centroid decoder on per-item correctness identifies every held-out run (100 percent). The gold answer's per-item log-likelihood identifies 10 of 10 runs on GSM8K and 10 of 10 on TriviaQA (chance 3.1 of 10, the mean over every relabelling of the runs), and 100 random TriviaQA items suffice (100 percent). Each of these tests sits at the floor of its exact permutation null ($p = 0.000635$). The per-item pattern is not the answer format: on 1,319 GSM8K items with every line-break token removed, arithmetic tokens

alone still identify every run. Three benchmarks with no shared items induce nearly the same run-by-run geometry once item difficulty is removed (Mantel $r$ 0.77 to 0.95, every pair $p \leq 0.0055$), so the pattern belongs to the checkpoint and not to the benchmark. The same instrument, applied to the eleven public OLMo-2-7B revisions, identifies the stage of 11 of 11 (exact floor $p = 0.00216$; Figure 12). A rank-eight reconstruction of the fork's change reproduces the full effect (1.05 to 1.16 of it over 5 pairs of fork and item set) where random subspaces do not, and the effect holds on a disjoint item set (Figure 6).

**Fast loss (Section 5).** Without replay, a capability formed over 50M tokens loses 0.67 of its level in the first 5M tokens of the background task (95% interval 0.65 to 0.68, $n = 36$ forks), and the rest decays with a time constant of about 132M tokens (65M to 186M, $n = 8$). One formed over 200M tokens loses 0.33 in the same 5M ($n = 8$; Figure 4a). Replay that starts after a few million tokens therefore comes after most of the loss.

**The one-percent comparison (Section 5).** We compare one percent uniform replay with no replay parent by parent, taking each parent's median over its forks within a 50M-token window. On the minimum of the synthetic score, one percent sits below no replay on 13 of 13 non-tied OLMo-2-7B parents (1 tie; one-sided sign test $p = 1.2 \times 10^{-4}$; median difference 0.078). The mean level over the window points the same way (12 of 14 parents, $p = 0.006$), but the final level does not resolve (10 of 14, $p = 0.090$). The sign reverses on a fully formed Qwen2.5-7B parent (0.395 against 0.301) and on the one OLMo-2-13B parent (0.066 against 0.047, one fork each; Table 3). We test at the parent level because forks of one parent share their batches (below). The deficit repeats on three further formed parents, where one percent falls 0.074 to 0.082 below no replay. Five percent keeps the capability, so retention is not monotone in the dose.

**The $\beta_2$ forks (Section 5).** By effective step we mean the size of each update after Adam divides by its bias-corrected second moment. Our forks inherit the parent's second moment and step counter ($t = 382$), accumulated at $\beta_2 = 0.95$; a fork run at 0.999 applies that value's bias correction to it, so each step is 1.77 times smaller. Our trainer also keeps Adam states in bfloat16, where multiplying by 0.999 can leave the second moment unchanged (below). Without replay for 200M tokens, the capability is lost in every run at $\beta_2 = 0.95$ and kept in every run at 0.999 (kept: minimum at or above 0.15; lost: final value below 0.08; 3 of 3 and 3 of 3, one learning rate and one recipe; Figure 4c). At twice the learning rate both lose it (one run each; 0.066 and 0.000), while the background task, answered with the same one-key JSON object on the same programs, stays at 0.94 or above, so the loss is not a lost answer format. In runs with one-percent replay, the step reduction applied through the learning rate gives minima of 0.262 to 0.293 (2 forks) and 0.999 with the bias correction removed gives 0.164 to 0.184 (4 forks), against 0.004 to 0.051 at the full rate (Table 4). Without replay, one run of $\beta_2 = 0.95$ at the matched rate, $1.13 \times 10^{-5}$, keeps the capability too (minimum 0.270 over 200M tokens), so we read the separation as mostly a smaller step. The learning rate affects forgetting (Ibrahim et al., 2024); in a fork, a $\beta_2$ change on inherited state is also a learning-rate change. On the GSM8K answer-only pool it does not separate the runs.

This section holds the 7B results that Sections 4 and 5 cite but do not show.

**The per-item pattern tracks the data, not the label (Section 6).** The per-item pattern tracks the data a run consumed, not its label: at 100M tokens, before the end-block runs see target data, grouping by data consumed separates the runs completely ($+0.127$ within-minus-between, exact $p = 0.029$) and grouping by condition label does not ($+0.051$, $p = 0.13$).

**Scoring rules.** GSM8K forks are scored offline on a fixed set of 256 held-out items under three rules: *exact match* (the number tokens, then the end-of-sequence token), *number right* (the number tokens alone), and a *lenient rule* that cuts at the first newline and takes the first integer. The exact-match rule is stricter than the strict-match filter of the evaluation harness, and the lenient rule is close to its flexible-extract filter (Sutawika et al., 2023). The synthetic task's answers are JSON objects, scored by parsing the object.

**One synthetic item.** A held-out item of the synthetic task, with its last instruction wrapped to fit the page (target {`"batch": -197`}):

````
Below is a Python program.

```python
def _h0(x, y, bias):
    return (((x * 1) + (y - 39)) + bias)

def main():
    gain = 42
    batch = 3
    span = -129
    delta = "gamma-amber"
    gain = _h0(batch, span, 0)
    gain %= 201
    if (span > -147):
        batch += 101
    else:
        batch -= 162
    if (span > -164):
        batch += -301
    else:
        batch -= 165
    return {"gain": gain, "batch": batch, "span": span, "delta": delta}
```

Run `main()` and report the final value of `batch`.
Reply with a JSON object whose only key is the variable name and whose
value is that final value, like {"name": value}.
Output only the JSON object.
````

**The stage difference is not the contaminated test set.** The released OLMo-2-7B checkpoint saw the GSM8K test questions in its mid-training mix. We therefore re-read the difference between the released and the last stage-1 checkpoint, at the parents and after 16 and 64 updates of fine-tuning with worked solutions (two seeds each). Between 0.81 and 0.94 of the GSM8K difference survives on GSM-Plus items whose numbers are changed, and 0.62 to 0.69 survives on SVAMP, a different problem set. The difference is present in every GSM-Plus perturbation type (Figure 5a).

**The stage difference is not a learning-rate match.** We repeated the 64-update fine-tuning with worked solutions at $4 \times 10^{-5}$ and $8 \times 10^{-5}$, one seed each; the $2 \times 10^{-5}$ values are the first seed above. The released checkpoint scores 0.646, 0.594 and 0.442 at the three rates, and the stage-1 checkpoint scores 0.372, 0.398 and 0.248 (five-shot, flexible extraction, 500 test items). After fine-tuning, the best released checkpoint leads the best stage-1 checkpoint by 0.248, against 0.274 at $2 \times 10^{-5}$ alone. Every rate lowers the released checkpoint below its parent's 0.690, and the stage-1 checkpoint scores highest at $4 \times 10^{-5}$. The new runs used a rebuilt training stack; on it, the stage-1 run at $2 \times 10^{-5}$ scores 0.360, against 0.372 before.

**A second scale.** OLMo-2-13B under the same recipe ($2 \times 10^{-5}$, 64 updates, one seed) gives the same sign. Before fine-tuning the released checkpoint scores 0.738 and the last stage-1 checkpoint 0.286; after 16 updates 0.686 and 0.378; after 64 updates 0.666 and 0.454, a gap of 0.212 (paired item interval 0.164 to 0.262, $n = 500$). On the GSM-Plus items with the numbers changed the 64-update gap is 0.240, and on SVAMP 0.090.

**The template control.** Before the synthetic-task fine-tune of the 2,400B checkpoint, 25M tokens of another task in the same template lift the level to 0.188 and 0.152 (two seeds), against 0.078 and 0.090 with no warm-up. The same 25M tokens of web text, with the token lengths of the template rows matched and the loss on the answer tokens only, give 0.160 and 0.094. The web-text mean, 0.127, lies 0.043 above no warm-up and 0.043 below the template run. The two web-text seeds differ by more than either gap, so the even split is imprecise. The web-text run ran on the rebuilt stack; the other two runs ran on the original one on 2026-09-19.

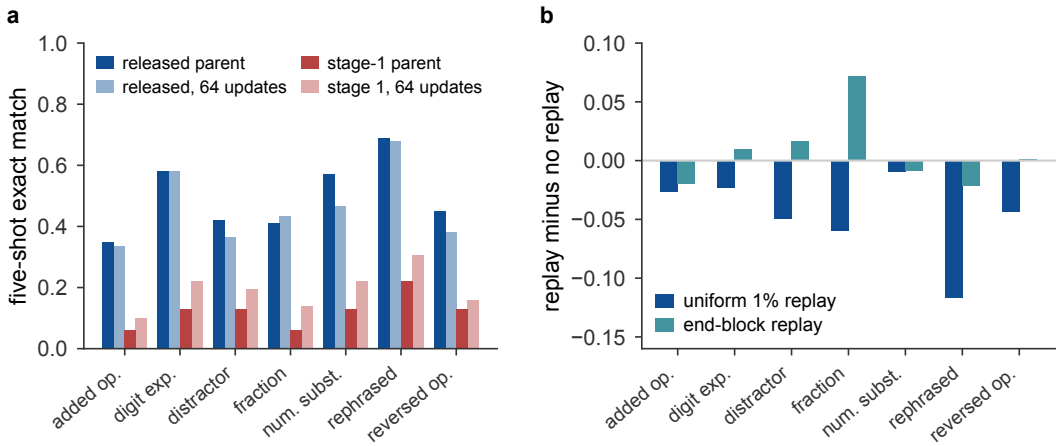



Figure 5: **GSM-Plus by perturbation type** (100 items per type, fixed five-shot prefix; the critical-thinking type has no numeric answer and is omitted). (a) The released and the last stage-1 parents and their 64-update fine-tunes. (b) Ten forks at 50M and 150M tokens: one percent uniform replay in the answer-only format minus no replay, and end-block replay minus no replay, by type. `numbers/gsmplus_by_type.py`, `analysis_timing/timing_numbers.py`.

**Where the replay falls.** Replay placed at the very end keeps a median of 0.094 ($n = 18$), below no replay (0.133), and recency alone correlates with retention at $-0.14$ ($p = 0.06$), so neither recency nor the largest gap orders retention alone.

**Own data keeps; other data does not.** From one OLMo-2-7B parent (seed 4), one percent uniform replay of the capability's own data over 50M tokens gives minimum levels of 0.012 to 0.051 ($n = 14$ forks). The same schedule with GSM8K problems in the replay slots gives 0.113 to 0.137 ($n = 4$), inside the no-replay band. The two sets do not overlap. The damage of one percent uniform replay needs the capability's own data.

**The pooled test and evaluation density.** Pooled over OLMo-2-7B forks, the medians of the minimum are 0.027 with one percent uniform replay and 0.125 without (one-sided Mann-Whitney $p = 5.0 \times 10^{-12}$, 56 and 43 forks). Forks of one parent share their batches, so Section 5 uses the parent-level sign test. Both conditions were evaluated at the same density (median 10 points per 50M tokens, range 10 to 96 in each condition), so the deeper minimum is not an artefact of denser evaluation.

**The one-percent deficit, parent by parent.** Table 3 gives the minimum level under one percent uniform replay and under no replay for every parent with both conditions. One percent falls below no replay in 13 of 14 OLMo-2-7B parents. On OLMo-2-13B and on Qwen2.5 at 1.5B and 3B it sits at or above no replay, from one to four forks per condition. On Qwen2.5-7B the sign depends on the parent, as Section 5 describes.

**The end-of-sequence token is lost, not the answer.** On GSM8K, the drop that the exact-match rule records in every no-replay fork is the loss of the end-of-sequence token after a correct answer, not the loss of the answer. The number is right on 0.113 to 0.121 of items, against the parent's 0.137. Of the 29 to 31 correct answers per fork, 24 to 27 continue with a unit word or a new sentence. Under teacher forcing, the newline class carries 0.951 of the shift in the gold log-likelihood. Under a five-shot prompt, every no-replay fork stops after its answer (share 1.00, parent 0.79) and scores 0.66 to 0.67, against the parent's 0.72. Web text usually follows a bare number with a unit word or more text, and the forks learn that continuation.

**Replay keeps the end-of-sequence token.** We use four matched settings, three learning rates ($5 \times 10^{-6}$, $10^{-5}$ and $4 \times 10^{-5}$) and one with $\beta_2 = 0.99$, each run without replay and with replay. At 200M tokens, the share of the 256 items with the right number and no end-of-sequence token is

| family | parent | 1% uniform | none | forks | none minus 1% |
|---|---|---|---|---|---|
| OLMo-2-7B | seed 3 | 0.016 | 0.094 | 1, 1 | +0.078 |
| OLMo-2-7B | seed 4 | 0.027 | 0.125 | 36, 24 | +0.098 |
| OLMo-2-7B | seed 5 | 0.031 | 0.094 | 1, 1 | +0.062 |
| OLMo-2-7B | seed 6 | 0.023 | 0.160 | 1, 1 | +0.137 |
| OLMo-2-7B | seed 7 | 0.008 | 0.082 | 1, 1 | +0.074 |
| OLMo-2-7B | seed 8 | 0.020 | 0.141 | 1, 1 | +0.121 |
| OLMo-2-7B | seed 9 | 0.043 | 0.117 | 1, 1 | +0.074 |
| OLMo-2-7B | seed 10 | 0.027 | 0.105 | 1, 1 | +0.078 |
| OLMo-2-7B | seed 11 | 0.035 | 0.098 | 1, 1 | +0.062 |
| OLMo-2-7B | seed 12 | 0.031 | 0.043 | 1, 1 | +0.012 |
| OLMo-2-7B | seed 13 | 0.027 | 0.141 | 1, 1 | +0.113 |
| OLMo-2-7B | seed 4, 1,479 distinct examples | 0.004 | 0.004 | 1, 1 | +0.000 |
| OLMo-2-7B | seed 4, formed over 200M tokens | 0.156 | 0.285 | 4, 3 | +0.129 |
| OLMo-2-7B | seed 4, formed over 200M tokens, second set of forks | 0.094 | 0.285 | 5, 5 | +0.191 |
| OLMo-2-13B | seed 3 | 0.066 | 0.047 | 1, 1 | -0.020 |
| Qwen2.5-1.5B | one parent | 0.203 | 0.207 | 4, 1 | +0.004 |
| Qwen2.5-3B | one parent | 0.344 | 0.332 | 3, 1 | -0.012 |
| Qwen2.5-7B | seed 3, fully formed | 0.395 | 0.301 | 5, 5 | -0.094 |
| Qwen2.5-7B | stopped at a level matched to OLMo-2 | 0.092 | 0.148 | 8, 7 | +0.057 |

Table 3: **Minimum synthetic-task level over a 50M-token window, by parent.** A parent is a checkpoint formed on the synthetic task over 50M tokens unless stated; seed is the formation seed. Medians over forks; uniform placement, learning rate $2 \times 10^{-5}$, no language-model background task. Descriptive; parents with one fork per condition carry no interval. `analysis_timing/family/family_sign.py`, `data/tokencost/records_all.csv`.

0.016 to 0.109 without replay, and it rises with the learning rate. With replay, uniform or in an end block, it is 0.004 to 0.016 (7 runs), below the matched no-replay run in all 4 settings. The share with the right number does not order the runs the same way across settings. On GSM8K, this is consistent with replay keeping the answer format more than the answer.

**Where the change lives.** If the parent receives a no-replay fork's residual stream after layer 8, it stops emitting the end-of-sequence token. On the first fork a single-layer sweep placed the change in layers 3 to 7, over the question tokens; the profile did not replicate on two of three forks (below). One update on target data restricted to layers 0 to 7 restores the exact-match score from 0.016–0.020 to 0.113–0.125. Eight updates restricted to layers 24 to 31 reach 0.031–0.059. Replay keeps the token but costs elsewhere: the one-percent replay forks sit 0.34 to 0.58 nats below the parent on the gold answer.

**The activation effect transfers to new items.** A replication repeated the residual-stream intervention on a disjoint GSM8K item set (Figure 6a). A rank-eight reconstruction of the no-replay change reproduces 1.05 to 1.16 of the full patch effect over 5 pairs of fork and item set; random subspaces do not (Figure 6b). The depth profile did not replicate on two of three forks (`data/xblock/x1/x1_score.json`). These interventions locate a compact change controlling this stopping behavior; they do not measure the dimension of the model's other capabilities.

**The $\beta_2$ separation, taken apart in runs with replay.** Forks inherit the parent's Adam step counter, $t = 382$, so the second-moment bias correction $1 - \beta_2^t$ is 0.318 at $\beta_2 = 0.999$ and each update is $\sqrt{1/(1-\beta_2^t)} = 1.77$ times smaller than at 0.95 at the fork, decaying towards one as $t$ grows. Table 4 gives the minimum synthetic-task level over a 50M-token window under one percent uniform replay from one OLMo-2-7B parent (seed 4), with each part of the optimizer state changed on its own: the step through the learning rate at $\beta_2 = 0.95$, the normaliser through $\beta_2 = 0.999$ with the step counter overridden to $10^5$ so that the bias correction is one, and both through plain $\beta_2 = 0.999$. The override at $\beta_2 = 0.95$ is the control; it changes nothing. At zero replay over 200M tokens, $\beta_2 = 0.95$ at the matched rate $1.13 \times 10^{-5}$ keeps the capability in one run (minimum and final 0.270).

**Optimizer state precision.** Retention is flat from 0.95 to 0.99 and changes only at 0.999, which the second moment's memory horizon does not predict and the number format of our trainer can. It keeps Adam's second moment in bfloat16, where multiplying a number by 0.999 returns the same

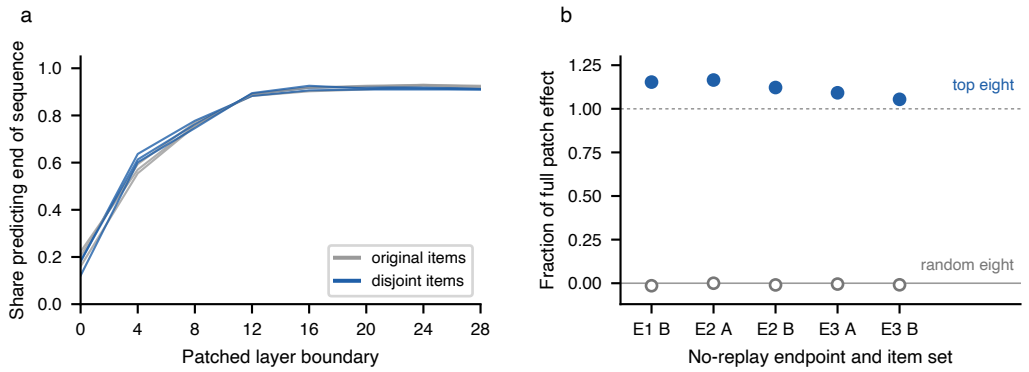



Figure 6: **The stopping effect on a disjoint item set and under a low-rank reconstruction.** (a) Share of items on which a no-replay fork predicts the end-of-sequence token after receiving the healthy parent's residual stream at each layer boundary. Each line is one fork; grey and blue use disjoint item sets. (b) Fraction of the full stopping effect reproduced when the parent receives the fork's change in the top singular directions or random directions. Each point is one pair of fork and item set. `figures/make_fig_v4_mechanism.py` reads `data/xblock/x1/x1_score.json`.

| part of the state changed | condition | minimum level | forks |
|---|---|---|---|
| neither | $\beta_2$ 0.95, lr $2 \times 10^{-5}$ | 0.004 to 0.051 | 23 |
| step only | $\beta_2$ 0.95, lr $1.13 \times 10^{-5}$ | 0.262 to 0.293 | 2 |
| step only, further | $\beta_2$ 0.95, lr $10^{-5}$; $5 \times 10^{-6}$ | 0.328 to 0.352; 0.395 to 0.398 | 3; 3 |
| normaliser only | $\beta_2$ 0.999, step counter overridden | 0.164 to 0.184 | 4 |
| both | $\beta_2$ 0.999 | 0.309 to 0.348 | 12 |
| control | $\beta_2$ 0.95, step counter overridden | 0.039 | 1 |

Table 4: **The $\beta_2$ separation by part of the optimizer state.** Minimum over a 50M-token window, one percent uniform replay, OLMo-2-7B parent, seed 4. `numbers/optstep_numbers.py`, from `data/tokencost/records_all.csv` (leaves `b7/optstep/*`, `lr_*`).

number for all 32512 positive normal values, while 0.99 and 0.95 lower every one of them. The threshold is sharp: multiplying by 0.996 changes every value (unchanged share 0.00), while 0.997 leaves a share of 0.30 unchanged and 0.998 a share of 0.95. These shares follow from the format. A bfloat16 value with significand $m \in [1, 2)$ keeps 8 significant bits, so multiplying it by $\beta_2$ returns it unchanged when $m(1-\beta_2) < 2^{-8}$, and the unchanged share is $\min\{\max\{2^{-8}/(1-\beta_2)-1,\, 0\},\, 1\}$. This reproduces each measured share to within one significand step ($1/128$). Under $\beta_2 = 0.999$ the second moment can grow but not decay: in PyTorch's Adam, a second moment of 1 with zero gradients is still 1.00 after 1000 steps, where exact arithmetic gives 0.37. A second moment that stays large keeps later updates small. We have not tested that this causes the protection; a run with float32 optimizer states would.

**Known and new capabilities.** On a capability the checkpoint already has, the later stage raises the starting level and leaves the rate of gain unchanged. At 1B, after the same 64-update fine-tune on MathGAP, the 8 mid-training checkpoints gain 0.012 more on multi-step items than the 13 stage-1 checkpoints (one seed each). Checkpoints taken after a decay can even fine-tune worse on standard benchmarks (Yano et al., 2026; Maskey et al., 2026), in Maskey et al. (2026) largely through a failure to stop generating, which is the format effect again.

**Weights small enough to move.** The share of MLP up-projection entries small enough to move at our rate grows from 0.186 at 412B to 0.219 at 2,400B, and it is flat only across the last boundary (0.3237 at 3,896B against 0.3242 after 5B mid-training tokens).

**Three contributing factors to the stage difference.** Three measurements take the difference apart, and none accounts for it alone. The first is the effective step size. The 7B trainer keeps

weights and optimizer states in bfloat16, so updates below half a weight's spacing are lost. The share of weights small enough to move at our rate is flat across the last boundary (above). At 1B, where the rate was varied at every checkpoint, the smallest rate that works falls with the checkpoint's age, while the best rate, $4 \times 10^{-5}$, is the same for all four (Figure 3c). At $5 \times 10^{-6}$ the 210B checkpoint reaches 0.023 and the 4,001B one 0.695, and at a rate chosen per checkpoint (one seed per rate) every stage-1 checkpoint reaches 0.785 to 0.883. The second is answer format. Before the same fine-tune, 25M tokens of a different task in the same prompt-and-answer template raise the 2,400B checkpoint from 0.078 and 0.090 to 0.188 and 0.152 on two seeds, 0.36 of the way to the first mid-training checkpoint. The same 25M tokens of web text reach 0.160 and 0.094, so extra training and the template each account for about half of the lift. The third is a head start on skills the model already has. On GSM8K, the last stage-1 checkpoint and the released one differ before any fine-tuning (0.22 against 0.69 five-shot). After 8.4M tokens of worked solutions, the stage-1 checkpoint gains 0.152 and 0.142 on two seeds while the released one changes by $-0.044$ and $-0.022$. At the best of three learning rates for each checkpoint, the released one still leads by 0.248 (one seed per rate), so at 7B the gap is not a learning-rate mismatch, unlike at 1B. It also holds at 13B: after 64 updates the released OLMo-2-13B checkpoint leads its last stage-1 checkpoint by 0.212 (0.164 to 0.262, $n = 500$) on GSM8K and by 0.240 on GSM-Plus items with the numbers changed.

**Readout sources.** GSM8K numbers come only from offline probes of saved checkpoints; synthetic-task numbers come from the runs' own evaluations, which reproduce offline at the final checkpoint we re-read. For the checkpoint-choice test, a checkpoint is dropped from a problem type when it writes an answer line on fewer than half of the items.

**The one-percent deficit on Qwen2.5.** A candidate reason for the sign change is how far the parent had formed the capability. A Qwen2.5-7B parent stopped at a level matched to OLMo-2 shows the deficit (median 0.092 against 0.148; 8 and 7 forks), while the fully formed Qwen2.5-7B parent shows the reverse (0.395 against 0.301; 5 and 5). The matched comparison is suggestive only, because that parent was still improving when it was forked. On Qwen2.5-3B one percent also sits at or above no replay.

**Weight space.** The per-item pattern of Section 6 is not carried by distance in weight space (AUC 0.50, $p = 0.25$).

**Cost of the readout.** Scoring 100 TriviaQA items by gold log-likelihood, in the prompt format of the per-item readout, takes 15,479 forward tokens under the OLMo-2 tokenizer. That is 0.06 percent of the 25M tokens between our first two evaluation points, and 0.02 percent of their training compute.

**Repair.** Fine-tuning on the capability's own data returns a lost capability to one plateau from every history, and the history sets the cost. In a test over 10 settings from 3 parents, the repair needs about 0.71M more tokens for every 50M tokens of unrelated training between formation and repair (permutation $p = 0.0042$). A replication on a second set of repair curves did not reach significance. Pooled over all 40 curves from 7 parents, a test over all curves gives 1.17M per 50M ($p = 0.0005$; Figure 7).

**The four quantities along the mid-training stage and across models.** Figure 8 follows the four quantities of Appendix A over three OLMo-2-7B revisions: the last stage-1 revision at 3,532B tokens and the mid-training revisions after 5B and 50B tokens (twelve runs, one seed). Table 5 collects them across models and scales.

## E CAPABILITIES ALONG THE PUBLIC RUNS

These readouts use data already on disk; none changes a claim in the main text.

**The loss carries little beyond the token count.** Along stage 1 every capability in Figure 11a correlates with the falling loss, because both move with tokens. With log tokens held out, the web loss has a partial correlation of at most 0.19 in size with one-step math, multi-step math and

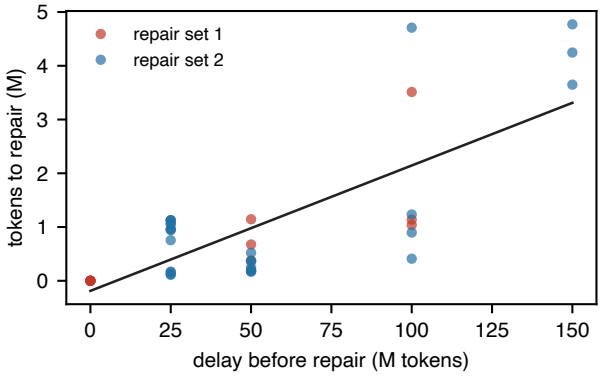



Figure 7: **Repair cost against delay.** Tokens to 80 percent of the repair's own gain against the unrelated tokens between formation and repair; 40 rescue curves from 7 parents with the pooled fit, and the three rescues along the mid-training stage. One of the two tests in the text is significant. `numbers/repair_debt.py`.

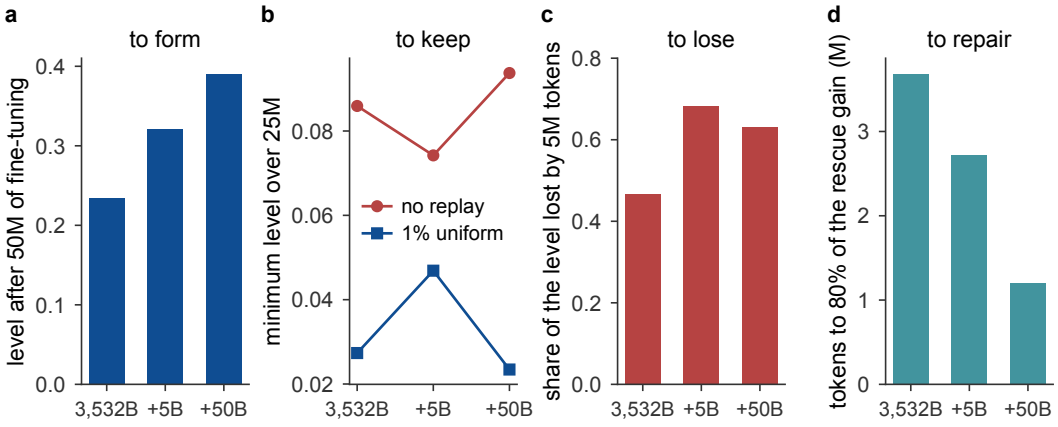



Figure 8: **The four quantities along the mid-training stage.** Checkpoints: the last stage-1 revision (3,532B tokens) and the revisions 5B and 50B tokens into mid-training. (a) Level after 50M fine-tuning tokens. (b) Minimum over 25M tokens with no replay and with one percent replay. (c) Share of the level lost in the first 5M tokens. (d) Tokens to 80 percent of the repair's gain. `numbers/ladder_score.py`.

multi-step relations (Figure 9a). Code output is the exception ($-0.37$, permutation $p = 0.046$). The math-domain loss has a partial correlation of 0.38 to 0.39 with math accuracy ($p \leq 0.04$), the opposite sign from the one a loss proxy assumes. None of these survives a correction for the 24 tests.

**Items keep changing late in stage 1.** Between adjacent stage-1 checkpoints, the median share of grammar items that change correctness falls from 0.19 over the first five transitions to 0.05 over the last five (Figure 9b). For code output about a tenth of the items still change at every checkpoint at the end of stage 1 (0.11), and for relational items it is 0.05 at the start and 0.06 at the end.

**The per-item pattern identifies the stage.** Per-item likelihood margins identify the stage of 106 of the 107 public OLMo-2-1B checkpoints at balanced accuracy 0.97, and which of three mid-training mixes produced each of 72 of the 73 mid-training checkpoints at 0.86 (chance 0.33, permutation $p = 0.001$; `data/v3/map_pack.json`). Figure 12 shows the same for position in training.

| platform | capability | to form | to keep | to lose | to repair |
|---|---|---|---|---|---|
| OLMo-2 7B | synthetic | stage-1 0.023 to 0.078, anneal 0.324 to 0.426 at 50M (n 4, 6) | 5% holds, 0.5 to 3% collapse (n 65) | fast share 0.665 (n 40) / 0.306 (n 9); $\tau$ 133.83M (n 9) | 4.33M to 80% of gain, all placements (n 28); +0.71M per 50M of age |
| OLMo-2 7B | GSM8K | installed by the anneal | zero 0.094 (n 3), uniform 1% 0.020 (n 3), end block 0.090 (n 7) | fast share 0.147 (n 3) | 0.000 to 0.122 in 8.39M from stage 1 |
| Qwen 1.5B | synthetic | 0.453 at 50M | zero 0.207 (n 1), uniform 1% 0.203 (n 4) | – | 3.69M, 4.16M (n 2) |
| Qwen 3B | synthetic | 0.566 at 50M | zero 0.332 (n 1), uniform 1% 0.344 (n 3) | – | (n 0) |
| Qwen 7B | synthetic | 0.703 at 50M | zero 0.178 (n 12), uniform 1% 0.148 (n 15) | – | 0.67M, 4.07M, 9.61M, 1.17M (n 4) |
| OLMo 13B | synthetic | 0.602 at 50M | zero 0.047 (n 1), uniform 1% 0.066 (n 1) | – | 1.84M, 2.59M, 3.01M (n 3) |

Table 5: **The four quantities across models and scales**, in tokens. `numbers/token_cost_table.py`.

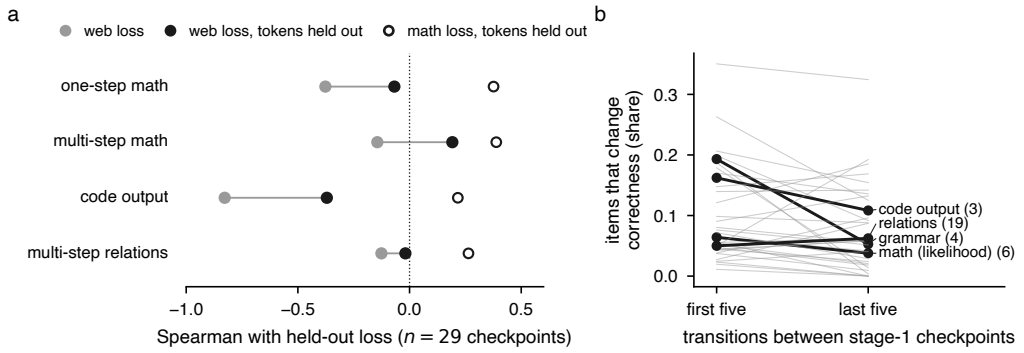



Figure 9: **The loss and the per-item state along OLMo-2-1B stage 1.** (a) Spearman correlation between likelihood-scored accuracy and held-out loss over stage-1 checkpoints from 84B tokens ($n = 29$): raw, and with log tokens held out (partial). (b) Share of items whose correctness changes between adjacent checkpoints, mean over the first and the last five transitions; grey lines are single item sets (number in brackets), black lines their medians. `data/v3/map_pack.json`, `data/v3/blimp_stability.json`.

**The onset is a switch, not a step.** Inside the onset window the trend is not a step but a switch. Scored every 1,000 steps on identical items behind the JSON instruction, one-step arithmetic reads 0.35 at 1452B, 0.04 at 1456B and 0.52 at 1464B. At the dip the model's most probable answer is the digit 1 on 152 of 200 items (full-vocabulary argmax accuracy 0.015), and the mean margin of the gold answer returns to its 110B value ($-3.15$ against $-3.23$). The gold is ranked first on 16 of 370 items, against 196 at 1464B (`numbers/acq_numbers.py`). The whole margin distribution moves, so the dip is not the accuracy-threshold noise of a low signal-to-noise benchmark. At the same checkpoint the program-output set behind the same instruction reads 0.74, so the failure is specific to one prompt family at that checkpoint: the format effect of Section 3, seen along pretraining. The bisection's program-arithmetic set switches in its own pattern (0.59 at 1527B, 0.22 at 1615B, 0.92 at 1791B). The 1B's bare program arithmetic ($n = 76$) rises from 0.09 to 0.70 between 1657B and 2748B and reads 0.18 at 4001B: it never settles. A ladder built from single public revisions near an onset samples a process that moves by fifty points inside 4B tokens; the early fine-tuning loss proxy of Section 4 is read on the parent directly, which is why it is needed.

**Grammar is kept for free.** The zero end of the same rule is observable on the public checkpoints. Grammar, which web text never contradicts, reads 0.392 at 1B tokens and 0.994 at 21B on the OLMo-2-1B run ($n = 1000$, native minimal pairs), and 0.997 at 4001B with 0 paired-significant steps of two points or more in between. On a second 1B model trained on a different corpus, grammar reads 0.396 to 0.988 over 400B tokens ($n = 1000$). A capability the stream agrees with has a cost to keep of zero over four trillion tokens.

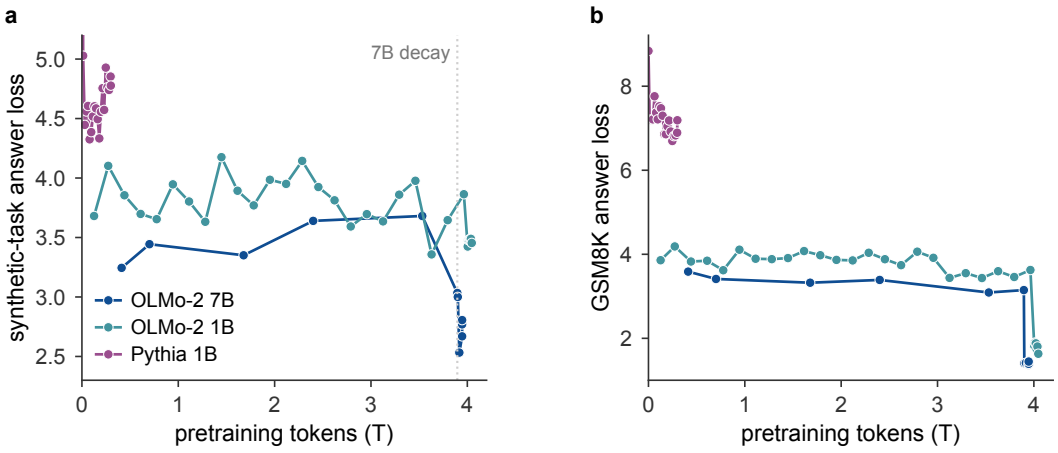



Figure 10: **Untouched-checkpoint answer loss along three pretraining runs.** Mean answer loss per token on 256 held-out items, with no fine-tuning, for the public revisions of OLMo-2-7B, OLMo-2-1B and Pythia-1B (Biderman et al., 2023): (a) the synthetic task, (b) the GSM8K answer-only pool. The 7B mid-training stage drops the synthetic-task loss to a band the 1B never reaches, which is where a 1B pilot sat on the floor; the GSM8K drop at 7B mid-training is the contaminated mid-training mix. `compute/xblock/readiness_meter_p4d.py`.

**Item identity transfers across the 7-fold scale gap.** On the 155 conditional-select program items scored with the bare prompt on both runs, the number of times an item flips between solved and unsolved across the 40 OLMo-2-1B checkpoints correlates with the first 7B checkpoint at which the item is solved (Spearman 0.48). Items the 7B solves at its first checkpoint, 110B, flip 4.2 times on the 1B ($n = 132$) and end at 1.00; the 23 it does not flip 10.9 times and end at 0.17. What is hard for the 7B is what is unstable for the 1B.

**The difficulty ordering is fixed by 84B tokens.** On the 1909 word-problem items of the 1B run, the Spearman correlation of the per-item gold margin between adjacent checkpoints has median 0.96 over 39 pairs. The ordering at 84B correlates with the ordering at 4001B at 0.93 (at 0B, $-0.09$). Pretraining sets which items are hard early and does not reorganise them; what moves afterwards is one position along that ordering.

**The format channel along a pretraining run.** The format cost is visible along a whole pretraining run without any training of ours. On the public OLMo-2-1B revisions, the same 155 program-output items are scored two ways at each of forty checkpoints. With the bare program as prompt, they read 0.77 at 42B tokens and stay between 0.75 and 0.90 from 126B to 4001B. With the instruction-wrapped prompt ("respond with only a JSON object"), they read between 0.17 and 0.89 over the same checkpoints. The share of items solved bare but not wrapped reaches 0.72 at 315B; the reverse never exceeds 0.09. The capability is in place by 126B and the wrapper withholds it intermittently for the remaining 3.9T tokens.

## F Likelihood proxies for the data choice, and what an intervention identifies

**Continuous proxies on the saved anneals.** We recompute the metrics of Magnusson et al. (2025) on the saved candidate log-likelihoods from the data-choice grid. This is an exploratory comparison on the same trained models, not a replication of their cross-scale prediction experiment. For item $i$, candidate $j$, and stored log-likelihood $\ell_{ij}$, let $q_{ij} = \exp(\ell_{ij}/d_{ij})$. We use $d_{ij} = 1$ for raw scores and the scored continuation's character count for character-normalized scores. The continuation includes the original closing JSON delimiter; the likelihood prompt and candidates are unchanged. Normalized $q_{ij}$ need not form a probability distribution over candidates.

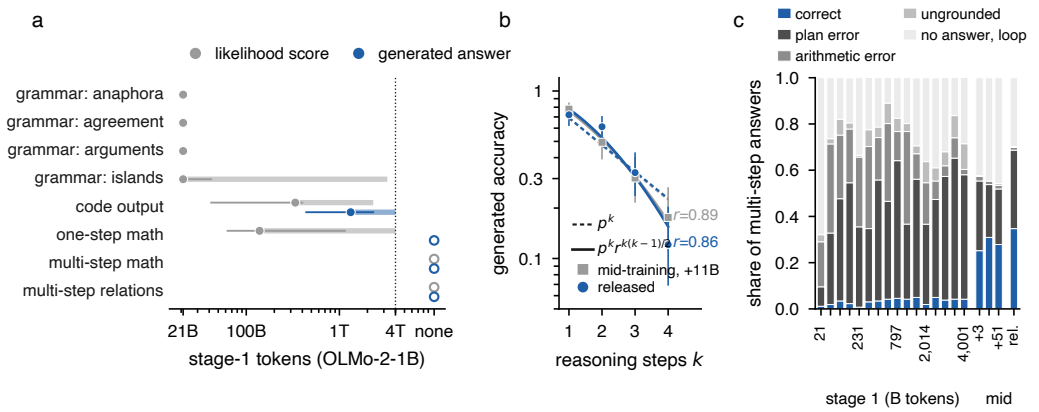



Figure 11: **Formation times by capability, and accuracy by number of steps.** OLMo-2-1B. (a) Stage-1 tokens at which each capability reaches half its stage-1 gain (dot; thin line, bootstrap 95% interval), with a bar to the point after which it stays above 90% of the gain; hollow markers at *none* have a gain inside two standard errors. Grey, likelihood score; blue, generated answers. (b) Generated accuracy on MathGAP by the number of reasoning steps at two mid-training checkpoints, Wilson 95% intervals, with fits of equal steps ($p^k$, dashed) and of steps that get harder ($p^k r^{k(k-1)/2}$, solid). (c) Outcome of each generated answer on items that need two to four steps ($n = 262$), by checkpoint.

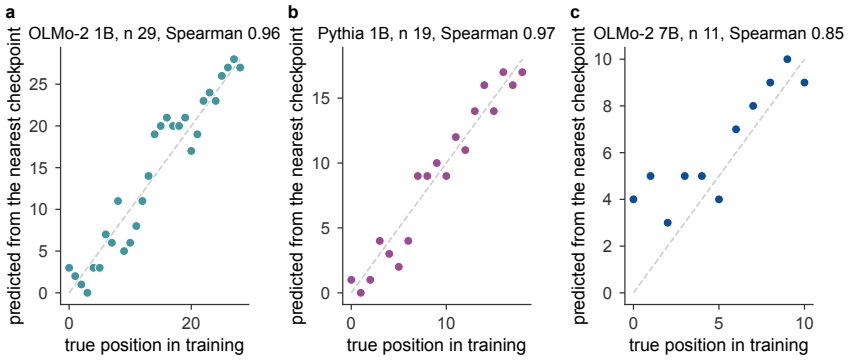



Figure 12: **Position in training from per-item scores.** For each checkpoint, the vector of gold log-likelihoods over 300 held-out items (GSM8K, TriviaQA, BoolQ (Clark et al., 2019)) is centred and normalised within the family; the checkpoint's position in training is predicted as the position of its nearest neighbour in correlation distance, leave-one-out. `numbers/clock_score.py`.

Correct Prob averages $q_{i,g_i}$, where $g_i$ is the gold candidate. Total Prob averages $\sum_j q_{ij}$. Norm Correct Prob averages $q_{i,g_i} / \sum_j q_{ij}$, and Margin averages $q_{i,g_i} - \max_{j \neq g_i} q_{ij}$. Accuracy averages the indicator that the gold candidate maximizes $q_{ij}$. Each metric is evaluated both raw and character-normalized. The saved accuracy decisions retain the evaluator's full precision; continuous metrics use the stored rounded log-likelihoods.

On one-step items, none of the 10 likelihood metrics prefers math data and 6 prefer code; none picks math among the five sources, and choosing by any of them costs 0.47 to 0.65 in generated accuracy against the best source. On multi-step items, 2 metrics (the two probability margins) prefer math and 7 prefer code. Continuous likelihood metrics therefore do not all behave alike, and the claim in Section 3 is about the metrics we list. The record contains all per-source means and every metric, including those whose intervals include zero.

**Low-rank control does not identify representation dimension.** The stopping interventions in Appendix D identify changes sufficient to alter a specified behavior at the patched site. They do not

| Score (math minus code) | One step | Two to four steps |
|---|---|---|
| Accuracy (raw) | -0.1279 [-0.2209, -0.0349] | -0.0038 [-0.0344, +0.0268] |
| Correct Prob (raw) | -0.0018 [-0.0032, +0.0001] | -0.0005 [-0.0006, -0.0004] |
| Total Prob (raw) | -0.0065 [-0.0086, -0.0041] | -0.0043 [-0.0046, -0.0039] |
| Norm Correct Prob (raw) | -0.0594 [-0.0958, -0.0247] | -0.0186 [-0.0234, -0.0140] |
| Margin (raw) | -0.0001 [-0.0016, +0.0018] | +0.0005 [+0.0003, +0.0006] |
| Accuracy (per char) | -0.0814 [-0.1628, +0.0000] | -0.0496 [-0.0840, -0.0153] |
| Correct Prob (per char) | -0.0260 [-0.0346, -0.0164] | -0.0279 [-0.0296, -0.0261] |
| Total Prob (per char) | -0.1515 [-0.1945, -0.1093] | -0.3440 [-0.3678, -0.3202] |
| Norm Correct Prob (per char) | -0.0179 [-0.0262, -0.0095] | -0.0085 [-0.0098, -0.0072] |
| Margin (per char) | -0.0024 [-0.0108, +0.0062] | +0.0076 [+0.0059, +0.0093] |
| Generated accuracy | +0.4651 [+0.3488, +0.5814] | +0.1718 [+0.1183, +0.2252] |

Table 6: **Likelihood metrics for the data choice, math minus code.** Math minus code at the same parent and annealing budget. Negative values favor code; positive values favor math. One-step items: $n = 86$; two-to-four-step items: $n = 262$. Brackets are unadjusted paired item-bootstrap intervals. Each data source has one training run, so these intervals do not measure variation across runs. Metrics have different scales and their magnitudes are not comparable. Generator: `numbers/proxy_choice.py`; record: `data/v3/proxy_choice.json`.

establish the dimension of the representation that originally computed the answer. For a scalar linear readout $z_i = w^\top h_i$ with $w \neq 0$, any desired change $\delta_i$ can be produced by $\Delta h_i = \delta_i w / \|w\|^2$. Every intervention lies in the span of $w$, regardless of the dimension of the unmodified $h_i$. This example shows why intervention rank alone cannot identify representation dimension. Our patches occur inside a nonlinear model, and their transfer across items and comparison with random subspaces provide additional behavioral evidence. They do not rule out all alternative internal computations, including the intervention illusions studied by Makelov et al. (2024).

## G How each claim is established

For each claim, the independent lines of evidence (different models, scales, designs or readouts), and for each alternative explanation the result that rules it out or bounds it.

| claim | independent lines | alternative → what rules it out or bounds it |
|---|---|---|
| Likelihood scoring misses multi-step skill that generation shows (§3) | 1B mid-training, three mixes, 10 checkpoints; 7B stage 1, four checkpoints; Open-1B, another group's run; 1B cooldown on own data; checkpoint choice over 18 to 20 checkpoints | likelihood scoring is broken → it agrees with generation on 0.93 of code pairs and 0.89 of one-step pairs; answer extraction fails → checkpoints below half extraction dropped; the scoring rule → holds under the lenient rule and wrapped prompts; contamination → the toy-car template is excluded |
| Much of a measured change is answer format (§3) | eight same-model, same-item pairs at 1B and 7B across forks, fine-tuning, cooldowns and public checkpoints (Table 2); token classes on forks at 200M; answer line by 84B | the capability itself changed → the right-number rate stays near the parent's (0.113 against 0.133), under a five-shot prompt every fork stops after its answer, the newline class carries 0.951 of the drift and arithmetic 0.039 |
| The cost to form differs by checkpoint at a fixed recipe (§4) | 7B ladder, 9 of 9 with seed replicates; GSM8K at the boundary, two seeds, and at 13B (gap 0.212, 0.164 to 0.262); early fine-tuning loss over 20 runs, four models; 1B cooldowns at seven parents, four on a second data draw | bf16 rounding → flat across the boundary GSM8K uses; the learning rate → at 1B a tuned rate removes most of the age effect, while at 7B on GSM8K the best of three rates per checkpoint leaves 0.248 of the 0.274 difference; answer format → extra training on a different task in the same answer template closes about a third, and web text of the same length reaches 0.127 against 0.170 with the template and 0.084 without warm-up (two seeds each); contamination → 85 percent of the gap survives GSM-Plus; a pre-existing level → on existing capabilities the later stage raises the starting level, not the rate (Appendix D) |
| Placement, not dose, keeps a capability; one percent leaves deeper troughs than none (§5) | 13 of 13 non-tied parents on the minimum, 12 of 14 on the mean, 10 of 14 on the final level; three more parents; three revisions; held-out prediction over 284 forks; spaced replay at 20 percent in Atreya et al. (2026) | the harness's in-line evaluation → the synthetic task's final level reproduces offline; draw-to-draw noise → 26 of 30 one-percent draws fall below the no-replay band; generality → opposite sign on Qwen2.5, stated as a boundary |
| Without replay a $\beta_2$ change separates survival, through the step size; format goes first (§5) | 3 of 3 against 3 of 3 over 200M; at $4 \times 10^{-5}$ both lost (one run each); fast share 0.67 against 0.17 under $\beta_2$ 0.999; stop-rule scorer offline | a capacity trade → the background task saturates identically in every run; a smaller effective step → present: the inherited bias correction makes the 0.999 run's steps 1.77 times smaller, and in runs with replay the step alone gives 0.262 to 0.293 against 0.164 to 0.184 for the frozen normaliser alone (Table 4); in the no-replay runs, 0.95 at the matched rate keeps it too (one run, minimum 0.270); the second moment's memory horizon → refuted by the flat ladder from 0.95 to 0.99 |
| Repair reaches one level; whether its cost grows with delay is unresolved (§D) | a test over 10 settings, 3 parents; 40 curves | effect size → the replication on the second set of repair curves did not reach significance, so we report the slope of the first test |
| Mid-training's token efficiency is capability-specific (§4) | three mid-training mixes; Open-1B | a stage-1 rate of zero → only lower bounds are reported for math |

## H UNEXPECTED RESULTS, AND HOW WE EXPLAIN THEM

Several results run against what a reader expects. For each we give the explanation we favour, say whether it was tested (an intervention that could have come out the other way), only consistent with the data, or untested, and name the observation that would falsify it. Status matters more than the story; an untested explanation is a hypothesis for the next experiment, not a finding.

**Likelihood scoring is flat on multi-step items while generation rises.** *Explanation.* Likelihood scoring ranks a final answer with no written steps, so every combination of quantities must happen inside one forward pass; under standard complexity assumptions, bounded-depth transformers cannot compute such compositions of growing length in one pass and can when they write the intermediate steps (Feng et al., 2023; Merrill & Sabharwal, 2024). *Status: consistent, and it predicts the pattern across step counts.* On one-step items, which need no composition, likelihood scoring does rise (0.128 to 0.477 over OLMo-2-1B mid-training) and agrees with generation on 0.89 of decisive pairs; at two and three steps agreement falls to 0.34 and 0.50. On relational items likelihood accuracy is above generated accuracy at all 14 checkpoints of the 1B data comparison, and there the generated answers lose on format: the answer line appears on 0.70 to 0.76 of relational items at the 4,001B parent and on 0.33 to 0.35 at the released model. A likelihood score cannot fail on format, so on those items it reads higher. On code items that need two or three steps likelihood accuracy is also higher at all 14, and format does not explain it: the answer line appears on 0.81 to 0.91 of those items at the parent. On one-step code items the two scores are within 0.012 at every checkpoint where generation leads. *Falsifier.* Likelihood scoring rising with generation on multi-step items when the prompt contains the model's own written steps with the final answer removed.

**One percent of own-data replay keeps less than none.** *Explanation.* Each replay batch is a full-size step on a capability that sits near its margin; the step overshoots, and the level after each refresh settles below where it started. *Status: the pieces are measured; the account is not tested as a whole.* In densely evaluated forks each replay event raises the level by 0.13 to 0.18 and the rise decays over about 10 to 25 updates, while the level before successive events is 0.141, 0.129, 0.066 and 0.074. The harm needs the capability's own gradient: the same schedule with other data in the replay slots lands in the no-replay band (0.113 to 0.137 against 0.012 to 0.051 for own data; Appendix D). At a smaller step the one-percent forks keep 0.395 to 0.398 ($5 \times 10^{-6}$; Table 4), above the no-replay median at the standard rate (0.133), and a parent far above its margin reverses it (Qwen2.5-7B, 0.395 against 0.301). *Falsifier.* The deficit persisting at a small learning rate with no replay run at that rate as the control, which we did not run.

**$\beta_2$ does nothing from 0.95 to 0.99 and everything at 0.999.** *Explanation.* Two properties of these forks change only at 0.999. The forks inherit the parent's step counter, $t = 382$; the bias correction $\sqrt{1/(1-\beta_2^t)}$ is 1.000 at 0.95, 0.97 and 0.98, 1.011 at 0.99 and 1.77 at 0.999. And in bfloat16, multiplying a second-moment entry by 0.999 returns the same value for all 32512 positive normal numbers, while 0.99 and 0.95 lower every one of them. *Status: the step part is tested (Table 4; without replay, 0.95 at the matched rate keeps the capability, minimum 0.270 in one run; at twice the learning rate, 0.999 loses the capability too, ending at 0.066 in one run); the bfloat16 part is untested.* A memory-horizon account, $1/(1-\beta_2)$, predicts a graded effect from 0.95 to 0.99 and is refuted by the flat ladder. *Falsifier.* A $\beta_2 = 0.999$ run with the bias correction neutralised and float32 optimizer states that still keeps the capability as well as the plain 0.999 run.

**Two thirds are lost in the first 5M tokens.** *Explanation.* 5M tokens are 38 updates at our batch size. The event-locked component measured above decays over about 10 to 25 updates (1.3M to 3.3M tokens), so a capability that rests largely on that fast component loses it within the first 5M tokens and keeps only the slow part, which decays with a time constant of about 132M tokens. *Status: consistent.* A capability formed over 200M tokens loses 0.33 rather than 0.67, as a larger slow part predicts. *Falsifier.* The same 5M-token loss under web text on a capability formed by pretraining rather than by fine-tuning at the same rate; most of our forks use a second synthetic task, so two-task interference is not excluded.

**What is lost first is the stop token, not the answer.** *Explanation.* The background stream teaches what follows a number, and web text almost never ends a line after a bare number. In one billion stream tokens, the GSM8K final-answer marker is followed by a number and a stop 0 of 3 times, against 7,470 of 7,470 in GSM8K training solutions; after a number, the stream continues with a word 0.329 of the time. *Status: the counts are measured; the causal link is consistent with the located change (an early-layer change, repaired by one update to layers 0 to 7) but not tested by editing the stream. Falsifier.* A background stream with the successor statistics of the solutions that still removes the stop token.

**Weight distance cannot tell the runs apart; a hundred items can.** *Explanation.* The forks share almost every batch, so their displacement from the parent is dominated by the shared stream; the condition-specific change is small in norm; for the stop token, a rank-eight reconstruction of it in the early residual stream reproduces the effect (Section 6). Mean weight distance between runs of the same condition and of different conditions differs by 0.19 to 0.29 percent of the mean, while the per-item readout reads the model's answers directly. *Status: consistent. Falsifier.* A weight-space distance restricted to layers 0 to 7 that identifies the condition as well as the per-item readout does.

**Stage-1 7B checkpoints do not learn the new task at our recipe.** *Explanation.* Stage-1 checkpoints do not fit their own fine-tuning pool (final loss 0.59 to 0.70 against 0.32 to 0.37 for mid-training checkpoints), so the difference lies in optimisation, and Section 4 takes it apart into the step size, the answer template and a head start. *Status: partly tested.* At 1B a larger learning rate ($4 \times 10^{-5}$, the best for every stage-1 checkpoint) removes most of the gap; at 7B on GSM8K the best of three rates leaves 0.248 of the 0.274 difference; the template closes 0.36 of it. Rounding in bfloat16 does not explain the boundary: the share of weights small enough to move is 0.3237 at the last stage-1 checkpoint and 0.3242 after 5B mid-training tokens. *Falsifier.* A stage-1 7B checkpoint that fits its fine-tuning pool as well as a mid-training checkpoint and still does not reach 0.3.

**The held-out web loss rises while the multi-step skill forms.** *Explanation.* Mid-training draws on a different mix of sources than stage 1 (OLMo Team, 2025), so loss on stage-1 web text can rise while the model improves on what the new mix contains. *Status: consistent; not tested.* It is why the loss curve in Figure 1 cannot be used to find the onset.

# I PROVENANCE OF THE NUMBERS IN THIS DRAFT

Every number in the main text is either a macro from `numbers/numbers.tex`, rebuilt from run records by the scripts in `numbers/`, or a literal that names its file here. Literals: the stage ladder and dose ladder, `data/tokencost/levers.json`; formation curves, `data/curves7b/UNIQUE_INDEX.json` (17 unique of 33 files); rescue curves, the same index (37 unique of 176 files); the repetition ladder, `data/tokencost/repetition/` and `numbers/repetition_ladder.py`; the GSM8K boundary gate, `data/x3c/x3c_score.json`; post-trainability in updates, record R1; the five-shot interference numbers, `data/perexample/t21_gsm8k5_25M_50M.json` and `t44_tradeoff/results.json`; the retained fraction by target, record R2; the token-class decomposition, record R3; the Qwen scale ladder, `data/perexample/t14_invisibility.json`; the integrity discrepancy, `data/r2/r2_probes/`.

**Files for the v3 main text.** Table 2: `data/termination.json`, `data/x3/x3_score.json`, `data/x3c/x3c_score.json`, `numbers/acq_numbers.py`, `data/v3/cool_score_three.json`, `data/v3/measure_pack.json`, `t44_tradeoff/results.json`, `data/v3/judge_real_analysis.json`. Own against other-data replay, the per-parent one-percent deficit and the GSM-Plus and SVAMP shares: `analysis_timing/timing_numbers.py`, from `data/tokencost/records_all.csv`, `data/gsmplus/contamination_score.json` and `data/x3c/x3c_score.json`. The trace-onset comparison reads `data/v3/trajectories.json` through `numbers/v3_numbers.py`. Figure 6 reads `data/xblock/x1/x1_score.json` through `figures/make_fig_v4_mechanism.py`. Section 6 literals: the condition control, record R4; the item-level transplant, `data/actpatch_items.json` (`cells.*.rho_own_vs_patched`, `null_p95`). The $\beta_2$ decomposition (Table 4 and the `\nOpt*` macros): `numbers/optstep_numbers.py`, from `data/tokencost/records_all.csv`. Figure 1: `data/v3/atlas.json`, `data/v3/mathgap_clean.json`. The 1B grid readouts cited in Section 3 and Appendix H: `analysis_timing/grid/grid.json`. Appendix H literals: record R5 (event-locked rises and pre-event levels), record R6 (successor counts), `data/perexample/t37_weightblind.json` (weight distances), record R7 (fine-tuning pool loss). The two-step example in Section 3 is item `mathgap-depth2-00454` of the MathGAP pool (gold answer 12). The doubled-rate $\beta_2$ runs (Section 5): `data/v3/b2_lr4e5_zero999_8131_run_record.json` and `data/v3/b2_lr4e5_zero95_8113_run_record.json` (field `curve`, `U_target` and `U_bg`; kept if the minimum is at least 0.15, lost if the final value is below 0.08). Appendix A, 65.5M likelihood interval: `data/final_pass/D_inversion.json` (`code_minus_math_depth1`). The single best 1B learning rate: `data/final_pass/J_lr.json`. The 1.05B data choice: `analysis_timing/chains/j3.json` (`research/compute/iclr_final/svamp_hf/svamp_hf.py` and `svamp_j3.py`) and `analysis_timing/grid/grid.json`. The matched-rate zero-replay run (Section 5): `data/v3/b2_lr113e6_zero95_8113_run_record.json` (field `curve`, `U_target`). All other v3 numbers: `numbers/v3_numbers.py`. Additional reanalyses (Sections 3 and 5): `data/review_2026-09-25/`, macros from `review_numbers.py`. Records R1 to R8 are released with the code as run notes. Section 4 learning-rate control at 412B (0.14, 0.04): `data/tokencost/records_all.csv`, runs `k2_lr5e5_412`, `k2_lr5e5_412_s5` and `k2_lr1e4_412`. Abstract: 0.47 and 24M–42M are the Section 3 math gain and the Section 4 token counts, rounded to two figures. Section 3 run-noise literals: 0.06 is the math gain on its two data draws (`analysis_timing/grid/grid.json`), 0.12 is the 4,001B cooldown on its two draws (`numbers/v3_numbers.tex`), and 0.13 is the Table 6 accuracy difference. The stop-token pair in Section 6: `data/v3/example_stop_token.json` (item `355d5cfb0af079e4`, the parent and the no-replay fork at 150M, copied verbatim from the stored generations). The towel trace in Section 4: `data/v3/example_trace_4001B.json` (item `mathgap-depth2-00175`, copied verbatim from the stored generations of the 4,001B checkpoint). The synthetic item in Appendix D is held-out item `001717384fb04b16` of the

released synthetic task pool; the bfloat16 unchanged shares are in `data/insight_bf16_cutoff.json`.

**Figure data.** Figure 1: `data/v3/atlas.json`, `mathgap_clean.json`. Figure 2: `data/v3/fa7b_clean.json`, `claims.json`, `measure_pack.json`, `analysis_timing/grid/grid.json`. Figure 11: `data/v3/map_pack.json`, `analysis_timing/compose/compose.json`, `data/v3/mathgap_clean.json`. Figure 3: `analysis_timing/cool/cool_score_auto_0924_1825.json`, `analysis_timing/teach/teach.json`, `data/tokencost/readiness_regression.json`. Figure 4: `analysis_timing/fall/fall.json`, `data/tokencost/records_all.csv`, `data/curves7b/all/`.

**Section 6 literals.** Layer-restricted repair, 0.113 to 0.125 after one update on layers 0 to 7: record R8. Condition control at 100M: record R4. Item-level transplant: `data/actpatch_items.json`.