# PREDICTING AND STEERING THE EMERGENCE OF CAPABILITIES IN NEURAL NETWORKS

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

## ABSTRACT

Neural networks often acquire capabilities abruptly during training, from in-context learning to arithmetic and multi-step reasoning. Scaling laws describe when a capability emerges on average, but not whether a particular run will acquire it, or whether that outcome can still be changed. Here, we introduce the training committor, the probability that a capability emerges within a fixed horizon from a given checkpoint, which we estimate by restarting training from the checkpoint many times. In small transformers and in a language model trained on natural text, we find that most runs are effectively decided before the capability appears, and that the committor predicts which runs will acquire it. Notably, on the controlled tasks, it is more accurate than predictors based on loss or weight statistics. Beyond prediction, the committor reveals where to intervene: a short burst of training data aligned with the capability can rescue nearly all of the runs it marks as unlikely to succeed. Since its branches can be trained under any intervention, the committor can also compare interventions on the same run and pick the most effective one. Overall, our results establish capability emergence as a predictable and steerable property of the training state.

## 1 INTRODUCTION

Neural networks can suddenly gain capabilities that are fundamental to what a model can do. For example, a language model can be trained for thousands of steps with no sign of in-context learning and then acquire it over a short window as induction circuits form (Olsson et al., 2022; Chan et al., 2022). Moreover, a small transformer on modular arithmetic can memorize its training set long before it generalizes (Power et al., 2022; Nanda et al., 2023), and across model scale whole task families move from chance to competence within a narrow range (Wei et al., 2022). As a result, the eventual outcome of a partly trained run is difficult to predict, since a run that has not yet acquired the capability may still do so before its budget runs out, and its training curves can look the same regardless of whether it eventually does.

Existing work on predicting emergence mainly focuses on populations of runs. Scaling laws describe average loss as a function of model size, data and compute (Kaplan et al., 2020; Hoffmann et al., 2022), and emergence laws extend this to the point at which a capability appears across a family of models (Snell et al., 2024). Within a single run, current precursors of a transition, such as loss dynamics, curvature and circuit formation (Tikeng Notsawo et al., 2023; Xu, 2026; Sivasankar, 2026), are each read from the one trajectory that was observed. However, deciding whether to keep training a run or to intervene on it requires something else: the probability that this saved run acquires the capability within its remaining budget, and, crucially, whether that probability can still be changed. Behind this practical question lies a scientific one: when during training is a run's outcome decided, and what in the saved state decides it?

Molecular simulation answers a surprisingly similar question for rare chemical transitions with the committor, the probability that a trajectory started from a given configuration reaches the product state or, in the finite-time version, reaches it before a deadline (Helfmann et al., 2020; Lucente et al., 2019). Chemists estimate it by launching about a hundred trajectories from the configuration and counting how many reach the product. The committor is useful because it measures how far a reaction has truly progressed: the states from which the system is equally likely to go either way define the transition state, and cheaper reaction coordinates are judged by how well they agree with



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.

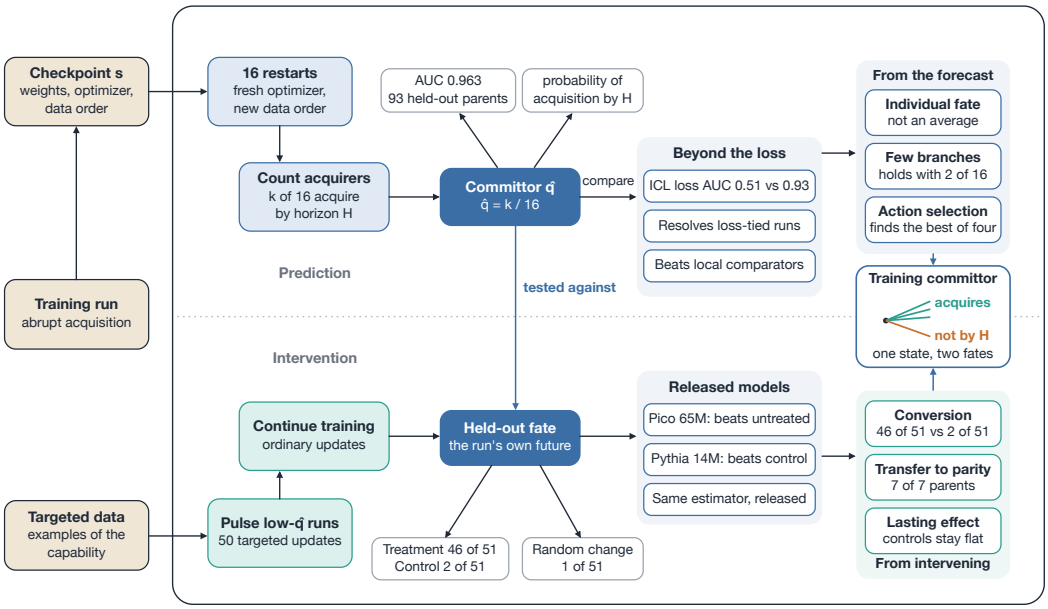


Figure 1: **The training committor links prediction to intervention.** Upper track: a saved state, the parent, is branched into sixteen reseeded restarts, whose acquiring fraction $\widehat{q}$ estimates the committor $q$ by horizon $H$, and the forecast is scored on the parent's own continuation (Section 4.1). Lower track: targeted data injected into low-$q$ states that do not acquire on their own leads them to acquire (Section 4.3), and the same estimator, run under each candidate action, compares the actions (Section 4.4). ICL, in-context learning.

it (Bolhuis et al., 2002; Peters & Trout, 2006). Since a training checkpoint holds everything needed to continue training, it can be treated in exactly the same way: the fraction of its continuations that acquire the capability estimates a probability we call the training committor. To our knowledge, this is the first work to estimate this probability for a training checkpoint, test it on a held-out continuation, and use it to intervene.

We study the training committor in small transformers trained on in-context learning, modular addition and permutation composition, and in a language model trained on natural text. From each saved run, the parent, we launch sixteen continuations with a fresh optimizer and a reseeded data order and record the fraction that acquire the capability (Figure 1). We then score this forecast on the parent's own continuation, which keeps the original optimizer state and data order and which the estimator never sees. Continuing the parent once would reveal only the one future it happens to take. Instead, the committor measures the spread of outcomes that the saved state still allows, and because its continuations can be run under any intervention, it can also compare interventions on the same state. Like committor estimates in chemistry and ensemble weather forecasts (Molteni et al., 1996), it spends compute on many futures of one state, and it serves as an instrument for studying and steering training rather than as a lightweight monitor (Section 4.2).

This instrument reveals that most runs are settled early and invisibly: for most parents all sixteen continuations agree, and in in-context learning this is already true at step 100 of a 10,000-step run, while the runs whose continuations all succeed are still well below the accuracy that defines the capability. The few parents whose continuations disagree are the runs still open, where the choice to continue, stop or intervene matters most. The forecast predicts which parents acquire the capability in all three task families, and in modular addition it even separates runs whose loss and weight statistics look alike. Because the continuations keep only the saved weights and the training clock, the weights must already carry information about the outcome beyond what standard training statistics show, and the committor reads it out. Although each forecast is expensive, it can be made much cheaper: a logistic readout of checkpoint behavior settles the clear cases, and restarting only the half of the parents it is least sure of matches the full estimate (Section 4.2).

However, a settled run can still be changed, and the forecast reveals where to intervene. In in-context learning, low-forecast parents that fail on their own respond to a short burst of training data aligned with the capability, which we call a pulse: it leads most of them to acquire it, while ordinary training and a random weight change of equal size almost never do. Parents with an intermediate forecast need a much shorter pulse than those with a zero forecast, so the forecast distinguishes unlucky runs from unlikely ones and sets how much help each one needs.

Our contributions are as follows:

- We define the training committor, the probability that a checkpoint acquires a capability within a fixed horizon, and estimate it from reseeded restarts (Section 3).
- Across 93 held-out parents from three task families, the forecast reaches AUC 0.963, higher than every checkpoint-local predictor that our saved records support (Section 4.1).
- A short aligned pulse converts 46 of 51 low-forecast in-context-learning parents that do not acquire on their own, against 2 of 51 under ordinary training, and in in-context learning and sparse parity alike the effect depends on alignment with the target (Section 4.3).
- Before any outcome is known, the committor chooses which runs to treat and identifies the most effective of four actions compared on the same state, and in a replay of recorded outcomes it sets the pulse length run by run (Sections 4.3 and 4.4).
- On natural text the forecast reaches AUC 0.864, and a targeted pulse moves a released Pythia checkpoint far past a matched control (Section 4.5).

## 2 Related Work

Our estimator comes from the study of rare transitions in molecular systems. In transition-path theory, the committor is the probability that a trajectory from a given state reaches the product basin before the reactant basin (Bolhuis et al., 2002; E & Vanden-Eijnden, 2006; 2010; Metzner et al., 2009), and it is widely regarded as the ideal measure of progress along a reaction. In high dimensions it is estimated from sampled paths, with trajectory models, or with neural solvers (Prinz et al., 2011; Husic & Pande, 2018; Mardt et al., 2018; Khoo et al., 2019; Li et al., 2019), and finite-time and control variants add a deadline or condition on the path (Helfmann et al., 2020; Lucente et al., 2019; Yuan et al., 2024; Du et al., 2026). Since a training run ends at a fixed budget, we use the finite-time form, and splitting methods, which restart only from states that reach intermediate interfaces (Allen et al., 2005; 2009; Cérou & Guyader, 2007), suggest how its estimate could be made cheaper (Section 5).

Restarting from a saved checkpoint is likewise a well-established way to study what a training state has already fixed. Frankle et al. (2020) and Fort et al. (2020) spawn copies of one checkpoint under different minibatch orders and find that the basin a run ends in is determined early in training, and Altıntaş et al. (2025) show that tiny early perturbations can send otherwise identical runs to different minima. Latent-state models, multi-seed studies and released training suites document how much transitions vary across seeds (Hu et al., 2023; Zhao et al., 2025; Ootani, 2026; Biderman et al., 2023; van der Wal et al., 2025; Magnusson et al., 2025). We use the same spawning step to ask whether one particular run will acquire a capability, and we check each answer against that run's own future.

When capabilities appear has mostly been studied across models rather than within a run. Scaling laws predict loss from model size, data, and compute (Kaplan et al., 2020; Hoffmann et al., 2022), and later work extends them to capabilities, for instance by measuring pass rates finely enough to reveal smooth progress (Hu et al., 2024), by fitting trends across model families (Ruan et al., 2024), or by linking emergence to a threshold in pretraining loss (Du et al., 2024), to the order in which discrete skills are learned (Michaud et al., 2023), or to the effect of finetuning on its onset (Snell et al., 2024). Whether the sharp jumps are artifacts of the metric remains debated (Wei et al., 2022; Schaeffer et al., 2023). However, these laws describe populations of models or runs, and individual runs can differ widely. In sparse parity, for example, whether a run succeeds depends on the luck of its initialization, which wider networks improve by searching more neurons in parallel (Edelman et al., 2023). Work on single runs has therefore looked for early warnings of a transition, mostly in grokking (Power et al., 2022; Nanda et al., 2023), reading them from the loss, through its early oscillations (Tikeng Notsawo et al., 2023) and the curvature of its landscape (Xu, 2026), and from inside the network, through hidden progress in the weights (Barak et al., 2022), synchronizing

circuits (Sivasankar, 2026), collapsing spectral entropy (Khanh et al., 2026), nucleation rates (Dong, 2026), and the local learning coefficient, a measure of effective model complexity (Lau et al., 2025). Each of these precursors, however, is read from the one trajectory that was observed, and so it cannot say how likely the other futures of the same state are. Our committor measures exactly this probability, and it ranks outcomes better than every precursor that our records support (Section 4.1).

Beyond predicting how training will go, many methods change it, for example by adding weight differences learned on other tasks (Ilharco et al., 2023), amplifying slow gradient components to accelerate grokking (Lee et al., 2024), swapping weight magnitude and direction between trajectories (Khanh, 2026), imposing representational priors that shorten the grokking delay (Howe, 2026), or rewriting the training data (Nadkarni et al., 2026). Interventions have also been used to uncover mechanisms. Singh et al. (2024) clamp activations during training to find the subcircuits that form an induction head and set its timing, and Raventós et al. (2023) find that in-context regression generalizes to unseen tasks only once pretraining covers enough distinct tasks, a finding our pulse builds on by adding unseen tasks to the training of a saved run (Section 4.3). Most related to our work, Xu et al. (2025) accelerate grokking by initializing a model with a weaker model's embedding, and in concurrent work Lin (2026) finds that the effect of short weight-decay perturbations across a grokking plateau becomes ordered by dose before generalization appears. We instead use the committor to decide, for each saved run, whether it needs help, which intervention to apply, and how much of it.

# 3 Method

## 3.1 The training committor

We begin by defining the training committor. A saved state $s$ contains the model parameters, the optimizer state, the training clock, and the data-loader state. Because one state can have different futures under different continuation rules, we make the continuation protocol part of the definition. For a capability event $E$, such as held-out accuracy crossing a threshold and staying above it, a horizon $H$, and a continuation protocol $\Pi$, the training committor is

$$q_{H,\Pi}(s) = \Pr_{\Pi}(E \text{ occurs within } H \mid S_0 = s)\,. \tag{1}$$

Here $S_0$ is the state at which the continuation starts, and $q_{H,\Pi}$ is the finite-time committor of transition-path theory (Helfmann et al., 2020; Lucente et al., 2019) applied to a training checkpoint: in place of a competing basin, the alternative to acquiring is not acquiring by the deadline, so $q_{H,\Pi}$ is a finite-horizon hitting probability. We set horizons on separate cohorts so that outcomes depend mainly on whether a run acquires rather than when (Appendix A). For the forecasts, $\Pi$ randomizes the post-checkpoint data order while fixing the architecture, data, objective, precision, event, and horizon. We use two such protocols, reset and carry. Both restore the weights and the training clock, and carry also keeps the optimizer moments, which reset rebuilds from scratch (Appendix A).

## 3.2 Estimation and a held-out test

We call a saved run whose outcome we forecast a parent, and each restart from it a branch. From a parent state $s_i$ we launch $B$ branches under $\Pi$, each with an independently reseeded data order, write $Y_{ib} = 1$ if branch $b$ reaches $E$ within $H$, and estimate

$$\widehat{q}_{H,\Pi}(s_i) = \frac{1}{B}\sum_{b=1}^{B} Y_{ib}. \tag{2}$$

We call $\widehat{q}$ the restart estimate and use $B = 16$ branches per parent; Appendix E.4 relates it to temporal-difference methods.

To use the estimate as a forecast, we score $\widehat{q}_{H,\text{reset}}(s)$ against $Y_H^{\text{own}}(s)$, the event indicator on the parent's own continuation, which carries the original AdamW and data-order state to the same endpoint and which no branch observes. Because the branches and the continuation follow different protocols, nothing guarantees that one predicts the other. We take reset as the primary protocol because it asks how much of the outcome the weights alone carry. Carry keeps more of the parent's state, and its forecasts come closer to the parents' own outcomes (Section 4.1).

The three task families each show an abrupt, well-studied transition in a small transformer trained from scratch, so the event is sharp and its mechanism is understood. In in-context learning the model is a single transformer block, each episode shows eight labeled examples of a mapping followed by a query, and the model acquires the capability when its accuracy reaches 0.90 on held-out mappings, which it can infer only from the episode. In modular addition and permutation composition in $S_5$, the model first memorizes its training pairs and later generalizes, and acquisition means training accuracy of at least 0.99 and test accuracy of at least 0.95. The controlled panel pools five cohorts and contains 93 parent checkpoints, 72 of them from in-context learning, drawn from 86 initialization clusters, groups of parents that share an initialization, and 31 of the 93 parents acquire the capability. The parent is the independent unit, and intervals resample whole initialization clusters (Appendix A). Section 4.3 also uses sparse parity, in which the target is the parity of three fixed input bits (Appendix D.3).

# 4 EXPERIMENTS

We ask three questions: whether the committor predicts held-out outcomes better than existing predictors (Sections 4.1 and 4.2), whether the runs it marks as unlikely can be steered and chosen for treatment (Sections 4.3 and 4.4), and whether the findings carry over to natural text and released models (Section 4.5). Forecasts use sixteen reset branches, and intervals are 95 percent whole-cluster bootstrap intervals (Appendix B).

## 4.1 PREDICTING HELD-OUT ACQUISITION

We first test whether the forecast predicts each parent's own outcome (Figure 2a). Across the 93 held-out parents from five controlled cohorts, 31 of which acquire, the forecast reaches AUC 0.963 and a Brier score of 0.046. It separates the two outcomes perfectly in four of the five cohorts, and within cohorts an exact permutation test that keeps clusters whole finds the ordering far beyond chance (Table 4, Appendix B).

Most saved states are already settled: all sixteen branches agree, so the estimate is exactly 0 or 1. Of the 56 forecasts of exactly zero, 2 acquire, and all 19 forecasts of one acquire, so a zero forecast means very unlikely rather than impossible. The two exceptions trace back to the optimizer state, which the parent keeps and reset branches discard, and carry branches, which keep it, close the gap (Appendix B.2). The other 18 parents, whose branches disagree, are the runs still open, and even among them the forecast orders outcomes with AUC 0.975. Nor does the forecast simply read off visible skill: in-context-learning parents with forecast 1 are still well below the 0.90 accuracy bar at the checkpoint, and some forecast-0 parents that never acquire already score higher than they do.

In practice, runs are judged by their loss and weight statistics, so a sharper test is whether the forecast separates runs that these statistics cannot tell apart. We test this on a separate cohort of 32 modular-addition states from 16 initializations. Without using outcomes, we divide the space of 18 weight and loss statistics into 24 cells. Two cells contain both outcomes, and within these cells the forecast orders all 117 pairs of states with opposite outcomes correctly, with no ties. Swapping outcomes within the nine sibling pairs that share an initialization but differ in outcome gives a one-sided probability of $1/512$, the smallest the design allows, so the forecast distinguishes runs that start from the same initialization (Appendix B.3).

A natural next question is whether an existing predictor, read from the saved snapshot alone, could do the same job. Of the 100 emergence predictors we surveyed, our saved records support eight, and we implemented all eight and checked each against its own positive control (Appendix C). The committor outperforms every one of them (Table 1): it has the higher AUC in all 23 paired comparisons, and in 11 the interval excludes zero. The strongest of them, attention-entropy collapse (a fall in the entropy of the attention weights), still trails the committor clearly. LC-PFN, a network pretrained to extrapolate learning curves (Adriaensen et al., 2023), is near perfect on its own prior but at chance on the full panel. Trajectory models do no better: the best of 112 Markov-state models built on the recorded modular-addition observables, an optimistic maximum selected on the outcome, reaches AUC 0.688 against 1.0 for the committor (Appendix C).

Table 1: **Checkpoint-local comparators against the committor on the same parents.** Gaps are paired committor-minus-comparator AUC estimates with 95 percent whole-cluster bootstrap intervals; positive values favor the committor, and a paired gap can differ slightly from the difference of the printed AUCs. Blue marks a gap whose interval excludes zero, and orange marks a comparator at or below chance. The in-prior LC-PFN row is a positive control on the model's own synthetic curves, and the loss rows come from a separate 104-state study (72 ICL and 32 modular-addition states) on which the committor is rescored. Best of 13 observables is the strongest single recorded statistic. ICL, in-context learning. LOO, leave-one-out.

| Comparator | Panel | Comparator AUC | Committor AUC | Paired AUC gap |
|---|---|---|---|---|
| Attention entropy | 93 | 0.794 | **0.963** | +0.169 [0.071, 0.286] |
| LC-PFN, zero-shot | 93 | 0.421 [0.271, 0.595] | **0.963** | +0.542 [0.344, 0.710] |
| Loss LOO, pooled | 104 | 0.691 | **0.967** | +0.277 [0.160, 0.401] |
| Loss LOO, ICL states | 72 | 0.511 | **0.927** | +0.419 [0.210, 0.619] |
| Loss LOO, modular states | 32 | 0.413 | **1.000** | +0.587 [0.380, 0.784] |
| Best of 13 observables | 32 | 0.834 | **1.000** | +0.166 [0.025, 0.298] |
| Fourier synchronization | 21 | 0.739 [0.412, 1.000] | **1.000** | +0.261 [0.000, 0.588] |
| LC-PFN, in-prior control | 256 curves | 0.994 [0.987, 0.999] | | |

### 4.2 Cost of the estimate

The main price of the committor is compute, since each forecast costs sixteen continuations of the parent. Two observations make it much cheaper. First, because most forecasts are 0 or 1, using fewer branches barely changes the ranking (Appendix B.2). Second, a cheap readout can settle the clear cases: a logistic model of checkpoint behavior, trained on 400 separately labeled runs, reaches AUC 0.816 on the 72 in-context-learning parents regenerated from the panel seeds, against 0.920 for full branching. Restarting only the half of the parents where the readout is least confident, a fraction we chose after seeing the full curve, then gives AUC 0.944, and two branches per uncertain parent give 0.912 at one sixteenth of the branch cost (Appendix E.2).

Continuing the parent once is a different kind of forecast: it watches the very run being scored, so at equal compute it predicts that run better than the committor does (Appendix E). What it cannot supply is the probability itself, which Section 4.3 uses to set pulse doses.

### 4.3 Steering low-probability runs

A zero forecast under ordinary training means that none of the sampled continuations of that protocol acquired the capability. However, it says nothing about what the same state would do under different data, so we ask whether low-forecast parents can still be made to acquire. The intervention is a pulse of data aligned with the capability, fifty updates long unless stated otherwise. In in-context learning it replaces 30 percent of each batch with episodes from three training mappings the model has not yet seen, and acquisition is scored on a disjoint bank of held-out mappings. We compare the pulse with ordinary training and with a random change to the weights of the same size.

We first apply the pulse to parents with a low forecast that fail to acquire on their own trajectory, a selection that uses each parent's outcome and that Section 4.4 later avoids. The ordinary-training arm is a new branch of each parent with its own data order, so its rate shows what these states do without help. Across four parent-disjoint blocks, the pulse converts 46 of 51 parents, against 2 of 51 under ordinary training (Table 2, Figure 2b), and the random change converts one. The result replicates on a fresh nine-parent block, and extending one 16-parent block tenfold, to 100,000 updates, leaves the gap unchanged (Appendix D). A low forecast therefore describes where ordinary training takes these runs, and under different data the same saved states can still acquire the capability.

How much pulse a run needs depends on its forecast. We measure this on a complete 48-parent cohort, plotting conversion against pulse length at every forecast level (Figure 2c). Longer pulses convert more, and at fifty updates the pulse exceeds ordinary training by 0.667 [0.521, 0.792]. However, parents with a zero forecast keep gaining from longer pulses, while those with an intermediate forecast gain most in the first ten updates (Table 7). The forecast therefore indicates how much help a run needs. Replayed on the recorded outcomes, a rule that sets each pulse length from the

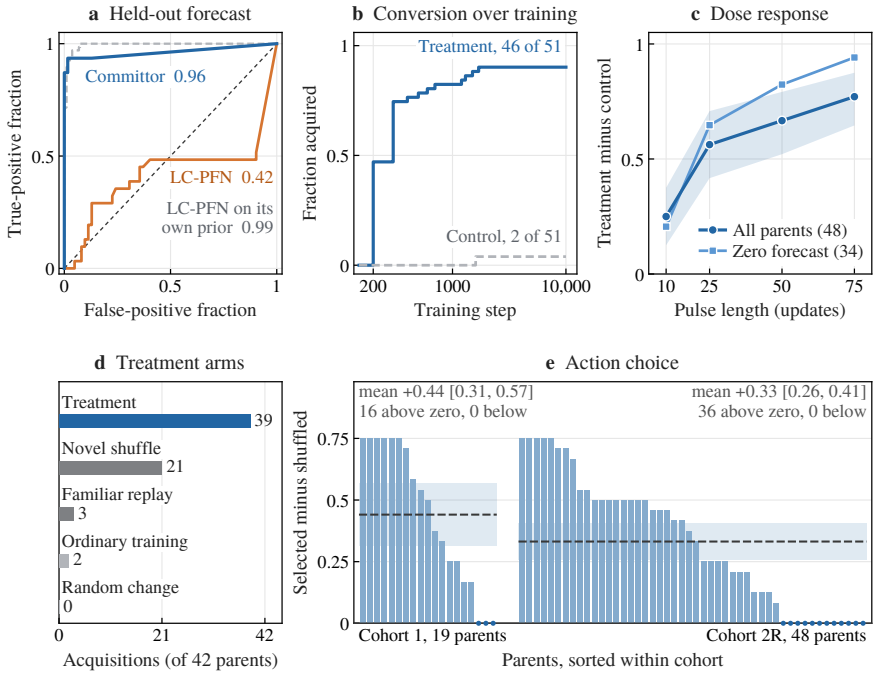


Figure 2: **The forecast predicts acquisition, and targeted pulses convert low-forecast runs.** **a**, ROC curves on the 93 parents. **b**, The 51 low-forecast parents that fail on their own. **c**, Dose response on one cohort. **d**, Acquisitions by arm. **e**, Fraction of branches converted by the selected action minus the shuffled-action average, with Cohort 2R the 48-parent replication; the top tick is the attainable maximum of 0.75, and dots mark ties. Shading shows 95 percent intervals (Appendix D).

forecast converts 47 of 48 parents with 27 percent fewer pulse updates than the best fixed dose (Appendix D.2).

We then ask which part of the pulse matters, and separate its novelty, its alignment with the target, and the weight movement it causes, on a complete 42-parent cohort (Figure 2d). Aligned new data converts almost every parent. In contrast, new inputs whose input-output mapping has been shuffled convert about half as many, familiar replay and ordinary training convert almost none, and the random change converts none. New data therefore helps on its own, extending to single runs the finding that in-context learning emerges when the training data are diverse enough (Chan et al., 2022; Raventós et al., 2023; Reddy, 2024), and alignment with the target adds most of the rest. Complementing the target labels in the pulse, conversely, removes most of the gain (Appendix D.2).

Sparse parity lets the control match the added inputs. There the intervention permanently enlarges the training set, from 700 to 2,000 target-labeled examples, and every untreated run fails. The matched control trains on the same inputs but labels the added examples by parity on other coordinates. Treatment converts 50 of 56 branches and the matched control converts none, and treatment beats it at all 7 parents, so the difference comes from alignment with the target (Appendix D.3).

### 4.4 Selection and action choice

So far we chose which parents to pulse using their own outcomes, which a practitioner cannot do. The real test of the committor as a guide is whether it can decide which runs to treat before any outcome is known. We therefore fix a rule in advance on a complete 72-parent panel: it gives the pulse to the lowest-forecast parents, those with $\widehat{q} \leq 1/16$, up to 57 slots. Compared with random allocation of the same budget, acquisition rises by +0.103 [0.050, 0.148], and the gain holds on a second runtime. It comes from the branches: a forecast rebuilt from a single branch per parent keeps most of the gain, while ranking by the cheap readout of Section 4.2, with no branches, does not reliably beat random allocation (Appendix D.4).

Table 2: **Acquisition in the intervention experiments.** Entries count parents that acquire, or branches where marked; decimal entries are expected counts. Ordinary training is the untreated continuation, and the random change moves the weights by the same amount as the pulse. Dose updates exclude the cost of the estimate. Per-block results and intervals are in Appendix D.

| Experiment | Treated | Ordinary | Other comparison |
|---|---|---|---|
| Low forecast, no own acquisition | 46/51 | 2/51 | Random change 1/51 |
| of which, fresh block outside the panel | 7/9 | 0/9 | Random change 1/9 |
| Components of the pulse | 39/42 | 2/42 | Novel shuffle 21/42, familiar replay 3/42, random 0/42 |
| Sparse parity (branches) | 50/56 | 0/56 | Labels on other coordinates 0/56 |
| Dose set by forecast, 2,630 updates | 47/48 | | Best fixed dose 47/48, 3,600 updates |
| Forecast chooses runs to treat | 63/72 | | Random allocation 55.6/72, treat all 66/72, wait then treat 69/72 |
| Forecast chooses action (branches) | 91/114 | | Shuffled over four actions 163/456, by loss 8/114 |

We also compare the forecast rule with two simpler strategies that need no forecast, and both end with more acquisitions. The forecast rule keeps 63 of the 66 acquisitions obtained by treating all 72 parents while using 57 slots (Table 2), and a rule that waits a quarter of the horizon and then treats the runs that have not yet acquired reaches 69 of 72 by the final deadline, because a later pulse converts more and the unaided acquirers are known by then. The forecast rule's advantage is timing: it pulses each run at step 100 of its own training, so it leads the waiting rule at every deadline before step 3,000, once its branches have finished (Appendix D.4).

Because branches can be run under any intervention, the committor can also compare actions on the same state. We score four candidate 25-update actions, which cross new or familiar mappings with intact or shuffled labels, on eight branches per parent and apply the highest-scoring one (Figure 2e). Across 19 parents, the chosen action converts 91 of 114 held-out branches, more than under shuffled scores or selection by loss (Table 2). The branch score picks the same action, new mappings with intact labels, for every parent, a 48-parent replication picks it every time, and the lowest-scored action reverses the gain (Appendix D.4). A per-parent oracle converts 92 of 114 against 91 for the single best action, so in this library only the dose needs to vary between runs (Section 4.3).

### 4.5 Natural text and released models

To test whether these findings extend beyond controlled tasks, we train a 10M-parameter GPT from scratch on FineWeb-Edu text (Penedo et al., 2024) and define acquisition as the induction score, the drop in loss from early to late positions of the context (Olsson et al., 2022), exceeding 0.4 nats (Appendix F.1). Across 60 runs from 8 initializations, restart estimates at the 170 checkpoints saved before each run acquires (later ones have a known outcome) predict whether it acquires within the next 50 updates with AUC 0.864 [0.752, 0.982]. As in the controlled tasks, most estimates sit near 0 or 1, and within one initialization the mean climbs from near zero to near one in about a hundred steps (Figure 3a).

In released training suites, onset is predictable from model size. We define onset as the step where zero-shot ARC-Easy (Clark et al., 2018) accuracy crosses 40 percent of the way from its lowest to its highest level (Appendix F). Across all eight released Pythia sizes from 70M to 12B, onset follows a tight log-log relation with size (Figure 3b). In a stricter test than the plotted fit, a fit to the four sizes up to 1B predicts the onsets of the four larger models, 1.4B to 12B, within a factor of 1.136, against up to 1.90 for a size-blind baseline. The ordering also replicates on DataDecide (Appendix F).

Finally, we ask whether the committor can find low-forecast states in released models that a pulse can steer. For a released Pico-65M model (Diehl Martinez et al., 2025), restart estimates jump from 0 of 16 at step 3,000 to 16 of 16 at step 4,000, and at step 3,000 repeated-token segments lift every pulsed branch above every untreated one. On Pythia 14M, where a lower induction score means stronger induction, restarts from 17 released checkpoints single out seed 4 at step 1,000: none of its restarts reach the preset bar. From that checkpoint a 250-update pulse of such segments ends at a mean of $-7.04$ nats, against $-2.62$ for a matched control (Figure 3c). The preset bar sits so near

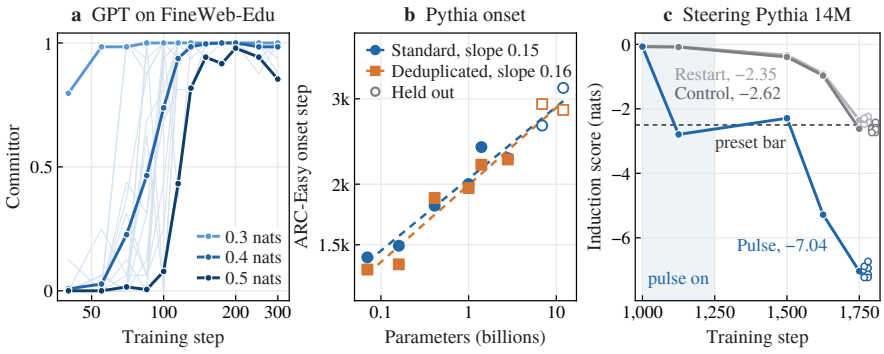


Figure 3: **The committor on natural text, onset across model size, and steering a released checkpoint.** **a**, Mean committor of 16 parent runs from one initialization, with bars of 0.3, 0.4 and 0.5 nats (12, 16 and 12 branches per run) on a log step axis; faint lines are single runs at 0.4 nats. **b**, Fits use sizes up to 2.8B; 6.9B and 12B (open markers) are held out. **c**, Branches from the seed-4 step-1,000 checkpoint; a lower induction score means stronger induction. The control permutes the pulse segments; the dashed line is the preset −2.5 nat bar.

the restart mean that most controls also cross it, but at every bar from −2.75 to −6.7 nats the pulse converts every pulsed run and no control. The pulse also lowers the natural-text repeated-span score by 0.31 nats (Appendix F).

## 5 Conclusion

In this work, we introduced the training committor, the probability that a saved checkpoint acquires a capability within a fixed horizon, and measured it by restarting training from the checkpoint. Looking at many possible futures of a run, rather than the one we happen to observe, changes what we can learn about it. Most runs turned out to be effectively decided before the capability appeared, and the committor predicted their outcomes on controlled tasks and on natural text. On the controlled tasks, it outperformed every checkpoint-local predictor we could implement, which shows that the saved weights already carry information about a run's future that standard training statistics miss. However, a decided run can still be redirected. A short targeted pulse led low-probability in-context-learning runs that did not acquire on their own to acquire the capability, and the committor identified which runs to treat, which action works best, and, in a replay of recorded outcomes, how long a pulse each run needs.

More broadly, our results suggest viewing training the way chemists view a rare transition: each run is one sample from a distribution of futures, and the probabilities of those futures can be measured and shaped. The natural next step is to measure the committor in large language models, for which Appendix E.6 estimates the compute of a 1B pilot. Cheaper estimates look within reach, since on another cohort a linear probe reads the outcome from pre-acquisition representations at AUC 0.93, or 0.78 at fixed training fraction (Appendix B.3), and splitting methods (Allen et al., 2005; Cérou & Guyader, 2007) could reduce the cost further. The committor also offers a principled way to find what decides a run, by testing whether a candidate variable such as attention entropy or the probe score captures it, as chemists test reaction coordinates (Bolhuis et al., 2002; Peters & Trout, 2006). We hope that measurements of this kind will make capability emergence something we can anticipate and guide during training, rather than only observe after the fact.

## AI use statement

Generative AI tools were used to edit software code and to draft or edit parts of the manuscript for readability. The authors reviewed the AI-assisted work and take responsibility for the final content.

## ETHICS STATEMENT

This work uses no human subjects or personal data. It studies controlled systems and released model checkpoints, reports no new frontier-scale training, and releases no model weights or turnkey training service. Because the same framework can steer the acquisition of different capabilities, its use should account for capability-specific risks.

## REPRODUCIBILITY STATEMENT

The anonymous artifact provides the code for the estimator, the controlled tasks, and the analyses. From parent- and branch-level records it recomputes the held-out forecast, the comparator tables, the intervention blocks, dosing, the alignment factorial, sparse parity, allocation with its cost-counted replay, action choice, the cheap readout, and the released-model analyses, and checks each value against the paper. Appendix A specifies the systems, events, and continuation protocols. Appendix B defines the scores and cluster-based uncertainty calculations. The remaining appendices give comparator implementations, intervention protocols, compute accounting, and released-model analyses, with parent-level outcomes and the scope of each comparison. Once its dependencies are installed, the claim replay runs offline on CPU and needs no model weights.

## REFERENCES

S. Adriaensen, H. Rakotoarison, S. Müller, and F. Hutter. Efficient Bayesian learning curve extrapolation using prior-data fitted networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

R. J. Allen, P. B. Warren, and P. R. ten Wolde. Sampling rare switching events in biochemical networks. *Physical Review Letters*, 94(1):018104, 2005. doi: 10.1103/PhysRevLett.94.018104.

R. J. Allen, C. Valeriani, and P. R. ten Wolde. Forward flux sampling for rare event simulations. *Journal of Physics: Condensed Matter*, 21(46):463102, 2009. doi: 10.1088/0953-8984/21/46/463102.

G. S. Altıntaş, D. Kwok, C. Raffel, and D. Rolnick. The butterfly effect: Neural network training trajectories are highly sensitive to initial conditions. In *International Conference on Machine Learning (ICML)*, 2025.

B. Barak, B. L. Edelman, S. Goel, S. Kakade, E. Malach, and C. Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning (ICML)*, 2023.

BigScience Workshop, T. Le Scao, A. Fan, C. Akiki, E. Pavlick, et al. BLOOM: A 176B-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. *Annual Review of Physical Chemistry, 53:291–318*, 2002.

F. Cérou and A. Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2), 2007. doi: 10.1080/07362990601139628.

S. C. Y. Chan, A. Santoro, A. K. Lampinen, J. X. Wang, A. K. Singh, et al. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018.

K. Clauw, S. Stramaglia, and D. Marinazzo. Information-theoretic progress measures reveal grokking is an emergent phase transition. *arXiv preprint arXiv:2408.08944*, 2024.

R. Diehl Martinez, D. D. Africa, Y. Weiss, S. Salhan, R. Daniels, and P. Buttery. Pico: A modular framework for hypothesis-driven small language model research. *arXiv preprint arXiv:2509.16413*, 2025.

L. Dong. The kinetics of training: A driven-nucleation rate law for emergence, plasticity loss, and circuit control in language models. *arXiv preprint arXiv:2607.27281*, 2026.

Y. Du, J. He, D. Zhang, E. Vanden-Eijnden, and C. Domingo-Enrich. Rare Event Analysis via Stochastic Optimal Control. *arXiv preprint arXiv:2604.13213*, 2026.

Z. Du, A. Zeng, Y. Dong, and J. Tang. Understanding emergent abilities of language models from the loss perspective. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

W. E and E. Vanden-Eijnden. Towards a theory of transition paths. *Journal of Statistical Physics, 123(3):503–523*, 2006.

W. E and E. Vanden-Eijnden. Transition-path theory and path-finding algorithms for the study of rare events. *Annual Review of Physical Chemistry, 61:391–420*, 2010.

B. L. Edelman, S. Goel, S. Kakade, E. Malach, and C. Zhang. Pareto frontiers in deep feature learning: Data, compute, width, and luck. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

S. Fort, G. K. Dziugaite, M. Paul, S. Kharaghani, D. M. Roy, and S. Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the Neural Tangent Kernel. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020.

L. Helfmann, E. Ribera Borrell, C. Schütte, and P. Koltai. Extending Transition Path Theory: Periodically Driven and Finite-Time Dynamics. *Journal of Nonlinear Science*, 30(6):3321–3366, 2020. doi: 10.1007/s00332-020-09652-7.

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, et al. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

G. L. Howe. Structure-specific representational priors causally control the grokking delay. *arXiv preprint arXiv:2607.04333*, 2026.

M. Y. Hu, A. Chen, N. Saphra, and K. Cho. Latent state models of training dynamics. *Transactions on Machine Learning Research (TMLR)*, 2023.

S. Hu, X. Liu, X. Han, X. Zhang, C. He, W. Zhao, Y. Lin, N. Ding, Z. Ou, G. Zeng, Z. Liu, and M. Sun. Predicting emergent abilities with infinite resolution evaluation. In *International Conference on Learning Representations (ICLR)*, 2024.

B. E. Husic and V. S. Pande. Markov state models: From an art to a science. *Journal of the American Chemical Society, 140(7):2386–2396*, 2018.

G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

T. X. Khanh. Cross-trajectory chimera interventions reveal dissociable roles of weight magnitude and direction in grokking. *arXiv preprint arXiv:2607.06628*, 2026.

T. X. Khanh, T. Q. Hoa, L. D. Trung, and P. T. Duc. Spectral entropy collapse as a phase transition in delayed generalisation: An interventional and predictive framework for grokking. *arXiv preprint arXiv:2604.13123*, 2026.

Y. Khoo, J. Lu, and L. Ying. Solving for high-dimensional committor functions using artificial neural networks. *Research in the Mathematical Sciences, 6:1*, 2019.

E. Lau, Z. Furman, G. Wang, D. Murfet, and S. Wei. The local learning coefficient: A singularity-aware complexity measure. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 258, pp. 244–252, 2025.

J. Lee, B. G. Kang, K. Kim, and K. M. Lee. Grokfast: Accelerated grokking by amplifying slow gradients. *arXiv preprint arXiv:2405.20233*, 2024.

Q. Li, B. Lin, and W. Ren. Computing committor functions for the study of rare events using deep learning. *Journal of Chemical Physics, 151(5):054112*, 2019.

Y. Lin. Canalization before generalization: Grokking as a dynamical probe. *arXiv preprint arXiv:2608.25813*, 2026.

Z. Liu, A. Qiao, W. Neiswanger, H. Wang, B. Tan, et al. LLM360: Towards fully transparent open-source LLMs. *arXiv preprint arXiv:2312.06550*, 2023.

Z. Liu, B. Tan, H. Wang, W. Neiswanger, T. Tao, et al. LLM360 K2: Building a 65B 360-open-source large language model from scratch. *arXiv preprint arXiv:2501.07124*, 2025.

D. Lucente, S. Duffner, C. Herbert, J. Rolland, and F. Bouchet. Machine Learning of Committor Functions for Predicting High Impact Climate Events. In *Proceedings of the 9th International Workshop on Climate Informatics (CI2019)*, Paris, France, 2019.

I. Magnusson, N. Tai, B. Bogin, D. Heineman, J. D. Hwang, L. Soldaini, A. Bhagia, J. Liu, D. Groeneveld, O. Tafjord, N. A. Smith, P. W. Koh, and J. Dodge. DataDecide: How to predict best pretraining data with small experiments. In *International Conference on Machine Learning (ICML)*, volume 267, pp. 42487–42502, 2025.

A. Mardt, L. Pasquali, H. Wu, and F. Noé. VAMPnets for deep learning of molecular kinetics. *Nature Communications, 9:5*, 2018.

P. Metzner, C. Schütte, and E. Vanden-Eijnden. Transition path theory for Markov jump processes. *Multiscale Modeling & Simulation, 7(3):1192–1219*, 2009.

E. J. Michaud, Z. Liu, U. Girit, and M. Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

F. Molteni, R. Buizza, T. N. Palmer, and T. Petroliagis. The ECMWF ensemble prediction system: Methodology and validation. *Quarterly Journal of the Royal Meteorological Society*, 122(529): 73–119, 1996. doi: 10.1002/qj.49712252905.

R. Nadkarni, Y. Elazar, H. Gonen, and N. A. Smith. Rewriting history: A recipe for interventional analyses to study data effects on model behavior. *Transactions of the Association for Computational Linguistics*, 14:1562–1588, 2026. doi: 10.1162/TACL.a.740.

N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations (ICLR)*, 2023.

C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

Y. Ootani. Grokking is conditional and fragile: A fully-tractable, multi-seed study at 12K parameters. *arXiv preprint arXiv:2607.05104*, 2026.

G. Penedo, H. Kydlíček, L. Ben Allal, A. Lozhkov, M. Mitchell, C. Raffel, L. Von Werra, and T. Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

B. Peters and B. L. Trout. Obtaining reaction coordinates by likelihood maximization. *The Journal of Chemical Physics*, 125(5):054108, 2006. doi: 10.1063/1.2234477.

A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

J.-H. Prinz, H. Wu, M. Sarich, B. Keller, M. Senne, M. Held, J. D. Chodera, C. Schütte, and F. Noé. Markov models of molecular kinetics: Generation and validation. *Journal of Chemical Physics, 134(17):174105*, 2011.

A. Raventós, M. Paul, F. Chen, and S. Ganguli. Pretraining task diversity and the emergence of non-Bayesian in-context learning for regression. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

G. Reddy. The mechanistic basis of data dependence and abrupt learning in an in-context classification task. In *International Conference on Learning Representations (ICLR)*, 2024.

Y. Ruan, C. J. Maddison, and T. Hashimoto. Observational scaling laws and the predictability of language model performance. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

R. Schaeffer, B. Miranda, and S. Koyejo. Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A. K. Singh, T. Moskovitz, F. Hill, S. C. Y. Chan, and A. M. Saxe. What needs to go right for an induction head? A mechanistic study of in-context learning circuits and their formation. In *International Conference on Machine Learning (ICML)*, 2024.

A. Sivasankar. Circuit synchronization precedes generalization: A causal precursor to grokking. *arXiv preprint arXiv:2606.12966*, 2026.

C. Snell, E. Wallace, D. Klein, and S. Levine. Predicting emergent capabilities by finetuning. In *Conference on Language Modeling (COLM)*, 2024.

Team OLMo, P. Walsh, L. Soldaini, D. Groeneveld, K. Lo, et al. 2 OLMo 2 Furious. *arXiv preprint arXiv:2501.00656*, 2024.

P. J. Tikeng Notsawo, H. Zhou, M. Pezeshki, I. Rish, and G. Dumas. Predicting grokking long before it happens: A look into the loss landscape of models which grok. *arXiv preprint arXiv:2306.13253*, 2023.

O. van der Wal, P. Lesci, M. Müller-Eberstein, N. Saphra, H. Schoelkopf, W. Zuidema, and S. Biderman. PolyPythias: Stability and outliers across fifty language model pre-training runs. In *International Conference on Learning Representations (ICLR)*, 2025.

A. Warstadt, A. Parrish, H. Liu, A. Mohananey, W. Peng, S.-F. Wang, and S. R. Bowman. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics, 8:377–392*, 2020.

J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

Y. Xu. Early-warning signals of grokking via loss-landscape geometry. *arXiv preprint arXiv:2602.16967*, 2026.

Z. Xu, Z. Ni, Y. Wang, and W. Hu. Let me grok for you: Accelerating grokking via embedding transfer from a weaker model. In *International Conference on Learning Representations (ICLR)*, 2025.

J. Yuan, A. Shah, C. Bentz, and M. Cameron. Optimal control for sampling the transition path process and estimating rates. *Communications in Nonlinear Science and Numerical Simulation*, 129:107701, 2024. doi: 10.1016/j.cnsns.2023.107701.

R. Zhao, T. Qin, D. Alvarez-Melis, S. Kakade, and N. Saphra. Random scaling of emergent capabilities. *arXiv preprint arXiv:2502.17356*, 2025.

# APPENDIX

## A SYSTEMS AND CONTINUATION PROTOCOLS

This appendix describes the three controlled systems behind the forecast panel, what a saved training state contains, and how each continuation protocol treats that state. We first list the cohorts and define the capability events, then describe the reset, carry and own-continuation protocols and check that restoration is exact, and finally separate the step at which a state can be scored from the step at which its forecast becomes available.

### A.1 CONTROLLED COHORTS AND EVENTS

Table 3 lists the five controlled cohorts. Together they hold 93 saved parents, the training states from which we branch, in 86 initialization clusters; parents that share an initialization seed form a cluster and are never treated as independent. For each parent we launch sixteen reset branches, estimate the committor as the fraction of them that acquire by the horizon, and score that estimate against the parent's own continuation, which keeps the parent's optimizer and data-order state. An acquisition counts only if it persists: the event criterion in the last column of the table must hold at five evaluations spaced 100 updates apart, and the first of the five must fall within the horizon.

The in-context-learning (ICL) system is a single causal transformer block of width 64 with four heads and a width-128 MLP. Each training episode shows eight labeled examples of a mapping followed by one query, and evaluation uses held-out mappings whose labels are specific to the episode, so that the model can answer only by reading the examples in its context. The two algorithmic systems share a four-head decoder block of width 128. Modular addition predicts $(a + b)$ mod 113 from all ordered pairs, and permutation composition predicts the composition of two elements of $S_5$, applying the right one first. The anonymous artifact contains the model, task and training code and the records behind every recomputed result.

The horizons were set on separate cohorts used only for that purpose, and chosen so that within the horizon a run's outcome depends mainly on whether it acquires rather than on when. Two systems mark the ends of the range. Sparse parity gives the clearest case of runs that never acquire: at training fraction 0.07, none of the 8 networks acquires within 80,000 steps, about 35 times the fastest acquisition seen in the same regime, and all 64 permanent failures among the 276 cohort runs are parity runs. Modular addition gives the opposite case, since non-acquirers at fraction 0.20 are mostly late rather than lost, with the within-horizon committor rising from 0.071 at 30,000 steps to 0.894 at 80,000. Within the chosen horizons the outcome label is stable, and 90.2 percent of 41 states keep their label from the half horizon to the full horizon, while 4 flip.

The forecast also holds outside the panel. In a separate fresh block of 16 modular-addition parents, 13 acquire by step 60,000. At step 10,000 the estimate reaches AUC 1.0 and Brier 0.060 against a base-rate Brier of 0.152, with a fixed-label perfect-separation probability of $1/560$. The $S_5$ result replicates at a lower training fraction as well: at training fraction 0.45, a second cohort of 16 states from 8 initializations has 2 acquirers, and the committor at step 2,500 separates them perfectly (AUC 1.0), although both acquirers share one initialization, so at the cluster unit this is 1 positive cluster of 8 with permutation probability only 0.125. Because this cohort reuses the initialization seeds of the panel's $S_5$ cohort, we report it separately rather than pooling the two.

### A.2 WHAT IS RESTORED AND WHAT IS RANDOMIZED

A saved state contains the model parameters, the AdamW state, the optimizer step, the training clock, the sampler state, and the random-number states. The three continuations differ in how much of this they keep. Reset restores the model and the clock, starts AdamW afresh with the same hyperparameters, and reseeds the future data order; it is the protocol behind every panel forecast. Carry keeps the AdamW state as well and reseeds only the data order. The parent's own continuation keeps everything, the AdamW state and the original sampler stream, and is the trajectory against which each forecast is scored. The three therefore define related but distinct distributions over futures, and we never pool them as replicates. All 93 scored labels are the parents' own uninterrupted continuations, not a seventeenth reset branch.

Table 3: **Controlled cohorts and capability events.** The horizon counts updates after the saved parent.

| Cohort | Seeds | Parents | Clusters | Step | Horizon | Persistent event |
|---|---|---|---|---|---|---|
| ICL full | 1400–1423 | 24 | 24 | 100 | 9,900 | held-out $\geq 0.90$ |
| ICL panel | 1600–1623 | 24 | 24 | 100 | 9,900 | held-out $\geq 0.90$ |
| ICL repeat | 1500–1523 | 24 | 24 | 100 | 9,900 | held-out $\geq 0.90$ |
| Modular addition | six seeds | 6 | 6 | 10,000 | 50,000 | train $\geq 0.99$, test $\geq 0.95$ |
| $S_5$ composition | eight seed families | 15 | 8 | 2,500 | 57,500 | train $\geq 0.99$, test $\geq 0.95$ |

Restoring a state with the same seed reproduces the minibatch stream and the endpoint exactly, while changing only the branch seed changes the future data. In a separate replay of 96 ICL branches, every binary outcome and every event step matched across devices.

Carry lets us measure how much the optimizer state matters. Swapping in another run's optimizer moments alone moves final accuracy by +0.0031 [0.0014, 0.0049] over 19 parents. In an eight-parent paired check, carry exceeds reset at seven parents, ties at one, and is lower at none (one-sided exact sign probability 0.0352 with the tie counted against carry). The same direction holds across the cohort: on all 72 ICL parents the mean $B = 8$ carry probability exceeds the $B = 16$ reset probability by +0.089 [0.052, 0.132], with 23 parents positive, 47 tied, and 2 negative. Carry changes AUC by +0.060 [−0.006, 0.157] and Brier by −0.023 [−0.064, 0.013], and both intervals include zero. Reset therefore gives lower probabilities than carry with a similar ranking, which is the pattern behind the calibration deficit examined in Appendix B.2.

The checkpoint also determines timing more tightly in some systems than in others. For acquiring branches, the starting checkpoint explains 0.987 and 0.981 of the variance in acquisition time in modular addition and permutation composition, but 0.547 and 0.626 in natural and intervened in-context learning. In the algorithmic systems the saved state nearly fixes when an acquiring branch acquires; in in-context learning much of that variation arises after the checkpoint. Even so, the forecast carries timing information that loss lacks: among the 16 in-context-learning parents that acquire, the Spearman correlation between the forecast and the acquisition step is −0.881 ($p = 6.6 \times 10^{-6}$), against −0.384 for loss, so a higher forecast also means earlier acquisition.

### A.3 Decision time and forecast availability

The committor is a property of the saved state, and a state at step zero has one as well, but its Monte Carlo estimate exists only once the branches have run. We therefore distinguish the step at which a state can be scored from the step at which its forecast is available. Released records provide neither: a released seed ensemble estimates how often a population acquires a capability, and a single released trajectory gives only its own onset.

When each method is scored from the first step at which it is defined, the committor of a subset of twelve ICL states rebuilt on a second device separates their outcomes at step zero with AUC 1.0, whereas the modular predictors we compare against first become defined at steps 2,000–4,000 for a decision at step 10,000. At step zero the scored state is the parent's initialization, and its forecast becomes available once its sixteen branches have finished.

## B Forecast scoring and uncertainty

In this appendix we define the scores and intervals used throughout the paper and give the calibration and information analyses behind Section 4.1. We first fix the unit of analysis and the scores, then evaluate the forecast as a probability and trace its one calibration failure to the protocol difference between the branches and the scored outcome, and finally ask what the forecast knows that the recorded observables do not.

### B.1 Units and scores

The parent is the independent unit throughout. We report AUC, the area under the receiver operating characteristic curve, for ranking, and the Brier score, the mean squared difference between forecast and outcome, for probability accuracy; the base-rate Brier is the score of a constant forecast at the cohort's acquisition rate. Confidence intervals resample all 86 initialization clusters with replacement, keeping sibling parents together, and paired comparisons resample once and recompute both methods on the same draw. Exact sign tests count only discordant parents. Neither branches nor repeated evaluations add to the parent count. Table 4 gives the scores for every cohort, and in particular two disjoint 24-parent in-context-learning cohorts each reach AUC 1.0, with Brier scores of 0.030 and 0.056 against within-cohort base-rate scores of 0.139 and 0.165. The one imperfect cohort, the ICL panel at AUC 0.840, is not a product of branch noise: a perfectly separating committor observed through $B = 16$ branches gives AUC at or below 0.8403 in 0 of 20,000 simulated cohorts, so its misses reflect the protocol difference discussed in Appendix B.2.

The observed orderings can also be tested exactly, by permuting outcomes rather than resampling parents. On the full panel, permuting outcomes only within each of the five cohorts and only in ways that keep initialization clusters whole gives a stratified AUC of 0.9421 and an exact one-sided probability of $9.09 \times 10^{-13}$. For the two perfectly separated in-context-learning cohorts, with 4 and 5 acquirers, the exact one-sided probabilities are $9.41 \times 10^{-5}$ and $2.35 \times 10^{-5}$, and $2.21 \times 10^{-9}$ jointly with the cohort as stratum. The recorded observables are less stable across the same two cohorts: memorization score predicts acquisition when low in the first cohort (AUC 0.931) but when high in the second (0.621), and held at one orientation across both cohorts the best observable, the in-context-learning score, reaches only 0.720. We chose the stratification and the exact tests after reading the cohorts, so these probabilities describe the observed orderings and are not prespecified tests.

### B.2 Calibration and finite branch counts

We next evaluate the forecast as a probability rather than as a ranking. Relative to a constant forecast at the panel's base rate of one third, it cuts Brier loss by 79.5 percent (whole-cluster interval [0.620, 0.936]). Part of this pooled skill comes from differences between cohorts: Brier skill within single cohorts ranges from 0.563 to 1.0, and the within-cohort stratified skill is 0.723. Figure 4 shows the forecasts behind these scores: how restart probabilities evolve over parent training steps (a), the forecasts separated by outcome (b), and the reliability diagram (c). The forecast is informative in the middle of its range as well as at the ends. Of the 18 parents with an intermediate forecast, 10 acquire, and the forecast orders them with AUC 0.975 over all 80 pairs with opposite outcomes. Carry branches, which keep the parent's optimizer state, close most of the gap left by the two zero-forecast parents that acquire: on the in-context-learning parents with eight branches, carry reaches AUC 0.988, against 0.927 under reset.

Every calibration bin except the zero bin lies inside its band, and the zero bin fails for a reason the protocol explains. Of the 56 parents with no successful branch, 2 acquire on their own continuation, which keeps the optimizer state that reset discards. Over the 93 parents and 86 clusters, the zero bin is tested against a simultaneous band of [0, 0], so any acquisition in that bin fails the test. The test uses the min-p statistic, the smallest exact binomial p-value across forecast bins, and for the calibration test behind Section 4.1, the global min-p statistic and its exact p-value are both zero. We therefore use the raw estimate for ranking and read a zero forecast as very unlikely rather than impossible.

The same protocol difference accounts for the small overall gap between forecasts and outcomes. The aggregate forecast-minus-outcome gap is $-0.0504$, with $z = -3.06$ and a fixed-seed matched-reset simulation probability of 0.0021. The gap is negative in all three ICL cohorts and exactly zero in the other two, and the calculation includes the uncertainty of both the sixteen-branch estimate and the single outcome. Keeping the optimizer state removes most of the deficit: on the same 72 ICL parents, reset minus outcome is $-0.065$ $[-0.123, -0.016]$, whereas carry minus outcome is $+0.024$ $[-0.019, 0.068]$. Within the reset protocol itself, bin-level reliability is 0.025 against an expected 0.017 ($p = 0.119$), consistent with the sampling noise of sixteen branches per parent.

Two further checks show what the forecast achieves when the outcome follows the same reset protocol as the branches. In a simulation in which it does, over 20,000 draws, AUC is 0.996, calibration-

Table 4: **Held-out forecast scores by cohort.** The pooled score is computed from all parent rows.

| Cohort | Parents | Acquired | AUC | Brier | Base-rate Brier |
|---|---|---|---|---|---|
| ICL full | 24 | 4 | 1.000 | 0.030 | 0.139 |
| ICL panel | 24 | 7 | 0.840 | 0.090 | 0.207 |
| ICL repeat | 24 | 5 | 1.000 | 0.056 | 0.165 |
| Modular addition | 6 | 2 | 1.000 | 0.001 | 0.222 |
| $S_5$ composition | 15 | 13 | 1.000 | 0.000 | 0.116 |
| Pooled | 93 | 31 | 0.963 | 0.046 | 0.222 |

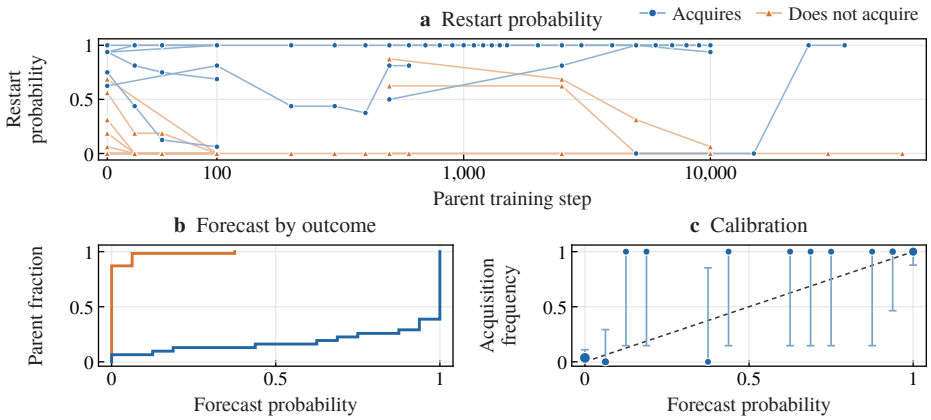


Figure 4: **Forecast behavior behind the headline score.** **a**, Restart probabilities across parent training steps. **b**, Held-out forecasts separated by observed outcome for all 93 parents. **c**, Observed acquisition frequency against forecast, with modified Jeffreys binomial intervals; marker area grows with the number of parents in each bin.

in-the-large is 0.000, and the analytic expected Brier is 0.022. We also ran the matched test empirically on a separate cohort of 24 in-context-learning parents, scoring each forecast against new held-out continuations run under the same protocol as its branches. There, reset reaches AUC 0.847 [0.620, 1.0] and Brier 0.149 (6 events), and carry reaches AUC 1.0 and Brier 0.005 (4 events), so the empirical reset AUC falls below the simulated 0.996, while carry separates the outcomes exactly. Estimator noise contributes little to the remaining error: of the pooled Brier score of 0.0456, Monte Carlo error from sixteen reset branches contributes 0.00139, so the rest reflects the gap between the reset probability and the parent's own continuation, which more branches would not close.

Calibration maps can repair the zero bin. Fixed maps leave the ranking untouched, while fitted maps cost ranking accuracy, so we report both without substituting either for the raw forecast. A fixed Jeffreys map, $(k + 1/2)/17$, gives finite log loss 0.175, expected calibration error (ECE) 0.059 [0.037,0.105], Brier 0.045, and unchanged AUC 0.963, although the deficit between protocols remains. A slightly stronger fixed map passes the calibration test outright: with $(k + 1)/18$, no bin falls outside its band and the exact global probability is 0.485, while AUC is unchanged at 0.963 and Brier changes by +0.0002 [−0.0041, 0.0036]. Fitted maps do worse on ranking: Cluster-held-out Platt, beta, isotonic, and Venn–Abers maps improve at least one calibration point estimate but reduce AUC by 0.032, 0.045, 0.035, and 0.029.

Smaller branch budgets can be tested exactly by subsampling the stored branches. The ranking remains stable at 8, 4, and 2 branches, with mean Spearman correlations of 0.962, 0.935, and 0.911. Calibration can be restored at these budgets with a pseudo-count scaled to the budget: with $k'$ of $B'$ subsampled branches acquiring, the estimate $(k' + B'/16)/(B' + B'/8)$ restores calibration at every budget down to 2, with the calibration test rejecting at most a fraction 0.0082 of subsamples.

### B.3 Information beyond recorded observables

Section 4.1 tests the forecast where the recorded statistics give no guidance, inside groups of states that those statistics cannot tell apart. We build that grouping without looking at outcomes. The grouping is a fate-blind $k = 24$ partition of 18 weight-space and loss observables, whose mixed cells split 9/8 and 9/5 (acquirers against non-acquirers). Within the mixed cells, the committor orders all 117 opposite-outcome pairs correctly.

The result survives the sibling structure of the sample. Among 16 initializations, nine informative opposite-fate sibling pairs define all $2^9 = 512$ swap assignments, and exactly one is as extreme, giving $1/512$, the design floor, and collapsing siblings to 16 independent initializations, 14 of which contain an acquirer, leaves the conclusion unchanged.

The forecast also arrives well before the event. Figure 4a shows how forecasts evolve within runs: In one six-parent modular-addition cohort, both acquirers' restart intervals stay above one half from step 2,500 onward on the logged grid, with leads of 51,700 and 32,100 steps before the event, while the forecasts for two other parents fall. In the 32-state modular-addition cohort, the 19 acquirers reach the event a median 29,000 updates after the step-10,000 checkpoint (minimum 6,500). These 19 acquirers come from 14 initializations, so they are not independent units.

The information that the recorded statistics miss can be read directly from the network. We fit a linear probe on the final-position penultimate representation of 96 MLP, sparse-parity, and permutation-composition runs, read at checkpoints at or before step 10,000 and therefore before any acquisition, and transfer it to 143 modular-addition transformer runs in 44 initialization clusters. It separates eventual acquirers from non-acquirers at AUC 0.93 [0.85, 0.97], with cluster-permutation probability $2 \times 10^{-4}$. The probe adds information beyond training fraction. Training fraction alone reaches 0.83, adding the probe raises it to 0.93, an increment of 0.100 with paired permutation probability below 0.01, and at fixed training fraction the probe alone reaches 0.78. This cohort is distinct from the 93-parent panel, and unlike the committor the probe is a fitted predictor.

## C Comparator protocols

In this appendix we describe how we chose, implemented, and checked the methods that Section 4.1 compares with the committor. Table 1 gives the main comparison and Table 5 every variant we ran. The 23 paired comparisons of Section 4.1 are the fourteen non-Fourier and seven Fourier rows of Table 5, together with the LC-PFN and Fourier synchronization rows of Table 1.

We surveyed 100 emergence predictors and kept the eight that our saved records support, and each of the eight passed its own positive control before we compared it with the committor. Most excluded methods require trajectory, loss, margin, or gradient channels that we did not save. Information-theoretic progress measures (Clauw et al., 2024), for example, read synergy and redundancy among neuron activations, which we did not record. Exclusion therefore reflects only what we recorded.

The comparators read different parts of the saved record. Attention entropy is read from the saved checkpoints. The loss model is fit leave-one-out on four features of the training loss, its value, its minimum, its slope, and its curvature, without seeing the held-out parent's outcome. Fourier synchronization follows the published frequency-domain construction at the sites where it is defined. LC-PFN, a published learning-curve extrapolator trained on a synthetic prior over curves, is run zero-shot on the available loss prefix and, as a positive control, on its own synthetic prior. Methods that observe the target run after the decision point, such as LC-PFN on the paid trace, the target's own post-checkpoint record read at its compute cost, and direct continuation, answer a different question, and we compare them on compute in Appendix E. The committor is not a restatement of the loss features: residualized on them, the committor still reaches AUC 1.000 in modular addition and 0.923 in in-context learning. Nor does adding the skill already visible at the checkpoint close the gap: against loss plus currently visible capability, the paired gap is +0.261 [0.104, 0.425] on the 72 in-context-learning states (AUC 0.666 against 0.927) and +0.577 [0.361, 0.769] on the 32 modular-addition states (AUC 0.425 against 1.000).

Two rows sit near chance for structural reasons. The ICL parents are saved at step 100 with a single logged loss value, so the value and its minimum coincide and the slope and curvature are constant; the ICL loss row is therefore in effect a one-feature fit. LC-PFN sees too little of each curve: Per

system, zero-shot LC-PFN reaches AUC 0.750, 0.744 and 0.695 on the three ICL cohorts, 0.625 on modular addition and 0.269 on $S_5$ composition, so even the cohorts with a 99-point or 25-point trace before the checkpoint do not lift it. Splitting the sibling pairs in half gives the same picture for the recorded observables. No recorded statistic that ranks one half of the sibling pairs holds up on the other half, and the best ICL and modular observable fields reach AUC 0.798 and 0.834, against 0.927 and 1.000 for the committor on the same parents.

Markov-state models are the standard coarse-grained estimate of a committor in molecular simulation: the observables are clustered into microstates, transitions between microstates are counted at a lag time, and the committor of each microstate follows from the resulting transition matrix. We build them on the recorded observables as well. On the 32-state modular-addition cohort at the step-10,000 anchor, we scan 112 configurations built on the 18 recorded observables (raw or time-lagged independent components with 2, 3 or 5 components, 8 to 60 microstates, and 4 lag times). The configuration selected by the VAMP2 score, the standard variational score for Markov-state models (Mardt et al., 2018), reaches anchor AUC 0.571, and the best configuration reaches 0.688, a maximum selected on the outcome and therefore optimistic. The committor reaches 1.0 on the same states, and the best leave-one-initialization-out supervised predictor on the same observables reaches 0.640.

Fourier synchronization (Sivasankar, 2026) is defined on 21 of 93 parents (six modular-addition and fifteen permutation-composition states in fourteen initialization clusters) and is undefined on ICL because those tasks have no supplied group answer coordinate. Our implementation reproduces the published behavior in its source regime, since modular addition synchronizes at step 6,000 before grokking at 8,000, and centered $S_5$ synchronizes at step 300 before grokking at 4,100, so it passes its own matched controls, but its interval on new runs includes chance. On the eligible target states, the committor has AUC 1.0 and Brier 0.00037, and Fourier minus committor is $-0.261$ $[-0.588, 0]$ in AUC and +0.220 $[0.133, 0.320]$ in Brier. Its pooled score comes mostly from differences between the two systems rather than from ranking runs within a system. The most favorable Fourier read site, on the MLP output, has AUC 0.85 [0.656, 1.0] and Brier 0.138 [0.055, 0.244], and we report it as a sensitivity analysis.

The two comparators defined on the full panel are LC-PFN and attention entropy. The committor–LC-PFN AUC gap is +0.542 $[0.344, 0.710]$ on all 93 parents, and LC-PFN's own interval contains chance. Attention-entropy collapse, the strongest newly implemented full-panel comparator, has AUC 0.794 [0.675, 0.889] and threshold-normalized Brier proxy 0.620 [0.523, 0.718].

Table 5 lists every comparator variant we ran, each with its paired AUC gap to the committor. Where a Fourier row's AUC interval touches zero, the Brier score settles the comparison: For the seven Fourier sensitivity rows, comparator-minus-committor Brier gaps range from +0.138 to +0.333, and every paired interval excludes zero in the committor's favor. Across the whole survey, every admissible comparator point estimate falls below the full-panel committor AUC 0.963.

# D Intervention protocols

In this appendix we give the protocols and full results behind Sections 4.3 and 4.4. We first describe the in-context-learning pulse, its control arms, and the four parent-disjoint blocks on which it was tested. We then separate what the pulse delivers from how much of it a run needs, turn to sparse parity and modular addition, where the intervention is a permanent change in the training set, and finally use the forecast to decide which parents to treat and which action to apply.

## D.1 Parent selection and treatment construction

The in-context-learning pulse is a brief change in the training data. At step 100 it replaces 480 of the 1,600 training sequences seen over fifty updates (30 percent of the sequences in those batches) with examples from three training mappings the model has not yet seen. Acquisition is scored on a separate bank of 64 held-out mappings that contains no training mapping. Four other arms isolate what in the pulse matters. Ordinary control continues standard training. Familiar replay uses mappings the model has already seen. The mapping-destroyed arm keeps the same tokens but breaks the correspondence between inputs and labels. The random change applies one isotropic displacement with the same norm as the weight change the treatment causes. Our control therefore covers one

Table 5: **Comparator sensitivity analysis.** Each gap is committor minus comparator AUC, so a positive gap favors the committor. Where a Fourier row's interval reaches zero, the Brier comparison in the text gives further evidence. Restricted and excluded loss are the progress measures of Nanda et al. (2023).

| Comparator and read site | Gap | 95 percent interval |
|---|---|---|
| Attention entropy collapse | 0.169 | [0.071, 0.286] |
| Source-threshold spectral entropy | 0.244 | [0.010, 0.589] |
| Source-code ILDR | 0.375 | [0, 1.000] |
| Restricted loss | 0.375 | [0, 1.000] |
| Commutator defect | 0.446 | [0.312, 0.581] |
| Excluded loss | 1.000 | [1.000, 1.000] |
| Preserved Fourier–Gini | 0.375 | [0, 1.000] |
| Initialization-relative ILDR, full | 0.303 | [0.148, 0.470] |
| Initialization-relative ILDR, source subset | 0.289 | [0.029, 0.618] |
| Initialization-relative spectral entropy | 0.261 | [0.128, 0.413] |
| Attention entropy, ICL only | 0.237 | [0.064, 0.419] |
| Attention entropy, modular only | 0.250 | [0, 0.750] |
| Commutator defect, modular only | 0.375 | [0, 1.000] |
| Spectral entropy, modular only | 0.750 | [0.250, 1.000] |
| Fourier synchronization, all pairs | 0.256 | [0, 0.594] |
| Fourier, attention output | 0.217 | [0, 0.517] |
| Fourier, MLP output | 0.150 | [0, 0.344] |
| Fourier, modular top-two/$S_5$ $K = 1$ | 0.300 | [0, 0.662] |
| Fourier, pre-ReLU expansion | 0.267 | [0, 0.606] |
| Fourier, source-synchronization indicator | 0.283 | [0.094, 0.532] |
| Fourier, training pairs | 0.228 | [0, 0.548] |

kind of random change, an isotropic matched-norm displacement. Every arm then continues with the carry optimizer, and the arms within a parent share the branch seed, so that within a parent the arms differ only in the treatment itself.

Table 6 lists the four parent-disjoint blocks. Treatment and control acquire in 6/0, 18/1, and 15/1 parents in the first three blocks and 7/0 on a fresh nine-parent block outside the forecast panel, a risk difference of +0.778 [0.444, 1.0] with one random-change acquisition. In total, 46 of 51 treated parents acquire against 2 of 51 controls. At the level of parents, pooled over the four ICL blocks, 44 parents are treatment-only, 0 control-only, and 7 tied, a one-sided sign probability of $5.68 \times 10^{-14}$. Against the matched-norm random displacement, 45 parents are treatment-only, 0 displacement-only, and 6 tied. Block by block, among the first seven low-probability parents, six acquire only under treatment and none only under control, and a disjoint 19-parent block has 17 treatment-only and no control-only outcomes. On the fresh block the sign probability is $1/128$ rather than the design minimum of $1/512$, because 2 parents tie, and 1 of 9 displacement controls acquires.

The effect can also be read at the level of the committor rather than of a single outcome per arm. On the first block, sixteen branches per arm on its seven complete parents give a committor-level reading, and the intervention raises the committor at 6 of 7 parents, ties at one, and lowers it at none, a one-sided sign probability of 0.0156, or 0.0625 with the tie counted against it. A longer horizon does not change the picture: Extending the 16-parent replication to 100,000 updates changes the difference by 0.0 [0.0, 0.0], although before the extension the original counts on 16 parents left room for any change from $-15/16$ to $+1/16$.

### D.2 Target alignment and dose

The 42-parent factorial crosses the content of the pulse, new versus familiar mappings, with the coherence of its labels, intact versus mapping-destroyed. For each parent we define the interaction as (NC − ND) − (FC − FD), where NC, ND, FC, and FD are the outcomes for novel coherent, novel mapping-destroyed, familiar coherent, and familiar mapping-destroyed data, so a positive value means that coherence helps more when the content is new. Each mapping-destroyed arm is run with four independent shuffles, and its score averages over them. Familiar data with a destroyed mapping helps about as much as the novel shuffle, while familiar replay with the mapping intact does not, so the interaction is larger than the increment. The novel-coherence increment, coherent treatment minus the average of the four novel shuffles, is +0.470 [0.345, 0.595], with 15 tied parents, and the interaction is +0.851 [0.720, 0.970], with 5 tied parents. Both stay positive on two disjoint

Table 6: **ICL intervention effects by parent-disjoint block.** T-C is the treatment-minus-control risk difference with its exact whole-parent 95 percent interval. The union row pools the blocks.

| Block | Parents | Treatment | Control | T-C | 95 percent interval |
|---|---|---|---|---|---|
| Seeds 1400s | 7 | 6 | 0 | 0.857 | [0.571, 1.000] |
| Seeds 1500s | 19 | 18 | 1 | 0.895 | [0.737, 1.000] |
| Seeds 1600s | 16 | 15 | 1 | 0.875 | [0.688, 1.000] |
| Fresh nine-parent block (1300s) | 9 | 7 | 0 | 0.778 | [0.444, 1.000] |
| Four-block union | 51 | 46 | 2 | 0.863 | [0.765, 0.941] |

evaluation sets, where the interaction is +0.631 and +0.643, and rerunning the factorial on CPU reproduces all 210 parent-arm outcomes and their event steps exactly.

The primary novel shuffle, a single shuffle on the same parents, separates the effect of new data from that of alignment. The mapping-destroyed novel shuffle acquires on 21 of 42 recipients against 2 under baseline, shuffle minus baseline is +0.452 [0.310, 0.595], and treatment still exceeds the shuffle by +0.429 [0.286, 0.571], with 18 positive, 0 negative, and 24 tied parents. An antithetic comparison, which pairs each random change with its negation, gives a difference of −0.0003 [−0.0018, 0.0012], with 20 positive, 20 negative and 2 tied parents, so displacement alone does not explain the effect of coherence.

A reversed pulse tests alignment from the other side. On a 48-parent cohort we complement the supervised query target on the 480 novel-mapping rows of the pulse, and compare it with an equal edit that flips one non-query context label and keeps the target. The complemented pulse acquires in 16 of 48 parents and the equal edit in 42 of 48, with 0 parents favoring the complemented pulse and 26 favoring the edit, a difference of −0.5417 [−0.6875, −0.3958] and an exact one-sided sign probability of $1.49\times10^{-8}$. The reversed pulse does not suppress acquisition below ordinary training, however: ordinary training acquires in 12 of 48, with 5 parents favoring the reversed pulse and 1 favoring ordinary training, a difference of +0.0833 [0.0000, 0.1875].

How much pulse a run needs depends on where it starts. On one shared 48-parent cohort, the treatment-minus-control difference rises from 0.250 to 0.563, 0.667, and 0.771 at 10, 25, 50, and 75 updates (Table 7). The 34 parents with a zero forecast drive this trend, while the parents with intermediate forecasts do not rise monotonically. Across parents, the dose a parent needs decreases with its starting committor (Spearman −0.679, one-sided fixed-seed permutation probability $5.0 \times 10^{-6}$).

The same cohort supports a dosing rule read off the forecast. We replay the recorded outcomes with no pulse for the six parents at $\widehat{q} \geq 0.5$, ten updates for the eight at $0 < \widehat{q} < 0.5$, and 75 for the 34 at $\widehat{q} = 0$. The rule reaches 47 of 48 with 2,630 pulse updates, matching the best fixed dose (75 updates for every parent, 47 of 48, 3,600 updates). The same doses shuffled across parents acquire 37.96 in expectation, and none of 200,000 shuffles exceeds 45, a difference of +0.188 [0.118, 0.244] per parent. Assigning the same doses by rank of the cheap readout of Appendix E.2 instead, computed on regenerated checkpoints of the same seeds, acquires on 41 of 48 with the same 2,630 pulse updates, +0.063 [−0.009, 0.135] per parent over shuffled doses, so here too the gain comes from the branches rather than from a snapshot score. Choosing each parent's dose from the other 47 parents returns the same rule. The replay uses recorded outcomes, the strata were fixed before the dose study, and the count excludes the cost of estimating the committor.

### D.3 Sparse parity and modular addition

Sparse parity is the system in which runs most often never acquire (Appendix A), and its intervention is a permanent change in the training set rather than a pulse. Inputs are drawn once into a fixed pool of 10,000 random 30-bit strings, training fractions refer to this pool, and the target is the parity of three fixed coordinates. The study uses seven eligible parents and eight branches per arm. Treatment expands the training set from 700 to 2,000 target-labeled examples for the rest of the run. The matched control uses the same 2,000 inputs as treatment, keeping the target labels on the first 700 and labeling the added 1,300 by parity on coordinates $[9, 11, 22]$, disjoint from the target

Table 7: **ICL dose response on the shared 48-parent cohort.** Entries are treatment-minus-control differences, with exact whole-parent 95 percent intervals in the All 48 parents column. The strata reuse the same parents and omit the six parents that start at or above 0.5.

| Updates | All 48 parents | Starting $\widehat{q} = 0$ (34) | Starting $0 < \widehat{q} < 0.5$ (8) |
|---|---|---|---|
| 10 | 0.250 [0.125, 0.375] | 0.206 | 0.625 |
| 25 | 0.563 [0.417, 0.708] | 0.647 | 0.625 |
| 50 | 0.667 [0.521, 0.792] | 0.824 | 0.375 |
| 75 | 0.771 [0.646, 0.875] | 0.941 | 0.500 |

coordinates $[28, 6, 15]$. Because it matches the inputs and the example count, the matched control isolates alignment with the target from the amount of data.

Table 8 lists every parent's treatment and control counts out of eight branches. Treatment converts 50 of 56 branches, and neither control converts any. For parent seeds 500–506, treatment minus control is $+0.375, +1, +1, +1, +0.875, +1, +1$. The mean parent effect is 0.893 with exact whole-parent interval [0.714,1.000], and all seven effects are positive, giving one-sided sign probability $1/128$. Against the matched control the seven effects, their mean, and its interval are the same, since the matched control converts 0 of 8 on every parent while treatment reaches the ceiling on five parents.

We fixed the seven-parent design after a one-parent pilot of the same treatment, in which 6 of 8 steered branches acquired against 0 of 8 controls, two-sided Fisher probability 0.007. Seed 507 was dropped from the roster before any outcome was seen, because seed 507's reproduced training accuracy, 0.9871, failed the fixed 0.99 eligibility threshold.

In modular addition the direction agrees but the test is underpowered. The original window was saturated at both ends, since at 50,000 updates all three arms were 0/8, and at 90,000 treatment and control were 8/8 and random change 7/8, so neither endpoint could test conversion. We therefore scored a separate horizon of 67,150 updates, at which treatment, control, and random change acquire in 5, 3, and 4 of eight parents. Every discordant pair favors treatment, but most of the eight parents tie, and treatment–control is +0.250 [0,0.625] and treatment–random change is +0.125 [0,0.375]. A decisive test needs more parents: An optimistic calculation requires 24 total parents against control and 51 against the random change for 80 percent power. Expanding the test would have needed about 86 candidate initializations and 244 core-hours against 55 available, which exceeded the compute available for this study.

### D.4 Action choice and allocation

Section 4.4 uses the forecast in two ways, to decide which parents receive a pulse and to choose which pulse each receives. We describe allocation first, then its cost and its comparison with simply waiting, and finally action choice.

Allocation ranks the 72 parents of a complete panel using only checkpoint information. The rule gives the pulse to at most 57 parents whose forecast satisfies $\widehat{q} \leq 1/16$, inclusive, with deterministic guidance and random tie orders. The cost of producing the forecasts is counted separately in Appendix E. On this panel, where the allocation study reran every forecast and every untreated continuation, the forecast reaches AUC 0.921 [0.823,0.995] and Brier 0.085 [0.033,0.147]. The guided rule acquires on 63 parents. We compare it with the exact distribution over all allocations of the same budget, rather than with a single random draw, which would add its own sampling noise: the exact expected acquisition count is 55.583, the 95 percent range is 53–59, and the probability of reaching at least the guided count of 63 is $1.15 \times 10^{-5}$, a difference of +0.103 [0.050, 0.148]. Guidance retains 63 of 66 treat-all acquisitions while using 57 rather than 72 slots, and among the recipients, target treatment minus guided random change is +0.667 [0.556,0.778], while random-change-all minus ordinary-all is 0.0 $[-0.042, 0.042]$.

The rule tolerates a cheaper forecast. Rebuilding each forecast from a random 8 or 4 of the 16 stored branches (500 subsamples per budget, seed 0), guidance acquires 62.8 [62, 64] parents at either budget, never fewer than 61, and exceeds 55.583 in every subsample. Forecasts permuted across parents under the same rule reach 63 in none of 500 subsamples (mean 55.7, range 53 to 59). The

Table 8: **Sparse-parity outcomes by parent.** Eight branches per arm. T-C and T-M are treatment minus ordinary control and treatment minus matched control.

| Parent | Control | Treatment | Matched | T-C | T-M |
|---|---|---|---|---|---|
| 500 | 0/8 | 3/8 | 0/8 | +0.375 | +0.375 |
| 501 | 0/8 | 8/8 | 0/8 | +1 | +1 |
| 502 | 0/8 | 8/8 | 0/8 | +1 | +1 |
| 503 | 0/8 | 8/8 | 0/8 | +1 | +1 |
| 504 | 0/8 | 7/8 | 0/8 | +0.875 | +0.875 |
| 505 | 0/8 | 8/8 | 0/8 | +1 | +1 |
| 506 | 0/8 | 8/8 | 0/8 | +1 | +1 |

forecast's AUC falls to 0.878 at 8 branches and 0.844 at 4, but most parents have no acquiring branch and stay at $\widehat{q} = 0$ under any subsample, so the ranking that the rule uses barely moves. Because the subsamples reuse the stored branches, these intervals reflect subsampling noise only.

The budget of 57 slots covers 79 percent of the panel, so we also replay the recorded outcomes at smaller budgets. Guidance pulses the $K$ lowest-forecast parents, averaging over uniform tie-breaks at the boundary, and the random policy pulses $K$ of the 72 uniformly. At $K = 5$, 10, 20, 30, 40 and 57 guidance acquires on 20.3, 24.5, 33.0, 41.6, 50.1 and 63.5 of 72 parents against random expectations of 19.5, 22.9, 29.9, 36.8, 43.8 and 55.6, a gain of 0.14 to 0.16 acquisitions per slot at every budget.

The forecast's own cost enters the comparison as follows. Rebuilding each forecast from one random branch per parent, one sixteenth of the 11,404,800 full-horizon branch updates of the 16-branch forecast, the rule acquires on 61.9 of 72 parents (60.3 to 63.5 across 5,000 branch draws), 6.3 above the exact random expectation of 55.583 (whole-parent interval [5.2, 7.4]). Two branches per parent acquire on 62.7, the readout alone on 59.0 with an interval that includes zero, and forecasts permuted across parents return 55.58. These replays use the regenerated panel described below and uniform averaging over tied forecasts, under which the full forecast gives 63.5. Even the one-branch forecast costs 950 times the 750 pulse updates saved by leaving 15 parents untreated, so in this setting the forecast pays for itself only when an intervention costs far more than a training update.

To check that the result is not tied to one runtime, we regenerated the 72 parents from seed on a second runtime and reran the policy with fresh 16-branch reset forecasts. The rule again acquires on 63 of 72 against an exact matched-budget random expectation of 55.58, the fresh forecast reaches AUC 0.920 against the parents' own outcomes, untreated own trajectories acquire on 16 of 72, and pulsing every parent at step 100 acquires on 66 of 72.

A natural alternative waits and watches the runs, since each run's own trajectory eventually reveals whether it acquires unaided. Waiting costs no forecast compute but delays the decision, so we compared the two on the same parents. A rule that waits until step 2,600, a quarter of the horizon, and pulses every parent whose own trajectory has not yet acquired, up to the same 57 slots, acquires on 69 of 72 against 63 of 72 for committor targeting at step 100, a paired committor-minus-waiting difference of −6 parents (bootstrap [−0.167, −0.014] per parent, 7 parents favor the waiting rule and 1 the committor). The committor's own 57 parents pulsed at step 2,600 acquire on 66 of 72, the waiting rule's set pulsed at step 100 acquires on 66 of 72, pulsing every parent acquires on 66 of 72 at step 100 and 69 of 72 at step 2,600, and 16 of 72 parents have already acquired by step 2,600. The gap splits evenly between timing and selection: the later pulse converts three more parents whichever set receives it, and by step 2,600 each run's own trajectory has already revealed the 16 parents that acquire unaided. Because the waiting rule acts a quarter of the horizon later, the order reverses at earlier deadlines. Committor targeting at step 100 has acquired on 60 of 72 by step 1,000 and 63 of 72 by step 2,000, while the waiting rule still stands at the 16 of 72 untreated acquisitions at step 2,600 and reaches 69 of 72 by step 3,000. At every deadline before step 3,000 the committor policy leads, and at every deadline from step 3,000 on the waiting rule leads by 6. The committor rule therefore gains by deciding earlier.

Action choice evaluates four 25-update actions (novel coherent, novel mapping-destroyed, familiar coherent, and familiar mapping-destroyed data) on eight scoring continuations per action and parent, and then applies the highest-rated action on six held-out branches per parent. In the first 19 parents, the chosen action converts 91 of 114 branches, against 163 of 456 when the scores are shuffled

Table 9: **Forecast scores at equal scheduled optimizer-update budgets.** Cells are AUC/Brier on the 93-parent panel. $c$ is the restart budget in units of one full-horizon branch per parent. For $c < 1$, Updates is $c$ times 1,875,300 plus the 93 times 400 persistence updates, paid once. For $c \geq 1$ every branch runs its own persistence window, so Updates is $c$ times 1,912,500. Trace columns give the fraction of the target trajectory read. The first column group uses only the checkpoint, and the second reads the target's own future. Unlike in Table 1, the loss model here reads the target trace.

| | | Prospective from checkpoint | | Reads target trajectory | | | | |
|---|---|---|---|---|---|---|---|---|
| $c$ | Updates | Restart | Base | Trace | Direct | Trace | LC-PFN | Loss LOO |
| 0.10 | 224,730 | 0.677/0.215 | 0.500/0.222 | 0.10 | 0.742/0.172 | 0.10 | 0.532/0.235 | 0.688/0.163 |
| 0.25 | 506,025 | 0.750/0.172 | 0.500/0.222 | 0.25 | 0.839/0.108 | 0.25 | 0.758/0.204 | 0.893/0.081 |
| 0.50 | 974,850 | 0.863/0.097 | 0.500/0.222 | 0.50 | 0.952/0.032 | 0.50 | 0.968/0.194 | 0.898/0.087 |
| 0.75 | 1,443,675 | 0.887/0.075 | 0.500/0.222 | 0.75 | 0.984/0.011 | 0.75 | 0.982/0.194 | 0.906/0.086 |
| 1 | 1,912,500 | 0.903/0.065 | 0.500/0.222 | 1 | 1.000/0.000 | 0.75 | 0.982/0.194 | 0.906/0.086 |
| 2 | 3,825,000 | 0.933/0.056 | 0.500/0.222 | 1 | 1.000/0.000 | 0.75 | 0.982/0.194 | 0.906/0.086 |
| 4 | 7,650,000 | 0.948/0.050 | 0.500/0.222 | 1 | 1.000/0.000 | 0.75 | 0.982/0.194 | 0.906/0.086 |
| 8 | 15,300,000 | 0.962/0.047 | 0.500/0.222 | 1 | 1.000/0.000 | 0.75 | 0.982/0.194 | 0.906/0.086 |
| 16 | 30,600,000 | 0.963/0.046 | 0.500/0.222 | 1 | 1.000/0.000 | 0.75 | 0.982/0.194 | 0.906/0.086 |

across actions. Of the 19 parents, 16 gain and none lose, and the selected action ties the best fixed action on all 19. Training loss is not a substitute for the branch score: Choosing instead the action that leaves the lowest training loss after its 25 updates converts 8 of 114 held-out branches. The 48-parent replication uses two parent sets. The primary set is chosen by the first cohort's low-forecast rule, and on it the rates are 0.677 versus 0.345, the gain is +0.332 [0.259,0.405] with 36 positive and 0 negative parents, and the worst action gives −0.300. On an unscreened set, the rates are 0.858 versus 0.620, a paired gain of +0.238 [0.162,0.317], with the worst-rated action reversing the gap to −0.196. In both sets the lowest-rated action reverses the sign, and on the primary set the selected action ties the best fixed action on all 48 parents without using held-out outcomes. A per-parent oracle bounds what a finer choice could add: picking each parent's best of the four actions on its held-out branches converts 92 of 114 in the first cohort against 91 for the single best action, and 247 of 288 and 195 of 288 in the two 48-parent cohorts, equal to the single best action. The oracle selects on the same noisy counts it is scored on, so even these small gains are upper bounds. Because one action is best for every parent in this library, testing whether the score can assign different actions to different parents requires a more varied library.

# E Compute and reduced estimators

In this appendix we give the compute accounting behind Section 4.2 and describe the cheaper estimators we tested: a readout of the checkpoint that branches only the uncertain parents, fewer or shorter branches, and estimators that learn from partial trajectories. We close with what running the protocol on released checkpoints would require.

## E.1 Equal update budgets and decision time

We count scheduled optimizer updates after the shared parent and count evaluation calls separately. Restart estimates never read the target's trajectory after the checkpoint, whereas direct continuation and the paid-trace predictors do, and this distinction organizes Tables 9 and 10, which give every budget and decision-time row behind the comparison in Section 4.2. Because direct continuation watches the run being scored, it wins at equal compute: it has higher AUC and lower Brier at every tested equal-update budget (Table 9), although at equal decision times the two differ significantly at only one deadline, three quarters of the horizon, where the continuation leads by 0.039 AUC (Table 10). The budgets in these tables count optimizer updates, while the token ratios below count input tokens. Reading the target directly resolves that one run, but it gives no probability for the other futures of the state.

Recorded observables cost far less than restarts, and the best ICL and modular observable fields reach AUC 0.798 and 0.834, while the two-branch restart schedule costs 47.0 and 757.4 times as many input tokens, respectively. Loss costs nothing under this convention, and a leave-one-out loss model on the 93 panel parents, given only pre-checkpoint data, falls below chance, with AUC

0.443, so its advantage in Table 9 comes from reading the target. Branches must also run to the horizon: when they are stopped early and scored on whether they have acquired so far, most of the signal is lost. Across a complete $4 \times 4$ replay, transfer AUC is 0.526, 0.658, 0.868, and 1.000 at horizon fractions 0.25, 0.50, 0.75, and 1.00, unchanged across widths 2, 4, 8, and 16. At a quarter of the horizon the interval includes 0.5. The same holds at equal compute, where depth beats width, since four full-horizon branches cost 7,650,000 updates versus 8,096,400 for sixteen quarter-horizon branches, with paired AUC gap +0.131 [0.018,0.247] for depth. Adaptive thinning is a third route to lower cost: replayed forward on 16 held-out modular-addition parents, a Horvitz–Thompson two-phase rule saved 55.60 percent of tokens against full event-stopped branching, with a whole-parent interval of 49.38 to 60.10 percent, although the full-budget AUC there was already 1.000, so this replay cannot show that accuracy is preserved.

### E.2 A CHEAP READOUT WITH BRANCHING ON UNCERTAIN PARENTS

The readout is an L2-regularized logistic regression on ten behavioral features of the step-100 checkpoint: accuracy, loss, and margin on held-out in-context probes; loss and accuracy on memorization probes and on training batches; the stored loss of the last training batch; and the rate and margin with which the model follows the context on conflict episodes, whose context shows a training mapping with its labels flipped. Fitting it requires labeled parents, so we generated 400 new parents (seeds 3000 to 3399) through the panel's unchanged preparation path and labeled each by its own continuation to the horizon, and 133 acquire. We chose the feature set, model, and regularization by five-fold cross-validation, repeated five times, on these 400 parents only, and scored the panel once. Alone, the readout reaches AUC 0.816 on the 72 in-context-learning parents regenerated from the panel seeds, against 0.920 for the full 16-branch restart estimate. This is real signal rather than selection noise: rerunning the whole selection and refit on 200 permutations of the 400 training labels gives mean panel AUC 0.505 and at most 0.743 ($p = 0.005$), and the readout's paired gap to the committor is $-0.104$ [$-0.227$, 0.019].

The cascade spends branches only on the parents whose readout probability is closest to 0.5, where the readout is least sure. The remaining parents keep the readout's order, with the low tail placed below and the high tail above the branched middle, which the restart estimate orders. Table 11 gives every branched fraction and width, and each subsampled width averages 300 random branch subsets. Because we chose the fraction after seeing the results, we report the whole curve. With half the parents branched at 16 branches, AUC is 0.944, a difference of +0.023 [$-0.040$, 0.110] from the full estimate and +0.128 [0.048, 0.229] from the readout alone (whole-parent bootstrap). The costs exclude the 400 labeled runs, which are paid once and then serve every later forecast.

### E.3 FEWER BRANCHES

Beyond the cheap readout, two further routes to a cheaper estimate reproduce the full one on the cohorts we tested. Over 65 runs in 26 initialization clusters, two full-horizon branches reproduce the sixteen-branch committor with $R^2$ of 0.998 and Spearman 0.980 for one eighth of the forward-simulation cost, and four anchor checkpoints out of a dense grid of twelve reconstruct one language-model committor curve at mean absolute error 0.10 in $q$ with a mean onset error of 16 training steps for the natural-text model of Appendix F.1. Each was tested on one cohort.

### E.4 RELATION TO OTHER ESTIMATORS

Other estimators also run continuations from a checkpoint, and it is useful to say exactly when they coincide with ours. Because the estimator in Equation 2 counts only which branches have reached $E$ by the horizon, it equals a continuation ensemble with the same information and compute. With a matched checkpoint, deadline, record set, and reseeded branches, the endpoint-counting estimator and a matched-width continuation ensemble average the same endpoint indicators, and their parent scores agree exactly, with paired AUC gap 0.0 [0.0, 0.0] at every tested decision time and matched width. The equality need not hold for estimators that learn from partial trajectories, as temporal-difference (TD) methods do, and a reduced fitted-TD estimator breaks it by scoring held-out prefixes without advancing any held-out branch to the horizon or consuming any held-out endpoint indicator.

Table 10: **Forecast scores at equal decision times.** Depth is the number of updates each system has run at decision fraction $\tau$, and cells are AUC/Brier. Restart trails direct continuation by at most 0.039 AUC at every deadline.

| $\tau$ | ICL depth | Modular depth | $S_5$ depth | 16-way restart | Direct |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0.500/0.333 | 0.500/0.333 |
| 0.10 | 990 | 5,000 | 5,750 | 0.737/0.193 | 0.742/0.172 |
| 0.25 | 2,475 | 12,500 | 14,375 | 0.817/0.148 | 0.839/0.108 |
| 0.50 | 4,950 | 25,000 | 28,750 | 0.927/0.069 | 0.952/0.032 |
| 0.75 | 7,425 | 37,500 | 43,125 | 0.945/0.056 | 0.984/0.011 |
| 1 | 9,900 | 50,000 | 57,500 | 0.963/0.046 | 1.000/0.000 |

Table 11: **Cascade AUC by branched fraction and branch width** on the 72 in-context-learning panel parents regenerated from the panel seeds. Fraction 0 is the readout alone, and fraction 1 at width 16 is the full restart estimate. Branch cost relative to the full estimate is the fraction times the width over 16.

| Width | 0 | 0.2 | 0.3 | 0.4 | 0.5 | 0.7 | 1 |
|---|---|---|---|---|---|---|---|
| 16 | 0.816 | 0.830 | 0.879 | 0.896 | 0.944 | 0.920 | 0.920 |
| 8 | 0.816 | 0.830 | 0.876 | 0.888 | 0.933 | 0.905 | 0.897 |
| 4 | 0.816 | 0.830 | 0.874 | 0.883 | 0.924 | 0.893 | 0.877 |
| 2 | 0.816 | 0.829 | 0.872 | 0.878 | 0.912 | 0.871 | 0.843 |

### E.5 Short-prefix fitted TD

The reduced TD study fits a clock-indexed ridge value recursion on one development trajectory from each of sixteen modular parents, and scores sixteen held-out parents from two independent $0.05H$ prefixes, so that it sees only the opening of each held-out run. The held-out set contains seven acquiring and nine non-acquiring parents, and Table 12 lists the scores beside the continuation baselines.

Fitted TD correctly orders 53 of 63 acquiring–non-acquiring pairs, with AUC 0.841 [0.587,1.000] and Brier 0.283 [0.204,0.361]. Relative to the full estimate, its AUC gap is −0.159 [−0.413,0] and its Brier excess is +0.282 [0.203,0.361]. With the fit shared across 100 queries, TD uses 1.62 percent of full scheduled updates and improves AUC over direct continuation by +0.341 [0.087,0.500], but the strongest equal-cost Brier improvement is only +0.026 [−0.174,0.037]. The shortfall is specific to this fitted-TD estimator, and TD methods with richer fits remain open. Watching the target is a strong comparator for a different reason: At three-quarters of the target trace, 30 of 31 positives and 2 of 62 negatives have already crossed, so the costless already-crossed indicator has AUC 0.968, but it answers only after most of the horizon has passed.

### E.6 Running the protocol on released checkpoints

The protocol needs saved optimizer state, which holds the fp32 master weights of mixed-precision runs and the Adam moments that carry keeps, so the first question for any released model is whether that state was published. Public optimizer-bearing state is available for standard Pythia 1B at all 154 checkpoints and for standard 2.8B through step 25,000, for OLMo 2 (Team OLMo et al., 2024) at 1B with full trainer state, for DataDecide 1B at a few steps without trainer state, for CrystalCoder 7B (Liu et al., 2023) in its native format, and for terminal or near-terminal BLOOM (BigScience Workshop et al., 2022) states from 1.1B upward. Amber (Liu et al., 2023) never saved optimizer state and no usable public K2 optimizer state was found. Where the state exists it loads exactly: A CPU-only check of public Pythia 14M step 1,000 mapped all 76 non-rotary model tensors to the independently published safetensors and loaded four Adam moment tensors, with every compared tensor matching bitwise.

The cost model charges six times parameters times tokens per continuation at 25 percent utilization on four RTX 4090 GPUs, so one 1,000-update continuation of Pythia 1B costs about 91 GPU-hours (63 dollars). On that basis, a two-parent pilot of ten continuation equivalents costs about 913 GPU-

Table 12: **Short-prefix fitted-TD scores and continuation baselines** on the sixteen held-out modular parents.

| Method | AUC [95 percent interval] | Brier [95 percent interval] |
|---|---|---|
| Fitted TD, two $0.05H$ prefixes | 0.841 [0.587, 1.000] | 0.283 [0.204, 0.361] |
| Full endpoint count, $B = 16$ | 1.000 [1.000, 1.000] | 0.00024 [0, 0.00073] |
| Direct continuation, 100-query allowance | 0.500 [0.500, 0.500] | 0.438 [0.188, 0.688] |
| Development base rate | 0.500 [0.500, 0.500] | 0.309 [0.215, 0.402] |
| Direct continuation, realized 16-query allowance | 1.000 [1.000, 1.000] | 0.000 [0.000, 0.000] |

hours (630 dollars), the 560-equivalent held-out test about 51,145 GPU-hours (35,290 dollars), and the full fitting, calibration, and test program of 1,827 equivalents about 166,877 GPU-hours (115,145 dollars), or 125,140 GPU-hours at the DataDecide per-update compute ratio of about 0.75. These figures exclude evaluation, storage, and failed runs.

# F Released-model evidence

In this appendix we give the protocols and full results for the natural-text model and the released models of Section 4.5. We begin with a small language model that we train from scratch on natural text, then examine the order and shape of capability onset across released families, and end with restarts from released checkpoints and a steering pulse on one of them. All of this evidence is analyzed separately from the 93-parent panel.

## F.1 A language model trained on natural text

Unlike the released models in the rest of this appendix, this is a small model that we train from scratch, to carry the committor protocol from the controlled systems to natural text. It is a GPT with 4 layers, width 384, six attention heads, a context of 512 tokens, an 8k-token byte-pair vocabulary and tied input and output embeddings, about 10M parameters in all. It trains on FineWeb-Edu (Penedo et al., 2024) with AdamW at learning rate $3 \times 10^{-4}$, betas 0.9 and 0.95, weight decay 0.1, and batches of 64 sequences of 512 tokens. The capability we track is induction, the use of earlier tokens of the context to predict later ones. The induction score of a checkpoint is its mean loss at context positions 40 to 60 minus its mean loss at positions 480 to 512 (Olsson et al., 2022), so it grows as the model uses earlier tokens of the context. We evaluate it every 25 updates, and a run acquires when the score reaches 0.4 nats and holds for two evaluations. Each parent trains for up to 300 updates. From each of its checkpoints we launch 16 reset branches with fresh AdamW moments and a reseeded data order, and each runs 50 updates. Every estimate was sealed with a hash and a timestamp before any outcome was scored.

The forecast is scored on whether the parent's own run acquires within the 50 updates that follow the checkpoint. Over the 170 checkpoints taken before each parent's own event, from 60 parents in 8 independent initializations, the committor reaches AUC 0.864 [0.752, 0.982], with the interval from a bootstrap over initializations, and Brier 0.173 against 0.230 for the base rate. Most estimates lie close to 0 or 1, since 148 of the 170 are at most 0.15 or at least 0.85, and estimates of 0 acquire 2 of 77 and estimates of 1 acquire 42 of 59. Estimates of 1 overstate the parent's chance within the window because the horizon is short: the parent keeps its optimizer state while the branches restart theirs, and it sometimes crosses the bar a little after the 50-update window closes. Within one initialization the estimate tracks the approach to acquisition, since the mean over 16 parents rises from 0.008 at step 40 to 0.227 at step 70, 0.738 at step 100 and 0.996 at step 150. The anchor reconstruction of Appendix E.3 uses these curves.

## F.2 Onset across size and seed

Section 4.5 takes onset as the step at which a model's zero-shot ARC-Easy accuracy crosses 40 percent of its normalized range, which runs from the mean of its three lowest checkpoint accuracies to the mean of its three highest, with the crossing interpolated in log step. On the standard Pythia ladder the onset relation extrapolates: leaving out one size at a time, it predicts each held-out size

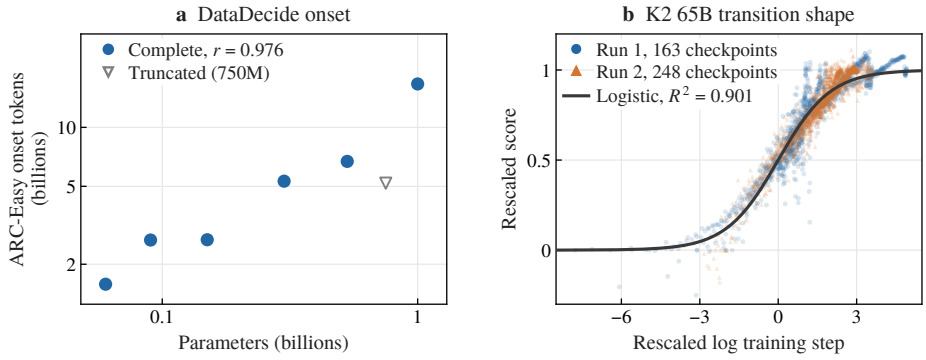


Figure 5: **Onset order and transition shape in other released families.** **a**, ARC-Easy onset in training tokens against parameter count on DataDecide, with the truncated 750M run as an open marker. **b**, Two K2 65B runs on seven benchmarks, rescaled to each run's onset, against a parameter-free logistic.

within a mean factor of 1.062, which none of 2,000 size-label shuffles reaches. For the Pythia onset relation, a size-label permutation null has median leave-one-size-out factor 1.347, against the observed 1.062. The relation also replicates on the deduplicated Pile, where the exponent is 0.16 with held-out errors of 1.08 and 1.04 times.

Three other released families extend this picture to independent runs (Figure 5). The ordering replicates on independent DataDecide records, where onset is strictly monotone across six sizes from 60M to 1B (Figure 5a), and on DataDecide the size-label permutation probability is 0.0014, and including the truncated 750M run gives Spearman $\rho = 0.89$ and probability 0.0015. We also test whether loss recovers this ordering. The DataDecide loss comparator works on complete size-by-corpus run triples, and among 350 such triples, 248 admit at least one loss-ranked endpoint. Across those 248, the primary loss AUC is 0.510 [0.462, 0.555] and the orientation-free leave-one-corpus-out AUC is 0.547 [0.495, 0.597], with the 25 corpora resampled as whole clusters.

Within a single size, released runs differ mainly in when they acquire. At a common BLiMP bar (Warstadt et al., 2020), all 47 released PolyPythias runs from 14M to 410M acquire by the end of released training, and the spread in their onset narrows with size, from a coefficient of variation of 20.1 percent at 14M to 5.5 percent at 410M. A stricter bar does separate PolyPythias seeds. Under a sustained-crossing rule, the same family splits at four of five sizes, with 8 of 10 seeds acquiring at 31M, 4 of 9 at 70M, 7 of 9 at 160M, and 8 of 9 at 410M, and the ten never-acquirers have flat tails, maximum absolute slope 0.00072 against a flatness bar of 0.002. We chose the per-size bars from a fixed grid after seeing the fractions, so this split is exploratory. It shows that a released family can differ in outcome from seed to seed, which the restarts below test at the level of single checkpoints. DataDecide shows the same across many recipes: in 397 of 3,500 task cells, a fraction of 0.1134 [0.1051, 0.1217], some but not all of three released reruns of the same recipe acquire the endpoint by the horizon. This split fraction peaks at 0.208 at 10M and falls to 0.032 at 1B, but at 1B the acquisition prevalence is 0.983, so the low 1B split is a ceiling effect.

At 65B the transition also keeps its shape from one run to the next. The 65B family is LLM360 K2 (Liu et al., 2025), with two training runs of 163 and 248 logged checkpoints. The seven benchmarks are GSM8K, HumanEval, MBPP, BBH, MMLU, ARC-Challenge and HellaSwag, and TruthfulQA-mc2 serves as the flat control. The cross-run overlay $R^2$ rescales each run's steps to its own onset and reports the $R^2$ of both runs' points against one parameter-free logistic in that coordinate. In that coordinate, across the two released K2 65B runs, seven benchmarks reach a mean cross-run overlay $R^2$ of 0.901, against 0.571 for a flat TruthfulQA control (Figure 5b), and the seven K2 benchmarks range from 0.774 to 0.964 in overlay $R^2$.

Table 13: **Released-model restarts.** Left, Pico-65M restart estimates from sixteen branches at six released checkpoints of one model. Right, Pythia 14M restarts of four parent seeds at step 1,000, scored at step 1,750 against an induction-score threshold of −2.5 nats. Clusters give the range of wave-1 endpoint scores in nats, and wave 2 uses disjoint data-order seeds. Seed 4 is the only parent whose branches mostly fail to cross the bar in both waves.

| Pico step | Acquired | Pythia seed | Wave 1 | Wave-1 cluster (nats) | Wave 2 |
|---|---|---|---|---|---|
| 2,000 | 0/16 | 1 | 4/4 | [−5.313, −5.213] | 2/2 |
| 3,000 | 0/16 | 2 | 4/4 | [−4.785, −4.732] | 2/2 |
| 4,000 | 16/16 | 3 | 4/4 | [−6.290, −6.042] | 2/2 |
| 5,000 | 16/16 | 4 | 1/3 | [−2.526, −2.348] | 0/2 |
| 6,000 | 16/16 | | | | |
| 7,000 | 16/16 | | | | |

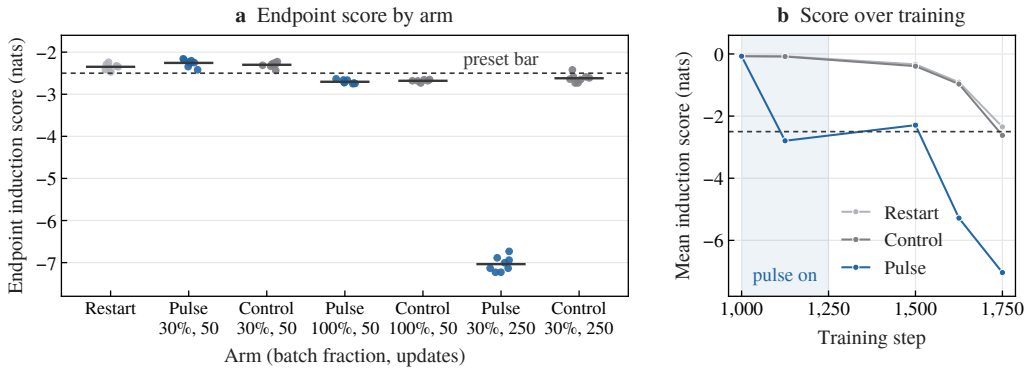


Figure 6: **Steering a released Pythia 14M checkpoint.** **a**, Endpoint induction score at step 1,750 for every branch from the seed-4 step-1,000 parent, with bars at arm means. Pulses inject repeated 48-token segments for the stated batch fraction and number of updates, and each control receives the same segments permuted within sequence. **b**, Mean score trajectories, with the pulse shaded. The dashed line is the preset −2.5 nat bar.

### F.3 Restarts and steering

We restart Pico-65M (64.6M parameters) from six checkpoints of one released model with sixteen branches each, and Pythia 14M from four parent seeds at step 1,000 in two waves. Table 13 gives both restart sets, and Figure 6 shows every arm of the steering experiment that follows from them.

**Pico.** The Pico induction score is the drop in negative log-likelihood on a repeated segment relative to its first occurrence, in nats, so a higher score means stronger induction, and acquisition requires the score to hold for three consecutive evaluations. At the step-3,000 parent, repeated-segment injection has sustained-three endpoints 1.616–1.938 nats, a unigram-identical shuffled arm has 0.180–0.365, and untreated continuation has 0.442–0.619. Each paired contrast is directional on all eight shared seeds, with per-contrast one-sided probability $1/256$. Every steered branch ends above 1.6 nats and every untreated branch below 0.62, and the natural committor is 0 of 16 at every bar from 0.8 to 2.0 nats and 1 of 16 at 0.6, so at any bar between 0.62 and 1.6 nats the steered arm converts 8 of 8 against 0 of 16. The declared 2.0-nat event lies above every branch, so we report the Pico result as a separation of endpoints at one parent. The control here is not matched in parameter norm, which the Pythia experiment below addresses.

**Pythia 14M restarts.** The Pythia score has the opposite sign, so lower means stronger induction. Each restart carries the fp16 weights and fp32 master weights of a step-1,000 parent, zeros both Adam moments, continues the learning-rate clock, and reseeds the data order. Against a threshold of −2.5 nats at step 1,750, seeds 1–3 each convert 4 of 4 continuations and seed 4 converts 1 of 3. The exact parent-by-outcome homogeneity probability is $1/35$, and the four endpoint-score clusters are disjoint. The largest within-parent range is 0.248 nats, smaller than every gap between adjacent parents. A second wave with disjoint branch seeds repeats this, and seeds 1–3 each convert 2 of 2 and

seed 4 converts 0 of 2. Every endpoint falls within 0.0424 nats of its parent's first-wave cluster, and the pooled homogeneity probability is $1/1771$. These are four independent parents with branches nested inside them.

**Pythia 14M steering.** A wider screen under the same protocol identifies the parent to steer. Four restarts from each of 17 released parents give 0 of 4 for all ten seeds at step 512 (endpoints $-0.13$ to $-0.36$ nats at step 1,262), 4 of 4 for seeds 0, 5, 6, 7, 8 and 9 at step 1,000 (endpoints $-5.0$ to $-6.6$ nats), and 0 of 4 for seed 4 at step 1,000 with endpoints $-2.20$ to $-2.32$ nats. This is the third independent restart set to place seed 4 near and mostly short of the bar. The step-512 parents are too early for a 750-update horizon, so seed 4 at step 1,000 is the low-probability checkpoint, and we steer it. The pulse repeats the Pico treatment: 48-token random segments are repeated to fill each 2,048-token sequence and replace part of each batch for a fixed number of updates. The control keeps the same segments and schedule but permutes each sequence, so the two arms differ only in repetition. Figure 6 shows every arm.

We tried two ways of strengthening the pulse, higher intensity and longer duration, and only duration separated the arms. At the ICL fraction (30 percent of the batch for 50 updates, here 160 of 512 sequences, or 31 percent), the pulse converts 0 of 8 against 0 of 8 controls and 0 of 8 fresh restarts, with endpoint means $-2.26$, $-2.30$ and $-2.35$ nats and no separation at any scoring step or on any secondary readout. On step-512 seeds 0 and 1 the same three arms convert 0 of 45 branches, and we stopped the remaining step-512 arms. Replacing every sequence for 50 updates does not help either, since 8 of 8 pulses and 6 of 6 controls cross, at $-2.70$ and $-2.68$ nats, with a pulse-minus-control difference of $-0.02$ nats. That shift comes from the injected tokens, which the control shares. Two of the eight control branches in this arm were not run. A longer pulse at the same fraction separates the arms. Holding that fraction for 250 updates converts 8 of 8 branches at $-7.04$ nats, range $-7.23$ to $-6.73$, against $-2.62$ nats for the matched controls and $-2.35$ for restarts, a pulse-minus-control difference of $-4.42$ nats with disjoint ranges on every branch. The pulse also lowers the natural-text repeated-span score by 0.31 nats, the period-24 and period-96 probes by 4.16 nats, and endpoint training loss by 0.092. The score crosses the bar while the pulse is on, rises back above it after the pulse stops, and then moves well beyond both controls, and the secondary readouts at other periods and on natural text move with it. At every bar between $-2.75$ and $-6.7$ nats the pulse converts 8 of 8 against 0 of 8 controls and 0 of 8 restarts. The $-2.5$ bar sits only 0.15 nats below the restart mean, so 7 of 8 controls also cross it, and we therefore report the effect as a separation of endpoints at one parent, with matched controls at every dose.