DECIDED EARLY, SEEN LATE: THE COMMITTOR OF NEURAL NETWORK TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training a neural network compiles one global number, the loss, into millions of weights, and one number cannot pin them all down: two runs of one recipe can fit their data equally well and still differ in whether they generalize (Zhao et al., 2025) or in which attention head carries a capability such as in-context copying (Olsson et al., 2022). We ask which source of chance makes that choice, when it is made, and whether anything in the network shows it while it is being made. To answer all three, we copy a checkpoint into replicas that differ only in the randomness that follows, as chemical physics does to estimate a committor, and divide the variance of an outcome into the part chance has yet to decide, the part already decided that a description of the network can read, and the part it cannot. The instrument recovers what was known by other means, such as early stability to training noise and the equal effect of every source of randomness on a ResNet's errors. It also shows that (1) equally good runs differ along only a few directions, along which chance moves a run, (2) which source of chance decides depends on the recipe, (3) outcomes which are decided at all are decided early, at a median of 7.5% of training in our recipes and long before they show in the run, and (4) only a task readout or a projection of the network's outputs reads the decision as it is made, whereas the diagnostics commonly used to watch training read it thousands of steps later yet pass the usual within-run test of a precursor. The instrument needs nothing more than the ability to restart a run from a checkpoint, so one can use it to date when an outcome of their own recipe is decided, to check whether a proposed diagnostic reads that decision or only its aftermath, and to forecast whether a run will generalize by letting a few replicas run briefly.

1 INTRODUCTION

Two runs of the same training recipe can fit their training data equally well and still end up as different networks (D'Amour et al., 2022; Sellam et al., 2022; van der Wal et al., 2025). One generalizes while the other never does (Zhao et al., 2025), a capability is implemented by a different circuit (Chughtai et al., 2023; Zhong et al., 2023) or carried by a different attention head, or a different set of test images is misclassified (Jiang et al., 2022; Jordan, 2024), and none of these differences shows in the training loss. The loss is one number, and one number cannot pin down millions of weights, so it fixes only a class of networks that fit the data equally well, and which member of that class a run reaches is left to the chance in the recipe, such as the initialization, the order of the minibatches, or even the convolution kernel that the numerical library selects (Dodge et al., 2020; Summers & Dinneen, 2021; Zhuang et al., 2022) (Figure 1a). Practitioners meet this as variance across seeds and handle it by training several runs (Bouthillier et al., 2021), which treats the choice as noise to average over rather than as an event that happens at some point in training and could be observed.

In one of our testbeds, 64 one-layer transformers trained on addition modulo 97 from one initialization and one split of the data, differing in the order of their minibatches, split into 42 that learned to generalize and 22 that never did; we call this recipe testbed F, and the same transformer shown more of the addition table, so that every run generalizes, testbed A (Table 1 lists three more outcomes left open in this way, and Appendix B describes all our testbeds). Understanding such a choice means answering three questions, each approached by one literature. Which source of chance makes it is studied through seeds and underspecification (D'Amour et al., 2022; Summers & Dinneen, 2021), when it is made through the stability of training and its critical periods (Frankle et al., 2020a; Achille

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.

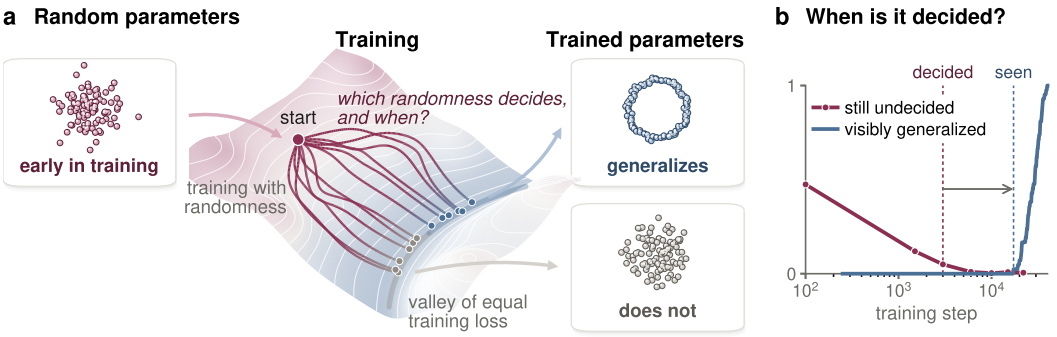


Figure 1: The question and, on one recipe, its answer. (a) Training turns random parameters into structured ones, and the loss alone does not decide which structure a run reaches: the token embeddings of testbed F, projected on their two principal directions, form a ring at the end of training in the roundest of the runs that generalize (right, top) and not in the run with the lowest test accuracy (right, bottom), although both fit their training data; the middle is a sketch, not data. (b) Over the 64 runs of testbed F, the share of the outcome that chance has yet to decide (red) is already small at the decision (red dashed line), long before the first run generalizes (blue dashed line).

et al., 2019; Kleinman et al., 2024), and whether anything shows it while it is made through hidden progress and developmental stages (Barak et al., 2022; Nanda et al., 2023; Lau et al., 2025). None of them measures how much of a named outcome is decided at a given moment, so none can date a decision, attribute it to a source, or tell whether a precursor reads the decision or its aftermath.

We supply that quantity. Because a training run can be restarted from exactly the state it was in, we copy a checkpoint, with its optimizer state, into replicas that differ only in the randomness that follows, as chemical physics releases trajectories from one configuration to estimate its committor (Dellago et al., 1998; Bolhuis et al., 2002; Du et al., 1998). Comparing the spread among the replicas with the spread across independent runs divides the variance of an outcome into the part chance has yet to decide, the part already decided that a description of the network reads, and the part it cannot read, and the share of the decided part that a description reads is the committor test for a reaction coordinate. With a measured detection floor and the rules the measurement must obey, this instrument is the contribution of the paper.

Applied to our testbeds, the instrument first recovers what those literatures found by other means: replicas of one checkpoint end early in one basin (Frankle et al., 2020a; Fort et al., 2020), every source of randomness, down to the convolution kernel, spreads a network's errors equally (Summers & Dinneen, 2021), and the local learning coefficient changes before a run generalizes (Lau et al., 2025). It then shows what they could not. (1) In the testbeds where we measured it, equally good runs differ along only a few directions, and chance moves a run along them (Section 2; Appendix C). (2) On sparse parity neither the learning rule nor the sample of the data moves this class, and which source then picks the member depends on the recipe: the initialization mostly picks which attention head carries a capability until the data offer a shortcut, whereas in a ResNet any perturbation does equally well (Section 3). (3) Outcomes that are decided at all are decided early, at a median of 7.5% of training in our recipes and long before they show in the run, and an outcome made of many small pieces is decided more slowly than a single number (Figure 1b; Section 4). (4) Only a task readout or a projection of the network's outputs reads the decision as it is made: the general diagnostics used to watch training read it thousands of steps later, yet pass the within-run test by which precursors are usually validated, as does a random projection of the parameters, because that test measures the lead over the visible event rather than over the decision (Section 5).

2 The committor of a training run

The variance of an outcome splits into the part chance still decides, the part a description reads, and the part that is decided but unread. Fix a training recipe and a fork time t, let θ_t be the full state of a run at t, its parameters together with its optimizer state, and let Y be a named outcome of the finished run, such as whether it generalizes or which attention head carries a capability. A description

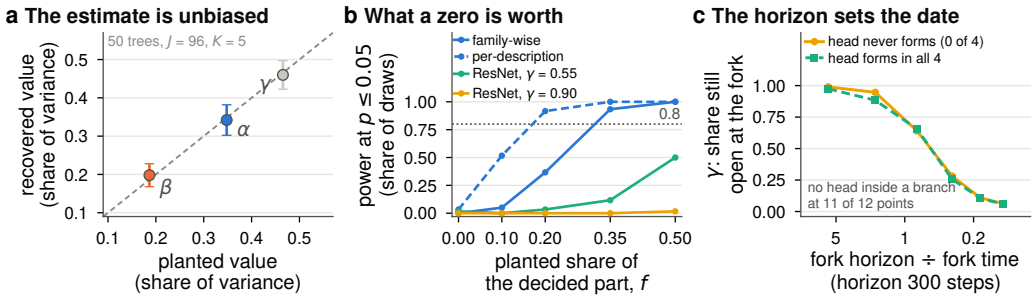


Figure 2: What a reading of the instrument is worth. (a) The estimator recovers all three terms on synthetic trees whose answer is known. (b) The power to detect a planted description carrying a share f of the decided variance, for testbed F under the family-wise and the per-description null, and for the ResNet. (c) With branches of fixed length, a recipe that never forms an induction head and one that always does fall on one curve, so the curve dates the branch length rather than the outcome.

of the network is any function $M_t = f(\theta_t)$ of that state, from the loss to a readout designed for the task. Taking the law of total variance first over the state and then over the description gives

$$\underbrace{\mathrm{Var}\big(\mathbb{E}[Y \mid M_t]\big)}_{\alpha\,\mathrm{Var}(Y)} + \underbrace{\mathbb{E}\big[\mathrm{Var}\big(\mathbb{E}[Y \mid \theta_t] \mid M_t\big)\big]}_{\beta\,\mathrm{Var}(Y)} + \underbrace{\mathbb{E}\big[\mathrm{Var}(Y \mid \theta_t)\big]}_{\gamma\,\mathrm{Var}(Y)} = \mathrm{Var}(Y), \qquad \alpha+\beta+\gamma = 1. \quad (1)$$

Here γ is the part on which replicas released from the same θ_t still disagree, which no forecast from the checkpoint can predict. For a binary outcome, $\mathbb{E}[Y \mid \theta_t]$ is the probability that a replica released from θ_t reaches the outcome, which is the committor of the checkpoint. Of the decided part $1-\gamma$, a description reads α and misses β, so we report its screening fraction $s(M_t) = \alpha/(1-\gamma)$, the share of what is already decided that it reads. A screening fraction of one is the test of a reaction coordinate in transition path sampling (Geissler et al., 1999; Peters, 2016) and of a macroscopic variable that screens off the microstate in theories of emergence (Rosas et al., 2024; Barnett & Seth, 2023), here stated on conditional means.

Each γ is measured on a fork tree: a set of independent runs, each copied at a grid of fork times into replicas that continue to the end of training. Because the replicas within a fork share θ_t exactly, a one-way analysis of variance gives γ without fitting anything, and α needs only a cross-validated forecast of each run's mean outcome from M_t, tested against a family-wise permutation null that repeats the whole search over descriptions, so that searching cannot inflate the result, and against a per-description null for each description on its own (Appendix B).

The estimate is unbiased, a zero means only that a description falls below a measured floor, and a date is valid only when every branch runs to the end of training. On synthetic trees whose answer is known, all three terms come back very nearly unbiased, and what bias remains pulls α down, so a screening fraction near one certifies screening off while one near zero certifies nothing by itself (Figure 2a). A zero means only that a description carries less of the decided variance than a detection floor, which we measure by planting descriptions of known strength: on testbed F a one-dimensional description must carry a share 0.35 of the decided variance to be found four times in five, and on the ResNet far more (Figure 2b). We call an outcome *decided* at the first fork time at which γ falls to 0.1 or below. Such a date is valid only when every branch runs to the end of training, because a branch of fixed length measures how far a perturbation spreads in that many steps, which shrinks with the fork time whatever the network does (Figure 2c).

3 Which source of chance decides the outcome

A fork holds everything fixed up to the fork time and then releases every later source of chance at once, which is what makes it a clean measurement of how much is still open. It is not, however, a measurement of which source made the decision. That question is answered by varying the sources in a factorial design, or by removing them one at a time, and the answer depends on the testbed.

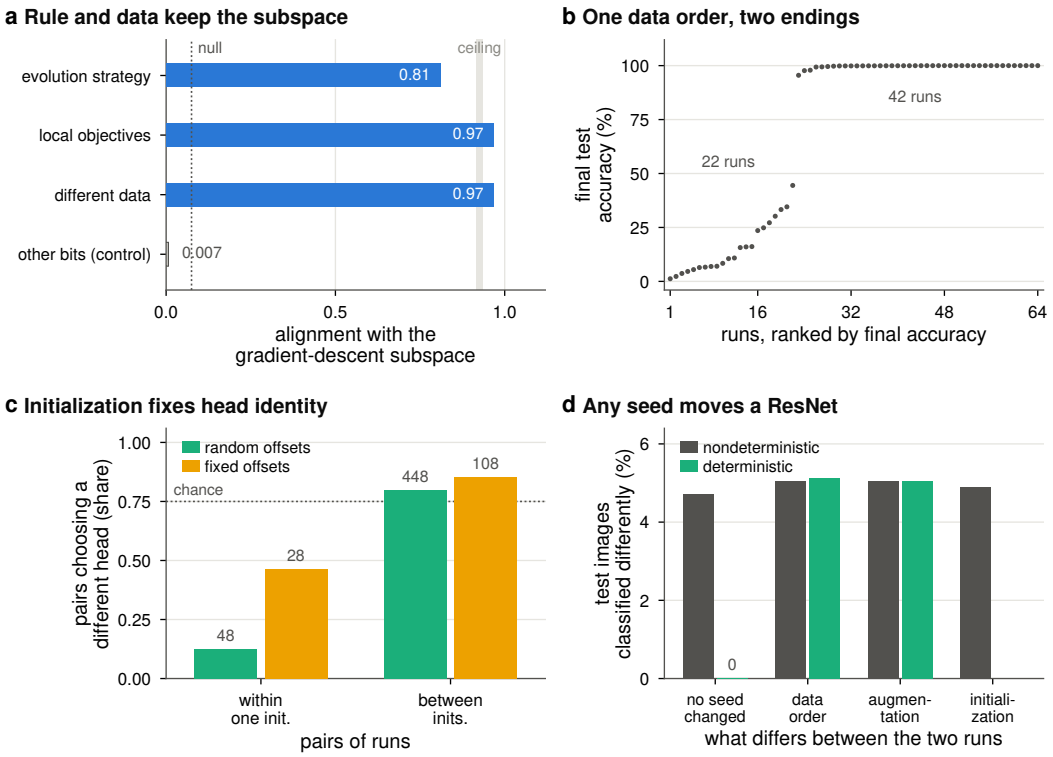


Figure 3: Which source of chance decides the outcome. (a) On sparse parity the subspace that separates independent runs survives a change of learning rule or of data sample, and moves, as a control, when the parity is on other bits. (b) Testbed F: 64 runs from one initialization and one data split, with the order of the minibatches as the only source of chance we vary, end in two groups. (c) Head identity: runs that share an initialization disagree far less often than runs from different initializations, until the data offer a positional shortcut. (d) A ResNet's errors: two runs differ on about one test image in twenty however they are perturbed, and not at all when the kernels are made deterministic and the seeds are held fixed.

Neither the learning rule nor the data sample moves the subspace a run ends up in. What fixes where a run ends up can be asked directly on sparse parity, by taking the subspace that separates independent runs and asking which ingredient of the recipe has to change before that subspace moves (Figure 3a). Replacing gradient descent by an evolution strategy that searches the same objective without gradients, replacing the shared objective by per-layer local objectives, or drawing an entirely different sample of the data leaves the subspace aligned far above two random subspaces of the same size, while a parity on a different three of the forty bits moves it, as the symmetry of the task requires, so that configuration only shows that the test can detect a move. This agrees with the observation that networks learning a sparse parity concentrate on its relevant coordinates (Barak et al., 2022), and adds that neither the learning rule nor the data sample changes which subspace a run ends up in, as far as random subspaces can tell. Which source of chance then picks the point inside that subspace changes from one testbed to the next. In testbed F every run shares one initialization and one split of the data, so the order of the minibatches is the only source of chance we vary, and varying it alone divides the runs into those that generalize and those that never do (Figure 3b); this is the cleanest answer in the section and also the narrowest, because the initialization was never allowed to vary. That the order of the data alone can decide the outcome agrees with Dodge et al. (2020), who found in fine-tuning that it matters about as much as the initialization.

The initialization mostly picks which head carries copying, until the data offer a shortcut. Induction heads (Olsson et al., 2022) and the conditions under which they form (Singh et al., 2024) are well studied, but we found no prior work that asks which of several interchangeable heads comes to carry the capability, or which source of chance decides it. In a small language model with six

layers of four heads each, the heads of a layer are interchangeable at initialization, so which of them comes to carry in-context copying can only be compared between runs that share an initialization. We trained several initializations under four data orders each, with the repeated spans at random offsets, and every run formed a genuine induction head in the same layer; runs that share an initialization rarely chose different heads, whereas runs from different initializations disagreed about as often as chance (Figure 3c). The initialization therefore largely, but not entirely, picks the head. The balance shifts when the data offer a competing solution: if the repeated span always begins at the same offset, a head in the first layer can copy by position instead of by content, a shortcut known to compete with induction heads (Kawata et al., 2025), some runs from a single initialization take that shortcut and others do not, and among the runs that still form a genuine head the data order splits the choice within an initialization. Because the two regimes also differ in batch size, learning rate and length of training, the comparison rests on the two regimes matched at a single initialization, described in Appendix B.

A ResNet's errors are decided by any perturbation, down to the choice of convolution kernel. Which of the test images a ResNet gets wrong is a question about which source of chance matters rather than how much is still open, so we answer it with a factorial design instead of a fork tree: 64 ResNet-18 runs, separate from the tree of Section 4, that combine four initializations, four data orders and four augmentation seeds. Changing any one of the three seeds by itself changes the answers on about one test image in twenty, and a random-effects analysis attributes almost all of that variance not to the seeds but to a residual (Appendix B). Two reruns identify the residual (Figure 3d): with all three seeds identical on the same model of GPU the answers still change on about one image in twenty, and with deterministic convolution kernels two such runs become bit-identical, so the residual is the kernel that the library happens to select. Yet under deterministic kernels, changing only the data order or only the augmentation seed still changes as many answers as before, so no seed is what selects the errors: any perturbation at all, whether a seed we set or a kernel the library chose, drives the pattern of errors to the same distance, while the accuracy that a practitioner watches barely moves. This agrees with Summers & Dinneen (2021), who found that every source of nondeterminism, down to a single bit of the initialization, produces a similar diversity of models, and with Zhuang et al. (2022), who found that the nondeterminism of the tooling alone leaves accuracy unchanged while moving the predictions on individual examples; what the fork tree adds, in Section 4, is that this pattern of errors is never decided anywhere on our grid of fork epochs.

4 When the outcome is decided

Wherever an outcome is decided at all, it is decided early in training, long before it becomes visible in the run. Knowing which source of chance decides an outcome says nothing about when it is decided, and the fork tree answers that second question directly. Every outcome has two dates, the fork time at which it is decided, as defined in Section 2, and the step at which it first becomes visible in the run itself, and the whole interest of the measurement lies in how far apart the two are. Across all our fork trees, about half of the cases, each one tree together with one named outcome, are decided on their grid, at a median of 7.5% of training, a share that reflects the training lengths we chose (Figure 4a; Appendix D). The gap to visibility is clearest in testbed F: no run generalizes before step 17250, and yet by step 3000 the replicas released from one checkpoint already agree about whether their run will ever generalize (Figure 4b). An early decision agrees with critical learning periods (Achille et al., 2019; Kleinman et al., 2024), and progress before it shows with Barak et al. (2022) and Nanda et al. (2023); what the fork tree adds is the date of each decision.

A ResNet-18 on CIFAR-10 sits at the far end of the pooled cases, since its runs agree almost exactly on how many test images they get wrong but not on which, and which images it gets wrong is still more than half open to chance at three fifths of training (Figure 4b). In a fine-tune of Qwen2.5-0.5B on Alpaca, a recipe we did not design, which injected facts a run retains is mostly decided soon after the injection stops (Figure 4b), adding a date per fact to studies of fact acquisition (Chang et al., 2024). On testbed A, the loss barrier between two replicas of one checkpoint vanishes soon after the outcome is decided, while that between independent runs does not, in agreement with the early linear mode connectivity of Frankle et al. (2020a) (Appendix B).

What differs between outcomes is how long the decision takes, and an outcome spread over many examples is decided more slowly than a single number. The comparison is cleanest inside a

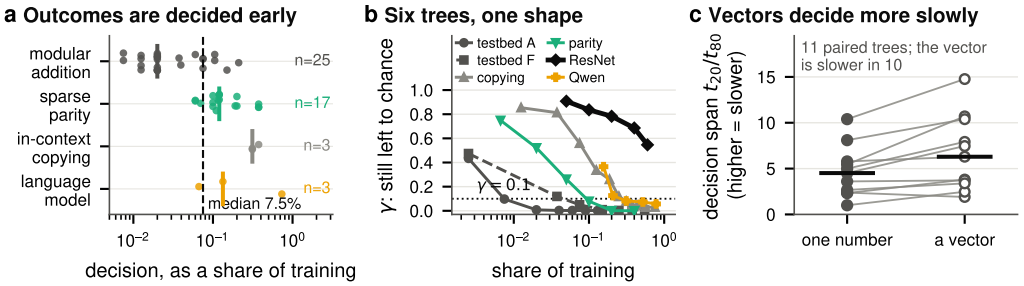


Figure 4: When the outcome is decided. (a) Every pooled case that is decided on its own grid of fork times, placed by the share of training at which it is decided and grouped by task family; the dashed rule is the median over all of them and the count beside each task family is the number of its cases that are decided. The ResNet appears in Table 4 of Appendix D but not here, because none of its cases is decided at all. (b) γ against the share of training for six trees: both modular-addition testbeds, the in-context copying tree, the reference sparse-parity tree, the ResNet-18 and the Qwen fine-tune. (c) Paired within a tree, an outcome that is a vector over examples or tokens takes longer to be decided than an outcome that is one number per run.

single tree, where an outcome that is one number per run, such as whether the run generalizes, and an outcome that is a vector over examples, tokens or frequencies are read from the very same runs. Measured by how many times the fork step must grow for γ to fall from 0.8 to 0.2, the vector is the slower in 10 of the 11 trees that carry both (Figure 4c). Whether the vector outcomes are also decided later these trees cannot tell, since in more than half of them both kinds of outcome are decided at the very same fork time and the few that differ are too few to tell, as Appendix D shows. An outcome made of many small pieces thus takes long to be decided, while the fate of the run as a whole is decided quickly.

On sparse parity the learning rule and the architecture move the date of the decision. What moves the date is a separate question, and Figure 5 answers it on sparse parity with a control and two contrasts. As a control, a parity on other bits or another sample of the data leaves the date at a tenth of training, as the symmetry of the task requires (Figure 5a). Changing the learning rule does move the date, whether it is counted in optimizer steps or as a share of training: per-layer local objectives decide only after a third of training has passed, and for a gradient-free search what matters is the layer into which the exploration noise enters, since with the noise in the first layer alone the outcome decides, while with the noise spread over every layer it never reaches the threshold anywhere on its grid (Figure 5b). Confining the noise to the last layer gives no date either, but that configuration carries no parity fingerprint and can be scored only on its final accuracy, so it is not a like-for-like comparison. The architecture moves the date as well, over a factor of thirty across six architectures when it is counted in optimizer steps but over only a factor of three when it is counted as a share of each architecture's own training length (Figure 5c,d). Since the length of training is our choice rather than a measurement, what this supports is that the spread in optimizer steps is much the wider of the two, and not that decisions fall at a fixed fraction of training.

5 What can see the decision while it is made

If the decision is made thousands of steps before anything visible happens, the natural question is whether anything in the network shows it at the time. A description is *readable* at a fork time when its share α is significant there and again at the next fork time. A precursor, in the sense the literature usually means, is a description that becomes readable before the outcome is visible; the question we ask is stricter, because it puts readability beside the decision rather than beside the visible event.

A task readout, or a projection of the outputs, reads the decision as it is made. Each of eight configurations of our testbeds carries a set of task readouts, descriptions designed for its own task, such as the Fourier spectrum of the embeddings for modular addition or the per-head induction scores for in-context copying, and in every one of the 11 cases in which both dates exist, the task readout is readable at or before the decision (Figure 6a). In 7 of these cases the outcome is itself the task

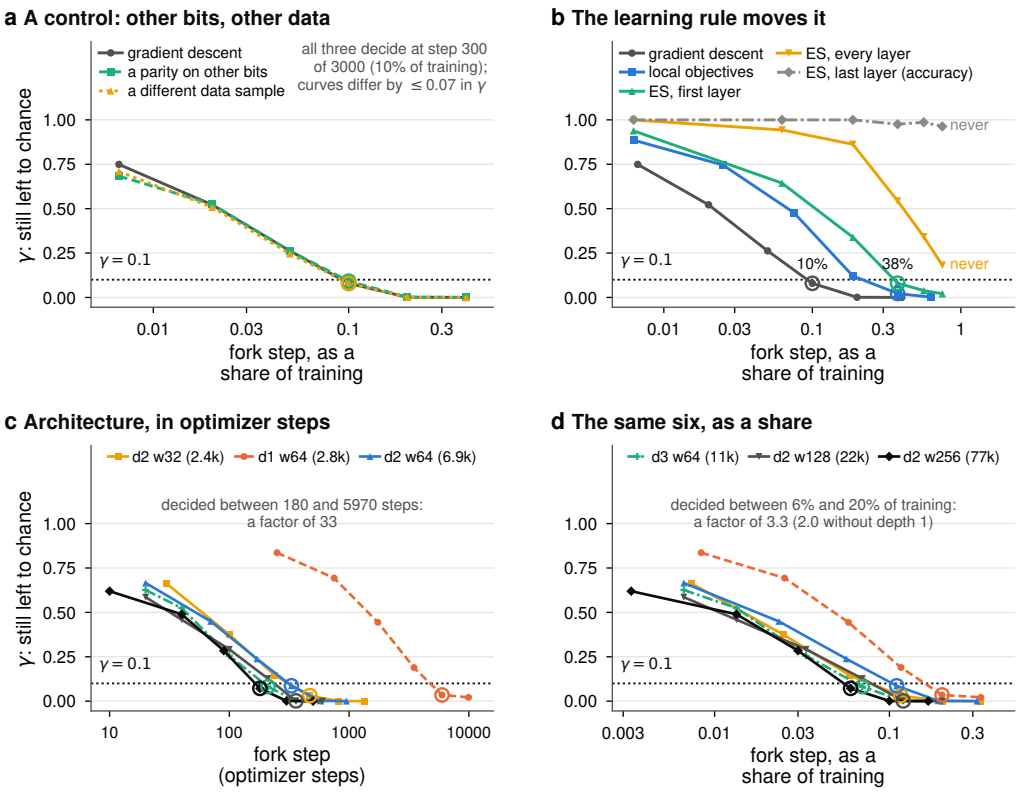


Figure 5: What moves the date at which an outcome is decided, and what does not; each curve is one fork tree on sparse parity, scored on its own grid of fork times, and rings mark the first fork time at or below the dotted rule at $\gamma = 0.1$. (a) Control: the reference parity, a parity on other bits and another data sample give one curve and one date. (b) The learning rule moves the date, and for a gradient-free search the layer into which the exploration noise enters decides whether there is a date at all; the last-layer configuration carries no structural fingerprint, so its curve is that configuration's final accuracy and is drawn dash-dotted. (c) Six architecture configurations against absolute optimizer steps. (d) The same six against their own training length, where the spread is far narrower.

readout's quantity at the end of training, so an early reading is expected, but the same holds in 4 of the 4 cases where it is not, such as a run's final accuracy and the step at which it generalizes. That such task readouts move before a run generalizes agrees with Barak et al. (2022) and Nanda et al. (2023), and the date adds that they move no later than the decision. The decision can be read from the network's outputs even through a random projection, but from its parameters only in coordinates aligned with the task: on testbed F a random projection of the outputs is readable as early as the Fourier summary, whereas a random projection of the parameters is readable only at the last fork time (Figure 6b; Appendix E).

The diagnostics used to watch training read the decision only thousands of steps later. On testbed F, at the decision the Hessian trace, the Fisher information and the gradient noise read nothing (Figure 6b), and because two independent estimates of each agree almost exactly, that zero is a real non-detection, bounded by the detection floor of Section 2. The local learning coefficient becomes readable before any run generalizes, in agreement with Lau et al. (2025) and Hoogland et al. (2025), which makes it a precursor in the usual sense, but thousands of steps after the outcome was decided. The three zeros are not a fault of the test, since at the same step of the same testbed the Fourier summary is read (Figure 6b), as is a task readout on a fine-tuned public model once its outcome is decided (Figure 6d; Appendix F).

The usual test of a precursor measures the lead over the visible event, not over the decision. The usual test asks, within each run, whether a description leaves the range spanned by the runs that never

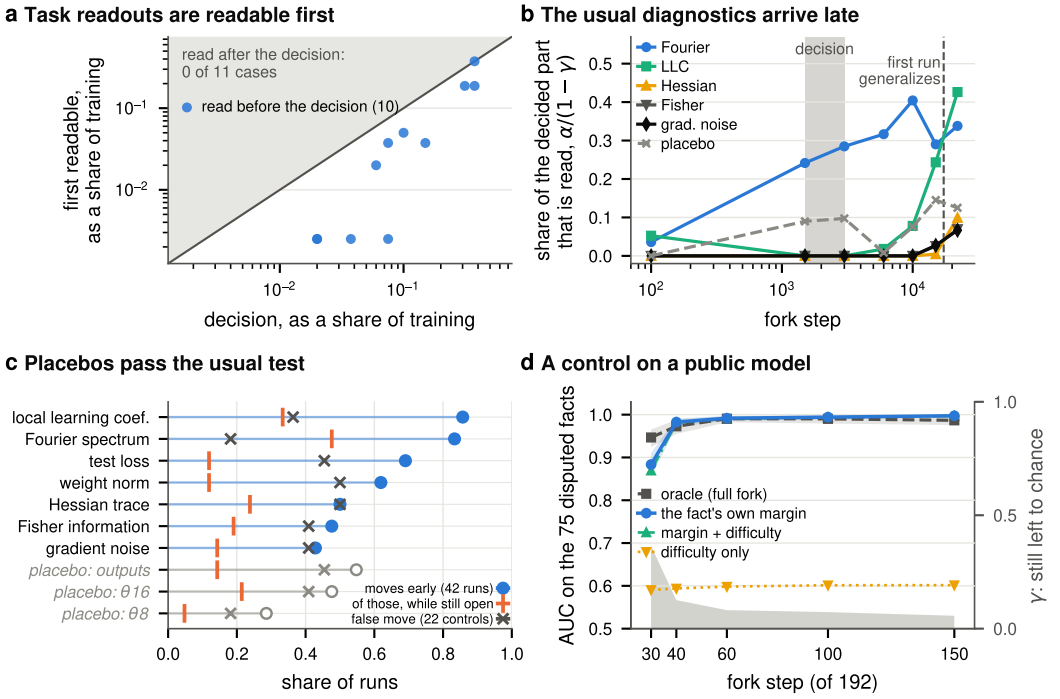


Figure 6: What can see the decision. (a) Across the 11 cases in which both dates exist, a task readout is readable at or before the decision. (b) Testbed F: the task readout carries the outcome at the decision, while the general diagnostics stay flat until thousands of steps later; the dashed placebo is a random projection of the outputs. (c) The test by which precursors are usually validated is passed by placebos, and mostly by moves that happen after the decision. The disc gives the share of the generalizing runs in which the description leaves the band spanned by the never-generalizing runs before that run itself generalizes, the tick gives the part of that share which happens while the outcome is still open, and the cross gives the same criterion applied to the runs that never generalize. (d) A positive control on a public model: a task readout reaches the committor oracle's own separation as soon as the outcome is decided.

generalize before that run itself generalizes (Figure 6c), and by that test the learning coefficient looks like a good precursor. The learning coefficient and the Fourier spectrum do beat a random projection of the outputs when the two are paired over runs, but a random projection of the parameters passes the test too, and more than half of the coefficient's early moves come only after the decision is complete. Running the test also on the runs that never generalize, as Boettiger & Hastings (2012a) prescribe for early warnings of critical transitions, exposes its false alarms, and the date of the decision adds a second check, which shows that the lead is counted from the visible event. The three tables behind these findings, with the controls and caveats that go with them, are in Appendix E.

6 Related work

Committors and reaction coordinates. Transition path sampling estimates the committor of a configuration by releasing trajectories from it, and accepts a coordinate as a reaction coordinate when configurations that share its value share a committor (Geissler et al., 1999; Peters, 2016). In glass physics the isoconfigurational ensemble releases many trajectories from one configuration and asks how much of the outcome its static structure predicts (Widmer-Cooper et al., 2004; Schoenholz et al., 2016; Bapst et al., 2020). In that literature neural networks appear as tools for computing committors and propensities. We reverse this: the training run is the system whose committor is computed, and dividing a named outcome's variance into the part still open, the part a description reads and the part decided but unread turns the reaction-coordinate test into a test of network descriptions.

Forking a training run. Copying a checkpoint and training the copies under different noise has shown that the copies come to lie in one linearly connected region early in training (Frankle et al., 2020a), that their functions stop drifting apart (Fort et al., 2020), and that a perturbation sends two trajectories apart when it is applied early but not late (Altıntaş et al., 2025). Critical learning periods show by intervention that what a network can still be taught is fixed early (Achille et al., 2019; Kleinman et al., 2024), and branching with the random state restored measures the response to an intervention (Lin, 2026). Our replicas agree with the first of these results, since two replicas of one checkpoint end in one basin soon after the outcome is decided. What these studies do not measure is how much of a named outcome is already fixed at a given checkpoint, and that quantity lets us date the decision for each outcome and separate it from the moment the outcome becomes visible.

Sources of chance. That two runs of one recipe can reach the same loss and still differ is underspecification (D'Amour et al., 2022), documented in seed studies of pretrained models (Sellam et al., 2022; Zhao et al., 2025) and in the circuits that networks build for modular arithmetic (Chughtai et al., 2023; Zhong et al., 2023). The variance of performance has been partitioned by source across benchmarks and in fine-tuning (Bouthillier et al., 2021; Dodge et al., 2020), every source of nondeterminism, down to the kernel the numerical library selects, is known to produce the same diversity of models (Summers & Dinneen, 2021; Zhuang et al., 2022), which our factorial design on a ResNet agrees with. These studies partition the variation of finished runs. By varying the sources of chance in a factorial design we identify which of them decides a named outcome, and find that the initialization mostly picks which attention head carries in-context copying (Olsson et al., 2022; Singh et al., 2024) until the data offer a positional shortcut (Kawata et al., 2025), and that on sparse parity neither the learning rule nor the data sample moves the subspace a run ends up in.

Precursors. Networks make progress before it shows in the loss (Barak et al., 2022; Nanda et al., 2023), grokking can be forecast from the loss landscape (Notsawo et al., 2023), and the local learning coefficient marks developmental stages (Lau et al., 2025; Hoogland et al., 2025), and such precursors are validated by their lead over the visible event within a run. In the study of critical transitions, validating a warning only on the systems that did undergo a transition makes it look more reliable than it is, and the remedy is to apply it also to systems that did not (Boettiger & Hastings, 2012a). We apply that remedy to training and add a second check that only a date for the decision allows: the diagnostics do change before a run generalizes, but they read the decision thousands of steps after it is made, and the within-run test that validates them is passed by placebos. Appendix A sets each result of the paper beside the work closest to it.

7 Discussion

Summary. Two runs of one recipe can fit their data equally well and still differ in whether they generalize, which head carries a capability, which test images they get wrong and which injected facts they keep. Releasing replicas from a checkpoint turns that observation into a measurement, which recovers what was known and adds four findings. (1) Equally good runs differ along only a few directions, and chance moves a run along them. (2) On sparse parity neither the learning rule nor the data sample moves the class a run ends up in, and which source of chance then picks the member depends on the recipe: in a fragile regime the order of the data, the only source we varied, splits the runs into those that generalize and those that never do, the initialization mostly decides which head carries copying, and any perturbation at all, down to the choice of convolution kernel, decides which images a ResNet gets wrong. (3) Outcomes that are decided at all are decided early, at a median of 7.5% of training in our recipes and long before they show in the run, and an outcome made of many small pieces takes longer to be decided than a single number. (4) The decision can be read while it is made, but only by a task readout or a projection of the network's outputs, and not by the general diagnostics used to watch training.

Implications. The instrument needs only the ability to restart a run from a checkpoint, and it serves three uses beyond this paper. It dates when an outcome of a given recipe stops depending on later chance. It tests a proposed precursor against the decision itself, which the usual within-run test cannot do, since that test is passed by diagnostics that read the decision thousands of steps late and by a random projection of the parameters. Where no description reads the decision, a few replicas released briefly from a young checkpoint forecast whether a run will generalize (Appendix F).

AI USE STATEMENT

In this work, we used generative AI tools only for language polishing, improving the grammar, wording, and clarity of text drafted by the authors. We have not used generative AI tools for research ideation, for the design of the proposed method or the experimental protocol, for implementation, for the generation or analysis of results, for the creation of figures and tables, or for literature discovery and citation; all of these are the original work of the authors. We have reviewed all AI-assisted work, checking every suggested revision against the authors' original text and the underlying results, and we take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix B gives the model, data, number of runs, fork grid and optimizer configuration of every testbed, and Section 2 with Appendix B gives the estimator, its nulls and the detection floor. All data are public or generated, the fine-tuned model is the public Qwen2.5-0.5B, and every fork restarts from the saved parameters and optimizer state of its run. Runs are specified by their seeds, but GPU kernels were made deterministic only where Section 3 says so, so a rerun reproduces the reported statistics rather than each run bit for bit. The code for training, forking and analysis will be made public upon publication.

REFERENCES

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2019.

Gül Sena Altıntaş, Devin Kwok, Colin Raffel, and David Rolnick. The butterfly effect: Neural network training trajectories are highly sensitive to initial conditions. In *International Conference on Machine Learning*, 2025.

Tatsuya Aoyama, Ethan Gotlieb Wilcox, and Nathan Schneider. Predicting the emergence of induction heads in language model pretraining. In *International Conference on Machine Learning*, 2026.

Victor Bapst, Thomas Keck, Agnieszka Grabska-Barwińska, Craig Donner, Ekin Dogus Cubuk, Samuel S. Schoenholz, et al. Unveiling the predictive power of static structure in glassy systems. *Nature Physics*, 2020.

Boaz Barak, Benjamin L. Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. In *Advances in Neural Information Processing Systems*, 2022.

Lionel Barnett and Anil K. Seth. Dynamical independence: Discovering emergent macroscopic processes in complex dynamical systems. *Physical Review E*, 2023.

Robert B. Best and Gerhard Hummer. Reaction coordinates and rates from transition paths. *Proceedings of the National Academy of Sciences*, 2005.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, et al. Emergent and predictable memorization in large language models. In *Advances in Neural Information Processing Systems*, 2023.

Carl Boettiger and Alan Hastings. Early warning signals and the prosecutor's fallacy. *Proceedings of the Royal Society B*, 2012a.

Carl Boettiger and Alan Hastings. Quantifying limits to detection of early warning for critical transitions. *Journal of The Royal Society Interface*, 2012b.

Peter G. Bolhuis, David Chandler, Christoph Dellago, and Phillip L. Geissler. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. *Annual Review of Physical Chemistry*, 2002.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems*, 2021.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. How do large language models acquire factual knowledge during pretraining? In *Advances in Neural Information Processing Systems*, 2024.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, 2023.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 2022.

Christoph Dellago, Peter G. Bolhuis, and David Chandler. Efficient transition path sampling: Application to Lennard-Jones cluster rearrangements. *The Journal of Chemical Physics*, 1998.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *International Joint Conference on Artificial Intelligence*, 2015.

Rose Du, Vijay S. Pande, Alexander Yu. Grosberg, Toyoichi Tanaka, and Eugene S. Shakhnovich. On the transition coordinate for protein folding. *The Journal of Chemical Physics*, 1998.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep learning versus kernel learning: An empirical study of loss landscape geometry and the time evolution of the Neural Tangent Kernel. In *Advances in Neural Information Processing Systems*, 2020.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, 2020a.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020b.

Phillip L. Geissler, Christoph Dellago, and David Chandler. Kinetic pathways of ion pair dissociation in water. *The Journal of Physical Chemistry B*, 1999.

Laura Gomezjurado Gonzalez. The long delay to arithmetic generalization: When learned representations outrun behavior. *arXiv preprint arXiv:2604.13082*, 2026.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Guy Hacohen, Leshem Choshen, and Daphna Weinshall. Let's agree to agree: Neural networks share classification order on real datasets. In *International Conference on Machine Learning*, 2020.

Erik P. Hoel, Larissa Albantakis, and Giulio Tononi. Quantifying causal emergence shows that macro can beat micro. *Proceedings of the National Academy of Sciences*, 2013.

Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape degeneracy and stagewise development in transformers. *Transactions on Machine Learning Research*, 2025.

Anne Hsu, Alexander Borst, and Frédéric E. Theunissen. Quantifying variability in neural responses and its application for the validation of model predictions. *Network: Computation in Neural Systems*, 2004.

Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020.

Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J. Zico Kolter. Assessing generalization of SGD via disagreement. In *International Conference on Learning Representations*, 2022.

Keller Jordan. On the variance of neural network training with respect to test sets and distributions. In *International Conference on Learning Representations*, 2024.

Ryotaro Kawata, Yujin Song, Alberto Bietti, Naoki Nishikawa, Taiji Suzuki, Samuel Vaiter, and Denny Wu. From shortcut to induction head: How data diversity shapes algorithm selection in transformers. In *Advances in Neural Information Processing Systems*, 2025.

Michael Kleinman, Alessandro Achille, and Stefano Soatto. Critical learning periods emerge even in deep linear networks. In *International Conference on Learning Representations*, 2024.

David C. Krakauer, Melanie Mitchell, and John W. Krakauer. Large language models and emergence: A complex systems perspective. *Philosophical Transactions of the Royal Society A*, 2026.

Edmund Lau, Zach Furman, George Wang, Daniel Murfet, and Susan Wei. The local learning coefficient: A singularity-aware complexity measure. In *International Conference on Artificial Intelligence and Statistics*, 2025.

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

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

Pascal Jr. Tikeng Notsawo, Hattie Zhou, Mohammad Pezeshki, Irina Rish, and Guillaume Dumas. Predicting grokking long before it happens: A look into the loss landscape of models which grok. *arXiv preprint arXiv:2306.13253*, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

Baron Peters. Reaction coordinates and mechanistic hypothesis tests. *Annual Review of Physical Chemistry*, 2016.

Fernando E. Rosas, Bernhard C. Geiger, Andrea I. Luppi, Anil K. Seth, Daniel Polani, Michael Gastpar, and Pedro A. M. Mediano. Software in the natural world: A computational approach to hierarchical emergence. *arXiv preprint arXiv:2402.09090*, 2024.

Samuel S. Schoenholz, Ekin D. Cubuk, Daniel M. Sussman, Efthimios Kaxiras, and Andrea J. Liu. A structural approach to relaxation in glassy liquids. *Nature Physics*, 2016.

Thibault Sellam, Steve Yadlowsky, Ian Tenney, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Turc, Jacob Eisenstein, Dipanjan Das, and Ellie Pavlick. The MultiBERTs: BERT reproductions for robustness analysis. In *International Conference on Learning Representations*, 2022.

Aaditya K. Singh, Ted Moskovitz, Felix Hill, Stephanie C. Y. Chan, and Andrew 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*, 2024.

Cecilia Summers and Michael J. Dinneen. Nondeterminism and instability in neural network optimization. In *International Conference on Machine Learning*, 2021.

Oskar van der Wal, Pietro Lesci, Max Müller-Eberstein, Naomi Saphra, et al. PolyPythias: Stability and outliers across fifty language model pre-training runs. In *International Conference on Learning Representations*, 2025.

Asaph Widmer-Cooper, Peter Harrowell, and Herb Fynewever. How reproducible are dynamic heterogeneities in a supercooled liquid? *Physical Review Letters*, 2004.

Rosie Zhao, Tian Qin, David Alvarez-Melis, Sham Kakade, and Naomi Saphra. Random scaling of emergent capabilities. *arXiv preprint arXiv:2502.17356*, 2025.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Advances in Neural Information Processing Systems*, 2023.

Donglin Zhuang, Xingyao Zhang, Shuaiwen Leon Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. In *Proceedings of Machine Learning and Systems*, 2022.

A Full related work

This appendix places the paper in the literature in the three steps of Section 6: where the instrument comes from and how it differs from earlier uses of the same construction, the known results it recovers, which calibrate it, and the results we could not find in prior work. We searched for prior and concurrent work on every result in September 2026, and where a result is known we credit it plainly, because an instrument is trusted only after it has reproduced what is already known.

A.1 The instrument

Chemical and glass physics. The committor of a configuration, estimated by shooting trajectories from it, and the test of a reaction coordinate by checking that configurations sharing a value of the coordinate share a committor, are standard in transition path sampling (Dellago et al., 1998; Bolhuis et al., 2002; Geissler et al., 1999; Best & Hummer, 2005; Peters, 2016). The isoconfigurational ensemble of Widmer-Cooper et al. (2004) releases many trajectories from one configuration with fresh momenta, which is the closest physical analogue of forking a training run, and how much of the resulting propensity static structure can predict, which is our question of how much of the decided part a description reads, has been asked of glasses with handcrafted structural features (Schoenholz et al., 2016) and with graph neural networks (Bapst et al., 2020). In that literature neural networks appear as tools for computing committors and propensities, and we found no work in which the training run of a network is itself the system whose committor is computed.

Statistics and emergence. The screening fraction coincides with standard quantities that we credit and do not claim. It is the R^2 of a model relative to the ceiling set by trial-to-trial variability (Hsu et al., 2004), and its reaching one is the closure condition of the operational theory of emergence (Rosas et al., 2024; Barnett & Seth, 2023; Krakauer et al., 2026), stated here on conditional means rather than on full distributions; it is not the effective information of Hoel et al. (2013).

Forking and partitioning in machine learning. Copying a checkpoint and training the copies under different noise has been done before, but to ask whether the weights or the function of the copies stay together. Frankle et al. (2020a) found that standard vision models become stable to SGD noise early in training, after which the copies end in one linearly connected region, and Frankle et al. (2020b) characterized the same early phase. Fort et al. (2020) spawned children at a sweep of times and measured how far their functions came to disagree, Summers & Dinneen (2021) traced how the sensitivity to each source of nondeterminism falls as training proceeds, and Altıntaş et al. (2025) showed that a perturbation of a single weight sends two otherwise identical trajectories apart when it is applied early but not when it is applied late. Lin (2026) branch a grokking run from saved checkpoints with the random state restored, so that the branches differ only in a weight-decay perturbation, which measures the response to an intervention and, because it releases no later randomness, no committor. The variance of final performance has been partitioned by source across benchmarks (Bouthillier et al., 2021) and in fine-tuning (Dodge et al., 2020), but across finished runs, not within the replicas of one checkpoint and not by what a description of the network can read. What we did not find in any of this work, and what the instrument adds to the construction, is the division of a named outcome's variance into the part still open, the part a given description reads and the part decided but unread, together with the reaction-coordinate test of a description that the division supports.

A.2 KNOWN RESULTS THE INSTRUMENT RECOVERS

The established results below are reproduced by the instrument, and in each case the priority belongs to the work cited; the agreement is what licenses the new results of the next subsection.

Outcomes the objective leaves open. That two runs of one recipe can reach the same training loss and still differ is underspecification (D'Amour et al., 2022), documented across vision, language and clinical pipelines. Pretraining one model under different seeds gives checkpoints that differ downstream (Sellam et al., 2022; van der Wal et al., 2025), emergent capabilities are bimodal across seeds well before most seeds reach them (Zhao et al., 2025), and on modular arithmetic which circuits a network builds, and in which order, varies with the seed and the hyperparameters (Chughtai et al., 2023; Zhong et al., 2023). The four disagreements of Table 1 are further instances.

Table 1: Four outcomes that the objective does not fix. In each case the objective is identical in the two runs and is identically satisfied, so something other than the objective decided them.

outcome the objective leaves open	what two runs of one recipe do
whether the run generalizes at all	42 of 64 one-layer transformers generalize on addition mod 97, while 22 never exceed half accuracy on held-out pairs
which attention head carries in-context copying	all 32 runs build the head in layer 6, and yet runs that share an initialization still disagree about which head in 0.125 of pairs
which test images the run gets wrong	a ResNet-18's accuracy agrees to $92.53 \pm 0.14\%$, while at least 2426 of the 10,000 CIFAR-10 test images are classified both ways
which injected facts survive fine-tuning	six fine-tunes of Qwen2.5-0.5B each retain between 37 and 73 of the 128 injected facts, and the six disagree about 75 of those facts

Early stability to a run's own noise. The instrument finds that two replicas of one checkpoint come to lie in the same basin soon after the outcome is decided, while two independent runs never do (Appendix B), which recovers the early stability found by Frankle et al. (2020a), Fort et al. (2020) and Altıntaş et al. (2025).

Every source of nondeterminism counts. Our factorial ResNet-18 design, in which any one seed, or the kernel selected at identical seeds, moves about one test image in twenty, recovers the finding of Summers & Dinneen (2021) that all sources of nondeterminism, down to a single bit of the initialization, produce similar diversity, and that of Zhuang et al. (2022) that the nondeterminism of the tooling alone leaves accuracy unchanged while moving performance on parts of the data. It is consistent with the observations that the disagreement between two runs approximately equals their test error (Jiang et al., 2022), that runs make approximately independent errors on individual test examples (Jordan, 2024), and yet that different runs learn the examples of a dataset in much the same order (Hacohen et al., 2020).

Early decisions. The instrument finds that outcomes are decided early, which recovers what critical learning periods show by intervening at different times (Achille et al., 2019; Kleinman et al., 2024), and what the break-even point (Jastrzebski et al., 2020) and the small subspace in which the gradient comes to lie (Gur-Ari et al., 2018) show in other terms.

Progress before it shows. The instrument finds that task readouts, and the local learning coefficient, change before a run generalizes. This recovers the hidden progress of Barak et al. (2022) on sparse parity, the progress measures that anticipate grokking (Nanda et al., 2023), the forecast of grokking from the loss landscape (Notsawo et al., 2023), the developmental stages marked by the local learning coefficient (Lau et al., 2025; Hoogland et al., 2025), the prediction of induction heads from statistics of the data (Aoyama et al., 2026), and the lead of learned representations over behaviour on arithmetic tasks (Gomezjurado Gonzalez, 2026).

A.3 NEW RESULTS

Each result below needs a part of the instrument that earlier methods did not have, and we could not find any of them in prior work.

Table 2: The testbeds and the measurements made on each. J counts the trunks of the fork tree, meaning the independent runs from which replicas were released, and K counts the replicas released from each checkpoint of a trunk; for the language model, which uses a factorial design rather than forks, J counts initializations and K data orders.

system	model	data	J	K	end	fork times
F	1-layer transformer, $d = 128$	$a + b \bmod 97$, 21%	64	4	40000	100, 1500, 3000, …, 22000
A	1-layer transformer, $d = 128$	$a + b \bmod 97$, 30%	80	4	20000	50, 150, 400, …, 13000
P	2-layer MLP, width 64	sparse parity, 3 of 40 bits	64	4	3000	20, 60, 150, …, 1200
IND	2-layer transformer, $d = 64$, 4 heads	repeated spans, $V = 64$, $L = 64$	48	4	8000	100, 300, 600, …, 6000
LM	6-layer transformer, $d = 384$, 4 heads	repeats at random offsets	8	4	3000	factorial, not forked
CNN	ResNet-18	CIFAR-10	32	4	20 epochs	epochs 1, 2, 4, 8, 12
SFT	Qwen2.5-0.5B	Alpaca, 128 facts	6	3	192 steps	30, 40, 60, …, 150

A clock for the decision. Earlier work shows that something is decided early; with the fork tree we can say what is decided and when, and separate that date from the moment the outcome becomes visible. A named outcome that is decided at all is decided at a median of 7.5% of training across the trees we built, an outcome spread over many examples is decided more slowly than a single number read from the same runs, and the learning rule and the architecture move the date (Figure 5). The outcome "which test images are wrong" is never decided on our grid of fork epochs, which adds a time axis to the results on nondeterminism above.

Which source of chance decides an outcome. A factorial design over the sources of chance, which a fork cannot separate, shows that the initialization mostly picks which attention head carries in-context copying (Olsson et al., 2022; Singh et al., 2024), and that the balance shifts toward the data order when the data offer a positional shortcut (Kawata et al., 2025). On sparse parity, measured against two random subspaces of the same size, a change of learning rule or of data sample leaves the subspace a run ends up in where it was.

What reads the decision, and what only appears to. The diagnostics listed among the known results change before a run generalizes, but with a date for the decision we can see that they read it only thousands of steps after it is made, and that the within-run test by which they are validated is passed by placebos, mostly through moves made after the decision. In the literature on early warning of critical transitions, Boettiger & Hastings (2012a) showed that studying only the systems that did undergo a transition makes warning statistics look more reliable than they are, a form of the prosecutor's fallacy that replicated experiments avoid, and Boettiger & Hastings (2012b) argued that warning indicators must report their error rates against a null. Our within-run test follows that remedy, since it is also run on the runs that never generalize, and the date of the decision adds a check that no such control provides, whether a move comes before the decision or only after it; we did not find either check applied to the diagnostics of neural network training before.

Forecasting a run by forking it. Stopping unpromising runs early is standard, whether by extrapolating learning curves (Domhan et al., 2015), by starting many fine-tuning runs and stopping the unpromising ones (Dodge et al., 2020), or by predicting memorization from smaller or shorter runs (Biderman et al., 2023). What we add is a forecaster that releases a few replicas for a short stretch, which at the earliest fork step beats every description available without forking (Appendix F). Our fine-tuning testbed relates to studies that inject facts into training and follow their acquisition and forgetting (Chang et al., 2024), and asks, fact by fact, when retention is decided.

B Experimental details

Table 2 gives each testbed's model, data, run counts and fork grid, and Table 3 gives how the optimizer was configured in each of them. The remainder of this appendix collects the design details of individual measurements, each of which is referred to from the section that uses it.

B.1 THE TESTBEDS AND THEIR RECIPES

Two of the seven testbeds are one-layer transformers trained on addition modulo 97, and they differ only in how much of the addition table they are shown. Testbed F sees the smaller share and therefore sits in a fragile regime, in which some runs come to generalize and others never do, so that the outcome worth asking about there is whether a run succeeds at all. Testbed A sees more of the table and always generalizes, so the outcome worth asking about is not whether but when. Because every run of either testbed starts from one initialization and one split of the data, the order of the minibatches is the only source of chance we vary, and that is what makes them the cleanest testbeds in the paper.

An outcome that is a number is easy to decompose, and we wanted outcomes that are not. The third testbed is a two-layer attention-only transformer that learns to copy a repeated span from its context, and the outcomes we read from it are a loss on held-out context and a fingerprint of the copying mechanism, both of which are vectors rather than numbers. The fourth is a six-layer language model in which four heads in a layer can each carry the same capability, so that the outcome is the identity of the winning head: an assignment among interchangeable alternatives, for which no scalar description can be adequate. That testbed combines every initialization with every data order instead of forking, which is what allows the two sources to be separated, and its heads are probed at an offset never seen in training, because a head that copies by position passes a probe placed where the training data repeated. The fifth is sparse parity learned by a small multilayer network, which is the only testbed we can run in many configurations cheaply, and it therefore carries the comparisons that need a sweep, shown in Figures 3a and 5. The sixth is a ResNet-18 on CIFAR-10, the one full-size vision network in the paper, and its outcome is the pattern of test images the finished network gets wrong. The seventh is a fine-tune of the public Qwen2.5-0.5B model on Alpaca, a published recipe and a model we did not train.

Table 3: The training recipe of each testbed of Table 2. The sparse-parity row is the reference configuration, and the configurations that vary it are those of Figures 3a and 5.

system	optimizer	batch	learning rate	weight decay	schedule
F	AdamW	512	0.001	1	constant
A	AdamW	512	0.001	1	constant
P	AdamW	256	0.001	0.01	constant
IND	AdamW	128	0.0003	0.01	constant
LM	AdamW	128	0.0006	0.1	warmup 500 steps
CNN	SGD, Nesterov 0.9	256	0.1	0.0005	1-epoch warmup, cosine
SFT	AdamW	128	2×10^{-5}	0	warmup 3% of steps

B.2 THE ESTIMATOR

The division of α from β in Section 2 forecasts each run's mean outcome from the description by leave-one-run-out ridge and kernel ridge regression, and subtracts from the variance that forecast explains the sampling noise of the per-run means, which are averages over a finite number of replicas. The family-wise null permutes the run labels and repeats the entire selection over candidate descriptions and regularization strengths inside every permutation, so that the best description in the real set is compared with the best description in a permuted set. The per-description null asks the narrower question of whether one particular description reads the outcome. The expectations and variances of Equation 1 are taken over both the draw of the run that produced θ_t and the randomness that follows the fork time, and the identity holds only for a description that is a function of the checkpoint, which is why each diagnostic that is itself a Monte Carlo estimate, such as the local learning coefficient, is computed twice with independent seeds, and its zero is read as a non-detection only where the two estimates agree.

B.3 THE TWO OFFSET REGIMES OF THE HEAD-IDENTITY TESTBED

The random-offset and the fixed-offset runs of Section 3 differ in more than the offset at which the span repeats, since they also differ in batch size, in learning rate and in the length of training, and the closest matched comparison is therefore the more persuasive one. A single initialization at the

smaller batch size produced the same head in all 4 of its random-offset runs, and split between two heads under fixed offsets.

B.4 The variance components of the ResNet's factorial design

The random-effects analysis of the factorial design of Section 3 attributes 1.7% of the per-image variance to the initialization, 0.7% to the data order and 0.2% to the augmentation, and 4.0% to the interaction between the order and the augmentation, which is the largest of the named terms. The two interactions involving the initialization came back negative and were set to zero, which is a warning about the precision of all four. The residual is 93.5%, and this design cannot separate it from the nondeterminism of the GPU kernels.

B.5 Linear mode connectivity

The measurement of Section 4 that agrees with the dating of testbed A in different coordinates is a linear interpolation in parameter space between two finished models, reporting the loss barrier along it. Between two replicas of one checkpoint the barrier stands at 1.62 at fork step 50, has fallen to 0.072 by step 400, and is 0.000 at the last fork time we measured. Between two independent runs it stays between 2.9 and 3.4 throughout.

C The class that chance moves within

The introduction pictures training as fixing a class of equally good networks and leaving the choice among them to chance. If that picture holds, finished runs should differ from one another along only a few directions, and the noise released at a fork should move along those same directions rather than at random. We test both on testbeds A and F and on in-context copying, whose outcomes are vectors of task features, the Fourier spectrum of the embeddings or the per-head induction scores. For each tree we compare the subspace in which replicas of one checkpoint come to differ with the subspace in which independent runs differ. On all three testbeds the two are aligned several times more closely than two random subspaces of the same size, at every fork time we measured, whereas in a control trained on random labels, where there is no capability to form, the alignment sits at chance (Figure 7a). The subspaces are also narrow: the runs of each testbed differ in a handful of the available directions, while those of the random-label control spread over most of them (Figure 7b). Where there is a capability to form, equally good runs therefore differ within a small class, and the later noise moves along it. This is a comparison of subspaces against a single null rather than an instance of Equation 1, so it supports the picture of the introduction without a claim of its own.

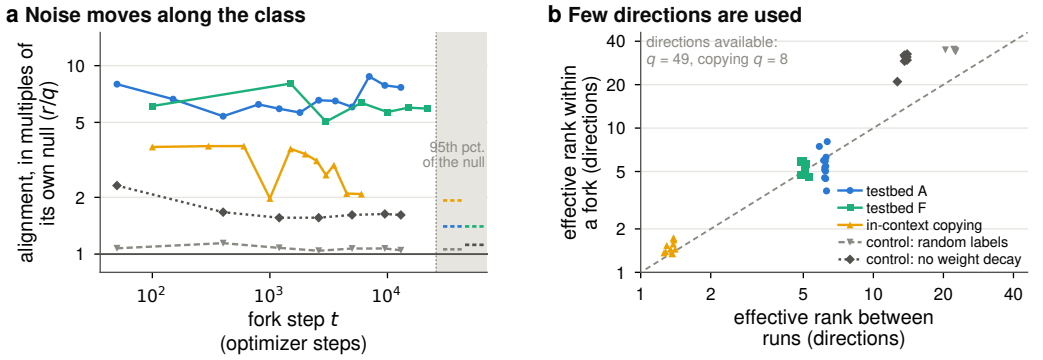


Figure 7: Equally good runs differ along a few directions, and the noise after a fork moves along them. (a) The alignment between the subspace in which replicas of one checkpoint differ and the subspace in which independent runs differ, in multiples of the alignment of two random subspaces of the same size, which is the solid line at one; the ticks on the right mark each configuration's 95th percentile under that null, and a control trained without weight decay sits only a little above chance. (b) How many directions each subspace uses; the dashed line marks equal numbers.

Table 4: The decision-time analysis by task family. A case is one fork tree together with one named outcome, and it counts as decided at the first fork time at which $\gamma \leq 0.1$; the percentage columns are shares of each tree's own training length, taken over the decided cases.

task family	trees	cases	decided	median %	quartiles %	range %	in first 10%
ResNet-18	1	3	0	—	—	—	—
in-context copying	1	4	3	31	31–34	31–38	0
language model	5	14	3	13	10–43	6.7–73	33
modular addition	11	49	25	2.0	2.0–7.5	0.8–21	84
sparse parity	13	24	17	12	10–20	6.0–38	35
all	31	94	48	7.5	2.0–14	0.8–73	58

D THE DECISION-TIME ANALYSIS IN DETAIL

Table 4 breaks down the pooled cases of Section 4 by task family. An outcome that every run reaches has no between-run variance to decompose, so the analysis leaves out every outcome whose total variance is not stable across the fork times of its tree. The pooled median depends on the mix of task families: modular addition supplies about half of the decided cases and decides within the first few percent of training, whereas the other task families decide between about a tenth and a third of the way through. The median is also bounded by how late we could afford to fork, but among the cases whose grids reach half of their own training it still falls within the first fifth. The small language models repeat their spans at a fixed offset, and one further tree scores testbed A a second time at a shorter horizon; removing these trees leaves the median unchanged. Where a tree carries both a single number and a vector outcome, the two usually decide at the same fork time, and too few differ to say which is decided first.

The six-layer language model has no tree to score, since it combines its sources of chance in a factorial design instead of forking, and the fine-tune of a public model is scored separately. In a second configuration of that fine-tune, with the facts present throughout training rather than only at its start, retention is decided far more slowly. Because the two configurations also differ in the number of facts, this is consistent with the reading that an outcome is decided once the data that bear on it stop arriving, without showing it.

E THE READABILITY ANALYSIS, TESTBED F AT ITS DECISION, AND THE WITHIN-RUN TEST

This appendix gives the three tables behind Section 5, with the detail that goes with each.

E.1 THE READABILITY ANALYSIS

Eight configurations of our testbeds carry task readouts, built for their own task: the Fourier spectrum of the embeddings for modular addition, the per-head induction scores for in-context copying, and a fingerprint of the parity function for sparse parity. Dating these task readouts in the same way as the decision gives 11 cases in which both a decision time and a readable time exist, and in 11 of them the task readout is readable at or before the decision, in 10 strictly before it (Figure 6a and Table 5). 2 of the cases come from the two modular-addition controls, one of them trained on random labels, where there is no capability for a task readout to align to, and without them the tally is 9 of 9. These dates are scored against a single permutation null, without the placebos, the per-description null and the detection floor used below, so each bounds how late a task readout becomes readable rather than fixing the date.

E.2 TESTBED F AT ITS DECISION

The full set of controls exists only on the two modular testbeds, and on testbed F it separates the network's outputs from its parameters (Table 6). Testbed F is decided at step 3000, where the Fourier spectrum of the embeddings reads a screening fraction of 0.27 ($\alpha = 0.256$), and a sixteen-component

Table 5: The readability analysis. Each case is a testbed together with a named outcome for which both a decision time and a readable time exist. The readable column gives the first fork time at which some task readout of that testbed reads the outcome there and again at the next fork time, and the decided column the fork time at which the outcome is decided, both in optimizer steps. The earlier column is the lead between them, counted in steps of that tree's own fork grid. The first task readout column names the task readout with the largest α among those readable first, and the number in brackets counts the further task readouts readable at the same fork time.

testbed	outcome	readable	decided	earlier	first task readout	α
IND	log-probabilities	1500	3000	3	per-head induction scores	0.043
IND	induction structure	1500	2500	2	per-head induction scores	0.099
A	Fourier structure	50	400	2	Fourier spectrum (+5)	0.045
F	per-example correctness	1500	6000	2	Fourier, 16 components (+1)	0.171
F	Fourier structure	100	3000	2	Fourier, 32 components	0.028
C_nowd	Fourier structure	50	400	1	Fourier, 8 components (+6)	0.087
C_rand	Fourier structure	400	1200	1	Fourier, 16 components (+3)	0.014
Pgd	parity fingerprint	150	300	1	parity fingerprint, 8 components	0.025
F	final accuracy	1500	3000	1	Fourier, 16 components	0.213
F	the step it groks	100	1500	1	Fourier, 8 components	0.119
Plocal	parity fingerprint	3000	3000	0	parity fingerprint, 4 components (+4)	0.150

summary of that spectrum is readable from step 1500. A random sixteen-dimensional projection of the network's outputs on probe inputs is readable from step 1500, whereas random projections of the parameters become significant only at the last fork time. The decision can therefore be read from the outputs even through a random projection, but from the parameters only in coordinates aligned with the task, and the test loss and the per-example losses, although they too are computed from the outputs, do not read it at the decision.

The diagnostics ordinarily used to watch training behave differently (Figure 6b). At the decision the Hessian trace, the Fisher information and the gradient noise all have $\alpha = 0$, and two independent estimates of each agree almost exactly, so the zero is a real non-detection. They become significant only at the last fork time under their own null, after the first run has generalized, and never under the family-wise null. The local learning coefficient (Lau et al., 2025) is readable from step 10000 under its own null and from step 15000 under the family-wise one, which is before any run generalizes but thousands of steps after the decision. Its α at the decision is zero as well, but that zero is not evidence, because two independent estimates of the coefficient correlate across runs at only 0.45. No ordering between two descriptions holds in four fifths of random subsets of the runs, so we cannot say which of them reads first on this testbed.

E.3 The within-run test

The same diagnostics pass the test by which precursors are usually validated, and so do placebos (Table 7). The test asks, within each run, whether a description leaves the range spanned by the runs that never generalize before that run itself generalizes. The local learning coefficient does so in 86% of the generalizing runs, a median of 17875 steps ahead, but the same criterion also fires in 36% of the runs that never generalize, and a random sixteen-dimensional projection of the parameters passes it in 48% of the generalizing runs. Paired over runs, the learning coefficient and the Fourier spectrum do beat the output placebo, with exact McNemar $p = 0.0044$ and $p = 0.0042$, so the test is not empty. What it measures, though, is the lead over the visible event rather than over the decision: 61% of the coefficient's early moves come only after the decision at step 3000 (Figure 6c). A within-run lead is therefore weak evidence that a description anticipates the decision.

Table 6: Testbed F at its decision, step 3000, where $\gamma = 0.050$. The first two numeric columns are the share of the outcome's variance that the description reads at that step and the same share expressed against the ceiling $1 - \gamma$. The two null columns give the first fork step at which the description is significant, under the family-wise null and under its own, with *never* where it is significant at no step and *last point* where it is significant only at the last one. The final column is *deterministic* for a description that is a function of the checkpoint, and otherwise the correlation across runs between two independent estimates of it at this step. The placebos are listed last.

description	α	$\alpha/(1-\gamma)$	family-wise null	own null	test–retest
Fourier spectrum	0.256	0.270	3000	100	deterministic
Fourier, 16 components	0.271	0.285	1500	1500	deterministic
output projection, 16 d	0.123	0.130	last point	3000	deterministic
per-example loss, 32 d	0.025	0.026	never	never	deterministic
scalars	0.043	0.045	15000	6000	deterministic
spectral entropy	0.043	0.046	last point	15000	1.000
loss-curve features	0.008	0.008	last point	10000	1.000
test loss	0.000	0.000	last point	10000	deterministic
weight-norm trajectory	0.000	0.000	last point	15000	1.000
learning coefficient	0.000	0.000	15000	10000	0.454
Hessian trace	0.000	0.000	never	last point	0.988
Fisher information	0.000	0.000	never	last point	0.998
gradient noise	0.000	0.000	never	last point	0.998
minibatch commutator	0.000	0.000	never	never	0.607
placebo: random outputs, 16 d	0.093	0.097	not tested	1500	deterministic
placebo: random θ, 16 d	0.009	0.010	not tested	last point	deterministic
placebo: random θ, 8 d	0.000	0.000	not tested	last point	deterministic

Table 7: The within-run test on testbed F. "Moves" is the share of the 42 generalizing runs in which the description leaves the band spanned by the never-generalizing runs before that run itself generalizes, and "false moves" is the same criterion applied to the 22 runs that never generalize. The median lead is the gap in optimizer steps between the move and the run's own transition, resolved only to the grid of fork times. The last column is the share of the 42 whose move comes only after the decision at step 3000. The three ladder rows collapse a description that was scored at several truncations, and give the range over them.

description	moves (%)	false moves (%)	median lead	after decision (%)
learning coefficient	86	36	17875	52
Fourier spectrum	83	18	18250	36
test loss	69	45	11250	57
minibatch commutator	62	55	24400	21
weight-norm trajectory	62	50	9125	50
spectral entropy	60	32	18750	43
scalars	57	45	8250	45
Hessian trace	50	50	20450	33
Fisher information	48	41	19150	36
gradient noise	43	41	16000	33
loss-curve features	29	18	24125	17
Fourier spectrum ladder (6 truncations)	60	36	12000	45
output projection ladder (6 truncations)	60	36	17750	48
per-example loss ladder (6 truncations)	55	50	19000	38
placebo: random outputs, 16 d	55	45	17625	48
placebo: random θ, 16 d	48	41	20500	36
placebo: random θ, 8 d	29	18	10625	29

F A POSITIVE CONTROL, AND A FORECAST OF WHETHER A RUN WILL GENERALIZE

The fine-tuned public model of Section 4 supplies the positive control that makes the zeros of Section 5 interpretable (Figure 6d). Two things about that testbed are worth separating. The first is what a practitioner would see. In the configuration with the facts injected only at the start, the six

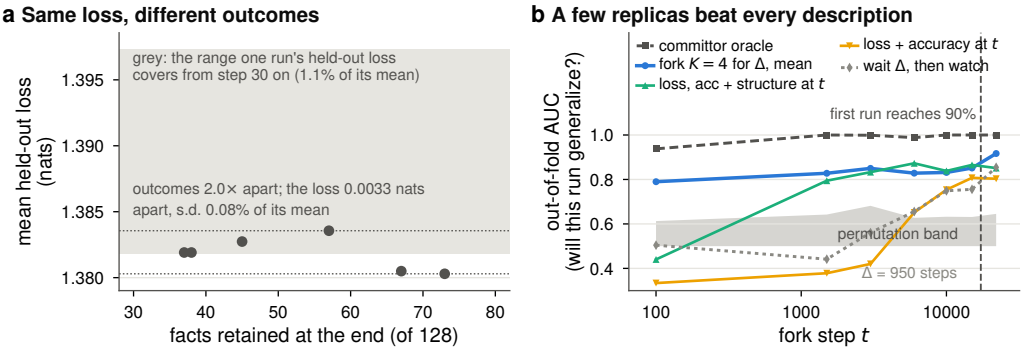


Figure 8: The number a practitioner watches is blind to the outcome, and a few replicas are not. (a) In the fine-tuning configuration with the facts injected only at the start, the 6 runs keep between 37 and 73 of the 128 injected facts and disagree about 75 of them, yet their aggregate held-out loss varies by a standard deviation of only 0.08% of its mean; the grey band is the range the same loss covers along one run from step 30 on. (b) Testbed F, out-of-fold area under the curve for "will this run ever generalize?": at fork step 100 every description available without forking falls below its own permutation band, whereas 4 replicas forked there and run 950 steps reach 0.790 and the full-fork committor reaches 0.938.

runs keep between 37 and 73 of the injected facts, yet the final value of the aggregate held-out loss varies across them by only 0.08% of its mean, so that it separates nothing (Figure 8a). The second is what a task readout can see. The outcome is which of the injected facts survive, and the task readout is each fact's own logit margin at the fork step. It forecasts survival among the disputed facts with an area under the curve of 0.88 at step 30 and 0.98 at step 40, and at both steps the committor oracle itself reaches a comparable value. Difficulty baselines computed from the other runs reach only 0.59 and 0.59. Because survival at the end is itself read from that margin, this is the outcome measured early rather than a precursor claim, and we use it only as a control. Its job is to show that the test credits a task readout as soon as the decision is made, which is what makes a zero elsewhere worth interpreting.

The task a practitioner would actually want performed is to say whether a given run will ever generalize. Take testbed F at fork step 1500 and ask how well each available forecaster does (Figure 8b), measuring all of them against a chance band of 0.642. The run's own state reaches an area under the curve of 0.794, and simply waiting 950 further steps and watching the loss reaches 0.441. Forking 4 replicas and running them for those same 950 steps reaches 0.828 instead. The full committor does better still, separating the two groups perfectly from step 1500 onwards, which is 15750 steps before the first run reaches ninety per cent test accuracy. The paired bootstrap over runs puts the advantage of a short fork over the best description available without forking at $+0.330$ at the earliest fork step, with a 95% interval of $[+0.137, +0.515]$, and at every later step that interval includes zero, so we claim the advantage only where the runs are young. Chance becomes readable, in other words, by letting a little more of it happen.