Between memorization and generalization: an interpolation regime in learning dynamics

Mahi Aminu Aliyu[1,2] **Ahmad Imran**[1,3] **Karim Farid**[4]
[1]Fatima Institute for Global Research [2]Abubakar Tafawa Balewa University, Nigeria
[3]National University of Sciences and Technology, Pakistan [4]University of Freiburg, Germany
mahigital@gmail.com

Abstract

Grokking marks the onset of generalization, but not how it unfolds. We study the structure of this transition by separating held-out problems according to whether they can be solved by interpolating from what the model has already seen. In the in-context modular-arithmetic setting of He et al. (2024), who identified two algorithms mechanistically in trained transformers, we follow both through training. Transformers first acquire Ratio Matching, which rescales a single in-context example. Only later, and only with enough context, do they acquire Modular Regression, which combines several examples and reaches beyond interpolation. Shallower models remain confined to Ratio Matching. We observe the same ordering in a controlled family of compositional tasks. Transformers first solve the held-out tasks that are recoverable as least-squares blends of training tasks, and solve the remaining tasks only later. The standard held-out threshold is crossed before this second transition, and MLPs of the same size rarely progress beyond interpolation. The ordering persists under controls for threshold choice and group size and, for MLPs, is not accounted for by task difficulty. More diverse training tasks shorten the interpolation window, whereas harder tasks stretch it. These results suggest that the transition from memorization to generalization is not a single global event, but a sequence of staggered capability transitions that aggregate evaluation collapses into one. We discuss whether this intermediate regime may contribute to plausible-but-incorrect predictions in larger models, including hallucinations.

1 Introduction

Generalization is usually dated with one number. A model is trained, its average score on held-out problems is tracked, and generalization is said to occur when that average crosses a threshold. In grokking, the crossing comes long after the training data are fit [Power et al., 2022]. Later work dates rule learning and memorization with one threshold each [Wang et al., 2026], or maps the conditions under which transformers generalize to held-out modular-arithmetic tasks [He et al., 2024]. A single crossing says when generalization becomes visible. It does not say how it emerges, because the same crossing is produced whether all held-out problems are solved together or in stages.

The difference matters because held-out problems are not all alike. Some can be answered by reusing what the model has already seen, and others cannot. A model that solves only the first kind can still cross the threshold, while on the second kind its answers look right without being right. This is one possible source of hallucination [Kalai et al., 2025]. We therefore ask whether, during training, all held-out problems are solved together or the reusable ones are solved first.

To answer this, we split the held-out problems before looking at the model. A problem is *interpolable* if it can be answered by reusing seen material, and *non-interpolable* otherwise. We then time each group separately (§2). The closest prior work, *Learning to grok* [He et al., 2024], showed

Preprint.

mechanistically how trained transformers solve in-context modular arithmetic. It identified two algorithms, Ratio Matching, which rescales one in-context example, and Modular Regression, which combines several. It found evidence for them in the models' predictions and in their attention heads, MLP activations and internal representations, and showed which one a model uses depending on its depth and the number of examples. That analysis is of trained models. We use it to study the two algorithms during training, and show that they appear in sequence. We also give a split that needs no named algorithm, so that the same question can be asked in another setting, and we show that the standard threshold is crossed while the model is still interpolating.

We find the same order in two settings. In modular arithmetic, deep transformers solve the interpolable queries first and the rest later, and only when given enough in-context examples, while shallow transformers never solve the rest (§3). In a controlled family of compositional tasks, transformers solve the interpolable tasks first in 188 of 189 runs, and MLPs of the same size rarely solve the others (§4). The order is not explained by group size or threshold choice, nor, for the MLP, by task difficulty. The usual threshold is crossed before the non-interpolable tasks are solved in all 196 runs where both can be timed. More training tasks shorten the interpolation regime and harder tasks stretch it. Individual transitions can still be sharp, so our finding refines the temporal structure of the transition without removing it.

Related work. The apparent suddenness of grokking can hide gradual internal change. Circuits form before the behavioural transition [Nanda et al., 2023], delayed generalization tracks a shift from lazy to rich dynamics [Kumar et al., 2024], and coarse evaluation can make continuous change look discontinuous [Schaeffer et al., 2023]. These works ask what changes inside the network before grokking becomes visible. We ask what the model can generalize to during that interval (§6).

2 Framework

We define interpolation first, then the clocks that time it and the regimes they mark. Every term used in the results is defined here.

Interpolable problems. Let S be the material a model has seen. A *reuse rule* R maps S to a set of answers $R(S)$ that can be built from S alone, which we call the *interpolation class*. A held-out problem with correct answer a is *interpolable* if a lies in $R(S)$ up to a fixed tolerance, and *non-interpolable* otherwise. The definition has three parts, namely the seen material, the reuse rule and the test for membership. Table 1 gives them for our two settings. The settings differ in what is seen and in how it is reused, but the definition is the same. We use "interpolation" in this sense only, and not for membership in the convex hull of the training inputs, which almost never holds in high dimension [Balestriero et al., 2021].

The split classifies problems and does not predict answers. Deciding whether an answer lies in the interpolation class requires knowing the answer, as any membership test does. Algorithm 1 gives the test for the hyperteacher. It finds the best least-squares combination of the training tasks' functions for a given held-out task, and asks how well that combination reproduces the task. This differs from a predictor such as a kernel smoother, which sets its weights without seeing the target and whose job is to produce an answer. The blend sees the target, and its job is to label the task. A model could not run it, and we do not use it as a model. Two details guard the test. The weights are fitted on one half of the probe inputs and scored on the other half, so the blend cannot overfit the probe. The blend is also fitted separately to each target, which makes it generous. A task labelled non-interpolable is therefore one that even the best combination of training functions does not solve.

Clocks. A *clock* is the first training step at which a score reliably reaches a threshold. We use one threshold throughout, $\theta = 0.9$, following Wang et al. [2026], and report $\theta = 0.75$ as a sensitivity check. *Reliably* means that the score stays at or above θ for a fixed stretch after the crossing, so that a single noisy crossing does not count. The stretch is the next 1,000 steps in the hyperteacher and the next four checkpoints in modular arithmetic. A task is *solved* when the model's score on it is at least θ. We use the six clocks of Table 2. Two are the standard ones, measured on the mean score, and τ_{gen} is the measurement analogue of τ_{rule} in Wang et al. [2026]. The other four are ours, each measured on one group of held-out problems. A mean can reach θ while one group is unsolved, which is why each group gets its own clock.

Table 1: One definition of interpolation in two settings.

	Modular arithmetic (§3)	Hyperteacher (§4)
Seen material S	the in-context examples of the sequence	the functions of the training tasks
Reuse rule R	rescale one example (Ratio Matching)	take a linear combination (the blend)
Interpolable if	some example is a multiple of the query	the best combination reaches $R^2 \geq \theta$

Algorithm 1 The blend, which decides whether a held-out task is interpolable.

Require: held-out task A; training tasks $B_1, \ldots, B_N$; probe inputs $p_1, \ldots, p_{512}$; threshold $\theta = 0.9$
Ensure: $\mathrm{blend}(A) \in (-\infty, 1]$ and whether A is interpolable

$\mathcal{P}_{\text{fit}} \leftarrow \{p_1, \ldots, p_{256}\}$, $\mathcal{P}_{\text{score}} \leftarrow \{p_{257}, \ldots, p_{512}\}$
for $i = 1, \ldots, N$ **do**
  $Y_i \leftarrow f_{B_i}(\mathcal{P}) \in \mathbb{R}^{512 \times 16}$ ▷ outputs of training task B_i on the probe
end for
$Y_A \leftarrow f_A(\mathcal{P})$
$w^\star \leftarrow \arg\min_{w \in \mathbb{R}^N} \left\| \sum_{i=1}^{N} w_i \, Y_i[\mathcal{P}_{\text{fit}}] - Y_A[\mathcal{P}_{\text{fit}}] \right\|_F^2$ ▷ least squares, no intercept
$\hat{Y}_A \leftarrow \sum_{i=1}^{N} w_i^\star \, Y_i[\mathcal{P}_{\text{score}}]$
$\mathrm{blend}(A) \leftarrow \frac{1}{16} \sum_{o=1}^{16} \left(1 - \frac{\sum_s (\hat{Y}_{A,so} - Y_{A,so})^2}{\sum_s (Y_{A,so} - \bar{Y}_{A,o})^2}\right)$ ▷ $s \in \mathcal{P}_{\text{score}}$; same R^2 as for the students
return $\mathrm{blend}(A)$; A is interpolable if $\mathrm{blend}(A) \geq \theta$

Acquisition and regimes. We say that a model has *acquired* a capability at the first step from which it reliably solves the problems that the capability solves. The definition is behavioural. Acquiring Ratio Matching means reliably answering the queries that Ratio Matching answers, and it is dated by τ_{RM}. He et al. [2024] identified the algorithms mechanistically in trained models (§3). We date their acquisition by behaviour. The clocks divide training into three regimes. *Memorization* runs from τ_{fit} to τ_{interp}. The model has fit its training tasks but solves neither held-out group, as in the task-memorizing phase of Kirsch et al. [2022] and He et al. [2024], although the mean held-out score can already cross the threshold. *Interpolation* runs from τ_{interp} to τ_{beyond}, when the model solves the interpolable problems and not yet the others. *Generalization* starts at τ_{beyond}. A model that never reaches τ_{beyond} is *interpolation-bound*.

3 Modular arithmetic

Setting. We use the in-context modular-arithmetic tasks of He et al. [2024]. A task is a pair (a, b) that maps an input (x, y) to the output $z = ax + by \bmod 29$. A sequence shows k examples (x_i, y_i, z_i) of one task, followed by a query (x, y), and the model must output the query's z. The task is not given, so the model has to infer it from the examples. Held-out queries come from tasks that were not used in training, and chance accuracy is $1/29$. He et al. [2024] showed, with black-box tests on the models' predictions and with mechanistic analysis of trained models, that transformers solve these tasks in context with two algorithms. *Ratio Matching* (RM) finds an example that is a multiple of the query, so that $xy_i \equiv yx_i \pmod{29}$, and rescales its output. *Modular Regression* (MR) combines several examples, and is needed for queries that no example rescales to. A query is interpolable if Ratio Matching can answer it (Table 1). We call such a query *RM-solvable* and any other query *non-RM*. The label depends only on the query and its examples, and is fixed before the model is evaluated.

At the end of training, the depth-4 transformers of He et al. [2024] use Ratio Matching with few examples and Modular Regression with many, while their depth-2 transformers use Ratio Matching only. Their analysis describes trained models and does not say when, or in what order, the two algorithms appear. We use their identification to study the algorithms during training. The clocks τ_{RM} and τ_{MR} date when the RM-solvable and the non-RM queries are reliably answered. We ran the authors' public code in their released depth-4 configuration (384 tasks) and depth-2 configuration (512 tasks), with 3 seeds each and 200k steps, for 6 runs in total. We evaluate at $k = 4$, 8 and 16 examples.

Table 2: The six clocks. Each is the first step from which its condition holds reliably.

clock	condition	setting
τ_{fit}	mean score on training tasks $\geq \theta$	hyperteacher
τ_{gen}	mean score on held-out tasks $\geq \theta$ (the usual threshold)	hyperteacher
τ_{interp}	at least 90% of interpolable tasks solved	hyperteacher
τ_{beyond}	at least 90% of non-interpolable tasks solved	hyperteacher
τ_{RM}	accuracy on interpolable (RM-solvable) queries $\geq \theta$	modular arithmetic
τ_{MR}	accuracy on non-interpolable (non-RM) queries $\geq \theta$	modular arithmetic

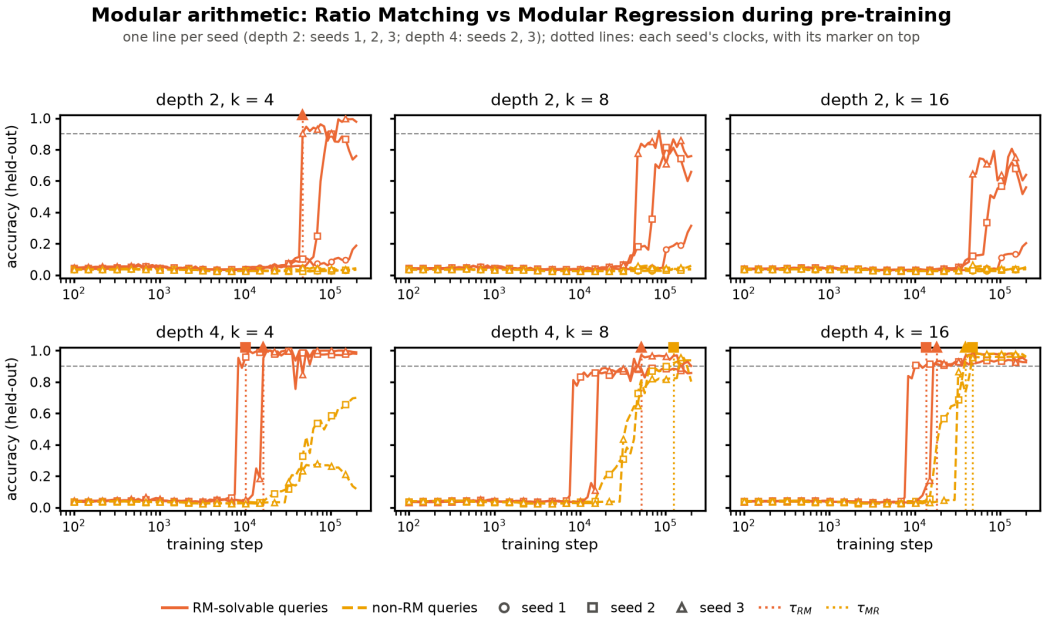


Figure 1: Modular arithmetic, accuracy on held-out queries over training for $k = 4$, 8 and 16 in-context examples. Solid lines are RM-solvable queries and dashed lines are non-RM queries, one line per seed with that seed's clocks marked. The rows are depth 2 (seeds 1–3) and depth 4 (seeds 2–3; seed 1 was the pilot and is in Table 3). At depth 4, RM-solvable queries are solved first at every k except seed 2 at $k = 8$. At depth 2, non-RM queries are never solved.

Depth 4 acquires Ratio Matching first. At $k = 4$ and $k = 16$, τ_{RM} precedes τ_{MR} in every depth-4 seed (Table 3, Fig. 1). With 16 examples, non-RM queries follow 19–34k steps later, and with 4 examples they never follow. In seeds 2 and 3 at $k = 16$, accuracy on RM-solvable queries leaves chance abruptly, and for the next 5–11k steps the model answers those queries and no others. The one exception at $\theta = 0.9$ is seed 2 at $k = 8$, where accuracy on RM-solvable queries plateaus at 0.86–0.89, just under the threshold, so that τ_{RM} is never recorded. At $\theta = 0.75$, the threshold used by He et al. [2024], RM-solvable queries come first at every k in every seed. Resolved by query type, the move from chance to generalization is two transitions, and the endpoint map of He et al. [2024] unfolds in time. With few examples the model stays between the two transitions, and more examples bring the second one earlier.

The depth-2 configuration stops at Ratio Matching. Accuracy on non-RM queries stays near chance for all 200k steps and never exceeds 0.14 at any k (Fig. 1, top row). RM-solvable queries are learned late and only partly. With 16 examples their accuracy peaks between 0.20 and 0.80 across seeds without crossing the threshold. Only seed 3 at $k = 4$ crosses it, and that run enters the interpolation regime and never leaves it. The authors' released depth-2 checkpoint ends in the same state, with 0.83–1.00 on RM-solvable queries and 0.02–0.09 on non-RM queries for $k \leq 16$. This is a contrast between configurations and not between depths alone. The depth-2 configuration also differs in the number of tasks, the fraction of inputs seen and the batch size, and our depth-2 runs fall

Table 3: Modular arithmetic, τ_{RM} / τ_{MR} in thousands of steps at $\theta = 0.9$. A dash means the clock is never reached in 200k steps. Wherever both clocks are reached, τ_{RM} comes first, and between the two clocks the model answers RM-solvable queries only. For example, at depth 4 and $k = 16$, non-RM queries are still at chance over steps 10–15k in seed 2 and 18–29k in seed 3.

	$k = 4$	$k = 8$	$k = 16$
depth 4, seed 1	15.9 / —	51.1 / 75.5	23.4 / 42.1
depth 4, seed 2	10.1 / —	— / 123.6	13.5 / 47.2
depth 4, seed 3	16.4 / —	52.0 / —	18.0 / 39.0
depth 2, seed 3	47.2 / —	— / —	— / —
depth 2, seeds 1–2	— / —	— / —	— / —

short of the released checkpoint at $k = 8$ and 16, unlike our depth-4 runs (App. A). Ratio Matching is specific to this task. Asking the same question in another setting needs a split that does not depend on a named algorithm, and the blend of §2 is such a split.

4 Hyperteacher

Setting. A hyperteacher [Schug et al., 2024, Redhardt et al., 2025] builds a family of regression tasks from M shared modules. Each task has a code z with K non-zero entries, which selects K modules and mixes them into the weights of a small target network, $W(z) = \sum_m (z_m/\sqrt{K})\,\Theta_m$. The task is the function computed by that network. The student receives the code z and an input, and predicts the target network's output. It is trained on N codes and tested on held-out codes, and each task is scored by R^2. The student is an MLP or a transformer, with about 3.2M parameters each (App. A). A held-out task is interpolable if the blend of the training tasks reaches $R^2 \geq \theta$ on it (Algorithm 1).

We vary the amount of data, the difficulty of the task and the depth of the student. Data is the number of training tasks, $N \in \{M, 2M, 4M\}$. Difficulty is the hidden width of the target network (16, 32 or 64), since a wider target is a harder function to learn. Student depth is 1, 2, 4 or 8 layers or blocks, with 4 as the default. Every setting is run on 9 task families ($M \in \{8, 16, 32\} \times K \in \{3, 4, 5\}$) with 3 seeds. This gives 972 runs, 486 per student (Table 4). Counts of transformer runs below pool all three arms unless a setting is named.

Do interpolable tasks come first? They do. In 188 of the 189 transformer runs that reach both clocks, τ_{interp} precedes τ_{beyond}. Seven further runs reach τ_{beyond} without reaching τ_{interp}. All seven are at $N = M$, where too few tasks are interpolable for their clock to be timed, so they do not bear on the order. The interpolation regime then ends. By the end of training the transformer solves all interpolable tasks, and 54%, 99% and 100% of the non-interpolable ones at $N = M$, $2M$ and $4M$.

Is the order an artefact of group size or threshold? It is not. The two groups can differ greatly in size, and a small group has a coarse solved fraction, so it might reach 90% first for that reason alone. If that drove the order, the smaller group would lead. It does not. Interpolable tasks come first in all 71 runs in which their group is at least 1.5 times larger than the other, the case least favourable to them (Table 7). Compared task by task, which does not depend on group size, an interpolable task is solved before a non-interpolable one more often than not in every default run of both students (App. B). With groups of exactly equal size, interpolable tasks still come first in both students (Fig. 3). Group size therefore affects how precisely the clocks are measured, but not their order. The threshold does not drive the order either. At $\theta = 0.75$, which changes both the split and the clocks, the order holds in all 78 transformer runs that reach both clocks, and the group means, which use no threshold, separate in the same way (Fig. 3, dotted lines).

Is the order explained by task difficulty? Interpolable tasks might simply be easier, in which case the order would reflect difficulty and not a separate stage. We test this in three ways (Table 6; details in App. C). First, we hold difficulty fixed. We measure a task's difficulty in ways that never look at the training pool, namely the number of modules it combines, how non-linear it is, and how well the same student learns it from the full pool. If interpolable tasks were just easier, comparing tasks of equal difficulty would remove their advantage. Second, we follow the same task as the pool grows. A task's intrinsic difficulty does not change when training tasks are added, but whether it is interpolable can change. If only difficulty mattered, tasks that become interpolable would gain no more than tasks

Table 4: Run accounting. Each level of each arm has 9 families $\times$ 3 pool sizes $\times$ 3 seeds $= 81$ runs. The lower part counts the transformer runs that reach each clock.

arm	MLP	transformer
default (width 16, depth 4)	81	81
teacher width 32 and 64	162	162
student depth 1, 2 and 8	243	243
total	486	486
reach both τ_{interp} and τ_{beyond}		189
of which depth 4 / depth 2 / depth 8 / depth 1		113 / 42 / 34 / 0
reach τ_{beyond} but not τ_{interp}		7
reach both τ_{gen} and τ_{beyond}		196

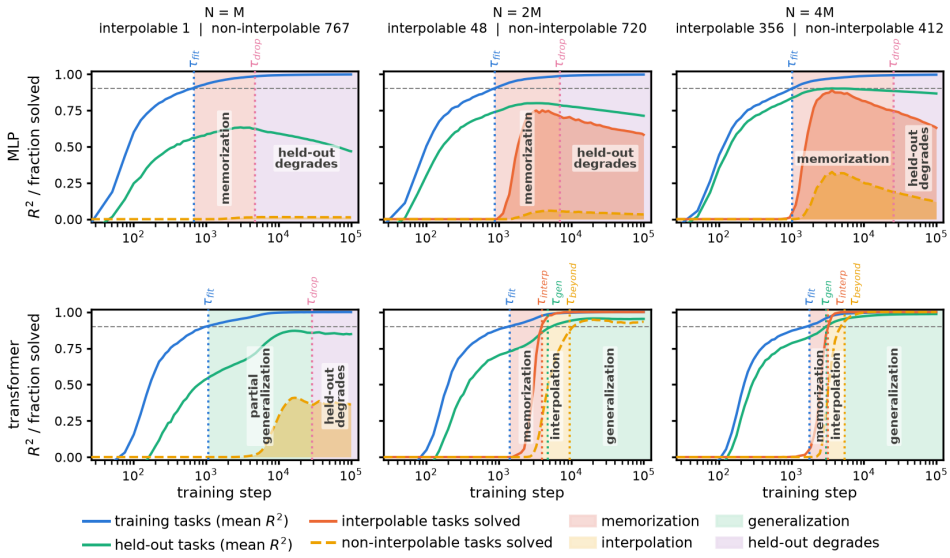


Figure 2: Hyperteacher dynamics against the number of training tasks N ($M = 32$, $K = 3$; means over 3 seeds). The transformer (bottom) reaches τ_{gen} before τ_{beyond}. The MLP (top) never reaches τ_{interp} here. At $N = M$ almost nothing is interpolable, and the transformer solves 40% of the other tasks without reaching τ_{beyond}.

that stay non-interpolable. Third, for the transformer, which solves nearly every task by the end, we repeat the first test on whether a task is solved at 5k steps.

For the MLP, difficulty does not explain the gap. An interpolable task is solved 36 points more often than another task in the same run, and at equal difficulty the gap is $+0.41$, so it does not shrink. A task that becomes interpolable gains 45 points more than a task that stays out of reach, although it had no head start (solved 0.07 of the time beforehand, against 0.12 for the others). For the transformer, little gap is left at the end of training because nearly every task is solved, so the timing test is the informative one. At 5k steps the gap is $+0.26$ at equal difficulty against $+0.30$ raw. Interpolable tasks are ahead early, and difficulty as we measure it removes little of that lead. Our first two difficulty measures explain 7–32% of the variation in full-pool error, so the control is firmer for the MLP, where the gap survives every test, than for the transformer. For the transformer the checks support interpolability as a contributor, without establishing that it is the only one.

What does the aggregate threshold see? It sees neither transition. The usual threshold τ_{gen} is crossed before τ_{beyond} in all 196 transformer runs that reach both (Fig. 2), so the model is declared to generalize before it has solved the non-interpolable tasks. In 109 of the 189 runs that reach both group clocks, τ_{gen} is crossed even before τ_{interp}, which is possible because a mean can reach θ while many individual tasks sit just below it. The threshold marks when generalization becomes visible. It does not show which regime the model is in.

Table 5: Transformer clocks over all 9 families at depth 4. Clock columns give the median step over the runs that reach the clock. The last column gives the runs that reach τ_{beyond}, out of those where it can be measured. Width is the teacher's hidden width.

setting	τ_{interp}	τ_{gen}	τ_{beyond}	reach τ_{beyond}
$N = M$	3,551	8,476	27,451	3/27
$N = 2M$	3,576	3,501	7,114	22/27
$N = 4M$	2,826	1,901	5,301	17/18
$N = 2M$, width 32	6,751	6,951	17,376	19/27
$N = 2M$, width 64	13,901	14,226	29,914	18/27

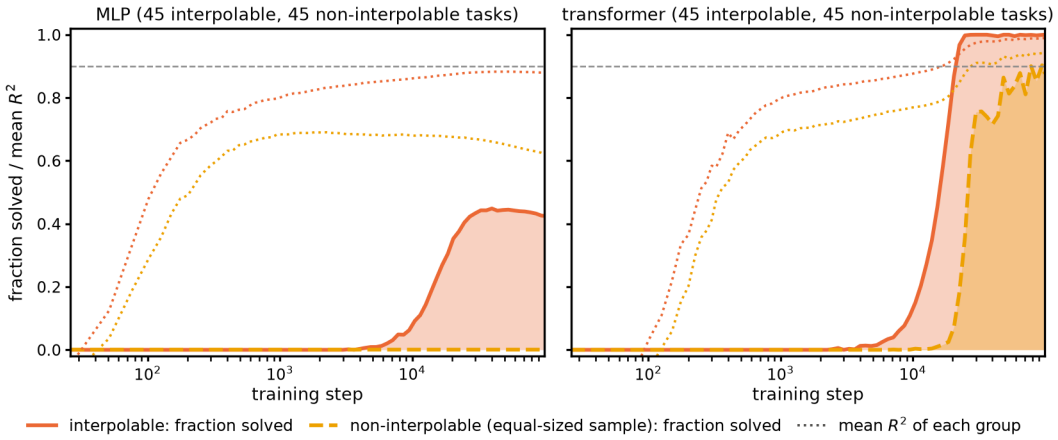


Figure 3: Equal-sized groups. Each seed's interpolable tasks are compared with an equal-sized random sample of its non-interpolable tasks ($M = 32$, $K = 3$, $N = 2M$, teacher width 64; mean over 3 seeds). Shaded areas show the fraction solved and dotted lines the mean R^2 of each group. The MLP never solves a non-interpolable task. The transformer solves interpolable tasks first and nearly all non-interpolable tasks later.

Is going beyond interpolation limited by capacity? Not within the range we tested. With the same number of parameters as the transformer, the MLP solves 40–82% of the interpolable tasks but only 5%, 10% and 32% of the non-interpolable ones at $N = M$, $2M$ and $4M$, and reaches τ_{beyond} in at most 2 of 27 runs per pool size (Fig. 2, top). Depth does not change this, since the MLP reaches τ_{beyond} in at most 2 of 27 runs at every depth from one to eight layers (App. D). The MLP is interpolation-bound in this family. Its limit is not explained by parameter count or depth, which points to inductive biase.

What sets the length of the interpolation regime? Data shortens it. More training tasks make more held-out tasks interpolable (a median of under 1%, 37% and 91% at $N = M$, $2M$ and $4M$). The first clock barely moves, while τ_{beyond} falls from 27k steps at $N = M$, where only 3 of 27 runs reach it, to 7.1k and 5.3k steps (Table 5). Difficulty stretches it. Each doubling of the teacher's width delays both clocks about twofold (Fig. 6), so the regime lasts longer in steps, with a median length of 2.8k, 5.6k and 7.1k steps at widths 16, 32 and 64, pooled over pool sizes. It does not last longer relative to its start, since the ratio $\tau_{\text{beyond}}/\tau_{\text{interp}}$ is 1.95, 1.81 and 1.55.

5 Discussion and limitations

The move from memorization to generalization decomposes into staggered transitions. Interpolable problems are solved first, and the rest later or never. Between the two lies an interpolation regime, and the standard threshold is crossed inside it without revealing it.

One explanation for why the clocks separate, which is consistent with our results but which we have not tested directly, is that reuse is the shorter route to low loss. Ratio Matching needs one example and

Table 6: Is "interpolable" just "easy"? The gap is how much more often an interpolable task is solved than another task in the same run ($\theta = 0.9$; 95% block-bootstrap intervals). The last column covers the two families with full-pool runs, before $\rightarrow$ after adjusting. For the MLP the gap survives every control.

	raw gap	at equal hotness and non-linearity	same task, pool grows	at equal full-pool error
MLP	$+0.36$	$+0.41$ [$+0.35$, $+0.46$]	$+0.45$ [$+0.39$, $+0.50$]	$+0.54 \rightarrow +0.55$
transformer	$+0.09$	$+0.09$ [$+0.05$, $+0.13$]	$+0.03$ [-0.01, $+0.08$]	$+0.03 \rightarrow +0.03$

one operation, whereas Modular Regression has to combine several examples. In the hyperteacher, the training functions already come close to many held-out targets, so recombining them reduces the loss before anything more general is learned. Two of our results fit this account. Adding training tasks makes more targets reachable by reuse, and the regime shortens. Making the target functions harder delays both clocks together, and the regime stretches in steps.

We see the interpolation regime as a possible source of plausible-but-wrong outputs, that is, hallucinations. An interpolating model is right where interpolation suffices and close but wrong elsewhere. In our hardest setting, non-interpolable tasks reach a mean R^2 of 0.61–0.82 while almost none is solved (App. B), and a threshold on the mean passes such a model. We do not test this in large models [cf. Kalai et al., 2025].

Limitations. The clocks are behavioural. They date when a class of problems is solved and do not show how the model solves it, and capabilities can precede behaviour [Park et al., 2024]. The label "interpolable" describes a problem relative to the training set and not the model's algorithm, and a learner that acquires its most frequent modules first could produce the same order. Both settings are synthetic. The hyperteacher is linear in its code, which is what makes a linear blend a natural reuse rule there, and modular arithmetic has 3 seeds per depth.

6 Related work and positioning

Grokking and what its suddenness hides. Grokking was identified as a delayed and sometimes sharp rise in held-out performance after the training data have been fit [Power et al., 2022], which has encouraged reading generalization as a relatively discrete transition. Prior work has shown that this apparent suddenness can conceal richer dynamics. Mechanistic studies find that abrupt generalization can emerge from gradual internal change, with circuits forming and being refined before the behavioural transition becomes visible [Nanda et al., 2023], and that delayed generalization can reflect a shift from lazy to feature-learning dynamics [Kumar et al., 2024]. Work on emergent abilities shows that coarse or threshold-based evaluation can make continuous change look discontinuous [Schaeffer et al., 2023]; it concerns emergence with scale rather than over training, but the caution carries over. Other work treats grokking as a genuine phase transition under particular definitions [Žunkovič and Ilievski, 2024, Rubin et al., 2024], so the field has no single settled interpretation. Together, these results caution against reading a sharp aggregate curve as evidence that the underlying learning process is itself instantaneous.

Our question. Our question is complementary but distinct. Prior work asks what changes internally before grokking becomes visible; we ask what the model can actually generalize to during that interval, that is, whether held-out generalization emerges uniformly across problems. The closest precedent, Nanda et al. [2023], resolves the transition inside the network; we resolve the behaviour itself, across problems. *Learning to grok* [He et al., 2024] identifies the algorithms, Ratio Matching and Modular Regression, mechanistically in trained models, and characterizes which is used under which architectural and context conditions, and how out-of-distribution generalization depends on the number of pre-training tasks. It leaves open a temporal question: are these capabilities acquired simultaneously during training, or in a systematic order? We use its identification to follow the algorithms through training. *The two clocks* [Wang et al., 2026] supplies the method, dating rule learning and memorization by threshold crossings, but does not resolve the order of capabilities within generalization. Intermediate stages are also known in in-context learning: task identification [Kirsch et al., 2022], a pre-training-task Bayesian solution below a diversity threshold [Raventós et al., 2023],

task retrieval versus learning [Lin and Lee, 2024], competing retrieval and inference phases [Park et al., 2025], differential learning kinetics [Nguyen and Reddy, 2025] and transient in-context learning [Singh et al., 2023]. Our split of held-out problems needs no named algorithm and can be timed against the aggregate threshold.

Positioning. The resulting picture is neither that grokking is an illusion nor that every capability emerges smoothly. Individual transitions can be sharp, yet the global transition from memorization to generalization decomposes into staggered capability transitions when observed at finer behavioural resolution. Our contribution is a behavioural refinement of the grokking transition: prior work has shown that smooth internal dynamics can underlie apparently sudden behavioural change; we show that the behavioural transition itself can contain an ordered intermediate structure that aggregate evaluation obscures. Grokking may appear as a single behavioural event under aggregate evaluation, while finer-grained analysis reveals an ordered progression of generalization capabilities.

References

Randall Balestriero, Jerome Pesenti, and Yann LeCun. Learning in high dimension always amounts to extrapolation. *arXiv preprint arXiv:2110.09485*, 2021.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2406.02550.

Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why language models hallucinate. *arXiv preprint arXiv:2509.04664*, 2025.

Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *International Conference on Learning Representations*, 2024. arXiv:2310.06110.

Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 30135–30188, 2024.

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.

Alex Nguyen and Gautam Reddy. Differential learning kinetics govern the transition from memorization to generalization during in-context learning. In *International Conference on Learning Representations*, 2025. arXiv:2412.00104.

Core Francisco Park, Maya Okawa, Andrew Lee, Ekdeep Singh Lubana, and Hidenori Tanaka. Emergence of hidden capabilities: Exploring learning dynamics in concept space. In *Advances in Neural Information Processing Systems*, 2024.

Core Francisco Park, Ekdeep Singh Lubana, Itamar Pres, and Hidenori Tanaka. Competition dynamics shape algorithmic phases of in-context learning. In *International Conference on Learning Representations*, 2025. arXiv:2412.01003.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-Bayesian in-context learning for regression. In *Advances in Neural Information Processing Systems*, 2023.

Florian Redhardt, Yassir Akram, and Simon Schug. Scaling can lead to compositional generalization. *arXiv preprint arXiv:2507.07207*, 2025.

Noa Rubin, Inbar Seroussi, and Zohar Ringel. Grokking as a first order phase transition in two layer networks. In *International Conference on Learning Representations*, 2024. arXiv:2310.03789.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems*, 2023.

Simon Schug, Seijin Kobayashi, Yassir Akram, Maciej Wolczyk, Alexandra Maria Proca, Johannes von Oswald, Razvan Pascanu, Joao Sacramento, and Angelika Steger. Discovering modular solutions that generalize compositionally. In *International Conference on Learning Representations*, 2024.

Aaditya K. Singh, Stephanie C. Y. Chan, Ted Moskovitz, Erin Grant, Andrew M. Saxe, and Felix Hill. The transient nature of emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems*, 2023.

Binxu Wang, Emma Lucia Byrnes Finn, and Bingbin Liu. The two clocks and the innovation window: When and how generative models learn rules. *arXiv preprint arXiv:2605.10019*, 2026.

Bojan Žunkovič and Enej Ilievski. Grokking phase transitions in learning local rules with gradient descent. *Journal of Machine Learning Research*, 25(199):1–52, 2024.

Appendix contents

A Experimental details (hyperteacher; modular arithmetic) 10
B Is the order an artefact of group size? 11
C Is "interpolable" just "easy"? 12
D Is going beyond limited by capacity? Student depth 12
E A threshold-free check: the coverage fingerprint 13
F Additional figures and tables 13

A Experimental details

Hyperteacher.

- **Teacher:** the configuration of Redhardt et al. [2025], with teacher hidden width 16 unless stated, 16 input/output samples per task, discrete magnitudes, and 25% of codes held out.
- **Training pools:** N codes chosen to have rank M (so they span all codes), to be compositional (every module appears) and to be connected (the module co-occurrence graph is connected); a frozen held-out set.
- **Training:** 100k steps at batch 128 (12.8M task instances); AdamW, learning rate $3 \cdot 10^{-4}$ ($3 \cdot 10^{-3}$ for one-layer students, chosen in a pre-run learning-rate check), weight decay 0.003, 1k warmup.
- **Students:** MLP with 4 hidden layers of width 1024 (3,215,376 parameters); transformer with 4 blocks, $d_{\text{model}} = 256$, 4 heads, feed-forward width 1024 (3,165,200 parameters). Depth ladder, 1/2/8 layers or blocks: MLP 66,576 / 1,116,176 / 7,413,776; transformer 803,984 / 1,591,056 / 6,313,488.
- **Scoring:** per-task R^2 with the per-output-unit variance of the targets. Endpoint scores average 64 repeats. Trajectories are scored every 25 steps with 4 repeats, on all held-out tasks at $M = 16, K = 3$ and on a fixed 256-task sample otherwise.
- **Blend:** Algorithm 1. It is computed on binary task codes, while the students see codes with random magnitudes.
- **Held-out degradation:** τ_{drop} is the first step, after the held-out peak, at which the held-out score is reliably 0.02 below that peak.
- **Registration:** the clocks, the interpolation split and the difficulty check (App. C) were registered before they were computed on these runs. The group-size checks (App. B) and the comparison of τ_{gen} with τ_{beyond} were added afterwards.

- **Hardware:** NVIDIA A10G GPUs.

The blend is Algorithm 1 in §2. Coverage (App. E) skips the fit: it is the best R^2 of a single training task used as is.

Modular arithmetic.

- **Code:** the public code at its released configurations. Depth 4: 384 tasks, 80% of each task's inputs seen, batch 1536. Depth 2: 512 tasks, 60%, batch 1024.
- **Shared settings:** $d = 512$, 4 heads, AdamW with learning rate $1.5 \cdot 10^{-4}$ and weight decay 2.0, cosine schedule, 10k warmup, 200k steps.
- **Our one change:** checkpoints are saved from step 100 rather than after warmup.
- **Clocks:** "reliably" means at that checkpoint and the next four, of 80 log-spaced checkpoints.
- **Pre-registration:** seeds 2–3 at depth 4 and all depth-2 seeds were analysed with readings fixed in advance for each possible order (RM first, together, MR first, or mixed). Seed 1 at depth 4 was a pilot, seen before the readings for seeds 2–3 were fixed. At $\theta = 0.9$ the depth-4 reading was "mixed" (RM first at $k = 4$ and 16 in both seeds; $k = 8$ as in §3); at $\theta = 0.75$, "an RM-first phase replicates". Depth 2, coded at $k = 16$, read "does not generalize in this configuration".
- **Reproduction:** on 90 held-out tasks that none of the models trained on, the released depth-4 checkpoint falls within the range of our three seeds (± 0.03) at every k. At depth 2 (28 common tasks) the release is better than our runs at $k = 8$ and 16. It was trained on a task list that the public code does not produce for any seed in $[0, 4999]$, so our results describe the public code's configuration.
- **Hardware:** one H100 per run (about 2.9 h at depth 4, 1.2–1.3 h at depth 2).

B Is the order an artefact of group size?

The two groups can differ greatly in size: at $N = 2M$ with $M = 32$, $K = 3$, 7.4% of held-out tasks are interpolable. A small group's solved fraction is coarse (with 17 tasks, one task moves it by about 6%), so the smaller group might reach 90% sooner for that reason alone. If that drove the order, the order would follow whichever group is smaller. Table 7 splits the 189 transformer runs that reach both clocks by the ratio of group sizes. The interpolable group comes first in every case, including all 71 runs in which it is at least 1.5 times larger than the other, the case least favourable to it. Group size affects the precision of the clocks, not their order. At $\theta = 0.75$, which also changes the split, τ_{interp} precedes τ_{beyond} in all 78 transformer runs that reach both.

Table 7: The order holds whichever group is larger: transformer runs in which τ_{interp} precedes τ_{beyond}, by the ratio of interpolable to non-interpolable tasks.

group sizes	τ_{interp} before τ_{beyond}
interpolable group smaller (ratio < 0.67)	68 / 68
roughly equal (0.67–1.5)	49 / 50
interpolable group larger (ratio > 1.5)	71 / 71

A size-free check. We also compare individual tasks. A task's solve time is the first step from which its $R^2 \geq 0.9$ holds reliably (never-solved tasks count as later than any solved task). Per run, p is the probability that a random interpolable task is solved before a random non-interpolable task; $p = 0.5$ means no order, and group sizes do not enter.

- **Default runs:** $p > 0.5$ in every run, for both students and at every pool size (Table 8).
- **All 636 runs, every arm:** $p > 0.5$ in 559. The other 77 solve essentially nothing: 74 are exact ties at 0.5, nearly all one-layer students; 3 sit at 0.47–0.50.

Table 8: Interpolable tasks are individually solved earlier. p = probability that a random interpolable task is solved before a random non-interpolable task (median over runs); solve times are medians over the tasks that are solved. Default students, $\theta = 0.9$.

student	N	p (runs with $p > 0.5$)	solve time, interpolable	solve time, non-interpolable
transformer	M	0.96 (8/8)	2.2k	5.9k
	$2M$	0.91 (27/27)	2.7k	4.6k
	$4M$	0.90 (18/18)	2.5k	3.3k
MLP	M	0.81 (8/8)	1.0k	2.5k
	$2M$	0.81 (27/27)	1.2k	2.4k
	$4M$	0.85 (18/18)	1.4k	3.5k

Equal-sized groups. Finally, we remove the size difference outright (Fig. 3). In the hardest setting ($M = 32$, $K = 3$, $N = 2M$, teacher width 64), we take each seed's interpolable tasks (45 in total over 3 seeds) and an equal-sized random sample of its non-interpolable tasks.

- **MLP.** 17 of the 45 interpolable tasks are solved, the first at about 3k steps; none of the 45 non-interpolable tasks is ever solved. Over 1,000 random samples, the non-interpolable solved fraction peaks at a median of 0.02, and never above 0.11. Yet the outputs are close: from 1k steps on, the non-interpolable tasks' mean R^2 stays at 0.61–0.70.
- **Transformer.** All 45 interpolable tasks are solved, and 40 of 45 non-interpolable tasks later: median solve step 20.2k against 27.5k. An interpolable task is solved before a non-interpolable one with probability 0.98 [0.94, 0.99]. Between 1k and 20k steps, while at most 2% of the non-interpolable tasks are solved, their mean R^2 is already 0.68–0.82.

With equal-sized groups, interpolable tasks still come first in both students. Only the transformer then goes beyond them. Group size therefore cannot explain the order.

C Is "interpolable" just "easy"?

If interpolable tasks were simply easier, holding difficulty fixed would remove their advantage, and a task that becomes interpolable as the pool grows would gain no more than one that stays out of reach. Within each run we regress "solved" on "interpolable" while controlling for three difficulty measures that never look at the training pool: hotness (modules combined), non-linearity ($1 - R^2$ of the best affine fit of the task) and, where available, the same student's error on the task after full-pool training. We also compare the *same* task across pool sizes, between tasks that become interpolable and tasks that stay out of reach (Table 6). For the MLP the gap does not shrink: when a task becomes interpolable, the MLP's chance of solving it rises 45 points more than for tasks that stay out of reach, although those tasks had no head start (solved 0.07 vs. 0.12 beforehand, at $M \to 2M$). For the transformer little gap is left at the end of training, but the order survives adjustment: at 5k steps the gap is $+0.26$ [$+0.21$, $+0.32$] against $+0.30$ unadjusted. Hotness and non-linearity explain only 7–32% of the variation in the students' full-pool error, so the transformer's result is the less firmly controlled one. Intrinsic difficulty, as we measure it, cannot explain the MLP's gap; for the transformer it is not excluded as a contributor.

Details. The main test is a per-run linear-probability model of "solved" on "interpolable", hotness indicators, non-linearity and its square; the pooled coefficient is the mean over eligible runs (both groups ≥ 5 tasks). Intervals: block bootstrap over the 27 family–seed blocks, 10,000 resamples. The same-task test is a difference-in-differences over the pool-size pairs $M \to 2M$ and $2M \to 4M$. Full-pool error is available for $M \in \{16, 32\}$, $K = 3$. The timing outcome is "solved" averaged over the 1,000 steps ending at 2k, 5k, 10k and 20k steps.

D Is going beyond limited by capacity? Student depth

If the MLP stayed interpolation-bound for lack of capacity, deeper MLPs should go beyond. One layer or block never reaches τ_{interp}. Two transformer blocks reach every clock later (τ_{interp} 10.7k vs. 3.6k steps at $N = 2M$); four and eight behave alike (τ_{interp} 3.0k and τ_{beyond} 7.1k with eight

blocks). The MLP reaches τ_{beyond} in at most 2 of 27 runs at every depth (Fig. 4). Depth changes when the transformer's clocks fire, but not whether the MLP goes beyond: in this range, its limit is not explained by depth or parameter count.

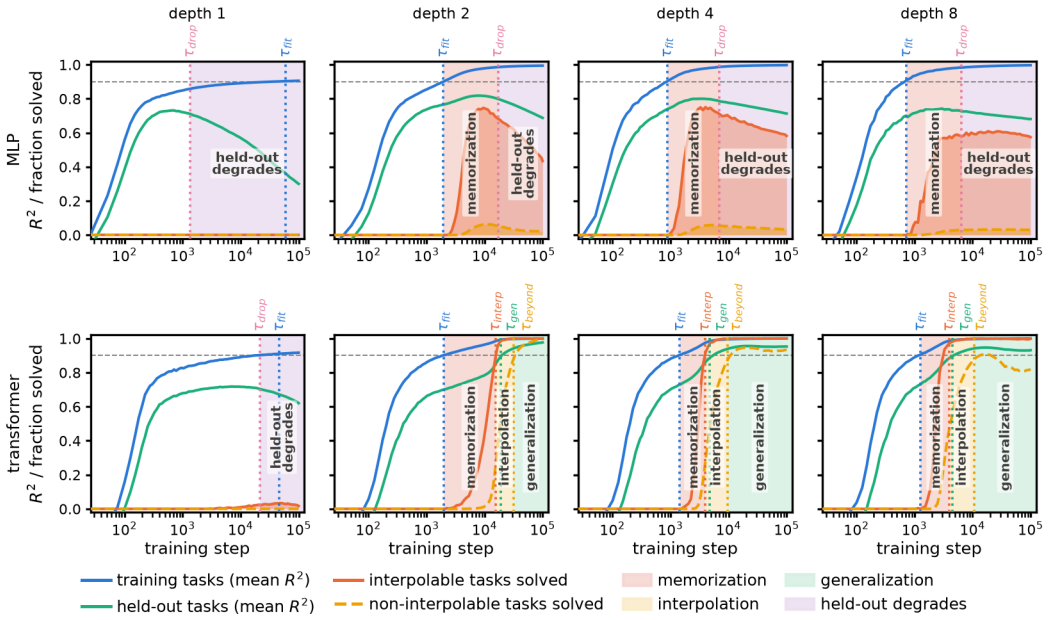


Figure 4: Learning dynamics against student depth, 1, 2, 4 and 8 layers or blocks ($M = 32$, $K = 3$, $N = 2M$, teacher width 16; 48 interpolable and 720 non-interpolable tasks over 3 seeds in every panel; curves and clocks as in Fig. 2).

E A threshold-free check: the coverage fingerprint

The clocks depend on a threshold; coverage gives a view without one. The coverage of a held-out task A is the R^2 of the single best training task used as is to predict A, on the 512 probe inputs. At every checkpoint we regress the student's per-task held-out R^2 on coverage across held-out tasks; the *coverage effect* is slope $\times$ range of coverage. A large effect means the model does worst where its training tasks cover least, as an interpolator would. Over the 81 default runs per student, the effect decays in all 81 transformer runs and in 53 MLP runs. It re-grows late, together with the degradation of the held-out score, mostly at $N = M$ (MLP 26 of 27 runs, transformer 24 of 27). At larger pools it re-grows far more often for the MLP (21 and 14 of 27 runs at $2M$ and $4M$, against 10 and 2 for the transformer; Fig. 5).

F Additional figures and tables

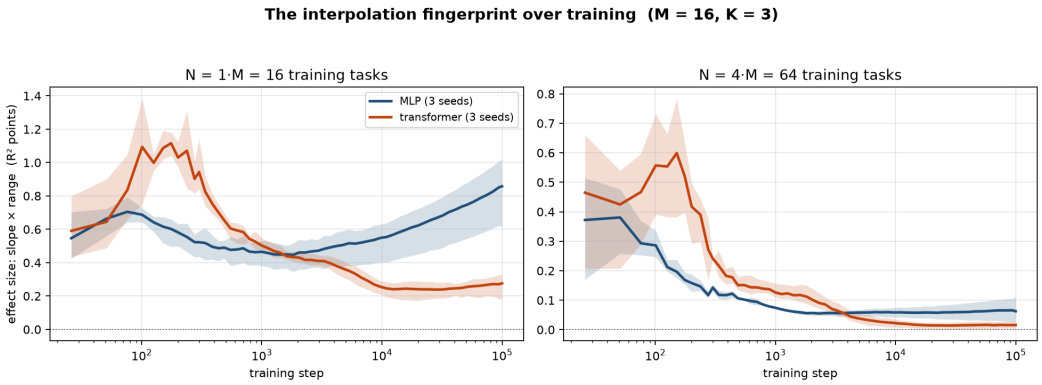


Figure 5: The coverage effect over training, $M = 16$, $K = 3$, mean over 3 seeds, band = seed range. With 16 training tasks ($N = M$, left) the transformer's effect decays while the MLP's re-grows; with 64 ($N = 4M$, right) both decay, the transformer's to near 0.

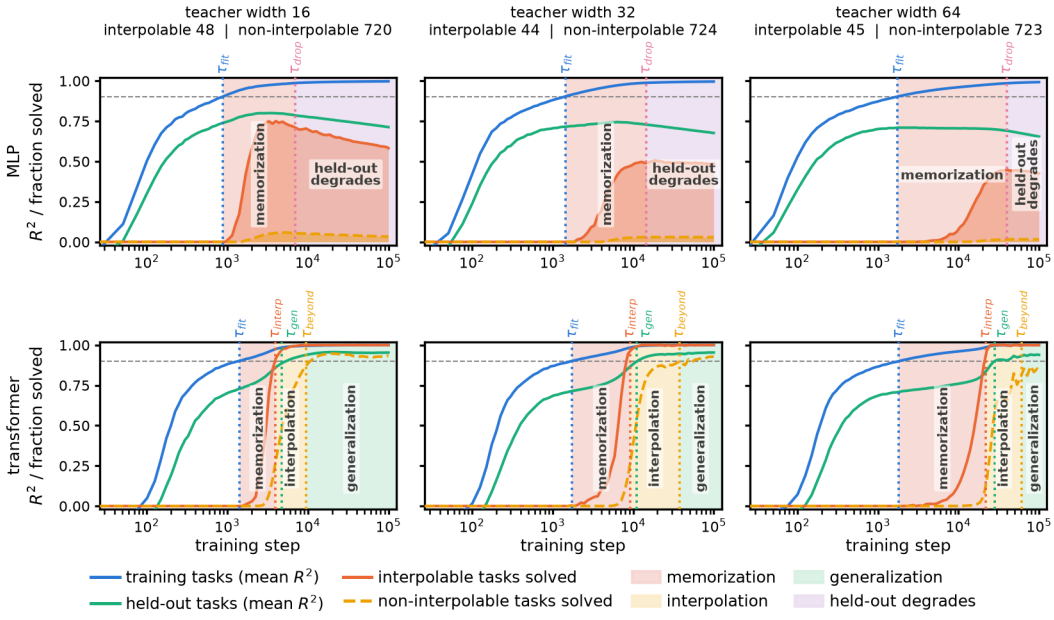


Figure 6: Learning dynamics against task difficulty, teacher hidden width 16, 32 and 64 ($M = 32$, $K = 3$, $N = 2M$; curves and clocks as in Fig. 2). Every clock moves later as width doubles; the transformer still interpolates first, and the MLP never reaches τ_{interp}.