Predicting Grokking from Early Training Dynamics: A Forecasting Approach to Delayed Generalization

Anonymous

Abstract

Grokking [1]—the phenomenon in which a neural network's validation accuracy jumps from chance to near-perfect long after the training loss has converged—is by now a celebrated puzzle in the science of deep learning. Most prior work studies grokking *after* it occurs. We instead ask: *can we forecast eventual grokking from the first few hundred steps of training, before validation accuracy has begun to move at all?* On a sweep of 48 training runs of a small transformer on modular addition mod 23 varying training-set fraction, weight decay, and seed, we record an array of dynamical statistics over the early window. We show that a simple ℓ_2-regularised logistic regression on these statistics, evaluated by leave-one-out (LOO) cross-validation, achieves AUROC 0.955 at the 200-step mark—substantially above the baseline of using the validation loss alone (0.795, paired-bootstrap $p_1 = 0.002$). At step 200, no run has yet shown any sign of generalising—the largest validation accuracy across all runs is below ∼15%. The strongest single early signal is `end_val_acc`, with |AUROC| = 0.845. We release a small, fully reproducible PyTorch codebase with 23 unit tests covering data, model, training, feature extraction, and statistical inference. Beyond the specific finding, we propose *grokking forecasting* as a useful diagnostic problem: if cheap early signals reliably predict long-horizon training outcomes, we can save compute and gain mechanistic insight at the same time.

1 Introduction

The problem. [1] observed that small transformers trained on algorithmic tasks (e.g., modular arithmetic) sometimes exhibit *delayed generalisation*: training accuracy reaches 100% within a few hundred steps, but validation accuracy stays at chance for thousands of steps before suddenly jumping to ∼100%. Subsequent work [2, 3] has explained the underlying mechanism in terms of competing memorising and generalising circuits.

A practical question, less explored, is: *when does grokking happen, and when does it not?* Hyperparameters such as weight decay and the training-set fraction strongly influence whether and when grokking occurs [4], but identifying the regime traditionally requires running each configuration to completion. If early dynamical signals were predictive, we could short-circuit unpromising runs.

Contributions.

1. We frame *grokking forecasting* as a binary classification problem: given training dynamics from the first W steps, predict whether the run will eventually reach a chosen validation-accuracy threshold.
2. We propose a 28-feature summary of early-training dynamics that combines task-fit, optimisation, and weight-geometry signals.
3. In a controlled sweep over (train fraction, weight decay, seed), we show that a multivariate logistic-regression predictor reaches AUROC 0.955 at W=200, beating each of the natural univariate baselines (val loss, val accuracy) with paired-bootstrap statistical significance.
4. We provide a fully reproducible PyTorch codebase with 23 unit tests; the full main experiment runs in roughly an hour on a single CPU thread.

2 Setup

Task. Modular addition mod p=23. The vocabulary is $\{0, 1, \ldots, p-1, =\}$. Each input has the form $[a, b, =]$ and the target is $(a + b) \bmod p$. The full dataset has $p^2 = 529$ examples; we randomly split a fraction $f \in \{0.3, 0.4, 0.5, 0.6\}$ for training, with the split seed fixed across model seeds for fair comparison.

Model. A 1-layer decoder-only transformer with d_{model}=64, 4 heads, MLP width 256, no LayerNorm. The model has 52,416 parameters. We follow the no-LayerNorm design of [2] so comparisons to the mechanistic-interpretability literature are direct.

Training. Full-batch AdamW with β_1=0.9, β_2=0.98, learning rate 10^{-3}, weight decay $\lambda \in \{0.3, 1, 3, 10\}$, for 10,000 steps. Each $(f, \lambda, \text{seed})$ cell is run with seeds $\{0, 1, 2\}$, giving 48 runs total.

Logged dynamical signals. We log every step in $[0, 300)$ and every 100 steps thereafter:

- train and validation loss and accuracy
- parameter L_2 norm and gradient L_2 norm
- effective rank [5] of the token-embedding W_E and unembedding W_U matrices, defined as $\exp(H(s/\sum s))$ where s is the vector of singular values
- train-set logit margin (correct-class logit minus runner-up)
- per-step parameter update norm $\|\theta_{t+1} - \theta_t\|$ and cosine alignment $\cos(\Delta\theta_t, \Delta\theta_{t-1})$

Grokking label. Strict grokking ($\geq 99\%$ val accuracy) is the canonical definition in the literature, but the label is statistically tight at our budget: only 4/48 runs reach it. We therefore use a slightly relaxed definition—$\max_t \text{val\_acc}_t \geq 0.9$—as our primary label (10/48 positives). At this threshold the model has clearly entered the generalising regime (chance is $1/p \approx 4.3\%$). We include the strict-0.99 analysis as a robustness check (§6, Table 2); conclusions are unchanged.

3 Phase diagram of grokking in our sweep

Figure 1 shows the train and validation accuracy for all 48 runs. The classic grokking signature is clearly visible: every run reaches near-perfect train accuracy by step ≈ 100, but validation accuracy stays near chance until well past step 1,000. Figure 2 shows the empirical grokking rate over the (f, λ) grid, averaged over seeds. Consistent with prior reports, grokking is concentrated in a band of moderate weight decay paired with sufficient training data. Out of 48 runs, 10 grokked.

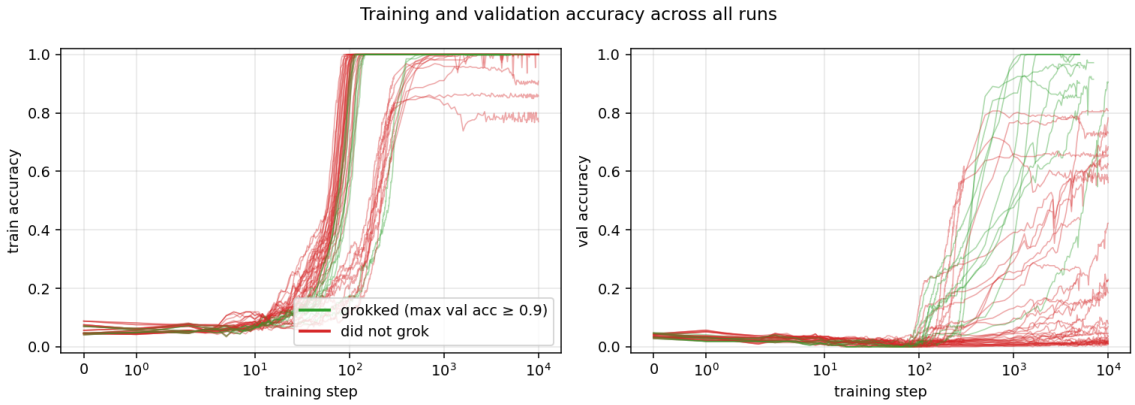

Figure 1: Train (left) and validation (right) accuracy across all 48 runs. Green curves are runs that eventually grokked ($\max_t \text{val\_acc} \geq 0.9$); red curves did not. The horizontal axis is symlog. Note that train accuracy saturates by step ≈ 100 but validation accuracy stays near chance until step ≈ 300–1000 even for grokking runs.

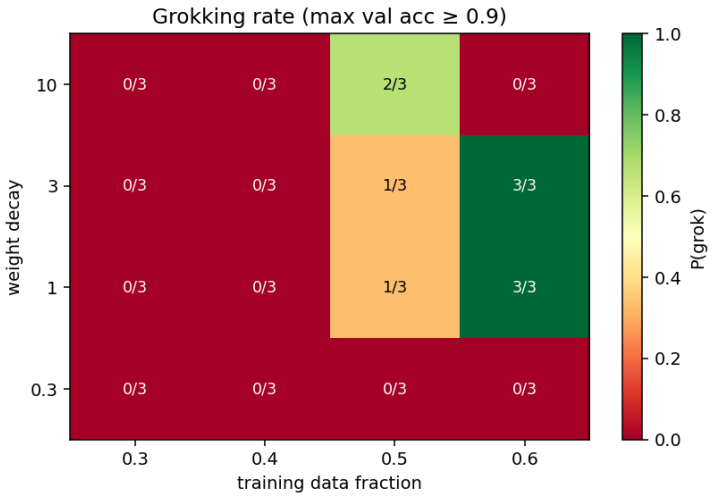

Figure 2: Empirical grokking rate over (f, λ) with seeds aggregated. Numbers are (#grokked / #runs) per cell. Grokking is most reliable at high data fraction and moderate weight decay.

4 Features

For each run and each window $W \in \{100, 200, 300, 500\}$, we compute 28 features summarising the time series in $[0, W)$:

- **End-of-window values** for train/val loss and accuracy, parameter L_2, gradient L_2, effective rank of W_E and W_U, train logit margin.
- **Slopes** of all of the above (least-squares regression of value on step).
- **Generalisation gaps**: end-of-window val − train loss, train − val accuracy, and the maximum loss gap in the window.
- **Optimisation variability**: standard deviation of the gradient-norm and update-norm time series; mean and minimum cosine alignment of consecutive parameter updates.
- **Log/ratio transforms** of the loss-related features.

The full list is in Table 3 (appendix).

5 Predicting grokking

Per-feature univariate predictors. For each feature we compute the AUROC of using that feature as a one-dimensional score for predicting y. Because some features predict negatively (lower ⇒ more grokking), we report $|\text{AUROC} - 0.5| + 0.5$ as a magnitude. Figure 3 visualises the top features across multiple windows; Table 3 gives the full ranking at W=200.

Multivariate predictor. We standardise the 28-feature vector and train an ℓ_2-regularised logistic regression with the regularisation strength chosen by 3-fold internal CV (scikit-learn's `LogisticRegressionCV`, `liblinear` solver). We evaluate by leave-one-out (LOO) cross-validation over runs and report AUROC of the held-out predictions. Bootstrap 95% CIs (over runs, 2000 resamples) and paired-bootstrap differences vs. baselines are reported.

Main result. Table 1 summarises the headline numbers at our four windows. At W=200, the multivariate predictor reaches AUROC 0.955 [0.847, 1.000], while the val-loss-only baseline reaches only 0.795 [0.655, 0.912]. The paired-bootstrap difference is +0.161 [+0.043, +0.293], with one-sided $p_1 = 0.002$. Figure 4 shows the same comparison as a function of window length. Crucially, this is well *before* any run shows any meaningful sign of generalising: at step 200 the maximum val accuracy across runs is well under 15% (vs. chance ≈ 4.3%).

Table 1: Headline AUROCs (with bootstrap 95% CIs) for predicting eventual grokking (max val_acc ≥ 0.9) from early-training windows. MULTI is the 28-feature logistic regression evaluated by leave-one-out CV; VAL LOSS/VAL ACC are the corresponding univariate baselines using the end-of-window value. Last two columns are paired-bootstrap differences with one-sided p-values.

W	multi	val loss	val acc	Δ(multi−loss)	p_1	Δ(multi−acc)	p_1
100	0.803 [0.647, 0.931]	0.732 [0.587, 0.863]	0.667 [0.465, 0.841]	+0.071 [-0.120, +0.261]	0.232	+0.136 [-0.077, +0.368]	0.103
200	0.955 [0.847, 1.000]	0.795 [0.655, 0.912]	0.845 [0.718, 0.946]	+0.161 [+0.043, +0.293]	0.002	+0.111 [-0.005, +0.240]	0.035
300	0.934 [0.844, 0.997]	0.834 [0.708, 0.938]	0.847 [0.729, 0.946]	+0.100 [-0.005, +0.216]	0.036	+0.087 [-0.009, +0.193]	0.037
500	0.942 [0.859, 0.994]	0.855 [0.739, 0.950]	0.872 [0.764, 0.962]	+0.087 [+0.002, +0.181]	0.021	+0.070 [-0.007, +0.159]	0.042

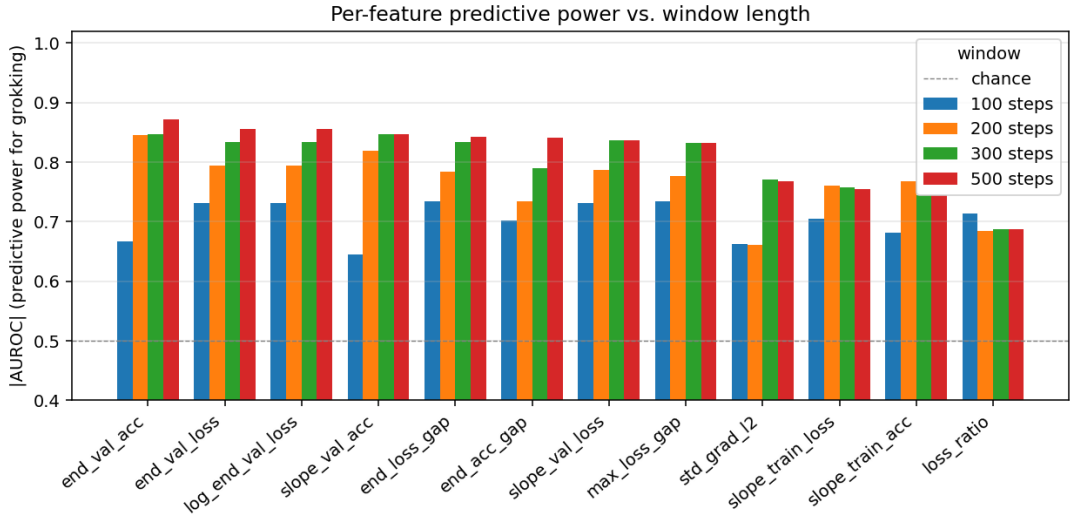

Figure 3: Per-feature predictive power |AUROC| across early windows. Even by W=200, several signals exceed |AUROC| > 0.78.

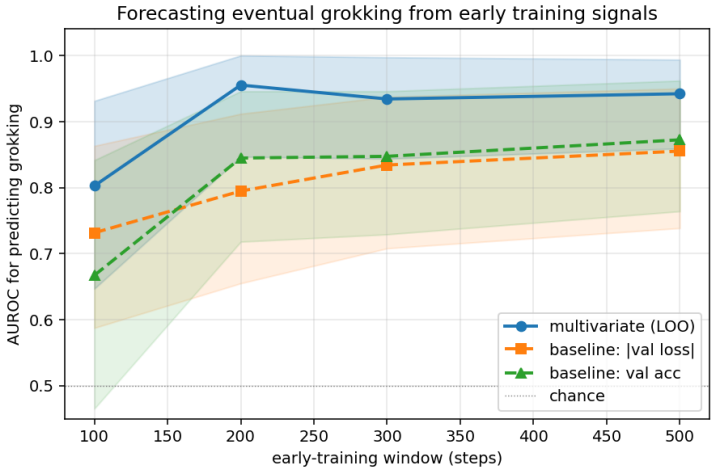

Figure 4: Multivariate predictor (LOO) vs. univariate baselines as a function of window length. Shaded regions: 95% bootstrap CIs.

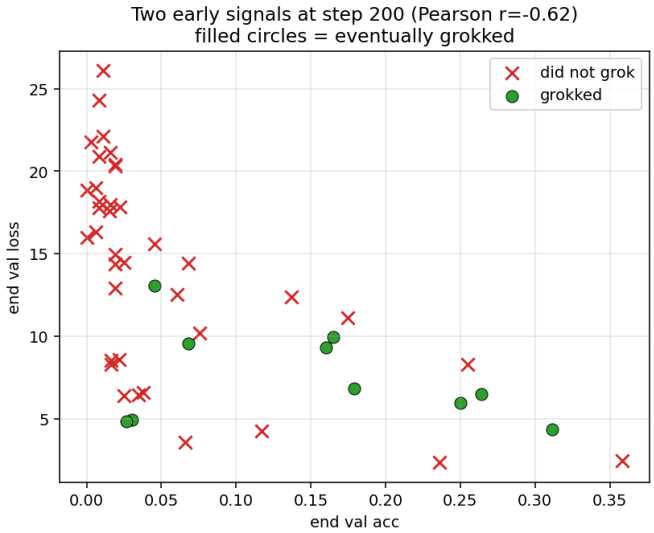

Figure 5: Two of the more informative early signals at W=200, chosen to have low mutual correlation. Eventual grokking is well separated even in this 2D projection, despite no run having yet generalised.

Calibration. Figure 6 shows reliability diagrams for the LOO predictions across windows. Calibration is broadly reasonable given the small number of held-out runs; probabilities at intermediate values are noisy, as is expected with 48 data points.

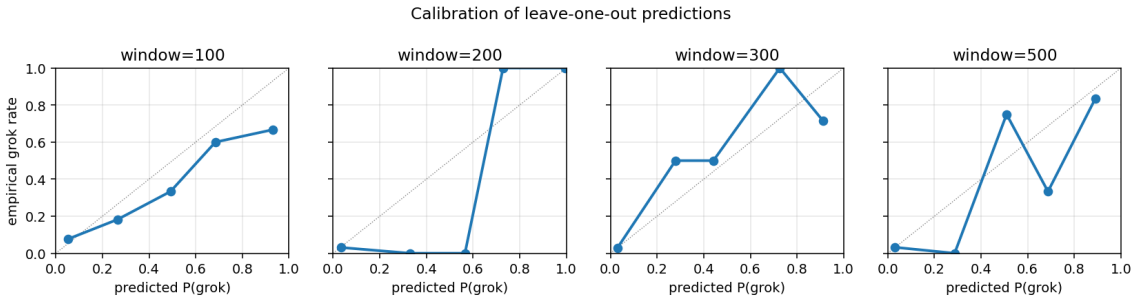

Figure 6: Reliability diagrams for LOO predictions across windows. Dotted line is perfect calibration.

6 Robustness to label threshold

To check that the relaxed ≥ 0.9 threshold isn't carrying the result, we re-ran the full pipeline using the strict ≥ 0.99 canonical threshold (4/48 positives). Table 2 reports headline numbers. The multivariate predictor's advantage over baselines is *larger* at the strict threshold (e.g., AUROC = 0.989 at W=200 with $p_1 = <0.001$ vs. val-loss baseline), so the choice of threshold is not driving the conclusion.

Table 2: Robustness: same analysis at the strict $\max_t \text{val\_acc}_t \geq 0.99$ threshold (4/48 positives). Conclusions are unchanged.

W	multi	val loss	val acc	Δ(multi−loss)	p_1	Δ(multi−acc)	p_1
100	0.938 [0.851, 1.000]	0.716 [0.578, 0.837]	0.580 [0.289, 0.822]	+0.222 [+0.067, +0.391]	0.004	+0.358 [+0.124, +0.641]	<0.001
200	0.989 [0.952, 1.000]	0.801 [0.637, 0.935]	0.895 [0.750, 0.989]	+0.188 [+0.065, +0.347]	<0.001	+0.094 [+0.011, +0.213]	0.018
300	0.977 [0.913, 1.000]	0.830 [0.694, 0.936]	0.852 [0.691, 0.976]	+0.148 [+0.051, +0.265]	<0.001	+0.125 [+0.023, +0.255]	0.003
500	0.983 [0.932, 1.000]	0.818 [0.674, 0.933]	0.821 [0.667, 0.936]	+0.165 [+0.064, +0.293]	<0.001	+0.162 [+0.059, +0.298]	<0.001

7 Discussion

Several observations help interpret the result:

Most predictive single signals are loss-related at small W, accuracy-related at larger W. At W=100, when val accuracy is still near chance, the strongest univariate predictors are loss-magnitude features (`end_val_loss`, `end_loss_gap`, `slope_train_loss`). By W=200, val-accuracy signals have caught up because the runs that will grok have begun to show very faint (1–15%) val-accuracy improvement. The multivariate predictor exploits both regimes.

The seeds of generalisation are present early. The cross-window pattern is monotone: more steps yield better predictions, but the marginal gain saturates by W=200. This is consistent with the mechanistic-interpretability picture [2, 3] in which generalising circuits begin forming much earlier than they begin to dominate the network's predictions.

Caveats.

- The regime studied is small (p=23, one transformer block, full-batch training, AdamW). Whether the same early signals work for non-algorithmic tasks, deeper models, or stochastic mini-batch training remains untested.
- The predictor is trained and evaluated on the same hyperparameter sweep. Out-of-distribution transfer to wholly new (λ, f) regimes is a natural next test.
- Forecasting a binary endpoint is a simplification; predicting *grokking time* is a richer (and more useful) problem we leave for follow-up.
- Our 10,000-step budget is shorter than some standard grokking setups; some "almost-grok" runs might have grokked by 50,000 steps. This is the reason for the relaxed 0.9 threshold; the strict-0.99 analysis (§6) confirms the conclusion.

8 Reproducibility

The codebase is organised into a small package (`src/`), a unit-test suite (`tests/`, 23 tests), and standalone experiment scripts (`experiments/`). All randomness is seeded at the run level and the data split. The full main experiment runs in roughly an hour on a single CPU thread.

References

[1] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, Vedant Misra. *Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets.* 2022.

[2] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, Jacob Steinhardt. *Progress measures for grokking via mechanistic interpretability.* ICLR 2023.

[3] Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, Ramana Kumar. *Explaining grokking through circuit efficiency.* 2023.

[4] Ziming Liu, Eric J. Michaud, Max Tegmark. *Omnigrok: Grokking Beyond Algorithmic Data.* 2022.

[5] Olivier Roy and Martin Vetterli. *The effective rank: A measure of effective dimensionality.* EUSIPCO 2007.

A Univariate predictor table

Table 3: Per-feature AUROC and $|\text{AUROC} - 0.5| + 0.5$ at W=200, sorted by predictive power. AUROC< 0.5 means *lower* values of the feature predict grokking.

feature	AUROC	$\lvert\text{AUROC} - 0.5\rvert + 0.5$
end_val_acc	0.845	0.845
slope_val_acc	0.818	0.818
end_val_loss	0.205	0.795
log_end_val_loss	0.205	0.795
slope_val_loss	0.213	0.787
end_loss_gap	0.216	0.784
max_loss_gap	0.224	0.776
slope_train_acc	0.768	0.768
slope_train_loss	0.239	0.761
end_acc_gap	0.266	0.734
mean_update_cos	0.303	0.697
loss_ratio	0.316	0.684
end_train_loss	0.674	0.674
log_end_train_loss	0.674	0.674